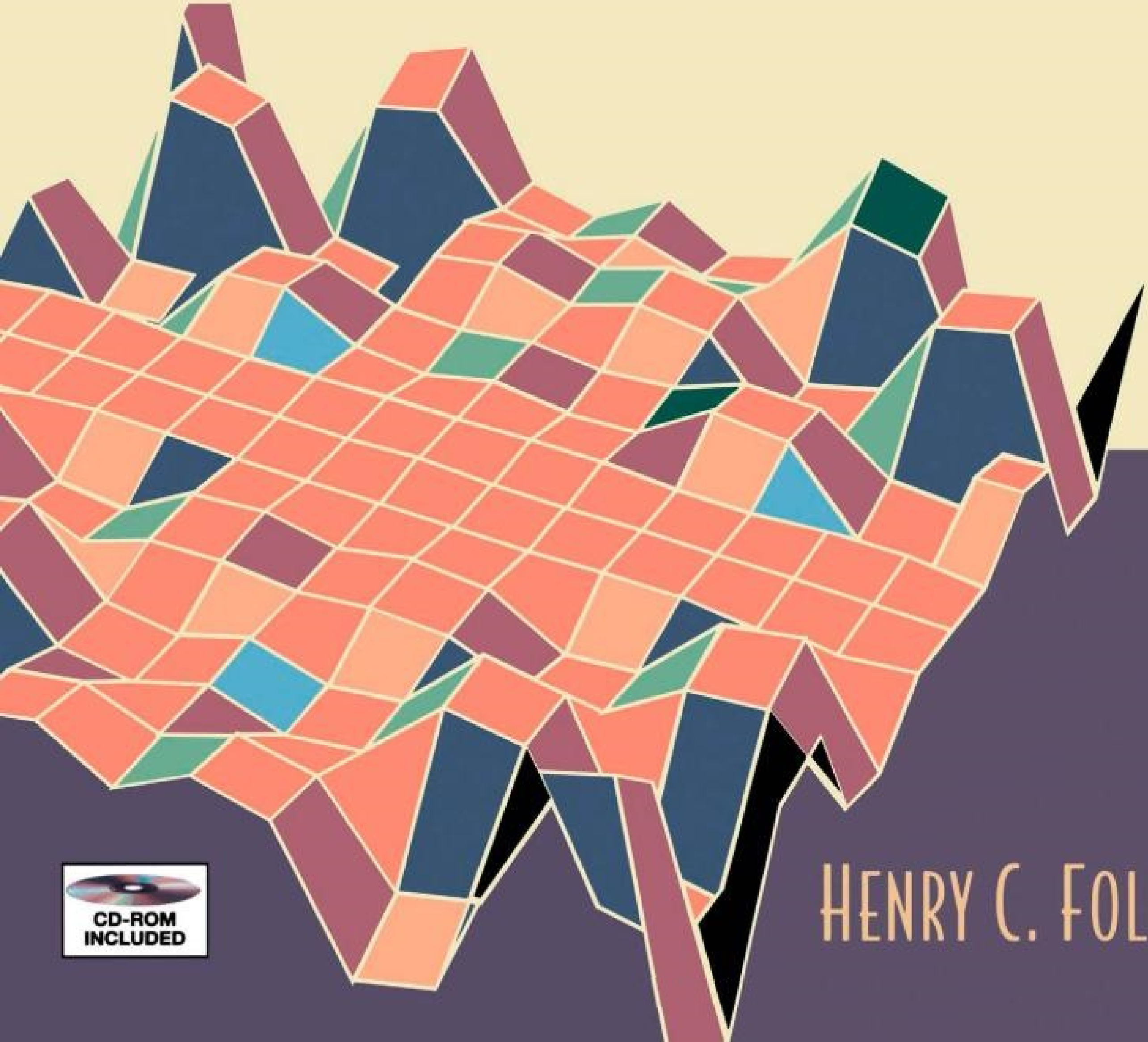
INTRODUCTION TO CHEMICAL ENGINEERING ANALYSIS USING MATHEMATICA®



CD-ROM INCLUDED

HENRY C. FOLEY

An Introduction to Chemical Engineering Analysis Using Mathematica

An Introduction to Chemical Engineering Analysis Using Mathematica

Henry C. Foley
The Pennsylvania State University
University Park, PA

ACADEMIC
PRESS

An imprint of Elsevier Science

San Diego San Francisco New York Boston
London Sydney Toronto Tokyo

This book is printed on acid-free paper. ♾

Copyright 2002, Elsevier Science (USA)

Academic Press
An imprint of Elsevier Science
525 B Street, Suit 1900, San Diego, California 92101-4495, USA
http://www.academicpress.com

Academic Press
An imprint of Elsevier Science
Harcourt Place, 32 Jamestown Road, London NWI 7BY, UK
http://www.academicpress.com

Library of Congress Catalog Card Number: 00-2001096535

International Standard Book Number: 0-12-261912-9

PRINTED IN THE UNITED STATES OF AMERICA
02 03 04 05 06 MV 9 8 7 6 5 4 3 2 1

For Karin, Erica, and Laura

Preface for an Instructor

This book is an experiment. To be precise, the book is not an experiment, but the approach of introducing and employing new concepts of chemical engineering analysis, concurrently with new concepts in computing, as is presented within this book, is experimental. Usually, the student of a first course in chemical engineering is presented with material that builds systematically upon engineering concepts and the student works within this linear space to "master" the material. In fact, however, the process is never so linear. For example, mathematics, in the form of geometry, algebra, calculus and differential equations, is either dredged back up from the student's past learning to be employed practically in the solution of material and energy balance problems or new math methods are taught along the way for this purpose. In fact a good deal of "engineering math" is taught to students by this means and not just at this introductory level — as it should be.

Therefore the critic might suggest that teaching computing simultaneously with introductory engineering concepts is not new, and instead simply adds, from the students' perspective, to the list of apparently "extra items" we already teach in a course and subject such as this one. That would be a fair criticism, if that were how this book had been designed. Fortunately, this book is intentionally not designed that way, but is instead designed with engineering and computing fully integrated — that is, they are introduced concurrently. I have purposely sought to avoid the simple addition of yet another set of apparently non-core learnings on top of the already long list of core learnings, by carefully staging the introduction of new computing methods with those of new types of engineering problems as they are needed. In this way the computing level rises with the engineering level in order to match the requirements of the problem at hand. Furthermore, the computing is not relegated to "gray boxes" or just to certain problems at the end of the chapter, but is integrated into the very text. By proceeding this way one actually leads the student and reader through a two-space of engineering and computing concepts and their application, both of which then reinforce one another and grow in sophistication with the complexity of the problem under consideration. However, this does not escape the fact that I have woven into the fabric of purely engineering material the new fibers of this computing. Why would I do so?

Simply put—I see many benefits to this approach, but will enumerate only a few. One major goal of the first course is to enable students to begin to do analysis. Doing so requires a formidable integration of skills from reading comprehension to physical conceptualization on through to mathematics and computing, and the student must do this and then run it all back out to us in a form that proves that he or she understood what was required. No wonder then that this first course is for many the steepest intellectual terrain they will encounter in the curriculum. It is simply unlike anything they have been called upon to do before! The student needs to be able to conceive of the physical or chemical situation at hand, apply the conservation of mass principle to develop model equations, seek the best method to assemble a solution to the equations and then test their behavior, most preferably against experiment, but short of that against logic in the limit cases of extremes of the independent variables or parameters. The first steps in this process cannot be facilitated by computing—the students must learn to order their thinking in a fashion consistent with modeling, that is, they must learn to do analysis. However, computing in the form of a powerful program such as *Mathematica* can facilitate many of the steps that are later done in service of the analysis. From solving sets of equations to graphically representing the solutions with systematic variations in initial conditions and parameters, *Mathematica* can do this better than a human computer can. So a major goal of the approach is to introduce computing and especially programming as a tool at an early stage of the student's education. (Early does not imply that the student be young. He or she could be a professional from another discipline, e.g., organic chemistry or materials science, who is quite experienced, but the material covered here may nonetheless be quite new to them and hence their learning is at an early stage.) The reason this is desirable is simple—programming promotes ordered thinking. Aside from the fact that computer codes allow us to do more work faster, this is typically hardly relevant to the beginning student for whom virtually every problem looks new and different, even if a more experienced eye sees commonality with previous problems, and for whom the problem rarely is number crunching throughput. Instead, the real incentive for learning to program is that in writing a few lines of code to solve a problem, one learns what one really does or does not know about a problem. When we seek to "teach" that to our CPU, we find our own deficiencies or elegancies, whichever the case may be, and that makes for good learning. Thus, ordering or disciplining one's thinking is the real advantage of programming in this way from an early stage—at least in my opinion. This need for ordered thinking is especially the case as the problems become more complex and the analyses tougher. By learning to program in the *Mathematica* environment, with its very low barriers to entry and true sophistication, one can carry over this ordered thinking and the methodologies it enables into other programming languages and approaches. In fact, it can be done nearly automatically for any piece of code written in *Mathematica* and which needs to be translated into C or Fortran code, etc. Computing and analysis begin to become more natural when done together in this way and the benefit is better thinking. Finally, I mentioned communication earlier as the last step in the process of analysis: I think that the *Mathematica* notebook makes an excellent medium for collecting one's thoughts and then communicating them back for others to read, understand, and even work with interactively.

In describing the approach, I have alluded to the benefits of *Mathematica* from my vantage and it seems appropriate at this point both to enlarge on this and give my reasons for choosing this program over others for the work we will do here. *Mathematica* is an astounding advance in computing. Within one environment one can do high-level symbolic, numeric and graphical computations. At the lowest level of sophistication it makes the computer accessible to anyone who can use a calculator and at its most sophisticated — it is a powerful programing language within which one can write high-level code. The width of the middle it provides between these two ends of the spectrum — computer as calculator and high-level production computing — is remarkable and worth utilizing more effectively. Beyond traditional procedural programming, one can use *Mathematica* to write compact, efficient, functional and rule-based code that is object oriented and this can be achieved with very little up-front training. It comes naturally as one uses the tool more completely. This functional and rule-based coding is a computational feature that truly makes the computer into the engineer's electronic work pad with *Mathematica* always present as the mathematical assistant. However, if one is to rely on an assistant then it better be a reliable assistant and one who can articulate reasons for failure when it cannot do something you have asked it to do. *Mathematica* is both. We have found that certain seemingly naive integrations that arise, for example, in the case of the gravity-driven flow from a draining tank can go awry in some programs when we attempt to solve these analytically and over regions in which they have discontinuities. When this happens students are rightly angry — they expect the software to get it right and to protect them from dumb mistakes; unfortunately, this is a serious mistake to make. This is one of the many fallacies I seek to hammer out of students early on because one has to test every solution the computer gives us in just the same ways we test our own hand-derived solutions. Yet we also do not want to find that we have to redo the computer's work — we want only to have to check it and hopefully go on. Both those graduate students who have worked with me as teaching assistants and I have found that *Mathematica* gave either inevitably reasonable results and comments as part of a problematic output or nothing at all — meaning the input was echoed back to us. The good news is that it is also relatively easy to check analytical solutions by the tried and true method of substituting back into the equation when using *Mathematica*. Otherwise daunting amounts of algebra are then a breeze and we never see it, unless we choose to, but the logical operations assure us that when the left-hand side equals the right-hand side we have arrived at a good solution.

An important outcome of this is that we can maintain continuity with the past within *Mathematica*, especially version 4.0 and beyond, in a way that is explicit and not achievable with packages that do only numerical computing. *Mathematica* does symbolic computing very well, better in fact than many (all?) of its human users. Although *Mathematica* is not the first symbolic computing package, it is one of the easiest to use and it is certainly the most advanced. Problems in analysis that were too tough to tackle analytically in the past can in many cases literally be solved now. However, the symbolic computing that made *Mathematica* so special is also well integrated with very powerful new numerical methods, which when combined with outstanding graphics capabilities create a complete computing environment. Hence a problem can be structured in such a way that by virtue of the constraints imposed it is readily soluble analytically, probably even by hand. But when the constraints are relaxed partially, the problem can still be solved analytically, but not readily by hand. Finally, the constraints can be nearly or fully removed and the problem admits no analytical solution, but is readily done numerically, which is almost as easy to convert to as is the procedure of changing from the statement DSolve to NDSolve. There are numerous examples of this kind in various contexts throughout the chapters of this book.

It is also worth mentioning what this book is not. It is not a book on *Mathematica* per se. There are many fine examples of this genre that have titles such as *Mathematica for the Scientist*, *Mathematica for the Engineer*, or *Learning Mathematica from the Ground Up*, all of which have already been published and are very well done. The most authoritative text on *Mathematica* is The *Mathematica Book*, by Steven Wolfram, so go to it when you need to do so. Remember that the Help menu will bring that book and other information directly to your monitor at any time. On the other hand, it is anticipated that many of the readers of this book will be tyros and will need some introduction to *Mathematica*. This is done in Chapter 1, which is in the form of a separate stand-alone primer at the beginning of the text. I have found that students and faculty who have read and used this chapter like it very much as a quick introduction. Through the next nine chapters new and more sophisticated *Mathematica* tools and programming techniques are introduced. Early on we are happy to have the student set up the models and run them interactively, employing a rudimentary toolset and the computer as a super-sophisticated calculator. By Chapter 8 the reader is encouraged to program at a more sophisticated level using, for example, Module, so that many calculations can be done, as well as rapidly and noninteractively through a wide range of parameter space. In the middle chapters tools are used that include solving differential equations analytically as well as numerically, solving sets of algebraic equations, also analytically and numerically, fitting models to data using linear and nonlinear regression routines, developing appropriate graphical displays of results, and doing procedural, functional and rule-based programming, and much, much more. Remember, however, that this is only the computing, and that we are also teaching engineering at the same time — so what is that content?

On the engineering content side, Chapter 2 begins with the word statement of the conservation of mass and its equivalent mathematical statement in the form of a rate equation. In teaching this material, it has been my experience that the conservation of mass needs to be introduced as a rate equation with proper dimensional consistency and not as a statement of simple absolute mass conservancy. Moreover, this must be done literally from day one of the course. The reasons are purely pedagogical. If mass conservation is introduced in terms that are time independent per the usual, then problems arise immediately. When rate equations are what is actually needed, but the statement has been learned in non-rate terms, there is an immediate disconnect for many students. The problems that come of this are readily predictable and usually show up on the first quiz (and often, sadly, on subsequent ones) — rate terms are mixed with pure mass terms, products of rates and times are used in place of integrals etc.

Therefore I do not start with the classical steady-state approach, but instead with rates and proceed to the steady state when it makes sense to do so, as a natural outcome of long-time behavior in a system with fixed inputs and outputs. From very simple examples of single component systems one can move to more complex problems including time-dependent flows and unique control volume geometries. Aside from being good fun, easy to visualize and downright interesting (Egyptian water clock design for instance), these problems accomplish two important goals: (1) they exercise the calculus while integrating in geometry and algebra; and (2) by design they focus on the left-hand side versus the right-hand side of mass balance rate equations. This works well too because it begins to build in the student's mind a sense of the real linkage between the physicality of the system and its mathematical description and where in the equations we account for issues of geometry versus those of mechanism of flow for example — a topic we cover explicitly in the subsequent chapter. The goal of this chapter and indeed the entire text is not just to assemble and solve these equations, but literally to "read" the mathematics the reader or someone else has written and in such a way that the equation or equations will tell you something specific about the system and that it will "say" what you want it to "say."

We rarely take the time in engineering to develop topics from an historical perspective — which is too bad. Our history is every bit as rich and the characters involved as interesting as any of those our colleagues in the humanities discuss. Why not talk about Fourier in Egypt with Napoleon for a little while when dealing with heat transport, or Newton's interesting and albeit bizarre fascination with the occult and alchemy, when discussing catalytic kinetics and diffusion? Doing so humanizes engineering, which is appropriate because it is as human an endeavor as philosophizing, writing, painting, or sculpting. Thus, Chapter 3 is an indulgence of this kind. From what I know of the story of Torricelli, his was a fascinating life. He was something like a modern Post-Doctoral Fellow to Galileo. He did for falling fluids what Galileo did for falling bodies, and of course so much more — which is fun to talk about because all of this was accomplished before Newton came along. In this chapter I take license in the way I present the "results" of Torricelli's experiments and his "work-up" of the data, but in essence it could not have been too far from this sort of thing — just a bit more grueling to do. I also find that this example works. It gets across the linkage between calculus and measurements in time — a linkage that is real and entirely empirical, but lost in much of our formal teaching of calculus. More important, we talk for the first time about the right-hand side and the fact that the mechanism of the flow or mass movement appears on this side of the equations. It also is the time and place to discuss the idea that not everything we need to complete a model comes to us from theoretical application of the conservation principle and that we may have to resort to experiment to find these missing pieces we call the *constitutive* relationships. Finally, we link the fundamental physics that students already know about falling bodies in a gravitational field to this topic through the conservation of energy. This shows that by applying a second and perhaps higher-order conservation principle to the problem, we could have predicted much of what we learned about Torricelli's law empirically, but Torricelli did not have the vantage point of four hundred years of Newtonian physics from which to view the problem.

To this point the problems have been rich, but lacking in the complexity that multiple components bring — namely multiple equations and eventually multiple coupled equations. Thus Chapter 4 introduces component material-balance rate equations. Much care is taken to present these equations as a subset which must sum to the overall material balance rate equation. The discussion moves to density effects and the expression of density as a function of concentration. This always takes time to work through. Students do not really understand density much beyond that which they learned in an introduction to physical science in eighth grade or thereabouts. The concept of concentration as taught at that point is also not on steady ground and is based solely on molarity for the most part. Having to deal with mass concentrations is one hurdle and then having to keep straight mass concentrations of individual components versus the total density is another and somewhat higher hurdle. However, it is surmountable if one takes the time to develop the concepts and to work out the mathematics of the coupled material balance. Throughout this chapter the assumption of perfect mixing within the control volume has been discussed and used both from the physical and mathematical points of view. The mathematics of the simple time-only dependent ordinary differential equations (ODEs) states that the system is well mixed with no spatial variation—so this is either the case physically, meaning that it is the case to as well as we can measure, or that it is approximately the case, meaning we can measure differences in concentration with position, but the differences are small enough to ignore, or it is really a bad approximation to the real system. For those seeking to bring in a bit more advanced concepts, say for an Honors student group, a section on mixing has been included here to get at these points more quantitatively. This section also shows some of the powerful objects that preexist in the *Mathematica* and which can be used creatively to solve problems and illustrate concepts.

At this point, the question that arises is whether to cover kinetics — batch, continuous stirred tank reactor (CSTR) and plug flow reactor (PFR) — next, and then to cover some problems in mass transfer later, or to do mass transfer first and then kinetics. The dilemma, if I may call it that, comes down to this. If one teaches kinetics first, the problems are all easily handled within a single phase, but the kinetics for the rate of chemical reaction become complex fairly quickly when one goes beyond the most rudimentary cases, which one wants, inevitably, to do. The simplicity of one phase then is offset by the complexity of nonlinearities in the rates. On the other hand, if one chooses to do mass transfer next, then one has immediately to introduce at least two phases that are coupled via the mass transfer process. However, the good news is that the mass transfer rate expressions are inherently linear and keep the math somewhat simpler. In the end I found that in teaching this material, and having taught it both ways, it was better to do mass transfer first because A remained A and B remained B throughout the problem even though they were moving between phases I and II. Linear transfer processes were easier for students to grasp than was A becoming B in the same phase, but by some highly nonlinear process. There is, I think, wisdom in "listening" to the ways in which the students tell us, albeit indirectly through their performances from year to year, how they learn better. Thus, this is why I present the material in this book in the way you see here — I was guided by the empiricism of the classroom and my own intuition derived therefrom. However, were the point to be pressed, I must state that I do not have hard outcomes data in hand (as of yet) to satisfy the unconvinced. In addition, when using a tool such as *Mathematica*, the issues of solving nonlinear versus linear systems mostly disappear and so it really is a toss up as to which to do first based on fundamentals. Hence Chapters 5 and 6 deal with mass transfer and then adsorption and both come before chemical kinetics and reactors. Adsorption is interesting to cover separately because one can get to a more molecular level and bring in physical chemistry concepts, as well as more complex rate expressions without chemical reaction. It is also very nice to distinguish mass action from mass transfer and to have the former in place before doing chemical kinetics, since one can then do interfacial kinetics with the proper physical foundation.

Chapters 7, 8, and 9 deal with chemical kinetics and idealized reactors. It should be quite familiar territory. Here as in previous chapters the focus is upon the interplay between analysis and experiment. Classical topics such as reaction stoichiometry are covered, but nondimensionalization is also introduced and taken up carefully with an eye toward its utility in the later chapters and of course in upper-level work. I also have found that rather than introducing the CSTR as a steady-state device, it makes more sense to develop the transient equations first and then to find the steady state at long time. Once one explains the benefits of this mode of reactor operation, it is moderately easy to see why we always use the steady-state algebraic equations. I also never fail to mention Boudart's point that it is easy to measure rates of chemical reaction with an experiment operated in a well-mixed stirred tank-type reactor. This another good time to teach the linkage between analysis and experiment with a system that is both quite easy to visualize and conceptualize. It is surprising to many of the better students that something as seemingly remote as that of the rate by which molecules are converted from one species to another at the nanoscale is so readily measured by quantities such as flowrate and conversion at the macroscale. That this should be the case is not obvious and when they realize that it is the case, well, it is just one of many such delightful epiphanies they will have during their studies of this discipline.

In teaching PFR, I find that the classical "batch reactor on a conveyor belt in heated tube" picture does not work at all (even though it should and does if you already get it). In fact, it leads some students in entirely the wrong direction. I am not happy when I find that the batch reactor equation has been integrated from zero to the holding time—even though it gives a good answer. Instead I very much favor taking one CSTR and rearranging the equation so that on the right-hand side the lead term is delta concentration divided by the product of cross-sectional area and a thickness (δz) and all this is multiplied by the volume flowrate. This becomes linear velocity multiplied by delta concentration over δz. Now we merely keep the total reactor(s) volume the same and subdivide it into n reactors with thickness $\delta z/n$. This goes over in the limit of δz taken to zero at large n to the PFR equation. We actually do the calculations for intermediate values of n and show that as n gets large the concentrations reach an asymptote equal to that which we can derive from the PFR equation and that for simple kinetics the conversion is larger than it would be for the same volume relegated to one well-mixed CSTR. This approach turns out to be fun to teach, seemingly interesting and actually useful, because the student begins to understand how a numerical algorithm works and that, for instance, the time-dependent PFR equation is a PDE that represents a set of spatially coupled time dependent ODEs.

Chapter 10, the last chapter, gathers together assignments and solutions that I have given to groups of honors-level students. I include these as further examples of what types of problems can be solved creatively and that these might serve as a catalyst for new ideas and problems. I also have homework, quiz and exam problems that I may eventually provide via the Internet.

Henry C. Foley
State College

Preface for the Student

In a place far away and long ago, people did calculations with paper, pencil, and slide rules. They wrote out papers, memoranda, and reports by hand and gave these to other people who would type them onto something called carbon paper in order to provide a copy of the work. In turn these could be duplicated on another machine called a mimeograph, the products of which were blurry, but had the sweet smell of ethanol when "fresh off the press." In about 1985 personal computers landed on our desks and things started to shift very fast. But many, even most people from this earlier era would still write out reports, memoranda, and papers in longhand and then either give it to someone else to "type into the computer," or if younger and lower in some ranking system do it themselves. The PC plus printer (first dot matrix, then laser) was used as an electronic combination of typewriter and mimeograph machine.

It took at least another few years before most of us had made the transition to using the computer as a computer and not as a typewriter. One of the greatest hurdles to this was being able to sit at the computer and enter your thoughts directly into a word processor program without "gathering your thoughts" first in a separate step. Even though this may seem absurd in hindsight, for those of us who grew up using pencil or pen and paper, we needed to adjust to the new technology and to retrain ourselves not to go blank when we sat in front of the computer. To my knowledge very few, if any, young people today whom I see ever do this or would consider doing it — they would consider it kind of absurd. They simply sit down and begin word processing. They make mistakes, correct them, then cut and paste, spell-check, grammar-check, and insert figures, tables, and pictures, etc. and paper is not involved until the last step, if at all. (The rendering of hardcopy step–by–step is becoming less necessary over time, which is a good trend — better to leave the trees out there to make oxygen and to soak up carbon dioxide than to cut them down for paper pulp — but it is still with us, despite the pundits' overly optimistic predictions of paperless offices and businesses.) Of course we all do this now — as I am currently doing. It is no big deal and it feels absolutely natural — now.

But it did not then. It felt strange and one wondered if it was even the right way to write. It was a very real paradigm shift.

Here is the point then: This same processing shift has never really happened in mathematics computing, at least not to the same extent, but it will. Most of us still work things out first on paper and then find a way to do number crunching on the computer, well after many other steps have been taken. This is why we see, for example, the use of spreadsheet programs having proliferated among engineering students over the last few years. They work out a model, derive the solution as analytical expressions, and plug them into the spreadsheet to make calculations for a given set of parameters. The analysis is done separately from the computing, in the same way we used to do writing separately from typing. It is the combined task that we now call word processing. The point of this book is to step away from that old, separated analysis and computing paradigm, to put down the pencil and paper (not completely or literally), and to begin electronically scribbling our mathematically expressed ideas in code by using up-to-date computational software. If there is any reason why this transition happened so much faster in word processing than in mathematics processing, it is because word processing software is less complex and mathematics "scribbling" is generally harder to do than is drafting a written document (not creative writing of course).

At this point I think we may have turned the corner on this shift. The mathematics processing software is so sophisticated that it is time to both embrace and use it — in fact students in engineering and science have, but not always with good results. We need to fix this problem and to do so, it makes very little sense to teach analysis in one place (course) and computing in another place (another course), when we can do the two concurrently. To do this requires a fully integrated environment, with symbolic, numeric and graphical computing and, surprisingly, word processing too. *Mathematica*, especially version 4.0 and beyond, does this extremely well, so it makes sense to use it. One review of the software written in *Science* magazine in Decompiler 1999, referred to *Mathematica* as the "Swiss army knife" of computing. In fact, I think it is much better than that analogy suggests, but the author meant that it is a high-quality and versatile tool.

In this book then you will find the concepts of engineering analysis as you find them elsewhere, but they will be presented simultaneously with the concepts of computing. It makes little sense to separate the two intellectual processes any longer and lots of sense to teach them as an integrated whole. In fact, this approach relieves the overburden of algebraic manipulation which I and others like me used to love to fill chalkboards with and it puts the emphasis back on engineering. Not a bad outcome, but only if we do it right. Here is the danger — that you will use the computer without thinking deeply, derive bad results, and go merrily on your way to disaster. This sounds absurd to you but it is not. For example, the public recently has had played out before its eyes just such an engineering snafu. A NASA space probe was sent crashing to its fiery demise because someone had failed to convert from feet to meters (i.e., English to metric system) in a trajectory calculation. A big mistake in dollar terms, but just a small mistake in human terms — the kind students often argue are not real mistakes and should be the source of at least partial credit when committed on exams or homework. Similarly, a bridge under construction near to where I am writing this was begun from two different ends and when it came time to close the gap with the final element of the structure, it could not be done — the two sides were not properly aligned. This happened despite the engineers having tools like lasers and computers at their disposal, which is really shocking given the shortness of the span and given that mighty gorges were spanned correctly in the late nineteenth century with nothing more than transits, plumb lines, and human computation! So whenever something new such as this tool is introduced something is gained, but inevitably we find later that something is also lost. This gives thoughtful people pause, as well it should. Therefore, to use this tool correctly, that is to do this right, we have to do things very carefully and to learn to check quite thoroughly *everything* that the computer provides. This is especially the case for analytical solutions derived via symbolic computation. If you follow the methods and philosophy of this text I cannot guarantee you will be error free because I am sure the text is not error free despite my best efforts, but you will definitely compute more safely and will have more confidence in your results.

The best way to use this book is in conjunction with *Mathematica*. Go through the first chapter and then try doing one of the things presented there for your own work or problems. Moving through the rest of the text will go faster if you take the time to do this up front. A nearly identical color version of this book has been provided on CD-ROM. I hope having this and being able to call it up on your computer screen while you have a fresh *Mathematica* notebook open will be useful to you and will aid your learning. Although it may be obvious, just reading this book will probably not do enough for you — you have to use the tool. If you own or have access to *Mathematica*, then you will be able to use the book as a progressive resource for learning how to program and how to solve real problems in real time. Good luck and happy concurrent computing and engineering analysis.

Henry C. Foley
State College

Acknowledgments

Along the way I have many people to acknowledge. If all of this bears more than a faint resemblance to the philosophy espoused in the earlier book *An Introduction to Chemical Engineering Analysis*, by T.W.F. Russell and M.M. Denn, well it should. I taught the introductory course many times at the University of Delaware from 1986 to 2000 and I always did so in conformity with this marvelous text. In particular, Fraser Russell taught me how to teach this material and what the original intent had been of the book and its approach. I was always very impressed by the stories he told of the time he and Mort spent on this topic thinking about their book and its philosophy through to the classroom. Fraser's enthusiasm for these matters was, as far as I could tell, limitless and his enthusiasm infectious. And as I arrived on the scene in 1986 as a Ph.D. in Physical Chemistry and not Chemical Engineering, I can attest to the efficacy of learning this approach—although I hope the reader is not learning the material literally the night before giving the lectures on it, as the present author did! In many ways I came to Delaware as a bit of blank slate in this regard (although I had read the original Notes on Transport Phenomena by Bird, Stewart, and Lightfoot while working at American Cyanamid between 1983 and 1984) and I had no preconceived notions about how this material should be taught. To say the least I enjoyed an excellent mentor and teacher in Fraser Russell and he did harbor a few notions and opinions on how this material should be taught. Fraser also gave me the push I needed to start this project. He realized that computation had come far and that one impediment to wider adoption of his book had been the steep gradient of mathematics it presented to both the instructor of the first course and the students. Using a computational tool to overcome this barrier was something we both felt was possible. However, when this was first conceived of in the late 1980s (~1988), *Mathematica* 1.0 was barely out on the market and it, as well as the other tools available, were in my judgment not up to the task. (Though I tried at that time.) The project was shelved until my first sabbatical leave in 1997. I must thank Dr. Jan Lerou, then of the Dupont Company's Central Research and Engineering Department at the Experimental Station, who provided me with an office and the wherewithal to start this project in the spring and summer of 1997. In fact, although I did get this book project off the ground, I was not at all happy with it. As my familiarity with the new version of *Mathematica* (4.0) grew during late 1998 and 1999, I realized I had to rewrite that which I had already written. As well, the experience of working with Honors ChE students helped me immensely to reconceptualize the material and its presentation. Furthermore, I had the good fortune to co-teach the first course in chemical engineering with Andrew Zydney. Andrew is a great teacher and he was the first person I had met who would literally battle me for lecture time in front of the class. This not only gave me more time to work out more ideas, but he also provided invaluable criticism and feedback on what I was trying to do. In the summer of 1999, I had the privilege of being a Visiting Fellow at Wolfram Corporation, the makers of *Mathematica*. Aside from having the good fortune to meet Steve in his own think tank, I spent six weeks alone that summer in Urbana–Champaign writing literally day and night with very few breaks. (I thank my spouse Karin for allowing such an absurd arrangement!) But I also had access to the brilliant young staff members who work every day on the new code and features of *Mathematica*. It was a broadening experience for me and one I thoroughly enjoyed. For making this possible, I want to thank Steve Wolfram personally, but also Lars Hohmuth, the jovial and ever helpful Director of Academic Affairs at Wolfram, who is also a great code writer and a power user! (He would spend his days doing his job and then get caught by me on his way out for the evening, only to spend hours answering what was a seemingly naive question — which usually began as “Lars, have you got a minute?”) In the latter stages of the work, I ran into a few issues associated with notebook formatting and answers to those questions always came to me promptly as exceptionally well written e-mail messages from P. J. Hinton, one of the many younger chemical engineers who have found their way into careers in computation. Finally, in the Spring of 2000, I had the opportunity to teach the whole of the book and its content as the first course in chemical engineering here at Pennsylvania State University. My partner in that was Dr. Stephanie Velegol. Stephanie is only the second person whom I have had to fight for lecture time in a course and whose suggestions and methods of using these things I had created were extraordinarily insightful. (I am pleased to note that the course was well received — largely due both to her efforts to smooth out the rough edges of both the materials and her co-instructor and to her pedagogical instincts, whose instincts told her when enough was enough.) Finally, throughout my career I have had the best of fortune to have a life-partner, my spouse Karin, who knows and understands what I am about and what I really need to do and get done. For her support and that of my daughters, Erica and Laura, who often strolled into my home office to find me hunched over the computer and to ask how my book was coming along, I offer my sincerest and deepest thanks.

The book represents a way to teach a first course in chemical engineering analysis that I think maintains a continuity with the past and yet steps right into the future with concurrent use of computational methods. The book and its techniques are battle tested, but are far from battle hardened. I am sure that there remain mistakes and misconceptions that will need to be considered, despite my best efforts to eliminate them and for those I take full blame and apologize in advance. Yet, I think there are seeds in this book from which can grow a new

and fruitful approach to teaching engineering analysis. The simple fact is that our students like using and being at the computer, perhaps more so than they enjoy hearing us lecture. We are going to have to face this paradigm shift, embrace it, and somehow integrate it into our pedagogy. To that end this book is my attempt to do so. I think the book may be used either as a textbook in its own right or as a supplementary textbook. I recommend that students each have a personal copy of *Mathematica* 4.0 or higher, which is moderately priced (about the same price as a textbook) or that they have ready access to the program in a centralized computer lab. In recent years I have gotten into the habit of sending homework out to students as *Mathematica* notebooks attached to an e-mail and then I also post the problem set and solutions on the course web site as notebooks. I also frequently receive via e-mail attached notebooks from students who are stuck or need some guidance. I personally like this approach because it allows me the opportunity to interact with the more motivated students at a higher level and in essence to e-tutor them on my own schedule. If you do this be prepared to be answering e-mails by the dozens, frequently and at all times of the day and night and week. Personally, I find it rewarding, but I can understand that some might consider this to be an imposition. I do, however, think this is more like the direction in which teaching will move in the future — the use of these electronic media technologies in real time seems to me to be inexorable and overall a good development — at least from the student's perspective.

Finally, who else can use this book? I clearly have in mind chemical engineering undergraduates, but they are not alone in potentially benefiting from exposure to this material. It seems as though industrial chemists and materials scientists could also find it useful to read and study on their own with a personal or corporate copy of *Mathematica*. I consider this level of self-study to be a very doable proposition. The mathematics used is fairly minimal, although it does expect grounding in differential equations and some intuitive sense of programming, but that is about all it requires. Formality is kept to a minimum — no — more precisely there is no mathematical formalism present here. For this reason then, I would hope that a few people in that category who really want to be able to discuss research and development matters with corporate chemical engineers on their own terms will find this background to be very useful. Finally, I suspect from my frequent excursions in consulting that there may be more than a few practicing chemical engineers who might not want to be seen actually reading this book in the open, but who might also benefit from having it on their shelves so that they might read it — strictly in private of course!

CHAPTER 1

A Primer of Mathematica

1.1 Getting Started in Mathematica

We will use *Mathematica* throughout the text. Most of what is necessary to know will be introduced at the time it is needed. Nonetheless, there is some motivation to begin with some very basic commands and structures so that the process is smooth. Therefore, this is the goal of this section—to make your introduction to *Mathematica* go smoothly. For more information of this type there are many texts that cover *Mathematica* in detail.

1.2 Basics of the Language

Commands in *Mathematica* are given in natural language form such as “**Solve**” or “**Simplify**” etc. The format of a command is the word starting with a capital letter and enclosing the argument in square brackets:

Command[argument]

Parentheses are used arithmetically and algebraically in the usual way:

$$3a(x-2)^2$$

On the other hand, braces are distinct. They are used to designate lists or vectors as in:

{1, 2, 3, 4 . . .}
{{1, 1}, {2, 2}, {3, 3} . . .}

The three must not be interchanged.

When you want to clear the value of a given named variable there are three options:

variable name =.

Clear[variable name]

Remove[variable name]

The first two simply clear the current value while the last removes the name entirely. You need to remember this because if you start a session and assign a value to a variable, then that value will be retained in that variable until you either change or clear it.

1.3 Simple Commands

The calculator level of *Mathematica* comes in the form of Palettes, which are very handy tools. Palettes are found under the **File** menu and there are several of them. If one wants to use a trigonometric function, for example, we can either type in its name or go to the **Basic Calculations** menu and then to the Trigonometric and Exponential Functions. Should we want to evaluate the sine of 2.3333π, then we can do so as follows:

```
In[1]:= Sin[2.33333π]

Out[1]= 0.86602
```

Should we need to know the sine of 120 degrees (120°), then we include this in the argument of the function:

```
In[2]:= Sin[120 Degree]
```

Out[2]= $\frac{\sqrt{3}}{2}$

To rationalize this fraction we need to evaluate it numerically. We do so by surrounding the **Sin** function with **N**:

```
In[3]:= N[Sin[120 Degree]]

Out[3]= 0.866025
```

For logarithmic and all other functions, we do the same as we have done with **Sin**. It is important to know, however, that the function **Log** in *Mathematica* is the natural logarithm and not the log base ten.

```
In[4]:= N[Log[10]]
        N[Log[100]]
        N[Log[1000]]

Out[4]= 2.30259

Out[5]= 4.60517

Out[6]= 6.90776
```

Mathematica has a huge number of built-in functions from the mundane to the exotic, but we can work with them more or less in the same way.

1.4 Table, Plot, Map, and Plot3D

These four commands are among the most useful because they do so much with so little. In contrast to a procedural language in which we would have to write a looping structure to evaluate a function at several different values or for a range of values, **Table** hides all this from us and gives us just the vector of output values. **Plot** does the same thing, except that we see the graph of the function's values rather than the values themselves. The output of a function is a **List**, that is, a vector. We can combine a set of values, such as the set of dependent values, with a set of independent variable values into a matrix. **ListPlot** allows us to display these results graphically. We will begin by working through these four topics. We begin with the **Table** command to evaluate *x* from zero to 20 at every integer:

```
In[7]:= Table[x^2, {x, 0, 20}]

Out[7]= {0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169,
         196, 225, 256, 289, 324, 361, 400}
```

Perhaps we wanted the values of x^2 for every other whole number between 0 and 20. We can obtain these too:

```
In[8]:= Table[x^2, {x, 0, 20, 2}]

Out[8]= {0, 4, 16, 36, 64, 100, 144, 196, 256, 324, 400}
```

Should we need all the values for every integer value and the midpoint between them, we would specify this:

```
In[9]:= Table[x^2, {x, 0, 20, .5}]

Out[9]= {0, 0.25, 1., 2.25, 4., 6.25, 9., 12.25, 16., 20.25, 25.,
         30.25, 36., 42.25, 49., 56.25, 64., 72.25, 81., 90.25,
100., 110.25, 121., 132.25, 144., 156.25, 169., 182.25,
196., 210.25, 225., 240.25, 256., 272.25, 289., 306.25,
324., 342.25, 361., 380.25, 400.}
```

It is also likely that we might need to assign this list or vector a name, call it "**ls1**."

```
In[10]:= ls1 = Table[x^2, {x, 0, 20, .5}]

Out[10]= {0, 0.25, 1., 2.25, 4., 6.25, 9., 12.25, 16., 20.25, 25.,
          30.25, 36., 42.25, 49., 56.25, 64., 72.25, 81., 90.25,
          100., 110.25, 121., 132.25, 144., 156.25, 169., 182.25,
          196., 210.25, 225., 240.25, 256., 272.25, 289., 306.25,
          324., 342.25, 361., 380.25, 400.}
```

This variable name is now assigned to this list until we either clear it or remove it:

```
In[11]:= ls1

Out[11]= {0, 0.25, 1., 2.25, 4., 6.25, 9., 12.25, 16., 20.25, 25.,
          30.25, 36., 42.25, 49., 56.25, 64., 72.25, 81., 90.25,
          100., 110.25, 121., 132.25, 144., 156.25, 169., 182.25,
          196., 210.25, 225., 240.25, 256., 272.25, 289., 306.25,
          324., 342.25, 361., 380.25, 400.}
```

In the next line we clear **ls1** and then show that it is no longer assigned to the list of values:

```
In[12]:= ls1 =.
         ls1

Out[13]= ls1
```

We may also have occasion to want to generate the vector of values and to assign these values to a list name, but we may not want to see all of them. For example, suppose we wanted all the values for x^x from between 1 and 100. This can be done and the list can be named, but we may not want this sent to the screen. To suppress it we place a semicolon after the command:

```
In[14]:= ls2 = Table[x^x, {x, 1, 100}];
```

To see what we have "missed" by not printing this out to the screen we can now do so by typing **ls2** as input:

```
In[15]:= ls2

Out[15]= {1, 4, 27, 256, 3125, 46656, 823543, 16777216, 387420489,
          10000000000, 285311670611, 8916100448256, 302875106592253,
          11112006825558016, 437893890380859375, 18446744073709551616,
          827240261886336764177, 39346408075296537575424,
1978419655660313589123979, 104857600000000000000000000,
5842587018385982521381124421,
341427877364219557396646723584,
20880467999847912034355032910567,
1333735776850284124449081472843776,
88817841970012523233890533447265625,
6156119580207157310796674288400203776,
443426488243037769948249630619149892803,
33145523113253374862572728253364605812736,
2567686153161211134561828214731016126483469,
205891132094649000000000000000000000000000000,
17069174130723235958610643029059314756044734431,
1461501637330902918203684832716283019655932542976,
129110040087761027839616029934664535539337183380513,
11756638905368616011414050501310355554617941909569536,
1102507499354148695951786433413508348166942596435546875,
106387358923716524807713475752456393740167855629859291136,
1055513495577778341407833008599583294612739608337019944251
 17,
1075911801979993982060429252856123779115487368830416064610
 0304,
1125951474620711925397894489888890599301921052191965170090
 951959,
1208925819614629174706176000000000000000000000000000000000
 00000000,
1330877630632711998713399240963346255985889330161650994320
 5137953641,
1501309375452965723567719721642544578140479705687387772350
 893533016064,
1734377336703026751990378128881203215830806253901209195300
 77767198995507,
2050773823560610053645205609172376035486179836520607547290
 4916966189367296,
2480636444513411454946491823954126897445305814926541643210
 720600128173828125,
3068034630079427423066043364764039789978817064507885328000
 82659754365153181696,
3877924263464448622666648186154330754898344901344205917640
 2325627886496385062863,
5007020782634593191745370252495708882467099553774002230210
 257741084821677152403456,
660097246862195508437683218183717716501470040592780694068⋱
 1419043656513182932506244 9,
888178419700125232338905334472656250000000000000000000000⋱
 000000000000000000000000000,
121921130509464847947319348187292783466757699259377071718⋱
 929822528439954197720823131505 1,
170676555274132171974277914691501574771358362295975962674⋱
 353045737940041855191232907575296,
243568481650227121324776065201047255185334531286856408445⋱
 0513087957672060915022330125615037 3,
354211804501063924032848133753332071263980863803681247321⋱
 1109743262552383710557968252383789056,
524744532468751923546122657597368049278513737089035272057⋱
 32464366860767768230289220809936523437 5,
791643248668629666078424060180632546719222453126466902233⋱
 624029184841704241043101695525920503234 56,
121581297366713640808862801923521362803054459089854018769⋱
 9033580010768658602308137775436770485568805 7,
190030638094159447976388394485939490393342173391549735102⋱
 6033862324967197615194912638195921621021097984,
302182066535432255614734701333399524449282910532282724655⋱
 138380663835618264136459996754463358299552427939,
488736779806892574893227522737746038656608501760000000000⋱
 000000000000000000000000000000000000000000000000,
803748056254594377406396163843525813945369338299102331167⋱
 037964742945238909157063019657136804802094856043166 1,
134364564515225004658302677932296937303529095376341154029⋱
 0906502671301148502338015157014479136799509522304466944,
228273036346967044979900512337165522400819024722490933829⋱
 95479307326771731500413559064280268724685077157913834284 7,
394020061963944792122790401001436138050797392704654466679⋱
 482934042457217714972106114142662548849156408066279903 06⋱
 816,
690825216476092085140553869446828608223037872425945418628⋱
 911729772998712910490187733003608627768699079751968383 78⋱
 90625,
122998480353523742535746057982495245384860995389682130228⋱
 631906566920771227021327602280884021030694269236652956 94⋱
 53244416,
222337020242360576812569226538683753874082408437758291741⋱
 262115823894811650848346334502642370010973465496690788 65⋱
 0052277723,
407949179542747833144743894229635944120105534129541880466⋱
 65939634971631296545460720786532465498226465248060567545⋱
 587093733376,
759604031216329727422244257820804323611227904183944130804⋱
 55142035956380302831768235397935875913722302301039331108⋱
 10192201741429,
143503601609868434285603076356671071740077383739246066639⋱
 24900000000000000000000000000000000000000000000000000000⋱
 00000000000000000,
275006373483461607657434076627252658495183350017755660813⋱
 75398177450890599808191940514056884835339723379661819264⋱
 5698819765129996471,
534490195473619995340253001400575385449406013931066115702⋱
 69540644280818850419033099696863861289188541180498511377⋱
 3393623416423223132 16,
105334051468072867203736594605020607857593791122125981160⋱
 64998418834781689316645387966435364502141349866164216580⋱
 5956097883251900620138 33,
210449190758543198861850228434282880911748656012122526352⋱
 86001514565478992866160785568445711391305050636166445827⋱
 73621942951905668236312576,
426181657761258833198605424151960757395791315610122269092⋱
 30019917908804339283405158889618455726386574838882026483⋱
 5885609500110149383544921875,
874647407767330977693561259365719780492040872417198817613⋱
 46374524717952404307119962211675102409649648957510056235⋱
 276523073007403698815894552576,
181880373878061983792773399155569296478074032831870486314⋱
 78337739929618787870634227045716719924575689062274471430⋱
 3688653882035406726660425309967 97,
383158981231346126213872650000641426814753403789311551232⋱
 59089391706871851454385790069500821953097058851346079904⋱
 186656073376329737705072368434544 64,
817598737071050959409276229318696698168591900537987468276⋱
 93207376890191209667334279321765760731642396831372649256⋱
 66736782739235660867861215513397759 19,
176684706477838432958329750074291851582748389687561895812⋱
 16062012926197760000000000000000000000000000000000000000⋱
 0000000000000000000000000000000000000000,
386621969787156332734047587900743169602142130961783196218⋱
 56934259807530937321861485192508542873470637501160980081⋱
 794035970219670238407078788135931371782481,
85651681910278991338310088485588763860782786752514138917⋱
 58617169692971014784447542255823577266886455881314507547⋱
 31704968996267139619369035601073162078388224,
192079787778504229782687634239832998136662613890310670723⋱
 96386230620731601620304963544415541870751106508384494531⋱
 08757445590084411555537438824653742747212640587,
435973436827325522360279881406914796368935566412408014666⋱
 80104726695921400093636969731839732875229357313838872128⋱
 9594366953995072735552848220101541587045199118336,
100140253328453899494506997059845948876248360208192710258⋱
 70334010718860779315506363581151510555924043061907775739⋱
 0331456723193970237417715907213278114795684814453125,
232737736870108098051032630552618777391020715805979404095⋱
 85933109624493442480014587281684425109432546907773222375⋱
 549181098538730989934437386098275807854764894176935936,
547236400751580609289084096221336193364655786735995545755⋱
 43693463433762205742631692905663619249992774511988021569⋱
 50364045812455566817070274944448633167362192918054601383,
130159283494297205518264830741731536453872507596006782791⋱
 53114847224523409663172158051068209591908333097049343465⋱
 17741237438752456673499160125624414995891111204155079786⋱
 496,
313119843606264301926053344163576361349265045995115651405⋱
 92969760191406230933171722203767186869842061905370495649⋱
 99303230341738506627657379866724844088015857197961365923⋱
 84409,
761773480458663923392897277206155617504248014023951967240⋱
 01565744957137343033038019601000000000000000000000000000⋱
 00000000000000000000000000000000000000000000000000000000⋱
 0000000,
187398754970444035883430239799421909138706990995859221061⋱
 52367184893220649019310617359174987694158429118066514085⋱
 32784617787067474359792929997061205566219581733294857302⋱
 9136642691,
466101087036369642390596621400310098213235393780243962934⋱
 25774112018587400879035854022570174490255580463084035551⋱
 28684298484146339920553893653953988411898447534660818749⋱
 990933364736,
117196384926544421041758258775124882470814614810980971003⋱
 33153423591117017616566024314352960493587163785179678960⋱
 50409107202745103300944452206991034477139649315017364735⋱
 008987336482893,
2978641516052715656715226918884874333982014782141043748 36⋱
 86344802018942169740653764805241893613019586796641682947⋱
 70215036703035475694094363170727692463342462659692676989⋱
 28260777661956096,
765142811538184924971089105229239398896084485704278030436⋱
 46059567958108943618778356292728753731576478313833091931⋱
 62363541428604718717978398581939982608934869290351343806⋱
 833028793334960937 5,
198627040519827975805761256394776123747083228931514412339⋱
 85491658847582706097318376646920317555554524971459613579⋱
 56707789253279272215867715207123334756347457728787131439⋱
 889933248847863716249 6,
521024593971836146804821104841449602253438957603391316494⋱
 00299130165682155803982962610720192317232798510072418380⋱
 11659882766685337218633992220688288491655299087016195985⋱
 2052183477115784857447 37,
138087834126148675065691180325230972687660410568672963807⋱
 27295432437014796705930332110080014435366263105359800775⋱
 44691196522513327846303307992442770355560270350429006522⋱
 58843340460238799209129574 4,
369729637649726772657187905628805440595668764281741102430⋱
 25997242355257045527752342141065001012823272794097888954⋱
 83265401194299967694943594516215701936440144180710606676⋱
 593013849997799991592004998 99,
100000000000000000000000000000000000000000000000000000000⋱
 00000000000000000000000000000000000000000000000000000000⋱
 00000000000000000000000000000000000000000000000000000000⋱
 00000000000000000000000000000000}
```

This leads to several other points. First, we could also operate on all of the values in **ls2** by operating on **ls2** alone. This property of **ls12**, called “listability,” is a very important attribute of such objects in *Mathematica*. For instance, then we could divide each value we just found by dividing **ls2** by 10^{100}:

```
In[16]:= N[ls2/10^100]

Out[16]= {1. × 10^-100, 4. × 10^-100, 2.7 × 10^-99, 2.56 × 10^-98, 3.125 × 10^-97,
          4.6656 × 10^-96, 8.23543 × 10^-95, 1.67772 × 10^-93, 3.8742 × 10^-92,
          1. × 10^-90, 2.85312 × 10^-89, 8.9161 × 10^-88, 3.02875 × 10^-86,
          1.1112 × 10^-84, 4.37894 × 10^-83, 1.84467 × 10^-81, 8.2724 × 10^-80,
          3.93464 × 10^-78, 1.97842 × 10^-76, 1.04858 × 10^-74, 5.84259 × 10^-73,
3.41428 × 10^-71, 2.08805 × 10^-69, 1.33374 × 10^-67, 8.88178 × 10^-66,
6.15612 × 10^-64, 4.43426 × 10^-62, 3.31455 × 10^-60, 2.56769 × 10^-58,
2.05891 × 10^-56, 1.70692 × 10^-54, 1.4615 × 10^-52, 1.2911 × 10^-50,
1.17566 × 10^-48, 1.10251 × 10^-46, 1.06387 × 10^-44, 1.05551 × 10^-42,
1.07591 × 10^-40, 1.12595 × 10^-38, 1.20893 × 10^-36, 1.33088 × 10^-34,
1.50131 × 10^-32, 1.73438 × 10^-30, 2.05077 × 10^-28, 2.48064 × 10^-26,
3.06803 × 10^-24, 3.87792 × 10^-22, 5.00702 × 10^-20, 6.60097 × 10^-18,
8.88178 × 10^-16, 1.21921 × 10^-13, 1.70677 × 10^-11, 2.43568 × 10^-9,
3.54212 × 10^-7, 0.0000524745, 0.00791643, 1.21581, 190.031,
30218.2, 4.88737 × 10^6, 8.03748 × 10^8, 1.34365 × 10^11,
2.28273 × 10^13,  3.9402 × 10^15, 6.90825 × 10^17, 1.22998 × 10^20,
2.22337 × 10^22,  4.07949 × 10^24, 7.59604 × 10^26, 1.43504 × 10^29,
2.75006 × 10^31,  5.3449 × 10^33, 1.05334 × 10^36, 2.10449 × 10^38,
4.26182 × 10^40,  8.74647 × 10^42,  1.8188 × 10^45, 3.83159 × 10^47,
8.17599 × 10^49,   1.76685 × 10^52, 3.86622 × 10^54, 8.56517 × 10^56,
1.9208 × 10^59,  4.35973 × 10^61, 1.0014 × 10^64, 2.32738 × 10^66,
5.47236 × 10^68,  1.30159 × 10^71,  3.1312 × 10^73, 7.61773 × 10^75,
1.87399 × 10^78,  4.66101 × 10^80,  1.17196 × 10^83, 2.97864 × 10^85,
7.65143 × 10^87,  1.98627 × 10^90, 5.21025 × 10^92, 1.38088 × 10^95,
3.6973 × 10^97,   1. × 10^100}
```

We also note that the output from this last computation is in scientific notation, whereas **ls2** was not; it was written in standard form. It is worth noting that we could have had **ls2** in scientific notation simply by changing the command as follows:

```
In[17]:= ls3 = Table[x^x, {x, 1, 100, 1.}]

Out[17]= {1, 4., 27., 256., 3125., 46656., 823543., 1.67772×10^7, 3.8742×10^8, 1.×10^10,
   2.85312×10^11, 8.9161×10^12, 3.02875×10^14, 1.1112×10^16, 4.37894×10^17, 1.84467×10^19,
   8.2724×10^20, 3.93464×10^22, 1.97842×10^24, 1.04858×10^26, 5.84259×10^27, 3.41428×10^29,
   2.08805×10^31, 1.33374×10^33, 8.88178×10^34, 6.15612×10^36, 4.43426×10^38, 3.31455×10^40,
   2.56769×10^42, 2.05891×10^44, 1.70692×10^46, 1.4615×10^48, 1.2911×10^50, 1.17566×10^52,
   1.10251×10^54, 1.06387×10^56, 1.05551×10^58, 1.07591×10^60, 1.12595×10^62,
   1.20893×10^64, 1.33088×10^66, 1.50131×10^68, 1.73438×10^70, 2.05077×10^72,
   2.48064×10^74, 3.06803×10^76, 3.87792×10^78, 5.00702×10^80, 6.60097×10^82,
   8.88178×10^84, 1.21921×10^87, 1.70677×10^89, 2.43568×10^91, 3.54212×10^93,
   5.24745×10^95, 7.91643×10^97, 1.21581×10^100, 1.90031×10^102, 3.02182×10^104,
   4.88737×10^106, 8.03748×10^108, 1.34365×10^111, 2.28273×10^113, 3.9402×10^115,
   6.90825×10^117, 1.22998×10^120, 2.22337×10^122, 4.07949×10^124, 7.59604×10^126,
   1.43504×10^129, 2.75006×10^131, 5.3449×10^133, 1.05334×10^136, 2.10449×10^138,
4.26182×10^140, 8.74647×10^142, 1.8188×10^145, 3.83159×10^147, 8.17599×10^149,
1.76685×10^152, 3.86622×10^154, 8.56517×10^156, 1.9208×10^159, 4.35973×10^161,
1.0014×10^164, 2.32738×10^166, 5.47236×10^168, 1.30159×10^171, 3.1312×10^173,
7.61773×10^175, 1.87399×10^178, 4.66101×10^180, 1.17196×10^183, 2.97864×10^185,
7.65143×10^187, 1.98627×10^190, 5.21025×10^192, 1.38088×10^195, 3.6973×10^197, 1.×10^200}
```

By changing the increment from 1 (the default value) to 1., we have gone over from integers to rational numbers, and when the latter are called for, then in this case *Mathematica* uses scientific notation.

Another very useful way to approach such calculations is to take advantage of the listability property by using the **Map** command. This command will evaluate a function at each of the list values. If we have an arbitrary function, f, and a vector of values {a,b,c,d,e,f}, then we can **Map** down this list:

```
In[18]:= Map[f, {a, b, c, d, e, f}]

Out[18]= {f[a], f[b], f[c], f[d], f[e], f[f]}
```

A more specific example is to **Map** the function square root, **Sqrt[ ]**, onto the first 10 values of **ls2**. To obtain the first 10 values we can use the **Take** command as follows:

```
In[19]:= Take[ls2, 10]

Out[19]= {1, 4, 27, 256, 3125, 46656, 823543, 16777216, 387420489,
          10000000000}
```

Now we can **Map** the square root function onto these values:

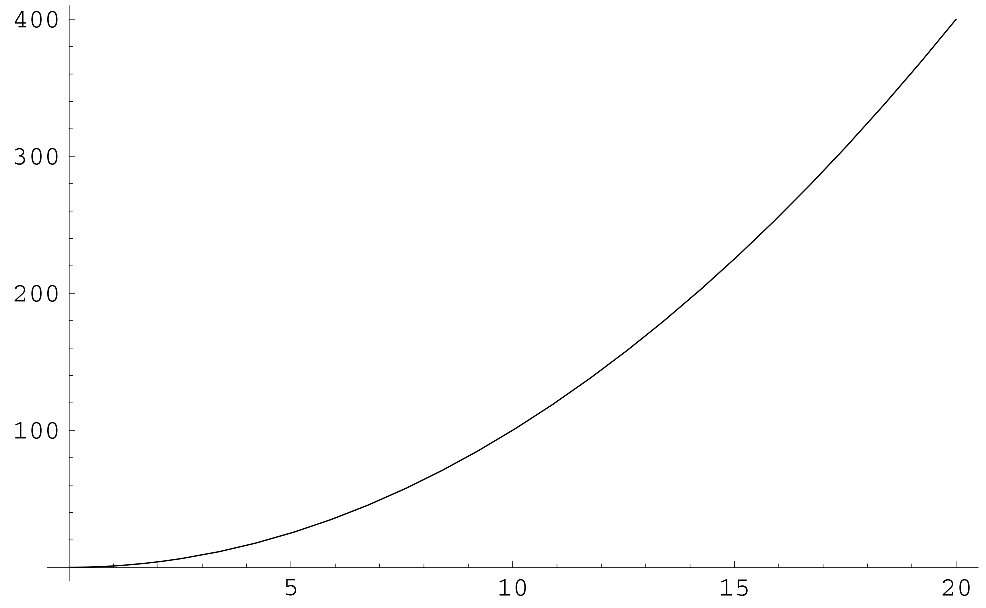
```
In[20]:= Map[Sqrt, Take[ls2, 10]]

Out[20]= {1, 2, 3√3, 16, 25√5, 216, 343√7, 4096, 19683, 100000}
```

We may use a built-in shorthand, referred to as "infix" notation, to accomplish this as well:

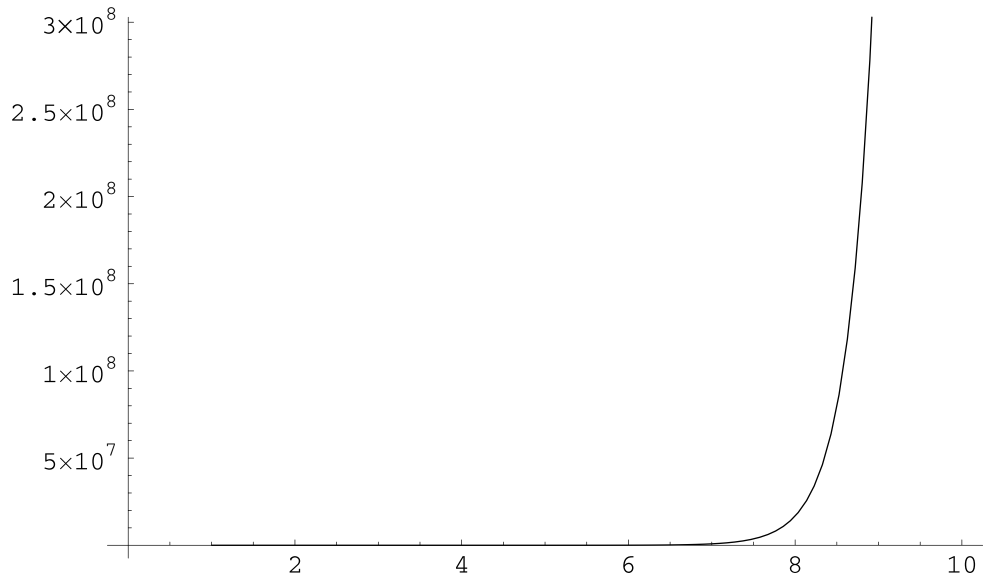
```
Sqrt/@ Take[ls2, 10]

{1, 2, 3√3, 16, 25√5, 216, 343√7, 4096, 19683, 100000}
```

We turn now from the **Table** and **Map** command to **Plot**. These have nearly identical syntax. Let us return to the example of x^2 and x^x to see how this works:

```
In[22]:= Plot[x^2, {x, 0, 20}];
```



```
In[23]:= Plot[x^x, {x, 1, 10}];
```



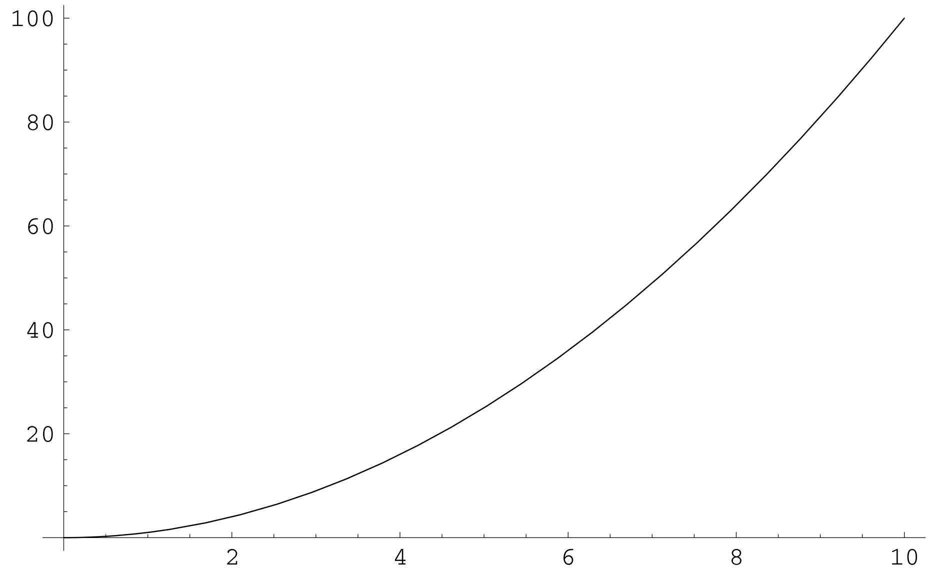
By placing the semicolon after each we suppress the output of the word "**Graphics**." We can spruce these plots up with axes labels and other attributes, but to do so at this point would lead us off the track. Notice that in both cases we did not specify an increment value. In fact with **Plot** we cannot. The reason is that *Mathematica* adjusts the increment as it moves through the function, making it smaller when the slope is large and larger when it is small. Hence, we do not have to set the increment; it is handled internally by the routine. We can be sure that in the vicinity of 16, the increment begins to become very small for x^x.

There are a host of different ways to adjust the look of the two-dimensional plots that we make in *Mathematica*. These adjustments are referred to as **Graphics Options**. To see what option we have in the **Plot** command we can use the double question mark command.

```
In[24]:= ?? Plot

          Plot[f, {x, xmin, xmax}] generates a plot of f as a
           function of x from xmin to xmax. Plot[{f1, f2, ...},
           {x, xmin, xmax}] plots several functions fi.

          Attributes[Plot] = {HoldAll, Protected}

          Options[Plot] = {AspectRatio → 1/GoldenRatio, Axes → Automatic,
           AxesLabel → None, AxesOrigin → Automatic,
           AxesStyle → Automatic, Background → Automatic,
           ColorOutput → Automatic, Compiled → True,
           DefaultColor → Automatic, Epilog → {},
           Frame → False, FrameLabel → None, FrameStyle → Automatic,
           FrameTicks → Automatic, GridLines → None,
           ImageSize → Automatic, MaxBend → 10.,
           PlotDivision → 30., PlotLabel → None, PlotPoints → 25,
           PlotRange → Automatic, PlotRegion → Automatic,
           PlotStyle → Automatic, Prolog → {}, RotateLabel → True,
           Ticks → Automatic, DefaultFont :→ $DefaultFont,
           DisplayFunction :→ $DisplayFunction,
           FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```

This shows us that we can change virtually everything about the appearance of these plots. The best way to demonstrate the use of these options' subroutines is to modify one of the standard plots that we have already made. We begin again with a plot of x^2 in its default format.

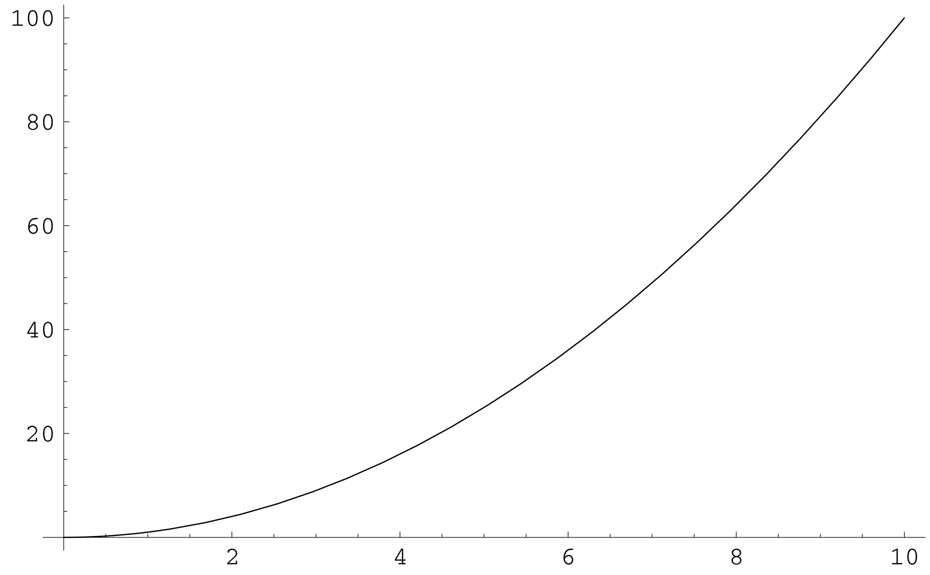
```
In[25]:= Plot[x^2, {x, 0, 10}]
```



```
Out[25]= -Graphics-
```

We notice that the axes lines are not dark enough, so we can enhance them by changing their **Thickness** parameter within the subroutine **AxesStyle**:

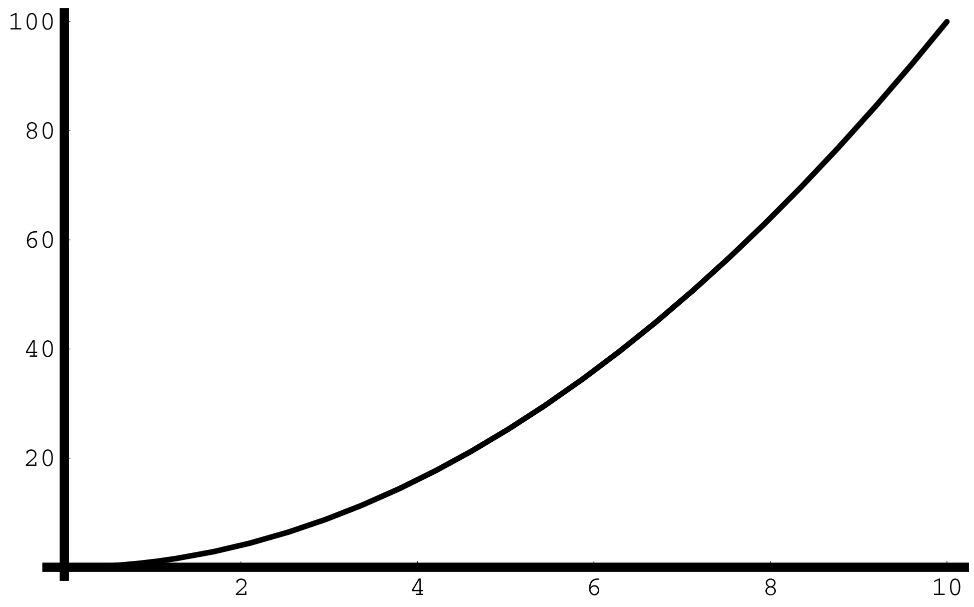
```
In[26]:= Plot[x^2, {x, 0, 10}, AxesStyle → Thickness[0.01]]
```



```
Out[26]= -Graphics-
```

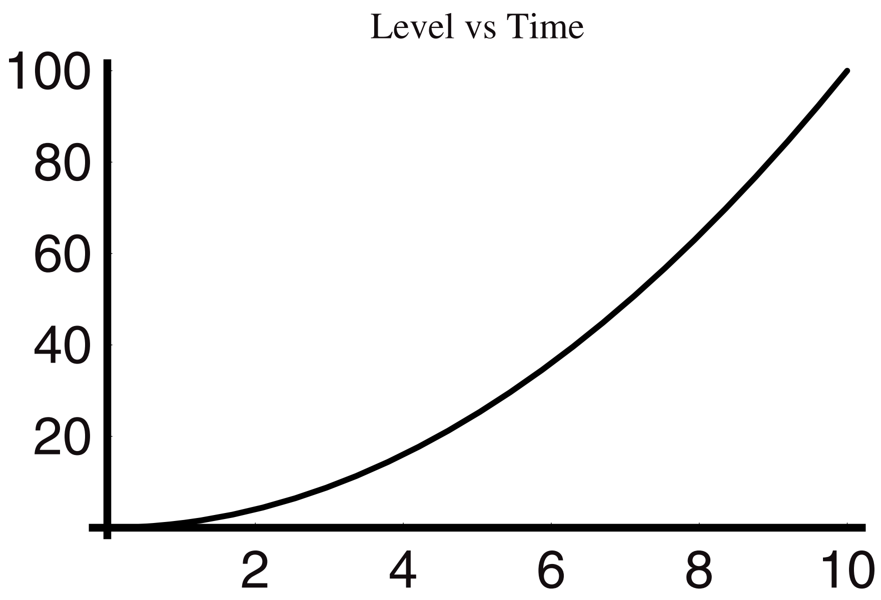
We can also enhance the plot of the function to make it more visible:

```
In[27]:= Plot[x^2, {x, 0, 10}, AxesStyle → Thickness[0.01],
          PlotStyle → {Thickness[0.006]}];
```



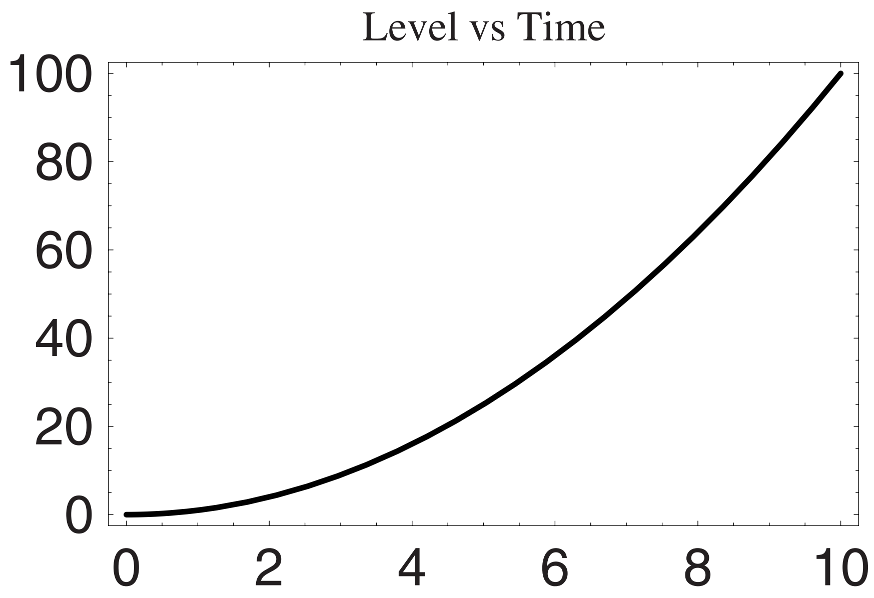
Next we change the font and the font size using **DefaultFont** and then add a label of different font type and size:

```
In[28]:= Plot[x^2, {x, 0, 10}, AxesStyle → Thickness[0.01],
          PlotStyle →{Thickness[0.0075]},
          DefaultFont →{"Helvetica", 20},
          PlotLabel →FontForm["Level vs Time", {"Times-Roman", 14}]];
```



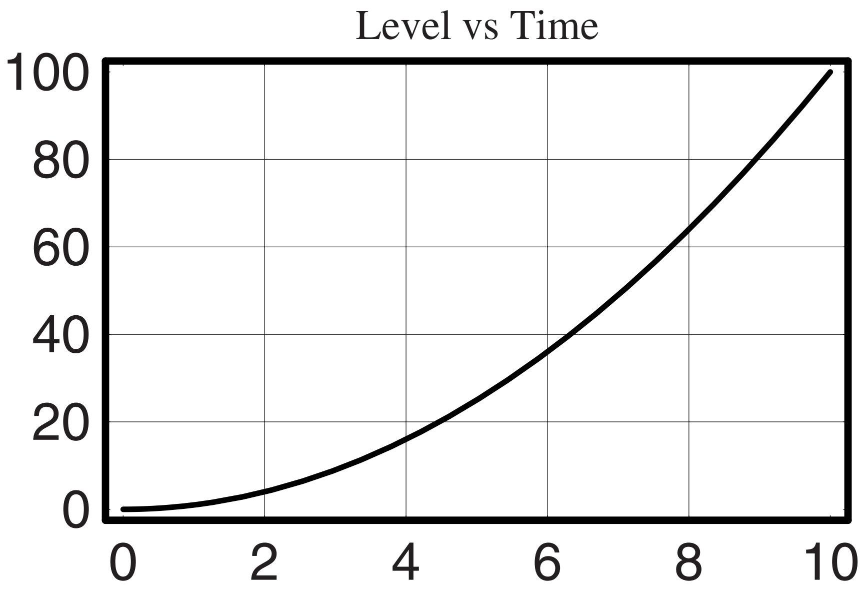
In the next instance we have changed from a simple graph to one with a frame around it:

```
In[29]:= Plot[x^2, {x, 0, 10}, AxesStyle → Thickness[0.01],
           PlotStyle → Thickness[0.0075]},
           DefaultFont → {"Helvetica", 20},
           PlotLabel → FontForm["Level vs Time", {"Times-Roman", 16}],
           Frame → True,
          ];
```



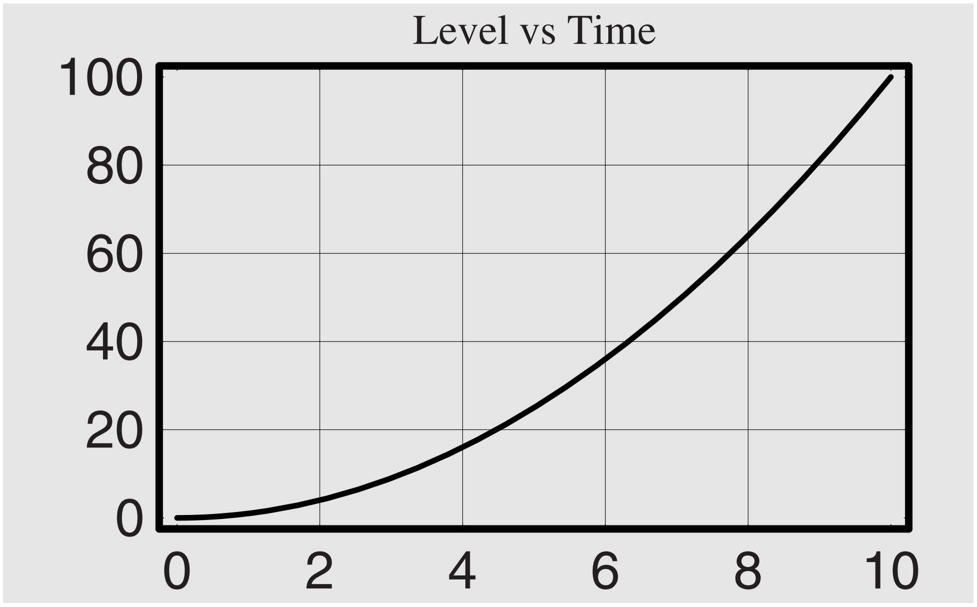
We can add a set of grid lines over the graph and to the frame as follows and thicken the latter:

```
In[30]:= Plot[x^2, {x, 0, 10},
           FrameStyle → Thickness[0.01],
           PlotStyle → {Thickness[0.0075]},
           DefaultFont → {"Helvetica", 20},
           PlotLabel → FontForm["Level vs Time", {"Times-Roman", 16}],
           Frame → True, GridLines → Automatic
          ];
```

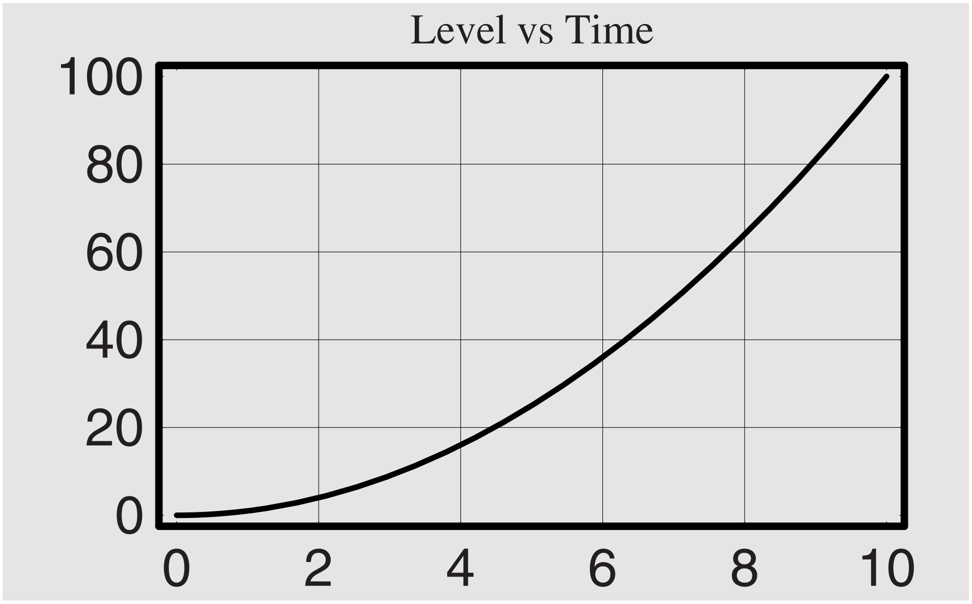


Here we add a gray background:

```
In[31]:= Plot[x^2, {x, 0, 10},
            FrameStyle → Thickness[0.01],
            PlotStyle → {Thickness[0.0075]},
            DefaultFont → {"Helvetica", 20},
            PlotLabel → FontForm["Level vs Time", {"Times-Roman", 16}],
            Frame → True,
            GridLines →Automatic,
            Background → GrayLevel[0.8]
          ];
```

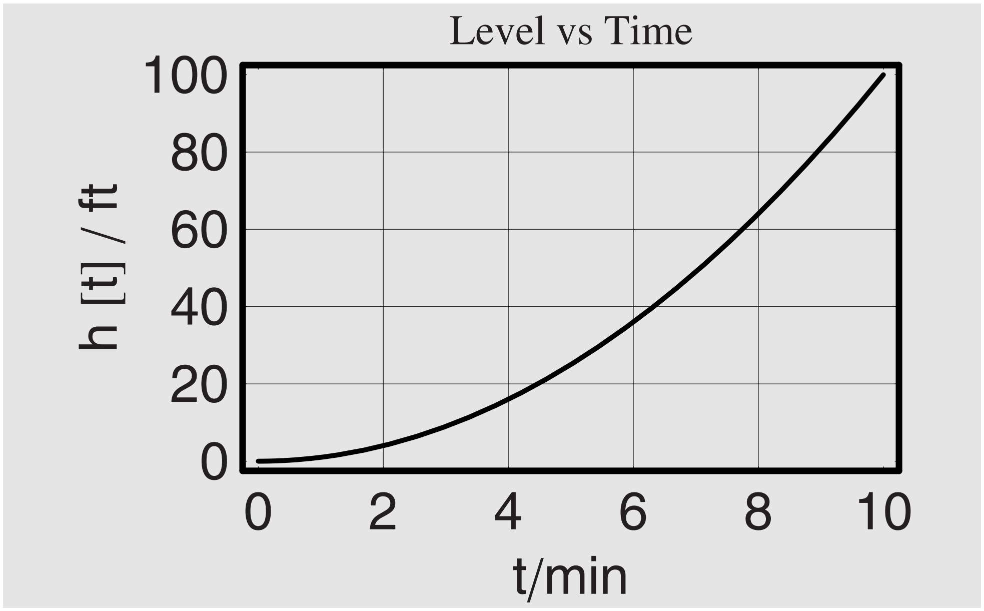


```
In[32]:= Plot[x^2, {x, 0, 10}, FrameStyle → Thickness[0.01],
           PlotStyle → {Thickness[0.0075]},
           DefaultFont → {"Helvetica", 20},
           PlotLabel → FontForm["Level vs Time", {"Times-Roman", 16}],
           Frame → True, GridLines → Automatic,
           Background → GrayLevel[0.8],
           AxesLabel → {"t/min", "h [ t ] / ft"}
          ];
```



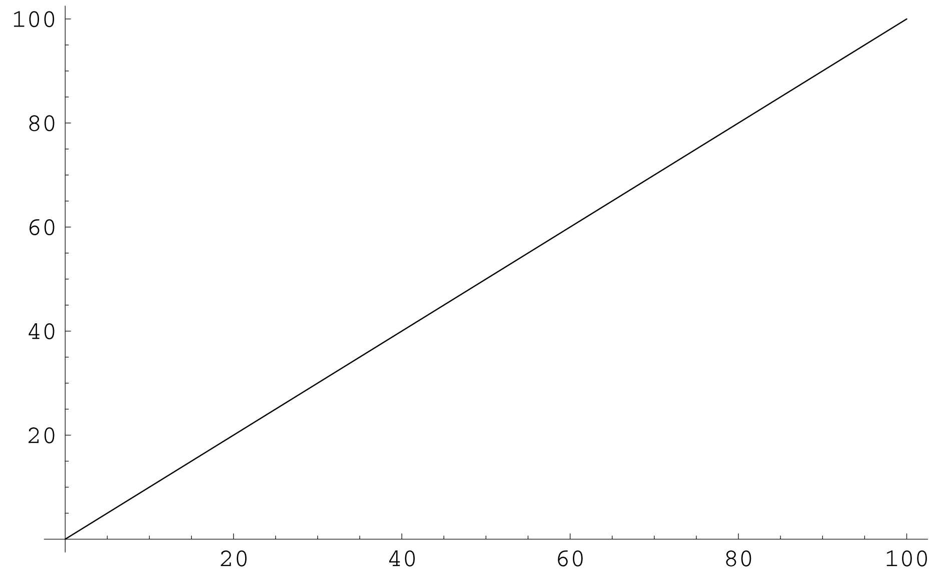
Finally, we add labels to the axes of the frame utilizing **FrameLabel**:

```
In[33]:= Plot[x^2, {x, 0, 10},
           FrameStyle → Thickness[0.01],
           PlotStyle → {Thickness[0.0075]},
           DefaultFont → {"Helvetica", 20},
           PlotLabel → FontForm["Level vs Time", {"Times-Roman", 16}],
           Frame → True,
           GridLines → Automatic,
           Background → GrayLevel[0.8],
           FrameLabel →{"t/min", "h [t] / ft"}, RotateLabel → True
          ];
```



Another very useful tool in formatting plots is the **SetOptions** command. This command allows us to set automatically the manner in which the graphs for a whole notebook will look. Let us see how this works. We begin with a simple default plot of a line, which looks as follows:

```
In[34]:= Plot[x, {x, 0, 100}];
```

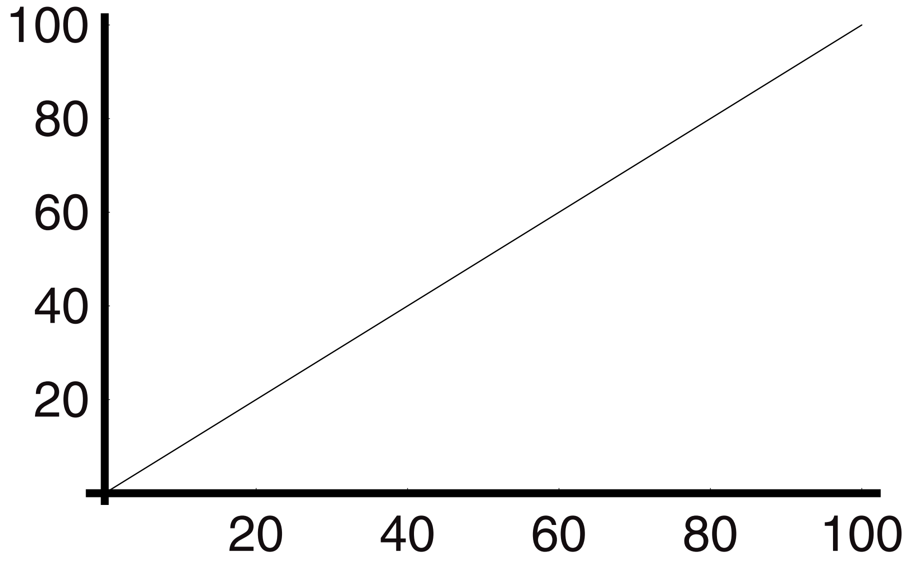


Now we can use **SetOptions** to change the thicknesses and color of the axes:

```
In[35]:= SetOptions[
          Plot, AxesStyle → {Thickness[0.01]},
          DefaultFont →{"Helvetica", 20}];
```

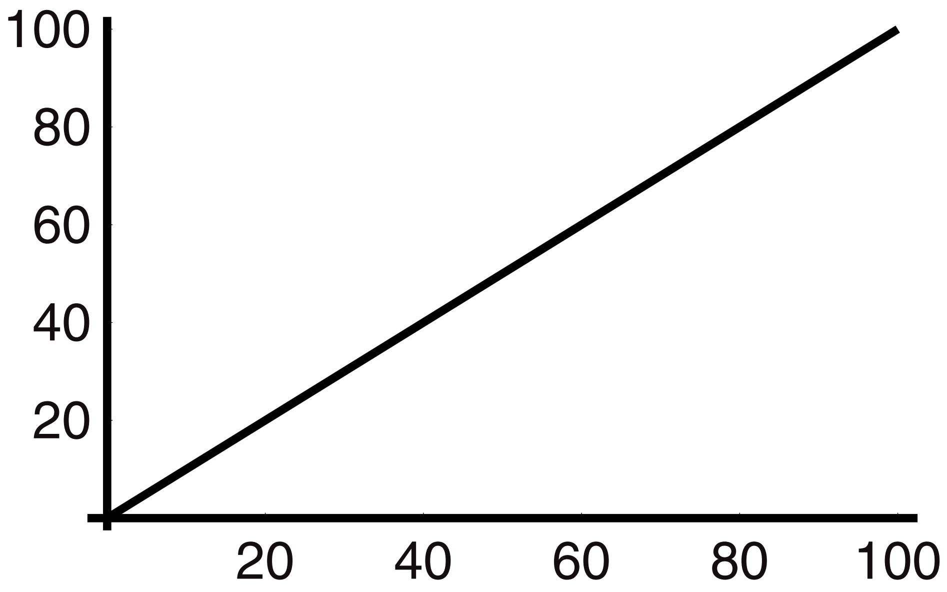
If we rerun the same command as before we now find:

```
In[36]:= Plot[x, {x, 0, 100}];
```



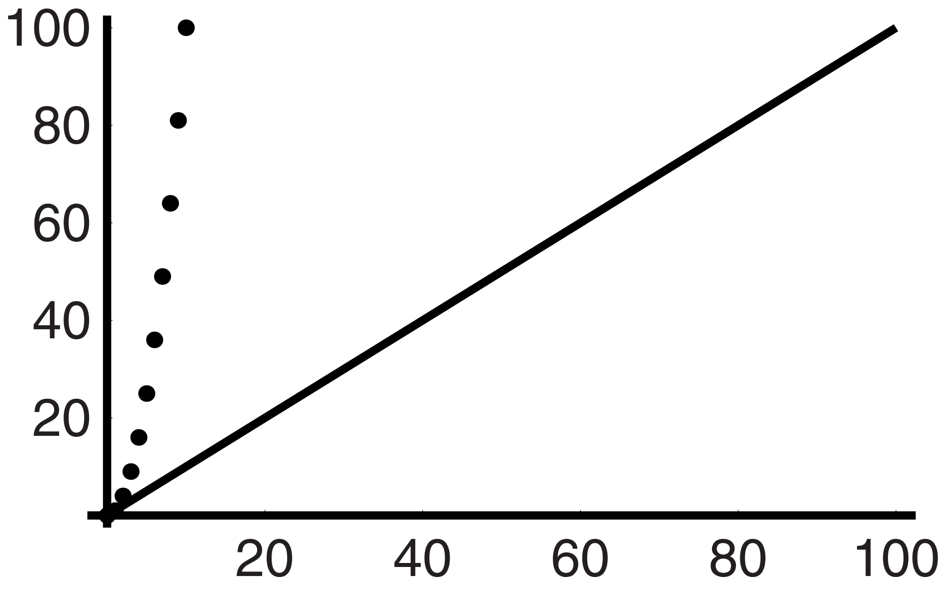
However, we can do more in fact to make our graphics look more like we may want them to look. For example, we can set the options in such a way that the plots within the graphic are more visible than at the default settings:

```
In[37]:= SetOptions[
         {Plot, ListPlot},
           AxesStyle → {Thickness[0.01]},
          PlotStyle → {PointSize[0.02],
            Thickness[0.01]},
          DefaultFont → {"Helvetica", 20}
         ];

In[38]:= Plot[x, {x, 0, 100}];
```



In the example that follows we modify both the data that will be presented as points and that which will be presented continuously. If we want to combine two graphs into one graph, then there are several ways to do this, but one of the easiest ways is to load the graphics subroutine called **DisplayTogether**. This subroutine is found within the library of subroutines called "**Graphics ‘Graphics‘**" and we load this using the << "**Needs**" command. You must call with **<<Graphics ‘Graphics‘** before you can use **DisplayTogether**. (If by chance you try to use **DisplayTogether** before calling **<<Graphics ‘Graphics‘**, then it will not work. You will need to clear the name, call the graphics commands and then use **DisplayTogether**.)

```
In[39]:= <<Graphics`Graphics`

In[40]:= DisplayTogether[ListPlot[Table[{x, x^2}, {x, 0, 10}]],
           Plot[x, {x, 0, 100}]];
```



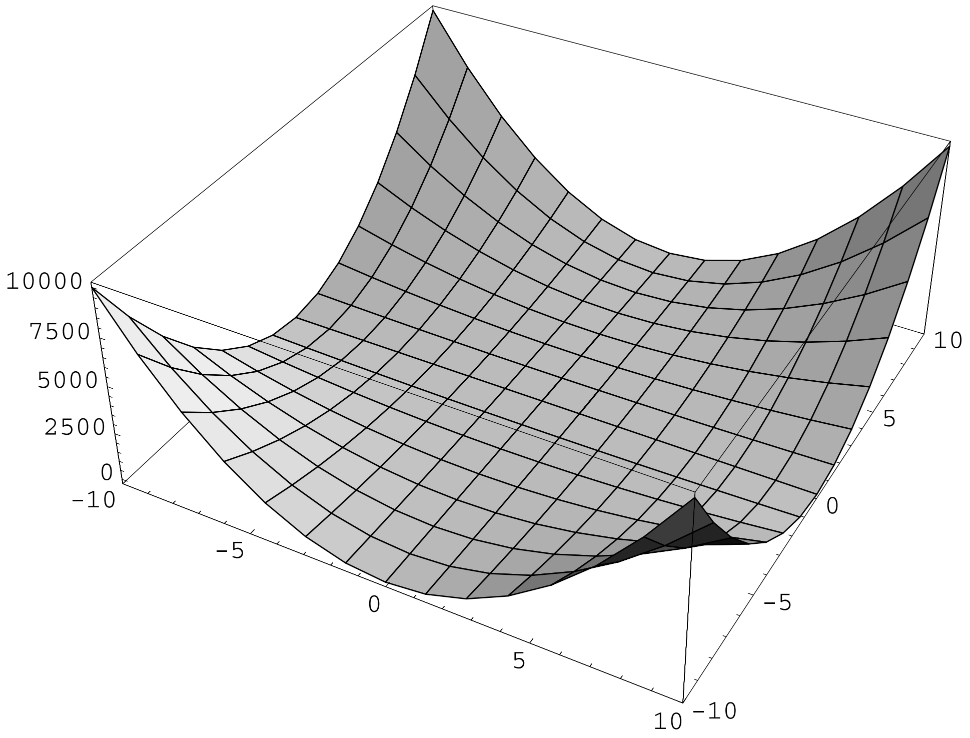
To reset the graphics options to their original present values we simply instruct the program to go back to **Default** settings for the axes and plot styles with the same command structure:

```
In[41]:= SetOptions[{Plot, ListPlot},
          AxesStyle → Automatic,
          PlotStyle → Automatic,
          DefaultFont → Automatic]

Out[41]= {{AspectRatio → 1/GoldenRatio, Axes → Automatic, AxesLabel → None,
           AxesOrigin → Automatic, AxesStyle → Automatic,
           Background → Automatic, ColorOutput → Automatic,
           Compiled → True, DefaultColor → Automatic, Epilog → {},
           Frame → False, FrameLabel → None,
           FrameStyle → Automatic, FrameTicks → Automatic,
           GridLines → None, ImageSize → Automatic, MaxBend → 10.,
           PlotDivision → 30., PlotLabel → None, PlotPoints → 25,
           PlotRange → Automatic, PlotRegion → Automatic,
           PlotStyle → Automatic, Prolog → {}, RotateLabel → True,
           Ticks → Automatic, DefaultFont → Automatic,
           DisplayFunction :→ $DisplayFunction,
           FormatType :→ $FormatType, TextStyle :→ $TextStyle},
          {AspectRatio → 1/GoldenRatio, Axes → Automatic, AxesLabel → None,
           AxesOrigin → Automatic, AxesStyle → Automatic,
           Background → Automatic, ColorOutput → Automatic,
     DefaultColor → Automatic, Epilog → {}, Frame → False,
     FrameLabel → None, FrameStyle → Automatic,
     FrameTicks → Automatic, GridLines → None,
     ImageSize → Automatic, PlotJoined → False, PlotLabel → None,
     PlotRange → Automatic, PlotRegion → Automatic,
     PlotStyle → Automatic, Prolog → {}, RotateLabel → True,
     Ticks → Automatic, DefaultFont → Automatic,
     DisplayFunction :→ $DisplayFunction,
     FormatType :→ $FormatType, TextStyle :→ $TextStyle}}
```

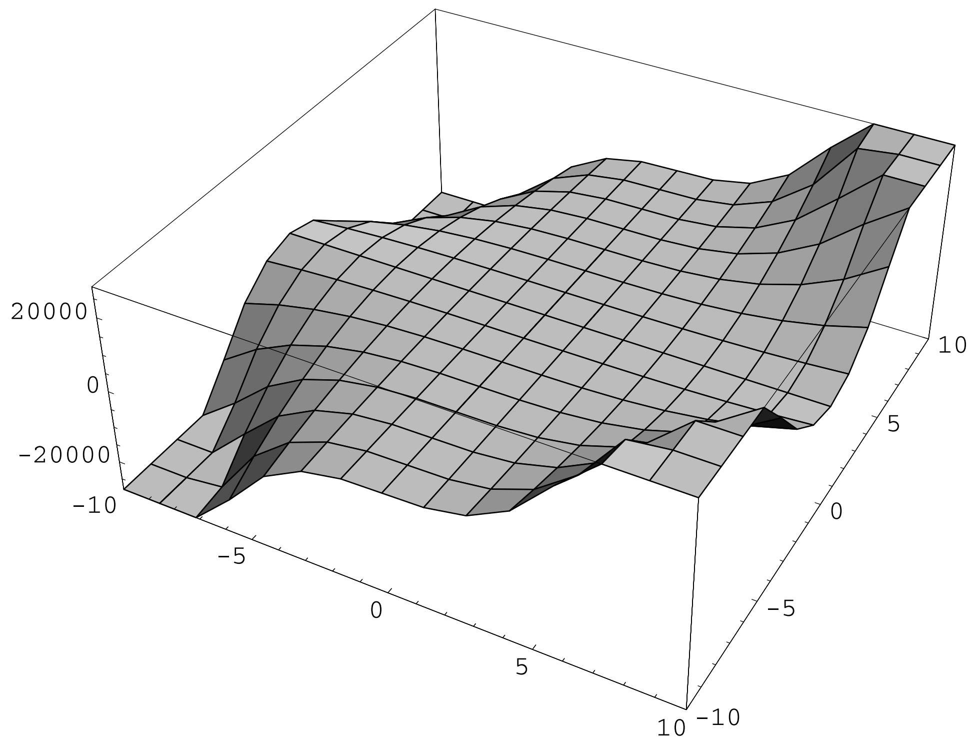
We can also plot in three dimensions. For example, if we have a function of two variables, then it is simple to see how it looks. For example, we can utilize the product of functions of **x** and **y** to see how they will appear in this space:

```
In[42]:= ?? Plot3D

         Plot3D[f, {x, xmin, xmax}, {y, ymin, ymax}] generates a
           three-dimensional plot of f as a function of x and y.
           Plot3D[{f, s}, {x, xmin, xmax}, {y, ymin, ymax}]
           generates a three-dimensional plot in which the
           height of the surface is specified by f, and the
           shading is specified by s.

         Attributes[Plot3D] = {HoldAll, Protected}

         Options[Plot3D] =
          {AmbientLight → GrayLevel[0], AspectRatio → Automatic,
           Axes → True, AxesEdge → Automatic, AxesLabel → None,
           AxesStyle → Automatic, Background → Automatic,
           Boxed → True, BoxRatios → {1, 1, 0.4},
           BoxStyle → Automatic, ClipFill → Automatic,
           ColorFunction → Automatic, ColorFunctionScaling → True,
           ColorOutput → Automatic, Compiled → True,
           DefaultColor → Automatic, Epilog → {},
           FaceGrids → None, HiddenSurface → True,
           ImageSize → Automatic, Lighting → True,
           LightSources → {{{1., 0., 1.}, RGBColor[1, 0, 0]},
           {{1., 1., 1.}, RGBColor[0, 1, 0]}, {{0., 1., 1.},
           RGBColor[0, 0, 1]}}, Mesh → True, MeshStyle → Automatic,
           Plot3Matrix → Automatic, PlotLabel → None, PlotPoints → 15,
           PlotRange → Automatic, PlotRegion → Automatic,
           Prolog → {}, Shading → True, SphericalRegion → False,
           Ticks → Automatic, ViewCenter → Automatic,
        ViewPoint → {1.3, -2.4, 2.}, ViewVertical → {0., 0., 1.},
        DefaultFont :→ $DefaultFont,
        DisplayFunction :→ $DisplayFunction,
        FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```

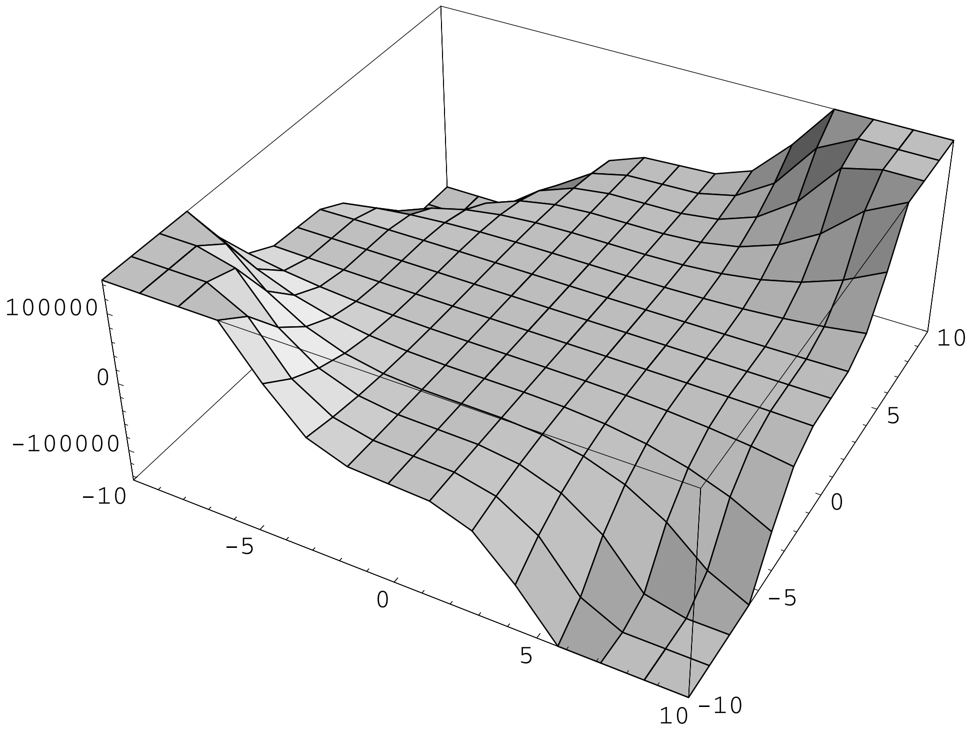
In[43]:= **Plot3D[x^2y^2, {x, -10, 10}, {y, -10, 10},**
ColorOutput → GrayLevel];



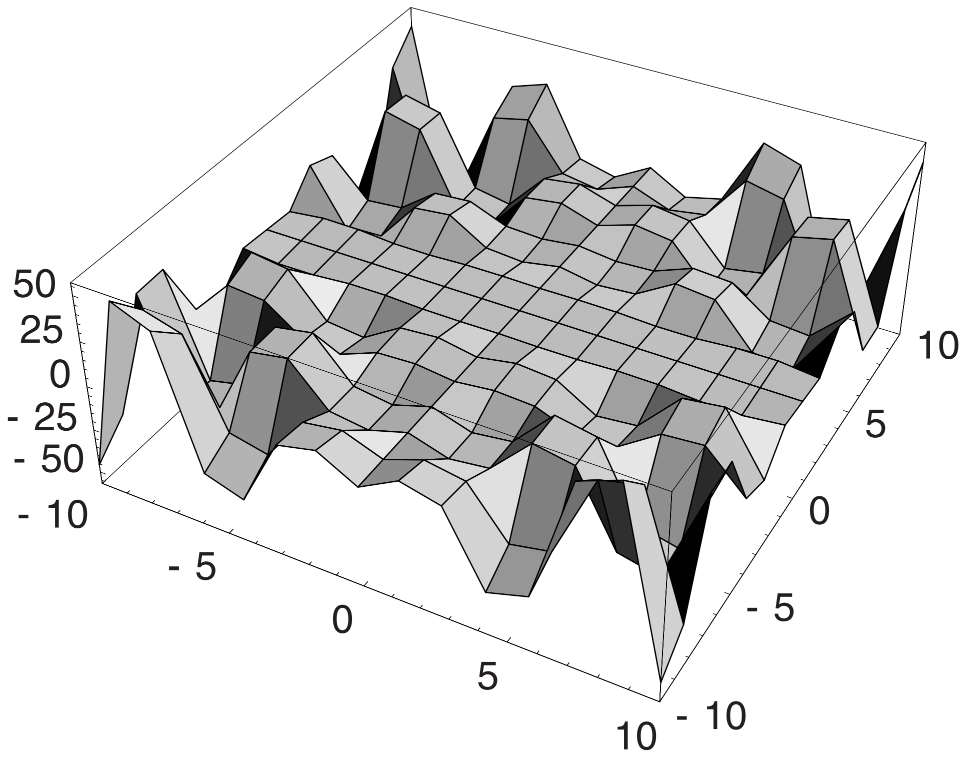
```
In[44]:= Plot3D[x^3y^2, {x, -10, 10}, {y, -10, 10},
           ColorOutput → GrayLevel];
```



```
In[45]:= Plot3D[x^3 y^3, {x, -10, 10}, {y, -10, 10},
           ColorOutput → GrayLevel];
```

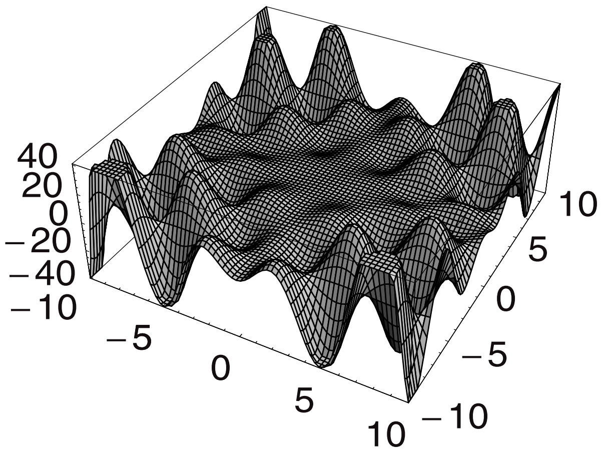


```
In[46]:= Plot3D[xSin[x] yCos[y], {x, -10, 10}, {y, -10, 10},
           ColorOutput → GrayLevel,
           DefaultFont → {"Helvetica", 15}];
```



We can also see that the structure of this plot is such that the resolution is relatively poor. Thus, it is not an adequate representation of the function. To enhance the graphical representation of the function we can increase the resolution by raising the magnitude of the attribute **PlotPoints** as follows:

```
In[47]:= Plot3D[x Sin[x] y Cos[y], {x, -10, 10}, {y, -10, 10},
           ColorOutput → GrayLevel,
           DefaultFont → {"Helvetica", 15},
           PlotPoints → 75];
```



What we see is that *Mathematica* has plotted the functions, fitted them with surfaces, placed a grid on the fitted surfaces and enhanced them with gray-level shadowing. All of this was done by routine operation in a default mode, that is, with a minimum of input from us. Here too we could spend time further enhancing these graphs, but instead we shall move on to the next subject.

1.5 Lists and ListPlot, Fit, and Show

Often we will have data rather than a function and we wish to plot it, so that we can find a function that describes the data by analysis. In such cases we can manipulate the data by bringing it into a matrix form and then plotting it with **ListPlot**. We also can compare it to the behavior of functions that are meant to represent the data. The following is a typical set of data obtained from an experiment, appropriately named "**data**." (This could have been imported to *Mathematica* by any number of different means.) The first column is time and the second is the value of the measured variable in the system:

```
0      10
2      8.2
4      6.7
6      5.5
8      4.5
10     3.7
12     3.
14     2.5
16     2.
20     1.4
```

24	0.9
28	0.6
32	0.4
36	0.3
40	0.2
44	0.1
50	0.1

First, we write a vector of time values (**tim1**) at which measurements were made and do the same with the dependent variable values (**dat1**) and input both:

```
tim1 = {0, 2, 4, 6, 8, 10, 12, 14, 16, 20, 24, 28,
       32, 36, 40, 44, 50};

dat1 = {10, 8.2, 6.7, 5.5, 4.5, 3.7, 3.0, 2.5, 2.0,
       1.4, 0.9, 0.6, 0.4, 0.3, 0.2, 0.1, 0.1};
```

To plot these we must join these into pairs of *x*,*y* values that can be plotted by **ListPlot**. We will use three commands **Join, Partition**, and **Transpose** to do this. Here is how it is done in stepwise fashion:

```
In[49]:= Join[tim1, dat1]

Out[49]= {0, 2, 4, 6, 8, 10, 12, 14, 16, 20, 24, 28, 32, 36, 40,
          44, 50, 10, 8.2, 6.7, 5.5, 4.5, 3.7, 3., 2.5, 2., 1.4,
          0.9, 0.6, 0.4, 0.3, 0.2, 0.1, 0.1}
```

The output from this operation is a single vector composed of time values and then the dependent variable values. We need them to be paired in order to plot them. Thus we first break this vector into two vectors within one. The first is for the time values and the second for the dependent variable values. To get this right we need to partition time only with time values, and therefore we need to state how many elements from the list should be in each partition. We can do this if we know the length of the time list. We get this information by asking for the number of elements in **tim1** with Length:

```
In[50]:= Length[tim1]

Out[50]= 17
```

We use this as follows:

```
In[51]:= pdata = Partition[Join[tim1, dat1], Length[dat1]]

Out[51]= {{0, 2, 4, 6, 8, 10, 12, 14, 16, 20, 24, 28, 32, 36, 40, 44, 50},
          {10, 8.2, 6.7, 5.5, 4.5, 3.7, 3., 2.5, 2., 1.4, 0.9, 0.6,
           0.4, 0.3, 0.2, 0.1, 0.1}}
```

Now we have two lists in one; in effect, we really have a matrix. We can see this by "//**Matrix Form**" after the **Partition** command:

```
In[52]:= pdata//MatrixForm
Out[52]//MatrixForm =
```

$$\begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 20 & 24 & 28 & 32 & 36 & 40 & 44 & 50 \\ 10 & 8.2 & 6.7 & 5.5 & 4.5 & 3.7 & 3. & 2.5 & 2. & 1.4 & 0.9 & 0.6 & 0.4 & 0.3 & 0.2 & 0.1 & 0.1 \end{pmatrix}$$

As it is a matrix we can do a very simple and yet powerful operation on it—we can transpose it. When we transpose a matrix we exchange the rows for columns. Here is a simple example:

```
In[53]:= m1 = {{a, b, c, d}, {1, 2, 3, 4}}
Out[53]= {{a, b, c, d}, {1, 2, 3, 4}}
In[54]:= m1 // MatrixForm
Out[54]//MatrixForm =
```

$$\begin{pmatrix} a & b & c & d \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

```
In[55]:= Transpose[m1] // MatrixForm
Out[55]//MatrixForm =
```

$$\begin{pmatrix} a & 1 \\ b & 2 \\ c & 3 \\ d & 4 \end{pmatrix}$$

Returning to our example, we can see that by transposing the partitioned set "**pdata**" we will have the pairs of independent and dependent variables we seek to plot:

```
In[56]:= dataset = Transpose[pdata]
Out[56]= {{0, 10}, {2, 8.2}, {4, 6.7}, {6, 5.5}, {8, 4.5},
          {10, 3.7}, {12, 3.}, {14, 2.5}, {16, 2.}, {20, 1.4},
          {24, 0.9}, {28, 0.6}, {32, 0.4}, {36, 0.3}, {40, 0.2},
          {44, 0.1}, {50, 0.1}}
In[57]:= dataset // MatrixForm
```

```
Out[57]//MatrixForm =
          0   10
          2   8.2
          4   6.7
          6   5.5
          8   4.5
          10  3.7
          12   3.
          14  2.5
          16   2.
          20  1.4
          24  0.9
          28  0.6
          32  0.4
          36  0.3
          40  0.2
          44  0.1
          50  0.1
```

Although we did each step interactively, we can do it all at once as follows:

```
In[58]:= Transpose[Partition[Join[tim1, dat1], Length[dat1]]]

Out[58]= {{0, 10}, {2, 8.2}, {4, 6.7}, {6, 5.5}, {8, 4.5},
          {10, 3.7}, {12, 3.}, {14, 2.5}, {16, 2.}, {20, 1.4},
          {24, 0.9}, {28, 0.6}, {32, 0.4}, {36, 0.3}, {40, 0.2},
          {44, 0.1}, {50, 0.1}}
```

Another way in which we could have done this takes advantage of **Table** and the listability of **tim1** and **dat1**, both of which are unidimensional vectors. To do this we make use of the fact that each element of the list is associated with a unique numerical position that we express as **tim1[[n]] or dat1[[m]]** as follows:

```
In[59]:= tim1[[5]]
         dat1[[5]]

Out[59]= 8
         4.5
```

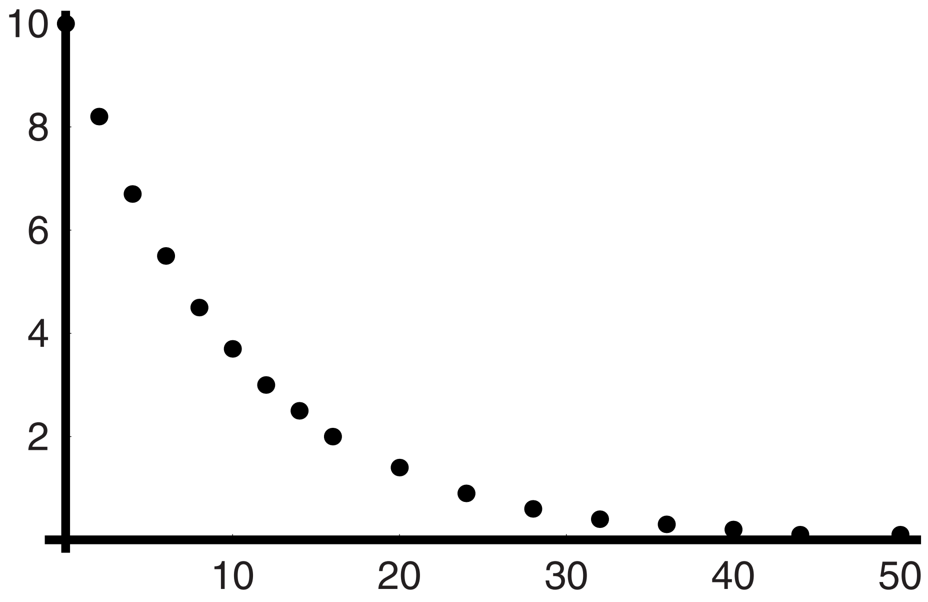
Now we can put the two lists together by placing the first element **tim1[[n]]** and the second element **dat1[[n]]** inside a set of braces, **{tim1[[n]],dat1[[n]]}**, which we then place inside the **Table** command:

```
In[60]:= dataset = Table[{tim1[[n]], dat1[[n]]}, {n, 1, Length[tim1]}]
```

```
Out[60]= {{0, 10}, {2, 8.2}, {4, 6.7}, {6, 5.5}, {8, 4.5}, {10,3.7},
          {12, 3.}, {14, 2.5}, {16, 2.}, {20, 1.4}, {24, 0.9},
          {28, 0.6}, {32, 0.4}, {36, 0.3}, {40, 0.2}, {44, 0.1},
          {50, 0.1}}
```

We can now use **ListPlot** to display this data:

```
In[61]:= SetOptions[
         {Plot, ListPlot},
           AxesStyle →{Thickness[0.01]},
           PlotStyle →{PointSize[0.02],
              Thickness[0.01]},
           DefaultFont →{"Helvetica", 15}
         ];

In[62]:= ListPlot[dataset];
```



Instead of a graph of a function we now have discrete points corresponding to the paired values of the independent and dependent variables. We can see that this data looks like an exponential decay of the **y** values with increasing **x**. A simple test of this would be to take the natural log of the **y-values** and plot them against **x**. Look once again at the paired values in the data set. We can see that if we take out one pair, then what we want is the first number paired with the **Log** of the second value of the pair:

```
In[63]:= dataset

Out[63]= {{0, 10}, {2, 8.2}, {4, 6.7}, {6, 5.5}, {8, 4.5}, {10,3.7},
          {12, 3.}, {14, 2.5}, {16, 2.}, {20, 1.4}, {24, 0.9},
          {28, 0.6}, {32, 0.4}, {36, 0.3}, {40, 0.2}, {44, 0.1},
          {50, 0.1}}
```

There are several ways in which we can proceed. We could go back to the set of **y-values, dat1**, take the log of these, and then redo all the steps we did in the preceding. That is an acceptable but inelegant approach. It is acceptable because it works; it is inelegant because we already have the dataset in the form in which we need only take the log of every second value. Therefore, a more elegant approach is to operate directly on the dataset using the power of *Mathematica*'s rule- and function-based programming language. In the process of doing this we will use more of the language and we will see why *listability* is so important.

When we want to take an element from a set it is simply a matter of using the correct syntax. For example, as we discussed before, to take the fifth element from the dataset we simply type **dataset** with a five after it in double square brackets:

```
In[64]:= dataset[[5]]

Out[64]= {8, 4.5}
```

Since the fifth element of dataset is a pair of numbers corresponding to the fifth point the output is this pair. If we wanted to take out of a data set the **y-value** of the 9th point, then we would type 9 and 2 separately in double square brackets after dataset:

```
In[65]:= dataset[[9, 2]]

Out[65]= 2.
```

Similarly, if we wanted to take out the **x-value** from the first data point in the set:

```
In[66]:= dataset[[1, 1]]

Out[66]= 0
```

It is clear that we are close to what we need in this function. We could extract all the **y-values** from dataset by incorporating this syntax into a **Table** function. For example:

```
In[67]:= Table[dataset[[n, 2]], {n, 1, Length[dataset]}]

Out[67]= {10, 8.2, 6.7, 5.5, 4.5, 3.7, 3., 2.5, 2., 1.4, 0.9,
          0.6, 0.4, 0.3, 0.2, 0.1, 0.1}
```

Taking the **Log** of this and using **N** to evaluate numerically, we can have a vector of the **Log** of **y**-values from dataset because it is *listable*:

```
In[68]:= N[Log[Table[dataset[[n, 2]], {n, 1, Length[dataset]}]]]

Out[68]= {2.30259, 2.10413, 1.90211, 1.70475, 1.50408, 1.30833,
          1.09861, 0.916291, 0.693147, 0.336472, -0.105361,
          -0.510826, -0.916291, -1.20397, -1.60944, -2.30259,
          -2.30259}
```

However, now we have violated our original goal, and we have taken **dataset** apart. We can be even more savvy than this and avoid having to **Join, Partition**, and **Transpose** again. We do this by writing a function in *Mathematica* that will do what we want from the start. The syntax for a function in *Mathematica* or a rule is **f[x_] := f[x]**. This function will take only single values for **x**. We have a set of paired values as the argument of our function, so we will follow the dummy variable on the left-hand side by a double underbar instead of a single underbar: **g[x__] := g[x]**. The function or rule that we want is written this way in English:

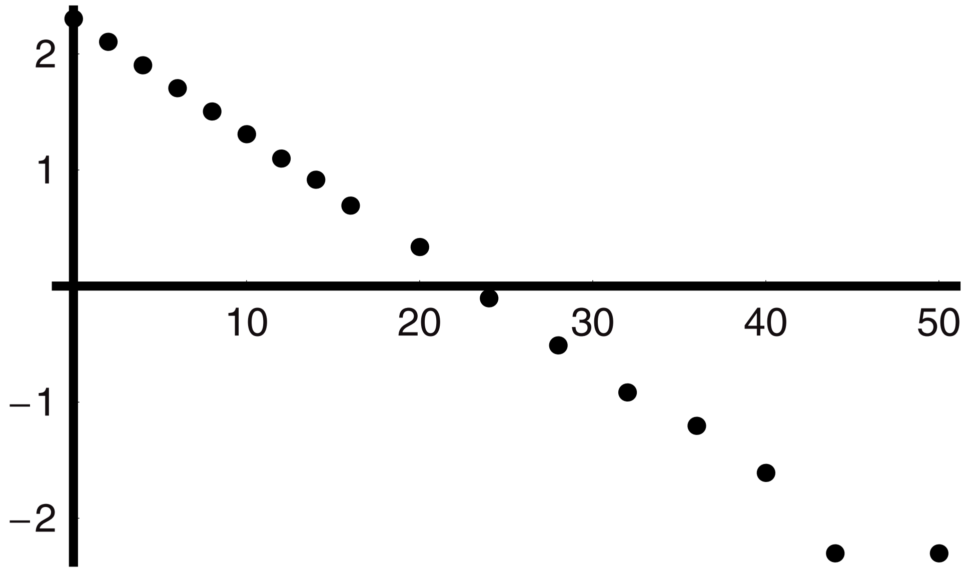
> **"Take an element from dataset, keep the x-value as it is, but take the Log of the y-value and automatically evaluate it, and keep the two values xn and yn paired as they originally were."**

Writing this in English first makes it fairly obvious what we need to do; this is our algorithm. In *Mathematica* we translate this algorithm directly into a rule or function. That rule will look like this for the *n*th element of any set:

```
In[69]:= lgf[x__] := {x[[n, 1]], N[Log[x[[n, 2]]]]}
```

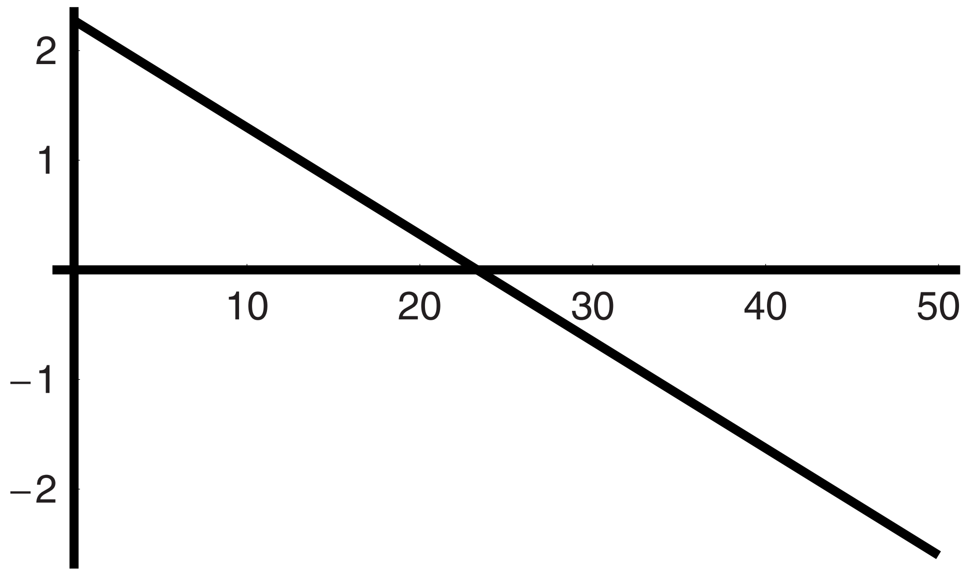
We should not move too fast on this because this rule is a program and it is rules like this one that form the bricks from which we can build larger structures later. Note that the left-hand side has function syntax with the dummy variable followed by a double underbar and set off from the right-hand side by a colon and an equal sign. This is called the *set delayed* structure in *Mathematica*. It means that until a specific argument is given within the brackets, this function in unevaluated, that is, its evaluation is delayed until we give it an argument. The form of the function is stored and will work with most any argument, provided we also given it a value for *n*. On the right-hand side we find a set of braces around two commands that now should look familiar. The first just takes the **x-value** of the *n*th element and pairs it with the log of the **y-value** of the *n*th element. If we now put this inside the table function, we can operate on dataset from *n* equals one to the end, which occurs at the value of **Length[dataset]**. Here it is:

```
In[70]:= lgdatset = Table[lgf[dataset], {n, 1, Length[dataset]}]

Out[70]= {{0, 2.30259}, {2, 2.10413}, {4, 1.90211}, {6, 1.70475},
          {8, 1.50408}, {10, 1.30833}, {12, 1.09861},
          {14, 0.916291}, {16, 0.693147}, {20, 0.336472},
          {24, -0.105361}, {28, -0.5108256}, {32, -0.916291},
          {36,-1.20397},{40,-1.60944},{44,-2.30259},
          {50,-2.30259}}

In[71]:= pllgdatset = ListPlot[lgdatset];
```



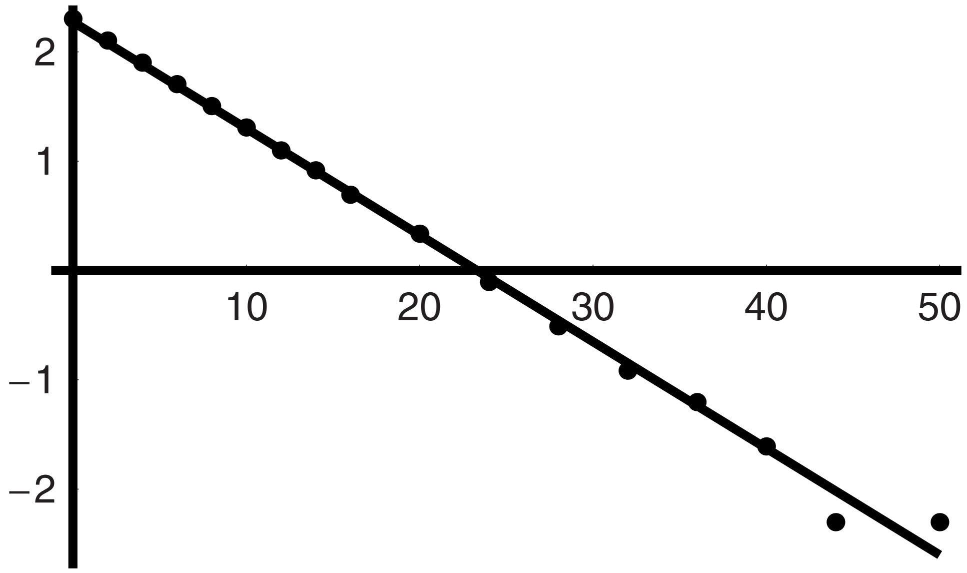
As we can see this looks quite linear, thereby indicating that the data follows an exponential decay. If we want to be more precise about this, we can use *Mathematica* to find a fit to the log data. That is, we can find the equation for the best fit line to lgdatset. After we have this function, we can then plot it and graph it with the data to once again visualize the goodness of fit. We introduce now the **Fit** command. The syntax for **Fit** is as follows—the argument consists of three elements, the first of which is the name of the matrix of data to be fit, the second of which is enclosed in braces and it states that we want to fit to a linear equation (we can use any polynomial we like), and the last of which names the independent variable. The output is a line (or polynomial) in **x**. As we will want to **Plot** this, we should give it a function name. We can call it **ftlg** for fit to the **Log** of the data and plot it from zero to 50 in **x**. (First we do it without the function name to show the output and then again with the function name. There is no reason to do the first, except to see the values of the slope and intercept.) We can give the plot a name, **plftlg**, plot of fit, to log:

```
In[72]:= Fit[lgdatset, {1, x}, x]

Out[72]= 2.27476 - 0.0975478x

In[73]:= ftlg[x_] := Fit[lgdatset, {1, x}, x]
         plftlg = Plot[ftlg[x], {x, 0, 50}];
```



Finally, we can put the data points and this line on the same graph by calling for the **Listplot, pllgdatset**, and the **Plot, plftlg**, within the **Show** command:

```
In[74]:= Show[plftlg, pllgdatset];
```

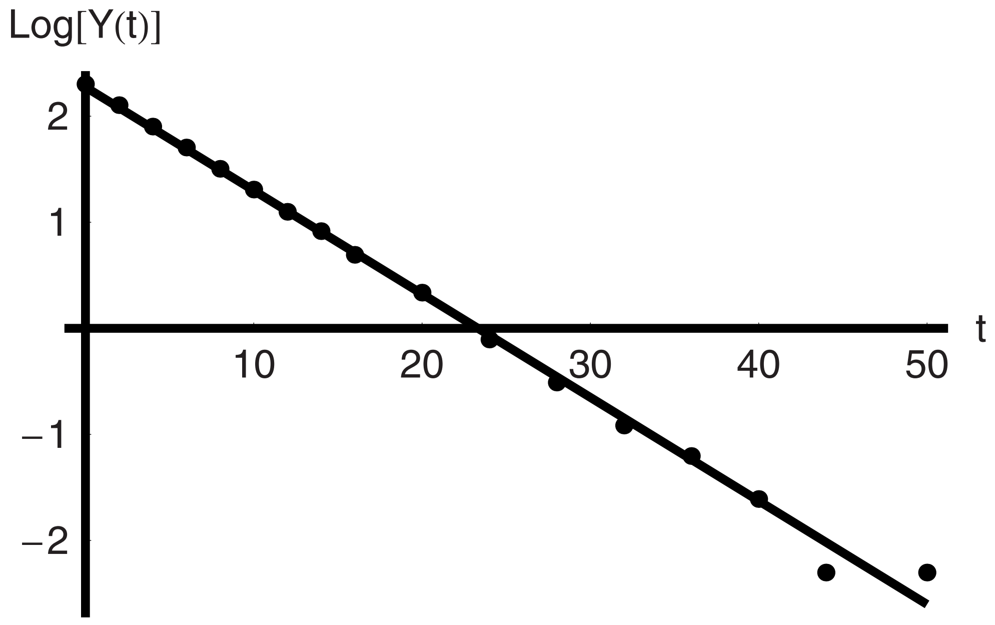


This looks much better than it did, but we still should give the **x**- and **y**-axes labels. Why not call them **t** for time and **LogY(t)** for the log of position **Y** as a function of time. To do this we need the command **AxesLabel**, which has the following attributes:

```
In[75]:= ?? AxesLabel

         AxesLabel is an option for graphics functions that
          specifies labels for axes.

         Attributes[AxesLabel] = {Protected}
```

We will put the label for each axis within a set of braces and then also within quotation marks so that they are not interpreted as a function to be evaluated but rather as simply strings.

```
In[76]:= Show[pllgdatset, plftlg, AxesLabel → {"t", "Log[Y(t)]"}];
```



1.6 Solve and NSolve

The **Solve** and **NSolve** commands are for algebraic equation solving. The **Solve** provides a symbolic result and **NSolve** numerically evaluates for the variable that is sought. These are used either for single or sets of equations. They are best illustrated by example. We can begin with **Solve**.

The syntax for **Solve** is quite simple. The argument consists of the equation or equations to be solved followed by the variable or list of variables we seek to define. An inquiry of *Mathematica* gives us the information more completely. Notice that there are **Options** we can set that allow us to deal with special situations when they arise. For the most part we can and will leave these at their default values, but it is important to know that the user has a considerable degree of control over most functions in *Mathematica*. The program is so powerful and the defaults work so well that one often gets the impression that nothing can be changed or fine-tuned by the user. In fact, this is an incorrect impression.

```
In[77]:= ?? Solve

         Solve[eqns, vars] attempts to solve an equation or set
          of equations for the variables vars. Solve[eqns, vars,
          elims] attempts to solve the equations for vars,
          eliminating the variables elims.

         Attributes[Solve] = {Protected}

         Options[Solve] = {InverseFunctions → Automatic,
          MakeRules → False, Method → 3, Mode → Generic,
          Sort → True, VerifySolutions → Automatic,
          WorkingPrecision → ∞}
```

To solve for one equation for one unknown we can examine how **Solve** works on a quadratic equation because we know that solution so well:

```
In[78]:= Clear[A1, B1, C1, x]
         Solve[0 == A1 x^2 + B1 x + C1, x]
```

$$\text{Out[78]= } \{\{x \to \frac{-B1 + \sqrt{B1^2 - 4\,A1\,C1}}{2\,A1}\},\ \{x \to -\frac{B1 + \sqrt{B1^2 - 4\,A1\,C1}}{2\,A1}\}\}$$

We could also have two quadratic equations in **x1** and **x2** with appropriate constant coefficients:

```
In[79]:= Clear[A1, B1, A2, B2, C1, C2]

In[80]:= Solve[{0 == A1 x1^2 + B1 x2 + C1, 0 == A2 x1^2 + B2 x2 + C2},
          {x1, x2}]
```

$$\text{Out[80]= } \{\{x2 \to \frac{A2\,C1 - A1\,C2}{-A2\,B1 + A1\,B2},\ x1 \to -\frac{\sqrt{B2\,C1 - B1\,C2}}{\sqrt{A2\,B1 - A1\,B2}}\},$$
$$\{x2 \to \frac{A2\,C1 - A1\,C2}{-A2\,B1 + A1\,B2},\ x1 \to \frac{\sqrt{B2\,C1 - B1\,C2}}{\sqrt{A2\,B1 - A1\,B2}}\}\}$$

If we move to a third-order equation, we obtain three solutions, two of which are imaginary as shown in what follows:

```
In[81]:= Clear[A1, B1, C1, x]

In[82]:= Solve[A1 x^3 + B1 x^2 + C1 x + D == 0, x]
```

Out[82]=
$$\{\{x\to -\frac{B1}{3\,A1}-(2^{1/3}(-B1^2+3\,A1\,C1))/(3\,A1\,(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2\,D+\sqrt{4(-B1^2+3\,A1\,C1)^3+(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D)^2})^{1/3})+\frac{(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D+\sqrt{4(-B1^2+3\,A1\,C1)^3+(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D)^2})^{1/3}}{3\,2^{1/3}A1}\},$$
$$\{x\to -\frac{B1}{3\,A1}+((1+i\sqrt{3})(-B1^2+3\,A1\,C1))/(3\,2^{2/3}A1(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D+\sqrt{4\,(-B1^2+3\,A1\,C1)^3+(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D)^2})^{1/3})-\frac{1}{6\,2^{1/3}A1}((1-i\sqrt{3})(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D+\sqrt{4\,(-B1^2+3\,A1\,C1)^3+(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D)^2})^{1/3})\},$$
$$\{x\to -\frac{B1}{3\,A1}+((1-i\sqrt{3})(-B1^2+3\,A1\,C1))/(3\,2^{2/3}A1(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D+\sqrt{4\,(-B1^2+3\,A1\,C1)^3+(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D)^2})^{1/3})-\frac{1}{6\,2^{1/3}A1}((1+i\sqrt{3})(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D+\sqrt{4\,(-B1^2+3\,A1\,C1)^3+(-2\,B1^3+9\,A1\,B1\,C1-27\,A1^2D)^2})^{1/3})\}$$
$$\}$$

We can find solutions to most equations even when transcendental functions **(Log, Sin, Cosh**. . . **)** are involved.

```
In[81]:= Clear[A1, B1, C1, x]
In[82]:= Solve[B1 Log[A1 x^2 + B1 x + C1] + Sin[C1] == D1, x]
```

Out[84]=
$$\{\{x\to\frac{-B1-\sqrt{B1^2-4\,A1\,C1+4\,A1\,e^{\frac{D1-Sin[C1]}{B1}}}}{2\,A1}\},\quad \{x\to\frac{-B1+\sqrt{B1^2-4\,A1\,C1+4\,A1\,e^{\frac{D1-Sin[C1]}{B1}}}}{2\,A1}\}\}$$

NSolve appears to work in very much the same way as **Solve**, but instead of working out a symbolic solution, it provides numerics. This syntax is essentially the same as that used for **Solve**. We put the arguments inside the brackets as the equations and the solution variable, but now of course the constants must be numerical. Here we take the first and last examples from the preceding **Solve** examples and put them together in one cell with the assignments of the constants.

```
In[85]:= A1 = 1;
         B1 = 10;
         C1 = 9;
         D1 = 8;
         NSolve[0 == A1 x^2 + B1 x + C1, x]
         NSolve[A1 x^3 + B1 x^2 + C1 x + D1 == 0, x]
         NSolve[B1 Log[A1 x^2 + B1 x + C1] + Sin[C1] == D1, x]
         Remove[A1, B1, C1]
Out[89]= {{x→-9.}, {x→-1.}}
```

```
Out[90]=  {{x→-9.10832}, {x→-0.445839 - 0.824346ⅈ},
           {x→-0.445839 + 0.824346ⅈ}}

Out[91]=  {{x→-9.2586}, {x→-0.741399}}

In[93]:=  Remove[A1, B1, C1, A2, B2, C2]
          A1 = 2;
          B1 = 5;
          C1 = 3;
          A2 = 3;
          B2 = 6;
          C2 = 2;
          Solve[{0 == A1 x1^2 + B1 x2 + C1, 0 == A2 x1^2 + B2 x2 + C2},
           {x1, x2}]
          N[%]
          NSolve[{0 == A1 x1^2 + B1 x2 + C1, 0 == A2 x1^2 + B2 x2 + C2},
           {x1, x2}]
          Remove[A1, A2, B1, B2, C1, C2]
```

Out[100]= $\{\{x2 \to -\frac{5}{3},\ x1 \to -2\sqrt{\frac{2}{3}}\},\ \{x2 \to -\frac{5}{3},\ x1 \to 2\sqrt{\frac{2}{3}}\}\}$

```
Out[101]= {{x2→-1.66667, x1→-1.63299}, {x2→-1.66667, x1→1.63299}}

Out[102]= {{x2→-1.66667, x1→1.63299}, {x2→-1.66667, x1→-1.63299}}
```

In the last case we solved first symbolically, but with values for the constants replaced into the solution. Then we evaluated these four solutions by using **N[%]**. This is a shortcut that is handy to use occasionally. The "**%**" symbol means the "**last result.**" We can do anything to the last result, but in this case we evaluate it numerically with **N[ ]**. For completeness we solve the problem once again using **NSolve** in place of **Solve** and we see that we obtain the very same result as on the previous line. In both cells the last statement is the **Remove** command. This is done to be sure that these symbols do not mistakenly appear with the same values once again in some work that we will do later in the session but in a different problem.

There is much more that can be done to manipulate equations and their solutions. For example, there is a set of commands for doing algebra that mimics what we do by hand (**Expand, Factor, Simplify, FullSimplify, PowerExpand**...). We observed here that we obtained imaginary roots to these equations. If our problems demand only real roots, then we can have *Mathematica* filter out the imaginaries and return just the real roots (**Miscellaneous**'**RealOnly**'). But we should not get too far ahead of ourselves. It is better that we learn *Mathematica* in natural stages that follow our level of need. In other words, we will find and introduce more sophisticated commands, routines, and procedures as we need them, so that their function is understood and retained, rather than trying to cover everything at once. With this in mind let us turn now to some Calculus functions.

1.7 Differentiate and Integrate

Chemical engineering is a science of chemical change and extents. When we need to treat change we are necessarily interested in rates of change either in time or in space or both. The language of change is Calculus. Here we will show how *Mathematica* provides with the bed-rock of applied Calculus—differentiation and integration. *Mathematica* will differentiate and integrate, both symbolically and numerically. Furthermore, it has many different ways to do numerical integration, methods that can be chosen by the user for any given application. We can begin with symbolic differentiation and integration.

Differentiation can be ordinary or partial. Here are two examples that illustrate how this is done. The syntax is simple we write **D[f[x], x]**, which means take the ordinary derivative of the function of **x** with respect to **x**. We can also use the **Basic Input** palette to do the same, but now we place the variable that we want to take the derivative with respect to in the subscript box under $\partial_x f[x]$**:**

```
In[104]:= ∂x(A1 x^3 + B1 x + C1)
          D[A1 x^3 + B1 x + C1, x]

Out[104]= B1 + 3 A1 x^2

Out[105]= B1 + 3 A1 x^2
```

To take higher-order derivatives we specify the order n in the argument, that is, we state **D[f[x], {x, n}]**:

```
In[106]:= D[A1 x^3 + B1 x + C1, x]
          D[A1 x^3 + B1 x + C1, {x, 2}]
          D[A1 x^3 + B1 x + C1, {x, 3}]
          D[A1 x^3 + B1 x + C1, {x, 4}]

Out[106]= B1 + 3 A1 x^2

Out[107]= 6 A1 x

Out[108]= 6 A1

Out[109]= 0
```

To take a partial derivative, we follow the same syntax. From the command line we type in, for example, **D[f[x, y], x]** or **D[f[x, y], y]** if we want the partial derivative of **f[x, y]** with respect to **x** or **y**. Using the input palettes we do as we did before:

```
In[110]:= ∂x(A1 x^2 y + B1 x y^2)
          ∂y(A1 x^2 y + B1 x y^2)
          D[A1 x^2 y + B1 x y^2, x]
          D[A1 x^2 y + B1 x y^2, y]

Out[110]= 2 A1 x y + B1 y^2
```

```
Out[111]= A1 x^2 + 2 B1 x y

Out[112]= 2 A1 x y + B1 y^2

Out[113]= A1 x^2 + 2 B1 x y
```

Taking second-order ordinary or partial derivatives follows much the same syntax:

```
In[114]:= ∂x,x(A x^2 y + B x y^2)
          ∂y,y(A x^2 y + B x y^2)
          ∂x,y(A x^2 y + B x y^2)
          ∂y,x(A x^2 y + B x y^2)

Out[114]= 2 A y

Out[115]= 2 B x

Out[116]= 2 A x + 2 B y

Out[117]= 2 A x + 2 B y
```

For higher-order partial derivatives, we use the command line syntax:

```
In[118]:= D[A x^2 y + B x y^2, {x, 2}]
          D[A x^2 y + B x y^2, {y, 2}, {x, 1}]
          D[A x^2 y + B x y^2, {y, 2}, {x, 2}]

Out[118]= 2 A y

Out[119]= 2 B

Out[120]= 0
```

Turning now to the antiderivative we can do symbolic integrations. Integration can be done either from the palette or from the command line and we will illustrate both. Here are two forms of the indefinite integral over **x** of (**A x** + **B**):

```
In[121]:= ∫(A x + B) ⅆx

          Integrate[A x + B, x]
```

$$\text{Out[121]= } B\,x + \frac{A x^2}{2}$$

$$\text{Out[122]= } B\,x + \frac{A x^2}{2}$$

We can integrate from **x1** to **x2**, that is, also as a definite integral:

$$\text{In[123]:= } \int_{x1}^{x2} (A\,x + B)\, dx$$

```
          Integrate[(A x + B), {x, x1, x2}]
```

```
Out[123]= -B x1 - A x1^2/2 + B x2 + A x2^2/2
Out[124]= -B x1 - A x1^2/2 + B x2 + A x2^2/2
```

$$\text{Out[123]} = -B\,x1 - \frac{A\,x1^2}{2} + B\,x2 + \frac{A\,x2^2}{2}$$

$$\text{Out[124]} = -B\,x1 - \frac{A\,x1^2}{2} + B\,x2 + \frac{A\,x2^2}{2}$$

The algebraic output in this case can easily been seen to be simplifiable. To find more simplified forms we request that *Mathematica* do the simplification for us. We can combine this into one command line:

```
In[125]:= Simplify[Integrate[(A x + B), {x, x1, x2}]]
```

$$\text{Out[125]} = -\frac{1}{2}(x1 - x2)(2\,B + A(x1 + x2))$$

Alternatively, we may have wanted to collect the terms in **A** and **B**; we would do that this way:

```
In[126]:= Collect[Integrate[(A x + B), {x, x1, x2}], {A, B}]
```

$$\text{Out[126]} = B(-x1 + x2) + A(-\frac{x1^2}{2} + \frac{x2^2}{2})$$

The function we are integrating may be one with two variables:

```
In[127]:= Simplify[Integrate[A x^2 y + B x y^2, {x, x1, x2}, {y, y1, y2}]]
```

$$\text{Out[127]} = \frac{1}{6}(A\,(x1^3 - x2^3)(y1^2 - y2^2) + B(x1^2 - x2^2)(y1^2 - y2^2))$$

Integration and differentiation can be done both numerically and symbolically. This becomes very important to us, because in many cases we need both approaches in engineering problems of the kind that we will deal with in this text. As we have seen previously, the syntax is kept very much the same when we compare the numerical command implementation to that of its symbolic analogue. This means that we will place an **N** in front of the command and we will specify a numerical range for the variable or variables we are integrating over in the argument. Also, as in the case of **NSolve**, we must be sure to have values for all the parameters. Examples are the best way to illustrate how this works:

```
In[128]:= A = 10;
          B = 0.5;
          C1 = 1;
          NIntegrate[A x^2 + B x + C1, {x, 0, 10}]

Out[131]= 3368.33
```

Numerical differentiation is about as simple to implement. We take the derivative and then evaluate it at a given point. The simplest way to do this is to add the evaluation command directly after the derivative, using "**/. x → a**" so that the derivative is evaluated immediately at **x** equal to **a:**

```
In[132]:= A = 10;
          B = 0.5;
          C1 = 1;
          D[A x^2 + B x + C1, x] /. x → 10

Out[135]= 200.5
```

From this vantage we are in a position to move to differential equation solving using **DSolve** and **NDSolve**.

1.8 DSolve

Most of the differential equations that we will be called upon to solve in this text are ordinary rather than partial. We will need to know the initial conditions in order to solve them for a function that describes the behavior of the system we are analyzing. Both **DSolve** and **NDSolve** can be used seamlessly to accomplish this. They can be used for multiple coupled equations as well as they can be for single equations. Their syntax follows essentially that which we have seen for the commands that we have used to this point.

Early on we will find that many of the differential equations that we seek to solve belong to a general class that can be "separated." This means that all the independent variables can be placed on one side of the equation and the dependent ones on the other. An example of such an equation is:

$$\frac{df(x)}{dx} = -\mathrm{C1}\, f(x)$$

This can be rewritten as:

$$\frac{df(x)}{f(x)} = -\mathrm{C1}\, dx$$

The solution can be found by integrating both sides—on the left over **f(x)** and on the right over **x**. Hence the first equations we will want to solve may be solved via separation and integration. We can solve this equation, even though **f(x)** is left unspecified, over some interval from **x1** to **x2**:

```
In[136]:= ∫_{f[x1]}^{f[x2]} 1/f[x] ⅆf[x] == ∫_{x1}^{x2} -C1 ⅆx

Out[136]= -Log[f[x1]] + Log[f[x2]] == x1 - x2
```

This can be simplified as follows if we seek to find **f[x2]** with the initial condition that **f[x1]** is **fo** at **x1** equal to zero:

```
In[137]:= Solve[-Log[fo] + Log[f[x2]] == -C1 (-x1 + x2), f[x2]] /.
           {f[x1] →  fo, x1 → 0}

Out[137]= {{f[x2] → ⅇ^(-x2+Log[fo])}}
```

We see that we have a solution, but we find that there is a **Log** in the argument of the exponential. We then can ask *Mathematica* to simplify the solution:

```
In[138]:= Simplify[%]

Out[138]= {{f[x2] →  ⅇ^-x2 fo}}
```

Therefore, **f[x2] = fo** $e^{-C1\,x2}$. We can test this solution by placing it back in the differential equation on the left-hand side to see if the derivative will equal the right-hand side of the equation. To do this verification, we define the function for **f[x]** and then take its derivative and finally test if the derivative of the solution is the same as the original right-hand side of the equation. We do this last operation by placing the derivative and the right-hand side of the equation astride the double equal sign and all of this is then placed within the **Simplify** command. If the two elements on either side of "==" are in fact the same then *Mathematica* returns a "**True**" statement.

```
In[139]:= f[x_] :=  ⅇ^-C1x fo
          Simplify[∂x f[x] == -C1 f[x]]
          Remove[f]

Out[139]= True
```

We have learned several important new concepts from this example.

- Many differential equations are separable and are nothing more than the integration of the left-hand and right-hand sides.
- We can use **Integrate** or the palette equivalent to carry out this operation on the separated form of the equation.
- The solution we obtain can be made specific for the initial conditions by adding them at the end of an appropriate **Solve** statement.
- Solutions can typically be simplified.
- The solution must be verified by testing its validity in the original differential equation.

The last point may not seem important at this point but it is, especially when we derive analytical solutions that are far more complex. A slightly more complex form of a separable equation is one that involves a sum on the right-hand side, such as:

$$\frac{dg(x)}{dx} = \mathrm{C1} + \mathrm{C2}\, g(x)$$

This is also separable, but we have to take the whole of the right-hand side to the left to show this:

$$\frac{dg(x)}{\mathrm{C1} + \mathrm{C2}\, g(x)} = dx$$

This is amenable to the techniques we have just used for the simpler equation, except that now that we know what we are doing we will combine the steps including the verification:

```
In[140]:= ∫_{g[x1]}^{g[x2]} 1/(C1 + C2 g[x]) ⅆg[x] == ∫_{x1}^{x2} ⅆx;
          Flatten[Simplify[Solve[%, g[x2]] /.
           {g[x1] → go, x1 → 0, x2 → x}]]
          g[x_] := Evaluate[g[x] /. %]
          Simplify[∂x g[x] == C1 + C2 g[x]]
          g[x]
          Remove[g, go]
```

$$\texttt{Out[143]=}\ \{g[x] \to \frac{-1 + e^{C2\,x}(1 + C2\ go)}{C2}\}$$

```
Out[145]= True
```

$$\texttt{Out[146]=}\ \frac{-1 + e^{C2\,x}(1 + C2\ go)}{C2}$$

In one set of statements we have solved the separated equation, rearranged for the function subject to the initial conditions, defined the function, verified it, and then restated the solution.

Generally, we can use **DSolve** to find an analytical solution when one is possible. This is more general because **DSolve** can find solutions to much more complex cases than we have examined to this point—that is, for those equations that are not separable. If no analytical solution exists, then we can solve the equation numerically with **NDSolve**. We will see here how these two powerful commands work.

We can redo the problem that we have just finished to see what is similar and different about using **DSolve**. The syntax is such that we place the equation and the initial condition in braces, followed by the name of the function we seek and the name of the independent variable:

```
In[148]:= DSolve[{∂x g[x] == C1 + C2 g[x], g[0] == go}, g[x], x]
```

$$\texttt{Out[148]=}\ \{\{g[x] \to \frac{-1 + C2\, e^{C2\,x}(\frac{1}{C2} + go)}{C2}\}\}$$

Verification can be done as we did before:

```
In[149]:= g[x_] := (-C1 + C2 ⅇ^(C2 x) (C1/C2 + go))/C2
          Simplify[∂x g[x] == C1 + C2 g[x]]
          Remove[g, go]

Out[150]= True
```

Another type of equation that we are likely to encounter is the linear first-order differential equation (**LFODE**). An example is given here:

```
In[152]:= DSolve[{∂x y[x] + C1 y[x] == g[x], y[0] == yo}, y[x], x]
```

$$\text{Out[152]=}\ \{\{y[x] \to e^{-x}\left(yo + \int_0^x e^{\text{DSolve`t}}\, g[\text{DSolve`t}]\, d\text{DSolve`t}\right)\}\}$$

Notice that the solution is implicit—meaning that it is not fully evaluated. We can see that this is so from the fact that on the right-hand side we have an integral that is over the function g and is left in terms of the dummy variable **DSolve‘t**. Notice also that the exponential involves this variable as well. Until **g[x]** is specified, we cannot find the full solution to this problem. We can see what happens when **g[x]**= x^2 or **Sin[x]**, that is, for specific functional forms:

```
In[153]:= Clear[C1, yo]
          DSolve[{∂x y[x] + C1 y[x] == x², y[0] == yo}, y[x], x]
```

$$\text{In[154]=}\ \{\{y[x] \to \frac{e^{-C1x}(2\,e^{C1x} - 2C1\,e^{C1x}x + C1^2\,e^{C1\,x}x^2 + C1^3(-\frac{2}{C1^3} + yo))}{C1^3}\}\}$$

```
In[155]:= Clear[C1, yo]
          DSolve[{∂x y[x] + C1 y[x] == Sin[x], y[0] == yo}, y[x], x]
```

$$\text{Out[156]=}\ \{\{y[x] \to \frac{e^{-C1\,x}(\frac{1+yo+C1^2yo}{1+C1^2}+\frac{C1^2(1+yo+C1^2yo)}{1+C1^2} - e^{C1\,x}Cos[x] + C1e^{C1\,x}Sin[x])}{(-i + C1)(i + C1)}\}\}$$

These solutions are involved and so it is critical that we verify them before applying them:

```
In[157]:= Clear["Global`*"]
```

$$\text{y1[x\_] :=}\ \frac{e^{-C1x}(2\,e^{C1x} - 2C1\,e^{C1x}x + C1^2e^{C1x}x^2 + C1^3(-\frac{2}{C1^3} + yo))}{C1^3}$$

```
          Simplify[∂x y1[x] + C1 y1[x] == x²]

Out[159]= True

In[160]:= Clear[C1, yo]
```

$$\text{y2[x\_] := }(e^{-C1x}(\frac{1 + yo + C1^2yo}{1 + C1^2} + \frac{C1^2(1 + yo + C1^2yo)}{1 + C1^2}$$
$$- e^{C1x}Cos[x] + C1\,e^{C1x}Sin[x]))/((-i + C1)\ (i + C1))$$

```
          Simplify[∂x y2[x] + C1 y2[x] == Sin[x]]

Out[162]= True
```

Both are valid solutions and can then be simplified further before we utilize them:

```
In[163]:= Simplify[y1[x]]
```

```
Out[163]= 
```
$$\frac{e^{-C1\,x}(-2 + e^{C1\,x}(2 - 2C1\,x + C1^2 x^2) + C1^3 yo)}{C1^3}$$

```
In[164]:= y1[x_] :=
```
$$\frac{e^{-C1x}(-2 + e^{C1x}(2 - 2C1\,x + C1^2 x^2) + C1^3\,yo)}{C1^3}$$

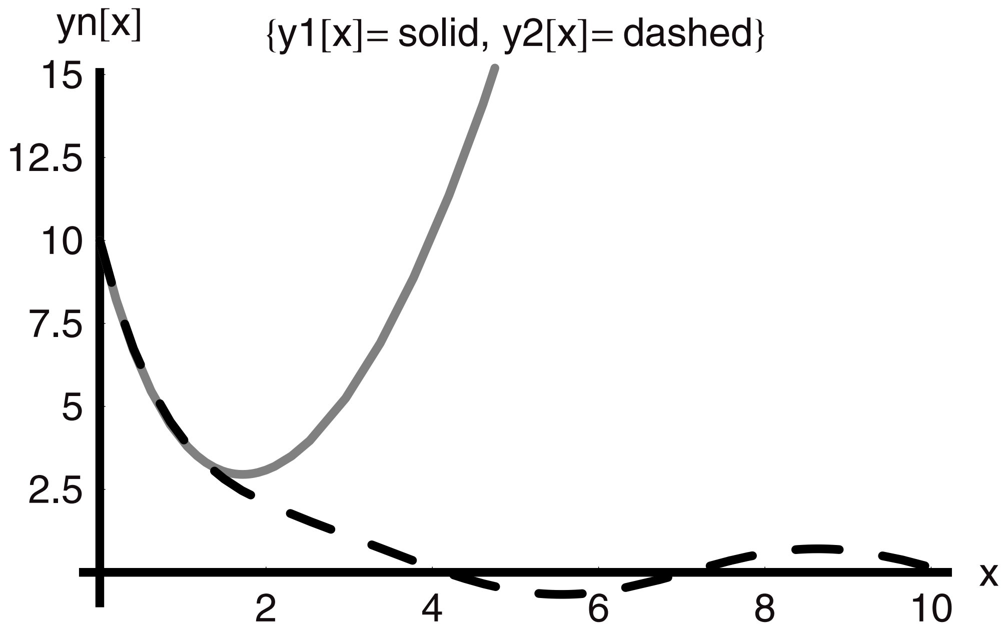
```
          Simplify[y2[x]]
```

```
Out[165]= 
```
$$\frac{e^{-C1x}(1 + yo + C1^2\,yo - e^{C1x}Cos[x] + C1\,e^{C1x}\,Sin[x])}{1 + C1^2}$$

```
In[166]:= y2[x_] :=
```
$$\frac{e^{-C1x}(1 + yo + C1^2\,yo - e^{C1x}\,Cos[x] + C1\,e^{C1x}\,Sin[x])}{1 + C1^2}$$

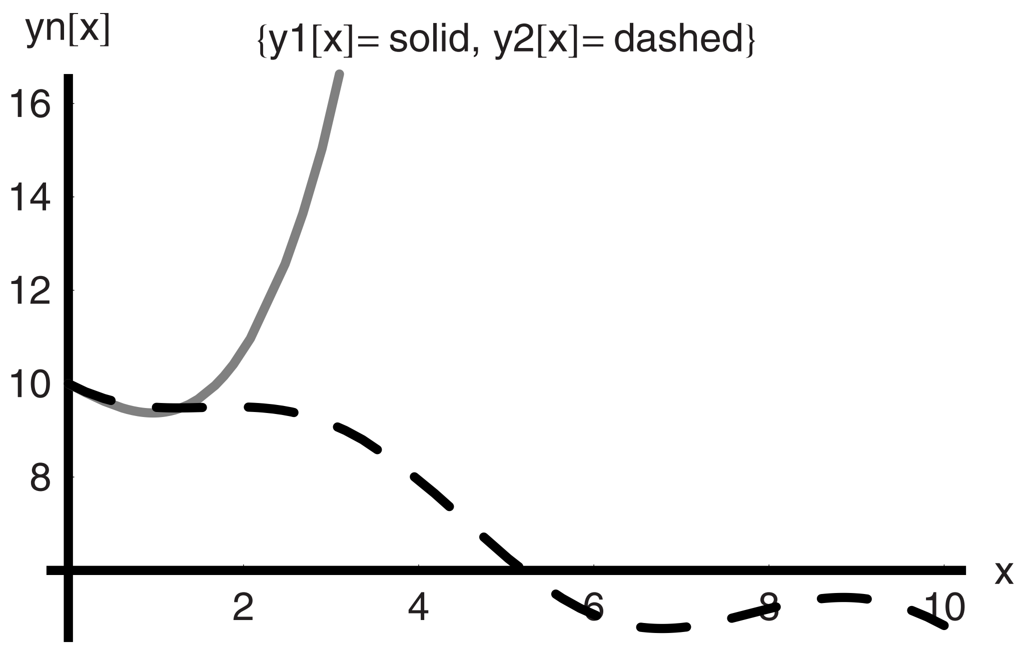
Finally, given a set of parameter values for **C1** and **yo**, we can **Plot** the two solutions simultaneously, with solid for **y1** and dashed for **y2** to see how they behave with increasing x in a specific range:

```
In[167]:= C1 = 1;
          yo = 10;
          Plot[{y1[x], y2[x]}, {x, 0, 10},
            PlotStyle → {{Thickness[.01], GrayLevel[.5],
             Dashing[{0}]}, {Thickness[0.01], Dashing[{0.05, 0.05}]}},
            PlotLabel → {solid "y1[x]=", dashed "y2[x]="},
            AxesStyle → {Thickness[0.01]},
            AxesLabel → {x, "yn[x]"}];
```

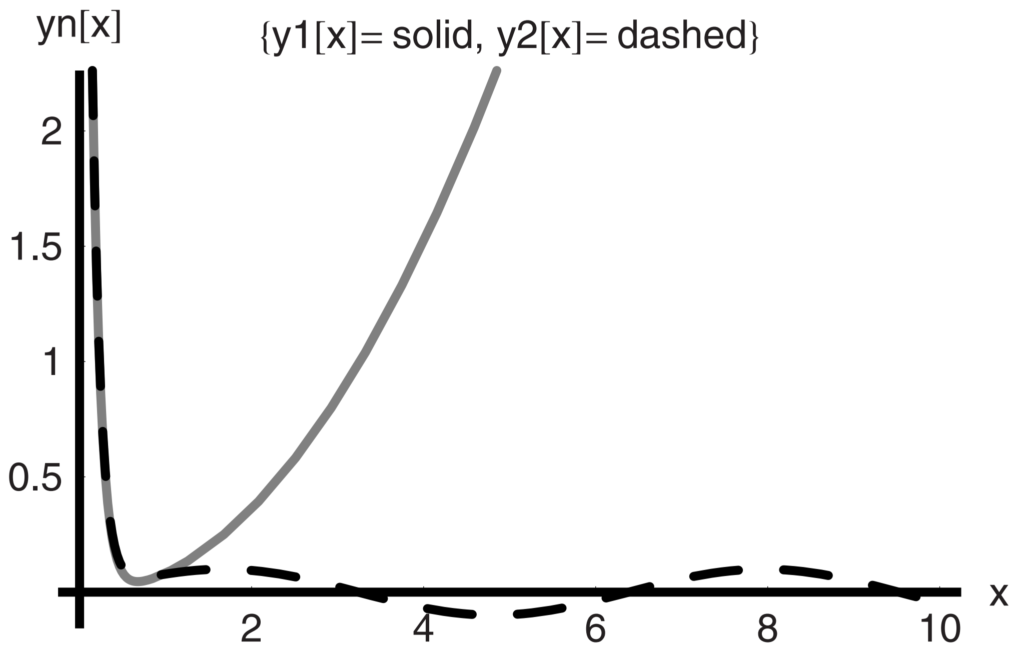


We can see that the two solutions are in fact quite different in their behavior. For these parameter values the first is dominated by the quadratic term and the second by the **Sin** function. If we want to obtain a sense of parametric sensitivity, we can drop the value of **C1** by 10^3 and then raise it by 10^2 and replot the graphs for these two cases:

```
In[170]:= C1 = .1;
          yo = 10;
          Plot[{y1[x], y2[x]}, {x, 0, 10},
            PlotStyle → {{Thickness[.01], GrayLevel[.5],
             Dashing[{0}]}, {Thickness[0.01], Dashing[{0.05, 0.05}]}},
            PlotLabel → {solid "y1[x]=", dashed "y2[x]="},
            AxesStyle → {Thickness[0.01]},
            AxesLabel → {x, "yn[x]"}];
```



```
In[173]:= C1 = 10;
          yo = 10;
          Plot[{y1[x], y2[x]}, {x, 0, 10},
            PlotStyle → {{Thickness[ .01], GrayLevel[.5],
             Dashing[{0}]}, {Thickness[0.01], Dashing[{0.05, 0.05}]}},
            PlotLabel → {solid "y1[x]=", dashed "y2[x]="},
            AxesStyle → {Thickness[0.01]},
            AxesLabel → {x, "yn[x]"}];
```



```
In[176]:= Remove[y1, y2, yo, C1]
```

1.9 NDSolve

We turn now to **NDSolve** for the solution of differential equations. A good starting point would be to begin to solve the equations that we have already solved symbolically with **DSolve**. Instead of simply solving the equation we are going to name the solution. We will call it **soln:**

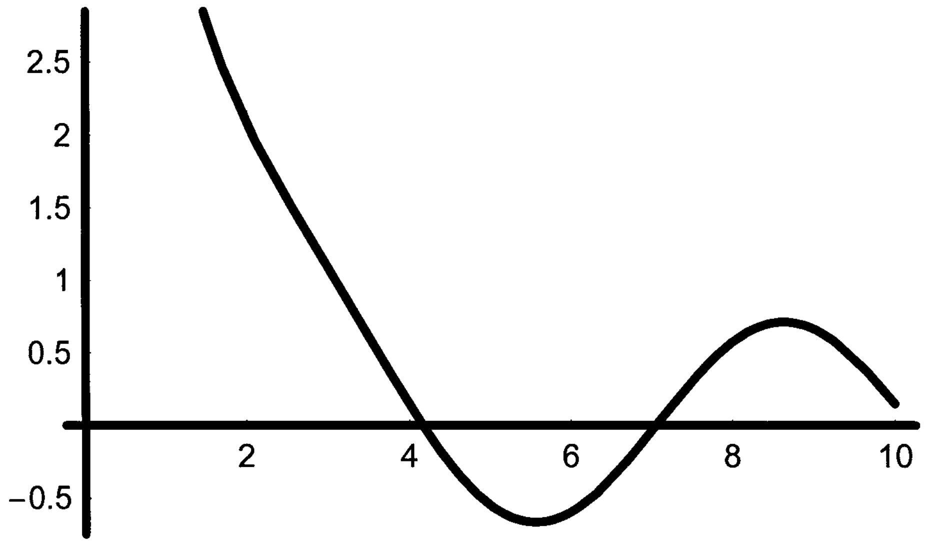
```
In[177]:= Clear[C1, yo, soln, y]

In[178]:= yo = 10;
          C1 = 1;
          soln = NDSolve[
            {∂x y[x] + C1 y[x] == Sin[x], y[0] == yo},
             y[x], {x, 0, 10}];
```

What we find is that the numerical solution is presented in the form of a tidy **Interpolation** function, which is good over the entire range of integration. This is much cleaner than having a table or list of values echoed to the monitor. But to use the interpolated function we must assign it a function name and then we can apply it and explore the numerical solution's behavior. To do this we use a command structure that we have utilized before; it looks like this:

```
In[181]:= nyb[x_] := Evaluate[y[x] /. soln]
```

What this says in simple terms is to assign to **nyb[x]** to the interpolating function **y[x]** found in the solution called **soln**. This function can now be plotted:

```
In[182]:= Plot[nyb[x], {x, 0, 10}, AxesStyle → {Thickness[0.01]}];
```



This looks identical to the plot we had before based upon the analytical solution. If we need to have a table of values for the function we can obtain this as follows:

```
In[183]:= Table[{x, nyb[x]}, {x, 0, 10, .5}] // TableForm

Out[183]//TableForm =
          0      10.
          0.5    6.16951
          1.     4.01333
          1.5    2.80625
          2.     2.08375
          2.5    1.5617
```

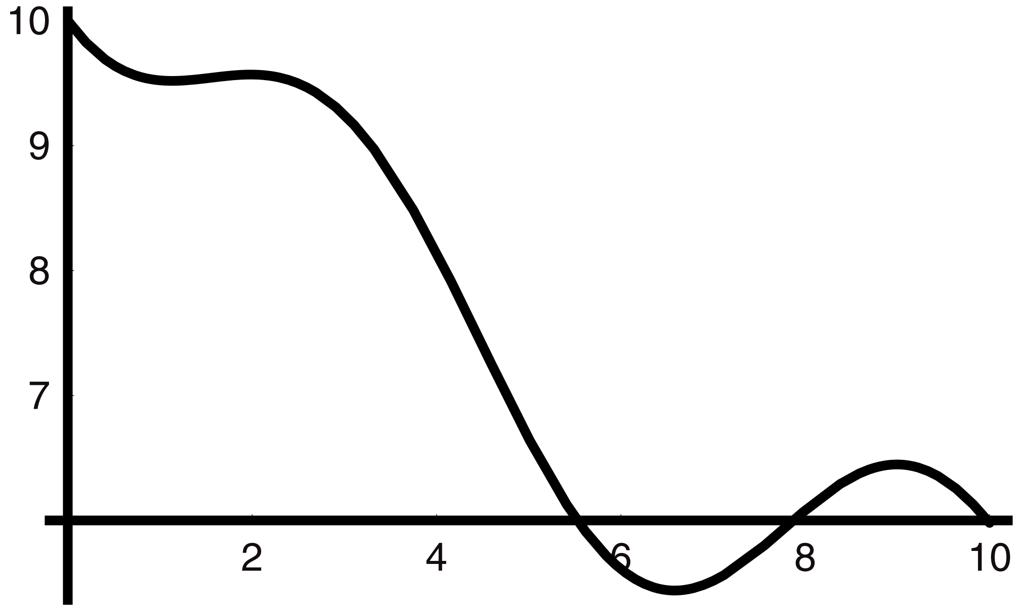
```
3.      1.08832
3.5     0.609911
4.      0.140736
4.5     −0.266721
5.      −0.550543
5.5     −0.664192
6.      −0.593765
6.5     −0.364947
7.      −0.0388831
7.5     0.30149
8.      0.570951
8.5     0.702385
9.      0.66292
9.5     0.461796
10.     0.148002
```

It is clear that for relatively simple linear equations such as these **DSolve** and **NDSolve** duplicate each other. When the equations become nonlinear, however, it may not be possible to find an analytical solution. At that point **NDSolve** no longer merely duplicates but, rather, it supplants **DSolve**. For example, if in the last differential equation **y[x]** appears quadratically rather than linearly, **DSolve** will not return a solution:

```
In[184]:= Clear[C1, yo]
          DSolve[{∂x y[x] + C1 y[x]^2 == Sin[x], y[0] == yo}, y[x], x]

Out[185]= DSolve[{C1 y[x]^2 + y'[x] == Sin[x], y[0] == yo}, y[x], x]
```

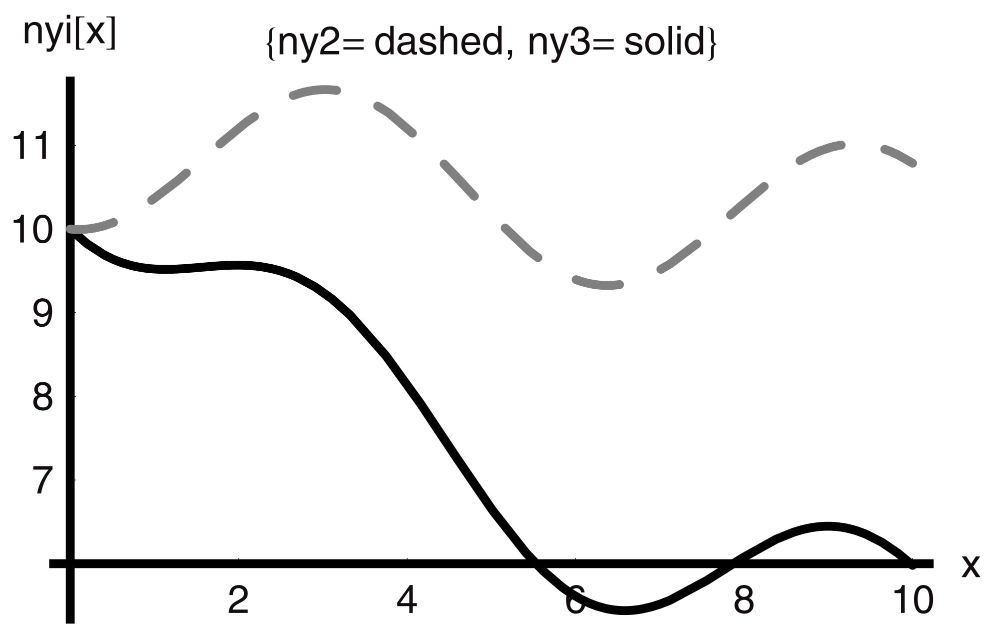
In contrast, NDSolve will do so and it will do it well:

```
In[186]:= Clear[C1, yo]
          yo = 10;
          C1 = 0.01;
          y3 = NDSolve[
              {∂x y[x] + C1 y[x]^2 == Sin[x], y[0] == yo}, y[x],
               {x, 0, 10}];
          ny3[x_] := Evaluate[y[x] /. y3]
          plny3 = Plot[ny3[x], {x, 0, 10},
              AxesStyle → Thickness[0.01],
              PlotStyle → Thickness[0.01]];
```



For the sake of learning we can now go back and compare the solution of this nonlinear equation to the linear version. To do so numerically, we must resolve the equation with the new value of **C1** set to 0.01:

```
In[217]:= Clear[C1, yo]
          yo = 10;
          C1 = 0.01;
          y2 = NDSolve[
              {∂x y[x] + C1 y[x] == Sin[x], y[0] == yo},
               y[x], {x, 0, 10}];
          ny2[x_] := Evaluate[y[x] /. y2]
          plny2 = Plot[ny2[x], {x, 0, 10},
             PlotRange → {{0,10}, {0, 12}},
             AxesStyle → Thickness[0.01],
             PlotStyle → {GrayLevel[.5], Thickness[0.01],
              Dashing[{0.05, 0.05}]},
             DisplayFunction → Identity];
          Show[plny3, plny2, DisplayFunction → $DisplayFunction,
             PlotLabel → {dashed "ny2=", solid "ny3="},
             AxesLabel → {"x", "nyi[x]"}];
```



1.10 Units Interconversion

Mathematica also provides a special package for the interconversion of units of measure. To access this functionality of the software we need to load the package named **Miscellaneous** ‘**Units**‘. We load this and other specialized packages from the *Mathematica* library with the following command:

```
In[199]:= <<Miscellaneous‘Units‘
```

This allows us to begin doing units interconversion immediately. The following are some examples of this utility:

```
In[200]:= Convert[5 Kilo Meter, Mile]

Out[200]= 3.10686 Mile

In[201]:= Convert[80 Year, Day]
          Convert[80 Year, Second]

Out[201]= 29200 Day

Out[202]= 2522880000 Second
```

```
In[203]:= Convert[2500 Kilo Joule, Calorie]

Out[203]= 597115. Calorie

In[204]:= Convert[25 Furlong/Fortnight, Mile/Hour]

          Convert[25 Furlong/Fortnight, Mile/Hour] // N
```

Out[204]= $\frac{25\ \text{Mile}}{2688\ \text{Hour}}$

Out[205]= $\frac{0.0093006\ \text{Mile}}{\text{Hour}}$

```
In[206]:= ConvertTemperature[19, Fahrenheit, Centigrade]
          ConvertTemperature[19, Fahrenheit, Rankine]
          ConvertTemperature[19, Fahrenheit, Kelvin]

Out[206]= -7.22222

Out[207]= 478.67

Out[208]= 265.928

In[209]:= Convert[1 Atmosphere, Bar]

Out[209]= 1.01325 Bar

In[210]:= Convert[1 TonForce, Dyne]
```

Out[210]= 9.96402×10^{8} Dyne

```
In[211]:= Convert[1 Ton, Gram]
```

Out[211]= 1.01605×10^{6} Gram

```
In[212]:= Convert[25 Angstrom, Micron]
          Convert[25 Angstrom, Micron] // N
```

Out[212]= $\frac{\text{Micron}}{400}$

```
Out[213]= 0.0025 Micron
```

This same utility will also allow us to specify a measurement in an arbitrary system and then convert this to a specified system, such as **CGS, MKS**, or **SI**:

```
In[214]:= SI[350 Atmosphere]
          MKS[300 Feet]
          CGS[1 Inch]
```

3.546375×10^{7} Pascal

```
Out[215]= 91.44 Meter

Out[216]= 2.54 Centimeter
```

For more examples of this kind and to see what units are available to use in these interconversions click on the Master Index in the Help Browser and Go To Miscellaneous ‘Units‘.

1.11 Summary

Now we have the basic tool kit that we need in order to get started with *Mathematica*. As we go through the next eight chapters and before we get to the Worked Problems in Chapter 10, we will build upon this foundation and add to these tools.

CHAPTER 2

Elementary Single-Component Systems

Elementary single-component systems are those that have just one chemical species or material involved in the process. Filling of a vessel is an example of this kind. The component can be a solid liquid or gas. Regardless of the phase of the component, the time dependence of the process is captured by the same statement of the conservation of mass within a well-defined region of space that we will refer to as the *control volume*.

In this chapter we will apply the conservation of mass principle to a number of different kinds of systems. While the systems are different, by the process of analysis they will each be reduced to their most common features and we will find that they are more the same than they are different. When we have completed this chapter, you will understand the concept of a control volume and the conservation of mass, and you will be able to write and solve total material balances for single-component systems.

2.1 The Conservation of Mass Principle and the Concept of a Control Volume

The conserved quantities that are of utmost importance to a chemical engineer are mass, energy, and momentum. It is the objective of this text to teach you how to utilize the conservation of mass in the analysis of units and processes that involve mass flow and transfer and chemical reaction. For each conserved quantity the principle is the same—conserved quantities are neither created nor destroyed. For mass this principle holds for all cases except those involving nuclear reactions. In all other situations, the principle is never violated. So we can use it to the utmost as you will see in developing both time-dependent and time-independent descriptions of chemical processes.

The principle is so seemingly obvious that you may wonder how it can be so useful to us. How does knowing that mass is neither created nor destroyed relate to a chemical process unit's behavior or to anything else for that matter? The key is that in order to use this principle we must translate it into mathematics so that we can work with it and derive the precise and accurate descriptions that we need.

If mass is neither created nor destroyed, that means if we seem to detect its apparent depletion or accrual in one region of space, this can only be the case if in some other region of space the same mass was either accruing or depleting. In other words, we always inspect some region of space and draw conclusions based on our measurements within that region. If mass is increasing within this space, it must be coming from somewhere else. Similarly, if we detect that mass is decreasing, then it is because it is leaving the region of our measurement. We have everyday experiences that correspond to these statements.The level of water in a glass left on a table at room temperature will slowly decrease as the water leaves via evaporation. Pulling the drain plug on a bathtub causes the water to flow out due to the force of gravity. When a stalk of corn grows all the mass that is accumulated in such complex forms within the plant had to be delivered to it from the soil and the surrounding atmosphere. Each of these, the glass of water, the tub, and the corn plant, can be considered a "system," and as such we can measure the rate of change that occurs within them whether it is through evaporative losses, flow, or growth. This is because each involves the transport and transfer of mass from outside of the system to inside of it or vice versa.

Another example is that of a living cell. Nutrients are transported across the cellular membrane and are utilized in metabolism. The by-products of metabolism are transported out of the cell and also back across the membrane to the surroundings. The young cell grows and increases in size and mass because the rate of by-product flow out is less than the rate of nutrient flow in. We know this because of the conservation of mass principle, and so we need no other information than to know that the cell grows in order to reach this conclusion. As the mass of the cell increases, the size of the cell also increases. If the cell is nearly spherical as is the case for some simple, single-cell organisms, then we can expect that its diameter or radius is also increasing. Hence, the simplest measurement to make to detect cell growth in an experiment may be to measure the cellular radii. When the cell matures, we find that the rate of nutrient flow in is balanced by the rate out, which is why the cell no longer is growing. This of course says nothing about the complex metabolic control mechanisms that lead to this situation, but it does define maturity explicitly in dynamical terms. In this condition, when the input rate is balanced by the output rate, there is no net accumulation of mass in the cell. The cell biologist refers to this as the *homeostatic state*; the chemical engineer calls it the *steady state*.

In these word statements we find that which we need to formalize at this point. The conservation of mass is applied to a system and more specifically to a ***control volume***, which is defined by a control surface that separates the control volume from its surroundings, either in actuality or abstractly. By defining the control volume and its boundaries, we know where "inside" is. The inside of the cell is that space within the membrane just as the inside of the glass lies within its regular walls. The same is true for the corn plant, even though it has a more complex geometry defining its control surface. Now we can begin to bring mathematical descriptions to bear on the problem, but not until we have accurately stated the conservation of mass in terms of the control volume and its boundaries:

> **The net rate of mass accumulation within a control volume is equal to the rate at which mass enters the control volume by any process minus the rate at which it leaves the control volume by any process.**

The mathematics that proceeds from this is at once simple and elegant. Since we are discussing rates we will write the mathematical statement in terms of rates also—the rate of change in mass within the control volume:

$$\frac{dm[t]}{dt} = \mathring{m}_{in} - \mathring{m}_{out}$$

$\frac{dm[t]}{dt}$ = the net rate of change in mass within the control volume, rate of accumulation; mass/time;

$\mathring{m}_{in}$ = the total rate of mass flow into the control volume by any means; mass/time;

$\mathring{m}_{out}$ = the total rate of mass flow out of the control volume by any means; mass/time.

This is the key unifying principle that we will use throughout this book. It will be all we need in order to analyze and model a wide array of elementary single-component systems and it is the foundation upon which everything else we do with more complex systems will be built. The best way to illustrate how to use this mathematical statement of conservation of mass is through examples.

Filling a Vessel with a Pelletized Solid: Conservation of Mass and the Constitutive Relationship

Many products come in the form of a powdered solid. The solid once produced is stored in a container. It may be a barrel, a bag, or a can depending on the volume. Powdered milk is a good example; so is lawn fertilizer. Catalytic solids are another. Catalysts promote the rate of chemical reaction and are used throughout the chemical and petroleum industries; they are usually small solid pellets of uniform size and shape. Catalysts are not consumed in the course of the reaction they promote. Nevertheless, catalysts do eventually need to be replaced. This is either because they were poisoned or their solid structures have become clogged with high molecular weight molecules that prevent access to the active sites. At the end of its lifetime, then, the catalyst must be replaced. The spent catalyst is removed from the reactor vessel, the reactor is cleaned, and the space left open is ready for a charge of fresh solid catalyst.

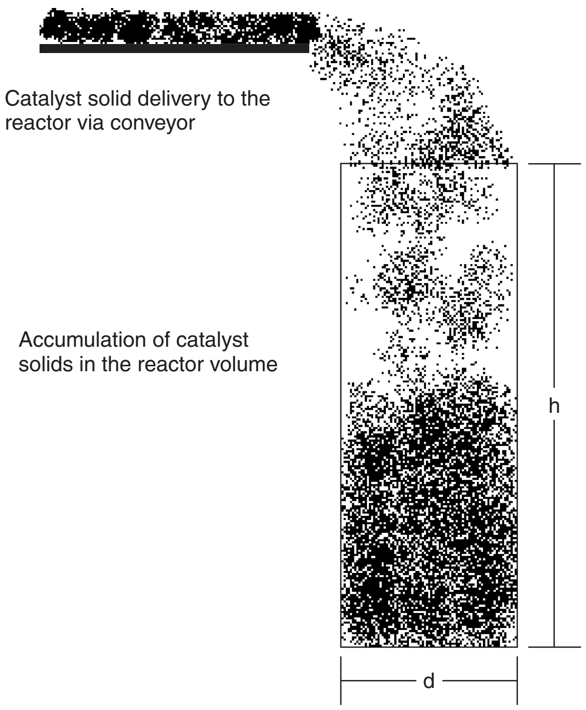

Catalyst solid delivery to the reactor via conveyor
Accumulation of catalyst solids in the reactor volume
h
d

Figure 1

The fresh catalyst will be delivered to the top of the reactor by a conveyor belt and dropped in. The process will proceed until the volume of the reactor vessel has been filled to the requisite level as is shown in Figure 1.

Reactors are often large in volume and cylindrical in shape. We will represent the reactor then as a simple cylindrical volume. The height of the vessel is **h** and its diameter is **d**. The overall volume to be filled by the catalyst is given as:

$$V == \pi d^2 h$$

Catalyst is being delivered by conveyor belt at a constant mass flow rate. The question we would like to be able to answer is: How much catalyst mass is in the reactor vessel at any time? The reason we care is that we will be paying for the catalyst on a per pound basis. If we look into the reactor at any time **t**, we may be able to measure the level to which the reactor is filled, and from that level measurement we could in principle compute the mass of catalyst if we had a density for the material. Remember though that this is solid and it packs irregularly into the reactor, as we can see from Figure 1. We can at best get an average value for the density and only after we have done an experiment in which the catalyst was carefully packed into a known volume and massed in order to find its so-called *compacted bulk density*.

Therefore, if we know the compacted bulk density, then it is possible to compute the mass in the bed using the mathematical statement for the conservation of mass. In this case the reactor and its physical dimensions define the control volume. The rate of catalyst delivery is a constant that we will call $\mathring{m}_{\mathbf{in}}$. The rate of mass flow out of the reactor is zero, that is, $\mathring{m}_{\mathbf{out}} = 0$. Therefore we have:

$$\frac{\mathrm{dm[t]}}{\mathrm{dt}} = \mathring{\mathrm{m}}_{\mathrm{in}}$$

This says that the rate of accumulation of catalyst in the reactor is just equal to the rate of delivery, which is exactly what we would have said based on common sense. An equation of this kind is the simplest type of differential equation. It is separable and we integrate from **t** = 0 to **t** and from **m[0]** = 0 to **m[t]**. The integrals that result from the separation of variables are shown in what follows. On the right-hand side we use the infix form " **/. m[0] → 0**" to tell *Mathematica* to use a lower-bound value of zero for m[0]:

$$\textit{In[1]:=}\quad \int_{\mathtt{m[0]}}^{\mathtt{m[t]}} d\mathbf{m[t]} == \int_0^{\mathbf{t}} \mathring{\mathbf{m}}_{\mathbf{in}}\, d\mathbf{t} \;/.\; \mathbf{m[0]} \to \mathbf{0}$$

$$\textit{Out[1]=}\quad \mathtt{m[t]} == \mathtt{t}\mathring{\mathtt{m}}_{\mathtt{in}}$$

We find that the mass of catalyst in the reactor is a simple linear function of time so long as the mass flow rate of catalyst via the conveyor remains constant. The dimensions on $\mathring{\mathbf{m}}_{\mathbf{in}}$ are **mass time**$^{-1}$, so we see that the resultant equation is dimensionally consistent.

```
In[2]:=  mass == (time) mass time^-1

Out[2]=  True
```

We can put some numbers into this result. Suppose that the reactor is fairly large in volume: it is 60 ft high and 20 ft in diameter. The catalyst delivery rate is 100 lb per hr. The compacted bulk density of the catalyst is 10 kg m^3. First, we want to know the mass of catalyst in kilograms in the reactor at any time **t**. We would also like to know to what level the reactor will be filled at time **t**, if the catalyst is packing in at its full compacted bulk density (**bd**). (As stated earlier this value can be obtained easily in the laboratory by simply filling a know volume with catalyst, being careful to leave no voids in the packing and then weighing the sample.) If we compare the actual volume in the bed to that which we calculate, then any difference between the two values will arise from the catalyst not packing at its **bd**.

To do this we will load a helpful package called **<<Miscellaneous 'Units'** from *Mathematica*. We also want graphs of the predicted catalyst mass as function of time, the theoretical level of catalytst in the reactor as a function of time, and the actual level that has been measured in the reactor at a few times during the loading process. Finally, we can compute the catalyst cost in $ flowing into the reactor volume per unit time. Here we calculate the mass flow in per unit time in metric units as well as the volume and cross-sectional area of the reactor.

```
In[3]:= << Miscellaneous`Units`
```

```
In[4]:= ṁin == NumberForm[Convert[1000 Pound/Hour,
            Kilogram/Minute], 2]
```

$$Out[4]= \ \mathring{m}_{in} == \frac{7.6\ \text{Kilogram}}{\text{Minute}}$$

```
In[5]:= Hreactor ==
         NumberForm[Convert[60 Feet, Meter], 3]
        Vreactor ==
```

$$\text{NumberForm}[\text{Convert}[\pi(\frac{20\ \text{Feet}}{2})^2\ 60\ \text{Feet},\ \text{Meter}^3],\ 3]$$

$$A_{reactor} == \text{NumberForm}[\text{Convert}[\pi(\frac{20\ \text{Feet}}{2})^2,\ \text{Meter}^2],\ 3]$$

```
Out[5]= Hreactor == 18.3 Meter
```

$Out[6]= V_{reactor} == 534.\ \text{Meter}^3$

$Out[7]= A_{reactor} == 29.2\ \text{Meter}^2$

Next we should compute the time it would take to fill the reactor if the catalyst were to pack in at its **cbd**. This time will be called $t_{\mathbf{max}}$. We find this time by setting the catalyst volume equal to the volume of the reactor in the mass balance and rearranging:

$In[8]:= V_{reactor} = 534.\text{Meter}^3$

$$\mathring{m}_{in} = \frac{7.6\ \text{Kilogram}}{\text{Minute}};$$

$bd = 10\ \text{Kilogram/Meter}^3;$

$$V_{cat}[t\_] := \frac{\mathring{m}_{in} t}{bd}$$

$\text{Solve}[V_{reactor} == V_{cat}[t_{mx}],\ t_{mx}]$

$Out[8]= 534.\ \text{Meter}^3$

$Out[12]= \{\{t_{mx} \rightarrow 702.632\ \text{Minute}\}\}$

```
In[13]:= N[Convert[702.6 Minute, Hour]]

In[14]:= NumberForm[Convert[%, Day], 3]

Out[13]= 11.71 Hour

Out[14]//NumberForm=
        0.488 Day
```

The time required to fill the reactor, if the catalyst packs in at its **bd**, is about five days or 117 hours. If the packing is at some bed density less than the **bd**, then the reactor volume will be apparently filled faster, but the catalyst load in mass will be below its design level due to voids in the bed. We will see later that if this were to go unnoticed and unrepaired, then the production rate for the reactor will fall below its design level because it does not contain the design mass of catalyst. Making the catalyst bed reach **bd** is important. The mass of catalyst that should be in the bed at t_{max} and **bd**, and the theoretical mass of catalyst in the bed at any time are found as follows:

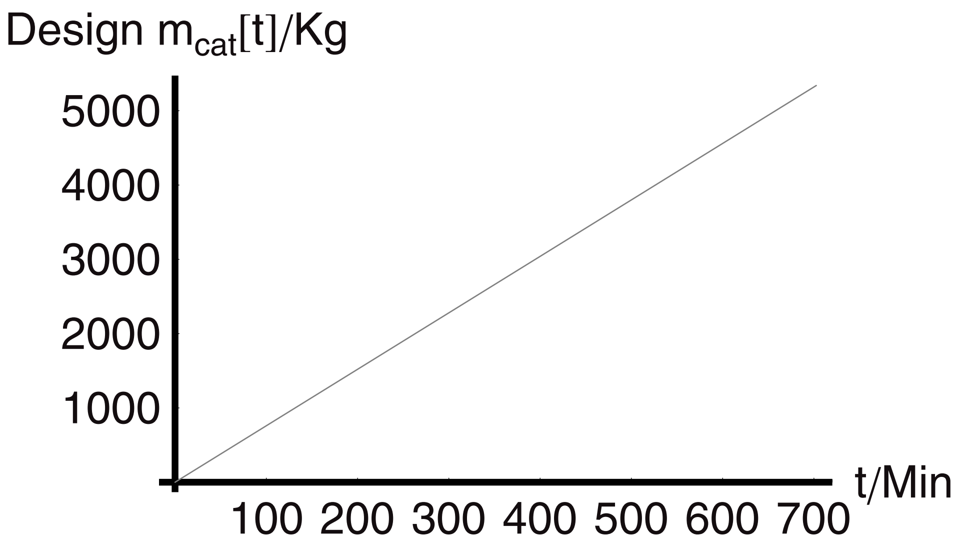
```
In[15]:= mcat[t_] := ṁin t
         tmxx = 702.6 Minute;
         mcat[tmx]
         Vcat[tmx]
```

$$Out[17]= \frac{7.6\,\text{Kilogram tmx}}{\text{Minute}}$$

$$Out[18]= \frac{0.76\,\text{Meter}^3\ \text{tmx}}{\text{Minute}}$$

At constant **bd** the catalyst will occupy the reactor fully and will have a total mass of 5340 kg. According to the model this mass will accumulate linearly in time:

```
In[19]:= SetOptions[{Plot, ListPlot}, AxesStyle → {Thickness[0.01]},
           PlotStyle → {PointSize[0.02], Thickness[0.006]},
           DefaultFont → {"Helvetica", 17}];

In[20]:= tf = 702.6;
         plmcat = Plot[(mcat[t] Minute/Kilogram), {t, 0, tf},
           AxesLabel → {"t/Min", "Design mcat[t]/Kg"},
           PlotStyle → GrayLevel[.5]];
```

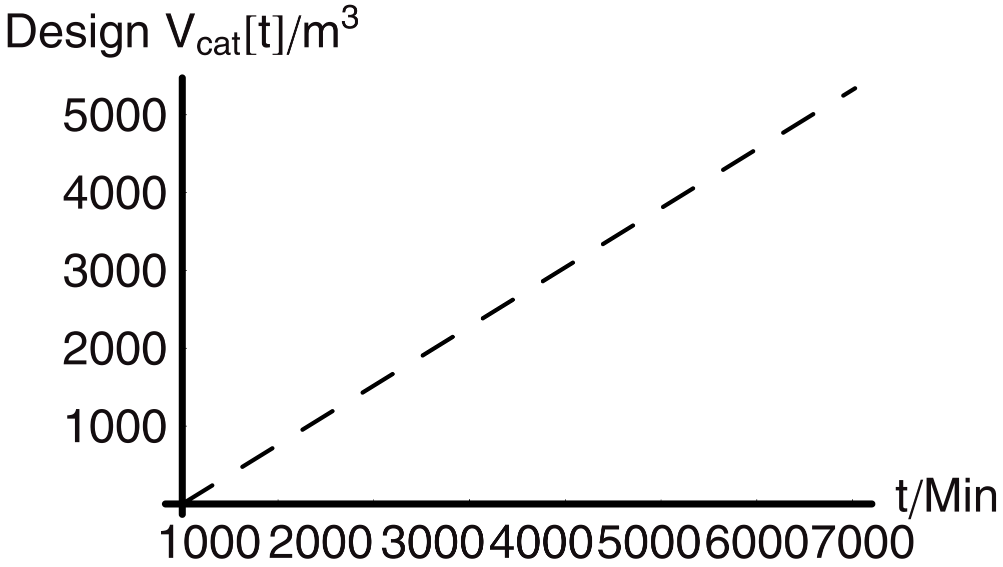


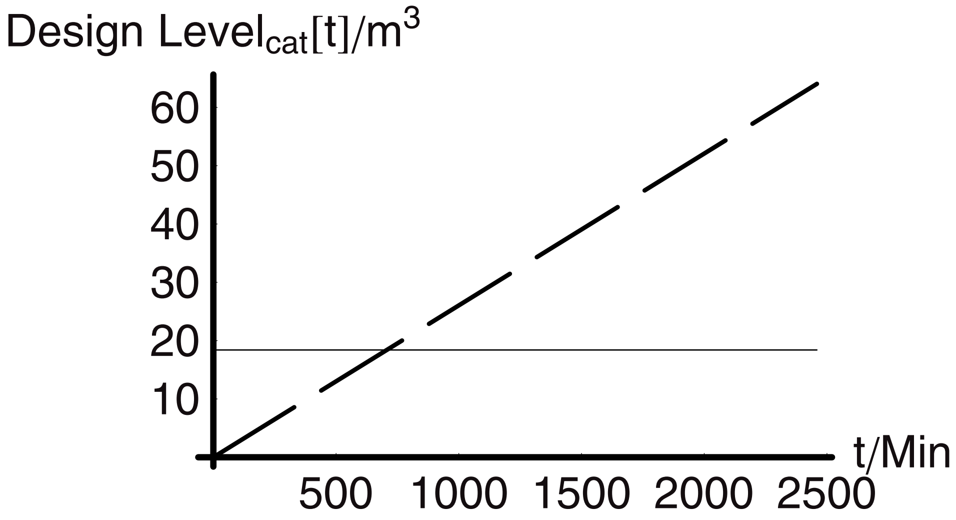
The volume and level of the catalyst bed will also vary linearly in time so long as the density remains constant:

```
In[22]:= tf = 702.6;
```

$$\text{lev}_{\text{cat}}[\text{t\_}] := \frac{\overset{\circ}{\text{m}}_{\text{in}}\text{t}}{\text{bd}}\left(\text{Convert}\left[\pi\left(\frac{20\ \text{Feet}}{2}\right)^2,\ \text{Meter}^2\right]\right)^{-1}$$

```
plvolcat = Plot[(Vcat[t]MinuteMeter^-3), {t, 0, 7026},
   PlotStyle → {Thickness[0.006], Dashing[{0.05,0.05}]},
   AxesLabel → {"t/Min", "Design Vcat[t]/m^3"}];

pllevelcat = Plot[(levcat[t] Minute Meter^-1),
   {t, 0, 3.5 tf}, PlotStyle → {Thickness[0.007],
    Dashing[{0.15, 0.05}]},
   AxesLabel → {"t/Min", "Design Levelcat[t]/m^3"},
   Epilog → Line[{{0, 18.4}, {3.5tf, 18.4}}]];
```



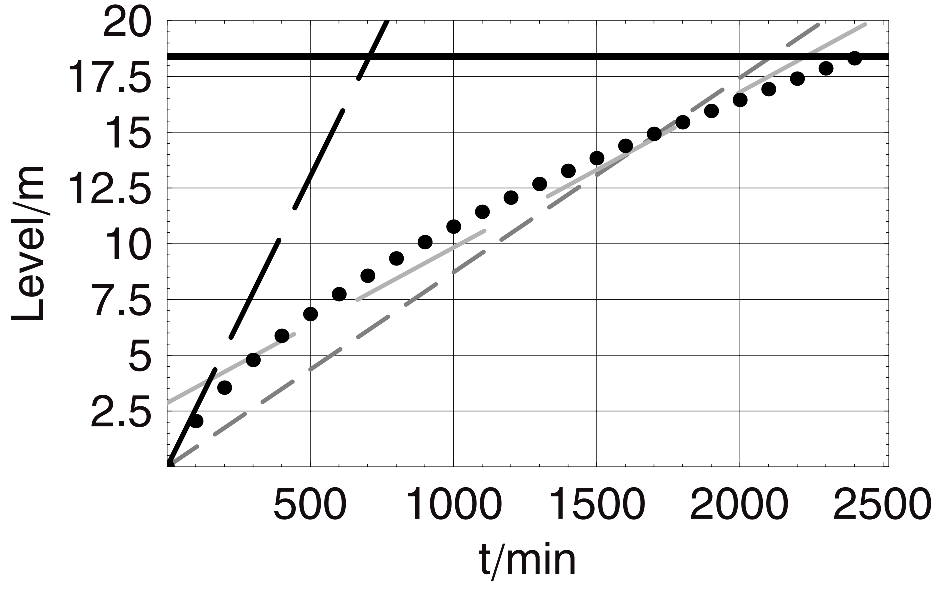


Now we will look at some actual data that accumulated as a function of time as the unit was being filled. We can enter this as follows and we name the data set "**levdata**":

```
In[26]:= levdata = {{0, "0."}, {100, "2.05"}, {200, "3.55"},
         {300, "4.79"}, {400, "5.87"}, {500, "6.85"}, {600, "7.74"},
         {700, "8.57"}, {800, "9.34"}, {900, "10.1"},
         {1000, "10.8"}, {1100, "11.4"}, {1200, "12.1"},
         {1300, "12.7"}, {1400, "13.3"}, {1500, "13.8"},
         {1600, "14.4"}, {1700, "14.9"}, {1800, "15.4"},
         {1900, "16."}, {2000, "16.4"}, {2100, "16.9"},
         {2200, "17.4"}, {2300, "17.9"}, {2400, "18.3"}};
```

It makes sense to try fitting this data to a line since that is exactly what our model suggests, that is, that we should have linear dependence upon time. We can do this by using the command **Fit**. We will fit the data to a line going through the point {0, 0} and also to a line with a nonzero intercept. We will also plot both of these results. We shall suppress the plots with **DisplayFunction → Identity** until we use the **Show** command, when we will use **DisplayFunction → $DisplayFunction** to render the graphic. To plot the actual data we use **ListPlot** and we suppress this also, and in the same way, until we use the **Show** statement. With **Show** we combine the two fitted function plots, the plot of level versus time from the analysis, and the data. The actual final level is the added horizontal line. We introduced this with the command **Epilog →Line[{{0, 18.4}, {3tf, 18.4}}** and we "turned on" each of these for display with the command **DisplayFunction →$DisplayFunction**:

```
In[27]:= tf = 720;

         Fit[levdata, {t}, t]

         ftpllevdat = Plot[%, {t, 0, 3.5 tf},
            PlotStyle→{GrayLevel[.5], Thickness[0.006],
             Dashing[{0.05, 0.03}]},
            DisplayFunction→Identity];

         Fit[levdata, {1, t}, t]

         ftpllevdat2 = Plot[%, {t, 0 , 3.5 tf},
            PlotStyle→{GrayLevel[.7], Thickness[0.006],
             Dashing[{0.2, 0.1}]},
            DisplayFunction→Identity];

         pllevdat = ListPlot[levdata, DisplayFunction→Identity];

         Show[{ftpllevdat, ftpllevdat2, pllevdat, pllevelcat},
           DisplayFunction→$DisplayFunction,
           Epilog→{Thickness[.01], Line[{{0, 18.4},
            {3.5tf, 18.4}}]},
           FrameLabel→{"t/min", "Level/m"},
           PlotRange→{{0, 3.5tf}, {0, 20}},
           Frame→True, GridLines→Automatic];

Out[28]= 0.00871916t

Out[30]= 2.86214 + 0.00696683t
```



The black dashed line is the result of the original model. We see that it crosses the fill line around 700 min. This is obviously a gross underprediction of the real time required to fill the unit, a time on the order of 2400 min. Now we fitted the data with two other lines, one with an intercept forced through zero (dark gray dashed line) and one in which the intercept was allowed to float (light gray dashed lines). Both do a better job of predicting the actual time to filling. Returning to the model projected line (black dashed lines): since it crosses the maximum level line at a time (~700 min) that is much less than the time that it actually took to fill the reactor, we must begin to question the physical premise that the catalyst bed remained at constant density, that is, **bd**, throughout the filling process. Constant density predicts that the reactor would be full of catalyst much too soon. As we know that the mass flow rate in is a constant, then the mass of catalyst in the reactor would be only one-third of the design level, which if left undetected (unlikely) would have disastrous consequences for the operability and economics of the process. When we look at the dark gray dashed line, we find that the prediction is much better as is than made with the light gray dashed line, but neither of these results is based upon a physical premise, and in fact the latter is unphysical in that the initial level in the unit was zero and yet its intercept is nonzero. Furthermore, both fitted functions under-predict and over-predict the level at different times, so neither would be useful for intermediate time predictions. Finally, neither the fitted model nor the physically based model captures the nonlinearity of the real data. Thus, we have a model based on physical reasoning that fits very poorly and two based on nonphysical reasoning that at best fit modestly. Clearly, we need to put more effort into this analysis.

What we need to realize is that as the catalyst level increases, more mass is present to bear down upon the underlying catalyst with more force. This causes the bed to compress. As the level rises the density at the bottom of the bed increases. With time and higher levels this occurs throughout the whole of the bed. Even though the density begins at **bd** then, it actually rises to a value above **bd**, especially at the bottom of the bed. The higher value at the bottom of the bed gives rise to an average across the bed that is higher than **bd** measured in the laboratory.

To handle this we can physically reason that the density of the bed must be a function of the level of filling of the bed. We need to bring this idea into the analysis quantitatively so that we might better predict the level as a function of time in the reactor. We begin with the statement of conservation of mass in the reactor:

$$\frac{\mathrm{dm[t]}}{dt} = \mathring{\mathrm{m}}_{\mathrm{in}}$$

We could integrate this expression and then convert it to the volume and level of catalyst by use of the **bd** but this assumes that the density in the bed always remained at **bd**, which we now realize to be incorrect. The next problem then is to find a way to bring this change in bed density into the original analysis. To do this we express the mass of catalyst as the product of the volume at any time and the bulk density, which in turn could be related to the level at any time:

$$\mathrm{m[t]} = \mathrm{bd\ V[t]} = \mathrm{bd}\ A_{\mathrm{reactor}}\ \mathrm{Lev[t]}$$

However, as we now know, the bulk density **bd** is not a constant. In fact, the average bulk density in the bed is a function of the mass of the bed and therefore time, giving at this point the following equation for the conservation of mass:

$$\frac{\mathrm{dm[t]}}{dt} = \frac{\mathrm{d[bd[t]}\,A_{\mathrm{reactor}}\,\mathrm{lev[t]]}}{dt} = \mathring{\mathrm{m}}_{\mathrm{in}}$$
$$\frac{\mathrm{d[db[t]\,lev[t]]}}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{\mathrm{A}_{\mathrm{reactor}}}$$

The equation as written cannot be solved. To solve it we need a relationship between the bulk density and the level of filling in the reactor. As we do not have a source for this we make an educated guess. It would be intuitive to assume that the bulk density at any level is proportional to the level:

$$\mathrm{bd[lev[t]] = k\,lev[t]}$$

However, this would be "unphysical" in that it would suggest zero bulk density at zero level. We can improve matters by letting the original bulk density vary linearly with level as follows:

$$\mathrm{bd[lev[t]] = bdo + k\,lev[t]}$$

This relationship has the benefit of providing a more physical result at zero level, but it suffers from the fact that the density continues to grow in an unbounded fashion with increasing level. We can instead imagine that the bulk density will increase with level or crushing force but that the compressive forces required in order for it to reach a maximum value are not attainable, since the expression is not bounded from above. We can substitute this relationship into the differential equation and then solve for the level as a new function of time:

$$\frac{\mathrm{d[bd[t]\,lev[t]]}}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{\mathrm{A}_{\mathrm{reactor}}}$$
$$\mathrm{bd[lev[t]] = k\,lev[t] + bdo}$$
$$\frac{\mathrm{d[(k\,lev[t] + bdo)\,lev[t]]}}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{\mathrm{A}_{\mathrm{reactor}}}$$
$$\frac{\mathrm{d[k\,lev\,[t]^2 + bdo\,lev[t]]}}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{\mathrm{A}_{\mathrm{reactor}}}$$
$$\mathrm{k}\frac{\mathrm{d\,lev[t]^2}}{dt} + \mathrm{bdo}\frac{\mathrm{d\,lev[t]}}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{\mathrm{A}_{\mathrm{reactor}}}$$

In fact, for the sake of solving this equation, *Mathematica* is perfectly capable of utilizing the equation in the form just after substitution of the linear equation for the bulk density. We see that this is the case in the computation that follows:

```
In[34]:= Remove["Global`*"]
         DSolve[{∂t((bdo + k lev[t])lev[t])== m̊in/Areactor,
          lev[0] == 0}, lev[t],t] // FullSimplify
```

$$\text{Out[35]}= \{\{\text{lev[t]} \to -\frac{\text{bdo} + \frac{\sqrt{\text{bdo}^2 \text{A}_{\text{reactor}} + 4\text{kt}\mathring{\text{m}}_{\text{in}}}}{\sqrt{\text{A}_{\text{reactor}}}}}{2\text{k}}\},$$

$$\{\text{lev[t]} \to -\frac{\text{bdo} + \frac{\sqrt{\text{bdo}^2 \text{A}_{\text{reactor}} + 4\text{kt}\mathring{\text{m}}_{\text{in}}}}{\sqrt{\text{A}_{\text{reactor}}}}}{2\text{k}}\}\}$$

Two solutions result from this equation because the level appears quadratically in the time derivative. The first solution will provide only a negative value of the level, so we must utilize the second solution. We can clean it up a bit algebraically:

$$-\frac{\text{bdo}}{2\text{k}} + \frac{\sqrt{\text{bdo}^2\text{A}_{\text{reactor}} + 4\,\text{kt}\mathring{\text{m}}_{\text{in}}}}{2\text{k}\sqrt{\text{A}_{\text{reactor}}}}$$

$$-\frac{\text{bdo}}{2\text{k}} + \frac{\sqrt{\text{bdo}^2\text{A}_{\text{reactor}} + 4\,\text{kt}\mathring{\text{m}}_{\text{in}}}}{\sqrt{4\text{k}^2\text{A}_{\text{reactor}}}}$$

$$-\frac{\text{bdo}}{2\text{k}} + \sqrt{\frac{\text{bdo}^2}{4\text{k}^2} + \frac{\text{t}\mathring{\text{m}}_{\text{in}}}{\text{kA}_{\text{reactor}}}}$$

This expression is one that can be tested against the experimental data. The rate of mass flow in, $\mathring{\mathbf{m}}_{\mathbf{in}}$, is constant as is the reactor cross-sectional area, $\mathbf{A}_{\mathbf{reactor}}$. Therefore, the only unknown is the value of **k**, the proportionality constant. We can try to fit this equation to the data:

```
In[36]:= levdata = {{0, "0."}, {100, "2.05"}, {200, "3.55"},
          {300, "4.79"}, {400, "5.87"}, {500, "6.85"},
          {600, "7.74"}, {700, "8.57"}, {800, "9.34"},
          {900, "10.1"}, {1000, "10.8"}, {1100, "11.4"},
          {1200, "12.1"}, {1300, "12.7"}, {1400, "13.3"},
          {1500, "13.8"}, {1600, "14.4"}, {1700, "14.9"},
          {1800, "15.4"}, {1900, "16."}, {2000, "16.4"},
          {2100, "16.9"}, {2200, "17.4"}, {2300, "17.9"},
          {2400, "18.3"}};
```

To do so we will load the package Statistics‘NonLinearFit’. Then we can fit this to the data by recognizing that there is only one parameter which we do not know and that is the value of **k**. The command **NonlinearFit** will do this for us. (One can learn all about this or any other command by inputting for example **??NonlinearFit.**) The syntax is straightforward: We provide the set of data by name, the expression or function to be fit to the data, the name of the independent variable, and the name of the parameter. There are many different control values we can set, including the method of minimization; in this case we have moved the number of iterations allowed from the default value of 30 upto 100.

```
In[37]:= << Statistics`NonlinearFit`

In[38]:= bdo = 10;
```

$$r_{reactor} = 10ft\,\frac{12\,in}{ft}\,\frac{2.54\,cm}{in}\,\frac{1\,m}{100\,cm}\,\frac{1}{m};$$

$$A_{reactor} = \pi r_{reactor}^2$$

$$\mathring{m}_{in} = 7.6;$$

$$\frac{\mathring{m}_{in}}{A_{reactor}}$$

$$\text{NonlinearFit[levdata,} -\frac{bdo}{2k}+\sqrt{\frac{bdo^2}{4k^2}+\frac{t\mathring{m}_{in}}{k\,A_{reactor}}}\text{, t, \{k\},}$$

```
    MaxIterations → 100]

Out[40]= 29.1864

Out[42]= 0.260396
```

$$\text{Out[43]=}\ -3.79651 + \sqrt{14.4135 + 0.197719t}$$

According to the fitting routine, the fitted function should be $-3.79651+\sqrt{14.4135+0.197719t}$. Clearly the value of **k** must be computed from the best fit parameters. We have magnitudes for $\frac{bdo}{2k}$ and for $\frac{bdo^2}{4k^2}$ and we can solve for **k** with each to be sure that the same value results.

$$\text{In[44]:= NSolve[}\frac{bdo}{2k} == 3.79\text{, k]}$$

$$\text{NSolve[}\frac{bdo^2}{4k^2} == 14.4\text{, k]}$$

```
Out[44]= {{k → 1.31926}}

Out[45]= {{k → 1.31762}, {k → -1.31762}}
```

Now we test this expression for its appearance of fit against the data set. We do so by creating the function lev [t] with it, computing the level as a function of time, and then plotting this with the actual data in order to visualize the fit.

```
In[46]:= bdo = 10;
```

$$r_{reactor} = 10\,ft\,\frac{12\,in}{ft}\,\frac{2.54\,cm}{in}\,\frac{1\,m}{100\,cm}\,\frac{1}{m};$$

$$A_{reactor} = \pi r_{reactor}^2$$

$$H_{reactor} = 60\,\text{ft}\;\frac{12\,\text{in}}{\text{ft}}\,\frac{2.54\,\text{cm}}{\text{in}}\,\frac{1\,\text{m}}{100\,\text{cm}}\,\frac{1}{\text{m}};$$

$$V_{reactor} = A_{reactor}\;60\,\text{ft}\;\frac{12\,\text{in}}{\text{ft}}\,\frac{2.54\,\text{cm}}{\text{in}}\,\frac{1\,\text{m}}{100\,\text{cm}}\,\frac{1}{\text{m}};$$

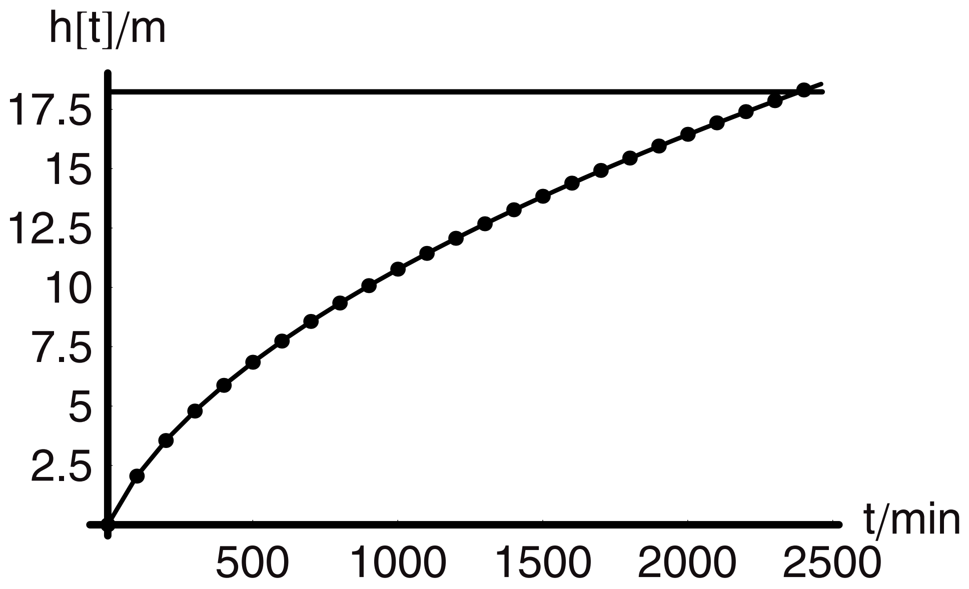
$\mathring{m}_{in}$ = 7.6;
k = 1.32;
tf = 820;

$$\text{lev}_{\text{cat,NL}}[\text{t\_}] := -\frac{\text{bdo}}{2\text{k}} + \sqrt{\frac{\text{bdo}^2}{4\text{k}^2} + \frac{\text{t}\mathring{m}_{in}}{\text{k}\,A_{reactor}}}$$

```
lev_cat,NL[t]
datpl = ListPlot[levdata, Epilog → {Thickness[0.006],
  Line[{{0, 18.2}, {3tf, 18.2}}]},
  PlotStyle → PointSize[0.02], DisplayFunction → Identity,
  AxesLabel → {"t/min", "h[t]/m"}];
plfitnl = Plot[lev_cat,NL[t], {t, 0, 3 tf},
  DisplayFunction → Identity];
Show[{datpl, plfitnl}, DisplayFunction → $DisplayFunction];
```

Out[48]= 29.1864

Out[55]= $-3.78788 + \sqrt{14.348 + 0.197269\ t}$



The model parameter has a value of 1.32 and the time that would be required to reach full capacity with this new model can be found by solving the following equation:

```
In[59]:= bdo = 10;
```
$$r_{reactor} = 10\,\text{ft}\,\frac{12\,\text{in}}{\text{ft}}\,\frac{2.54\,\text{cm}}{\text{in}}\,\frac{1\,\text{m}}{100\,\text{cm}}\,\frac{1}{\text{m}}$$
$$A_{reactor} = \pi r^2_{reactor}\text{//N}$$
$$H_{reactor} = 60\,\text{ft}\,\frac{12\,\text{in}}{\text{ft}}\,\frac{2.54\,\text{cm}}{\text{in}}\,\frac{1\,\text{m}}{100\,\text{cm}}\,\frac{1}{\text{m}}$$
$$V_{reactor} = A_{reactor}\,H_{reactor}$$
$$\mathring{m}_{in} = 7.6$$
$$k = 1.32$$

$$\text{In[59]:= t/. Flatten[NSolve[18.3 == } -\frac{bdo}{2k} + \sqrt{\frac{bdo^2}{4k^2} + \frac{t\mathring{m}_{in}}{kA_{reactor}}}\text{, t]]}$$

```
%/60
%/24
```

We find that the value predicted is 2400 min or 40 hr, which is much closer than our estimate of 78 hr based on the constant bed density.

Filling a Cylindrical Tank

Most often the mass flow that we are concerned with will involve a liquid. When a liquid is flowing we typically measure its flow rate in dimensions of volume per unit time. We consider next the flow of a liquid into a tank (another simple single-component problem), shown in Figure 2.

The control volume is the tank itself. This is because the liquid flowing into the tank is homogeneous, meaning that wherever we make a measurement of composition, density, or temperature it is everywhere the same in the liquid. The differential statement of the conservation of mass is the same as it was in the first case:

$$\frac{dm[t]}{dt} = \mathring{m}_{in}$$

Now, however, the expression for the mass flow rate into the tank is given by the product of the density of the liquid $\boldsymbol{\rho}$ and the volumetric flow rate **q**:

$$\frac{dm(t)}{dt} = \rho q$$

The mass accumulated within the control volume is the product of the density of the liquid, the cross-sectional area of the tank A_C, and the height of the liquid in the tank at any time **t**, **h[t]**. Replacing this in the time derivative and rearranging we find:

$$\frac{dh(t)}{dt} = \frac{q}{A_C}$$

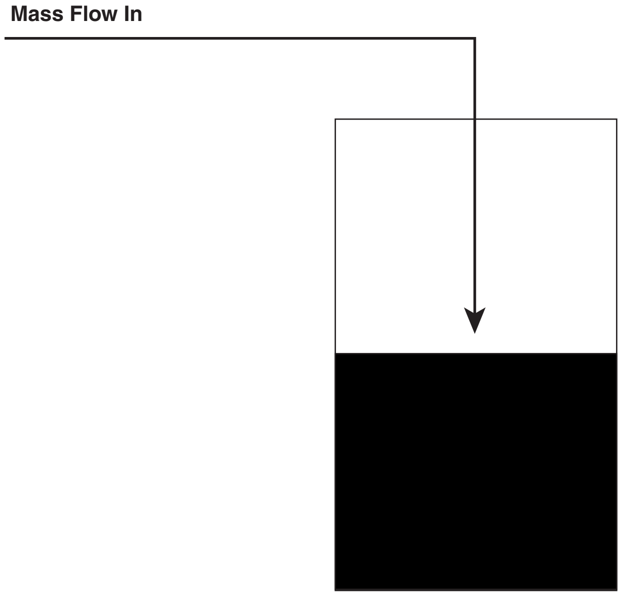


Figure 2

This equation states that the rate of change of liquid level in the tank is a constant. From this we know then that the change in level must be a linear function of time:

In[59]:= $\int_0^{h[t]} dh[t] == \int_0^t \frac{q}{A_C} dt$

Out[59]= $h[t] == \frac{qt}{A_C}$

Here too we can do the integration trivially because the flow rate into the tank is a constant. Notice also that the units are consistent in the final expression:

In[60]:= $\text{Length} == \frac{\text{Length}^3}{\text{time}} \frac{1}{\text{Length}^2} \text{time}$

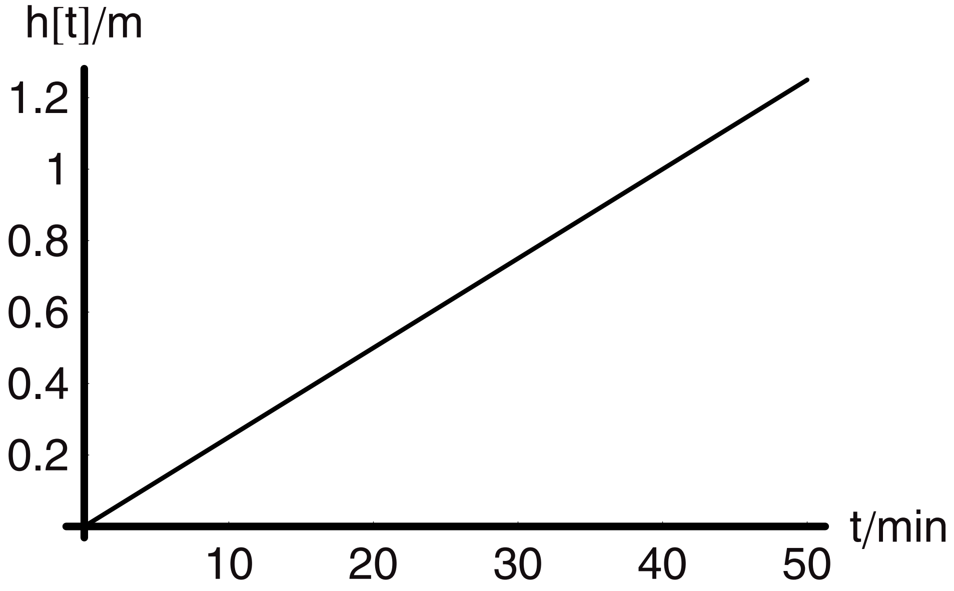
Out[60]= True

If we take the cross-sectional area of the tank to be 10 m^2 and the flow rate to be 0.25 m^3 min^{-1}, then a plot of the level of liquid versus time is as follows:

In[61]:= **<< Miscellaneous`Units`**

$\text{Plot}[(\frac{0.25\,\text{Meter}^3\,\text{Minute}^{-1}}{10\,\text{Meter}^2})\frac{\text{Minute}}{\text{Meter}}t, \{t, 0, 50\},$

AxesLabel → {"t/min", "h[t]/m"}];



If the aspect ratio of the tank is 4, that is, the ratio of the height to the diameter, then after how many minutes will it overflow under these conditions? We know the cross-sectional area so we can find the diameter of the tank because we know it is one-fourth of its height. Given this height we can solve for t_{critical}:

In[63]:= **Solve[10 == N[$\pi\frac{d^2}{4}$],d]**

Out[63]= {{d → -3.56825}, {d → 3.56825}}

In[64]:= **Solve[4*3.57 Meter == $\frac{.25\,\text{Meter}^3\,\text{Minute}^{-1}}{10\,\text{Meter}^2}$ t_{critical}, t_{critical}]**

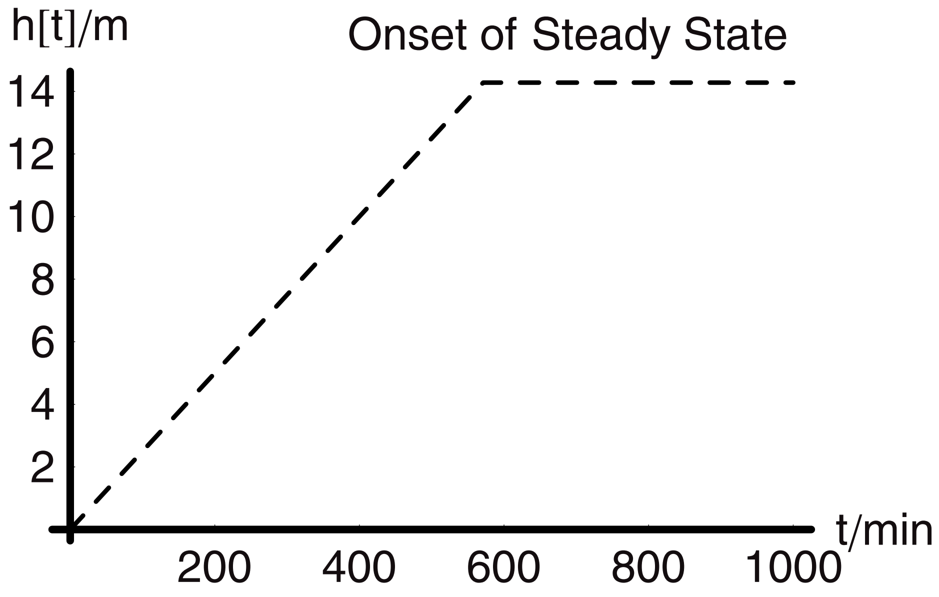
Out[64]= {{t_{critical} → 571.2 Minute}}

Thus the tank will begin to overflow after 571 min. How would we write the differential mass balance for that situation? We would do it just as we have before, except that now we would have the second term on the right-hand side:

$$\frac{\text{dm}[t]}{dt} = \mathring{\text{m}}_{\text{in}} - \mathring{\text{m}}_{\text{out}}$$

If we think about this, the answer is immediately obvious—the right-hand side is identically zero. This means that the net rate of change of level in the tank is also identically zero, meaning that it no longer can rise or fall, but stays at a steady-state value. Thus the overall behavior of the unattended tank under these conditions is simply this:

```
In[65]:= a = Plot[(0.25 Meter^3 Minute^-1 / 10 Meter^2) Minute/Meter t,
           {t, 0, 571.2}, AxesLabel → {"t/min", "h[t]/m"},
            PlotStyle → {Thickness[0.006],
             Dashing[{0.025, 0.025}]},
            DisplayFunction → Identity];

         b = Graphics[{Dashing[{0.025, 0.025}],
              {Thickness[0.006], Line[{{571.2, 14.28},
               {1000, 14.28}}]}}];
         Show[a, b, DisplayFunction → $DisplayFunction,
           PlotLabel → "        Onset of Steady State"];
```



We considered time-independent flow rates into the system, but what if we had to handle a situation in which the flow rates were time dependent? How would we handle the analysis of that situation?

Pressurizing an Initially Evacuated Tank with an Ideal Gas

Gas flows are common and offer another opportunity for us to apply this new tool we have found in the total mass balance. Gases can be simple or complex. By simple we mean that in some cases the atoms or molecules of which the gas is composed do not interact except at the point of collision. They behave as if they were nanoscopic ball bearings racing around, colliding with one another and the walls of the vessel in which they are contained. Furthermore, even as we increase their pressure within the vessel by increasing their number per unit volume at constant temperature or by raising temperature at constant number per unit volume, they continue to behave in the same way. We refer to gases of this kind as "ideal" and their characteristics are those of "hard-spheres." Most gases do not behave ideally. If their properties are nearly ideal at low pressure, we find that they deviate from ideality at higher pressure. The reason for this is that these molecules exhibit truly molecular behavior in all its rich detail and complexity. When they collide they do not do so as if they were merely billiard balls bouncing off one another. Instead they are sticky or they are repulsive. They have size and they have shape. Some are polar; others are nonpolar. It is these properties that give rise to much subtler and richer effects than are observed in "real" gases and which the hard sphere model could never predict. Nonetheless, the ideal gas is a good model and one from which we can learn a great deal. We can also use it to advantage here, because going into the theory of real gases is a subject in and of itself.

Recall that an ideal gas follows a very simple equation of state:

$$\mathrm{PV} = \mathrm{nRT}$$

where **P** is the pressure, **V** the volume of the vessel, *n* the number of moles of the gas, **R** the gas constant, and **T** the absolute temperature. With this we can calculate the pressure in a vessel of volume **V** as a function of pressure, the volume of a gas at fixed pressure and temperature, or the temperature at fixed pressure and volume by simple rearrangements:

$$\mathrm{P} = \frac{nRT}{V}$$

$$\mathrm{V} = \frac{nRT}{P}$$

$$\mathrm{T} = \frac{PV}{nR}$$

In addition, we can compute the concentration of the ideal gas in moles per unit volume or its density in mass per unit volume:

$$\mathrm{C} = \frac{n}{V} = \frac{P}{RT}$$

$$\rho = \frac{nMW}{V} = \frac{PMW}{RT}$$

where **MW** refers to the molecular weight of the gas. This provides the link we need between the gas phase material and the overall mass balance. Let's see how it can work.

Gases and liquids are both fluid phases, but they differ in density. For example, the density of water at ambient conditions is ≈1 g per cm^3. The vapor pressure of water in equilibrium with the liquid is 0.43 psia at 75°F. (Psia stands for pounds per square inch absolute—meaning above vacuum.) We can compute the concentration of water in the liquid phase and compare this to that of the equilibrium vapor phase, and then we can compute the density of the equilibrium vapor phase and compare that to the liquid density. This way we can have a better sense of the magnitudes of these quantities and by how much they differ. We might call this having a "physical feel" for the numbers.

```
In[68]:= << Miscellaneous`Units`

                                   3
                   Gram Mole (1000 Centimeter )
In[69]:= CncH2OLiq == N[----------------------------]
                             3
                   Centimeter  (18 Gram) Liter

         CncH2OEqVap ==

                                   0.43 psia Atmosphere
                                 N[--------------------]
                                        14.7 psia
         ---------------------------------------------------------------------
         .08205 (Liter Atmosphere) ConvertTemperature[73, Fahrenheit, Kelvin]
                                 Mole Kelvin

         DensH2OEqVap ==

                                   0.43 psia Atmosphere
                                 N[--------------------]
                                        14.7 psia
         ---------------------------------------------------------------------
         .08205(LiterAtmosphere) ConvertTemperature[73, Fahrenheit, Kelvin]
                                 Mole Kelvin

            18 Gram       1 Liter
         × -------  ----------------
                                 3
             Mole   1000 Centimeter

                             55.5556 Mole
Out[69]= CncH2OLiq == ------------
                                Liter

                               0.00120472 Mole
Out[70]= CncH2OEqVap == ---------------
                                   Liter

                                0.000021685 Gram
Out[71]= DensH2OEqVap == ----------------
                                             3
                                   Centimeter
```

The concentration of water in liquid water is on the order of 55 mol per L. The concentration of water in the vapor that is in equilibrium with the liquid is less than 1×10^{-3} mol per L. From this we see that at ambient conditions the gas phase is on the order of three to four orders of magnitude less concentrated than the liquid. The density of the gas makes this even clearer.

A common operation in a pilot plant or laboratory, as shown in Figure 3, is the pressurization of a batch reaction vessel with a gas such as hydrogen. A batch reactor is one that does not have flow into or out of it during reaction. It does have to be charged with reactants prior to operation. We can consider this process to be one that is amenable to the techniques we have at this point for analysis. We will assume that the gas remains ideal throughout the pressurization, not too bad an approximation for a gas like hydrogen. We will see that once we account for

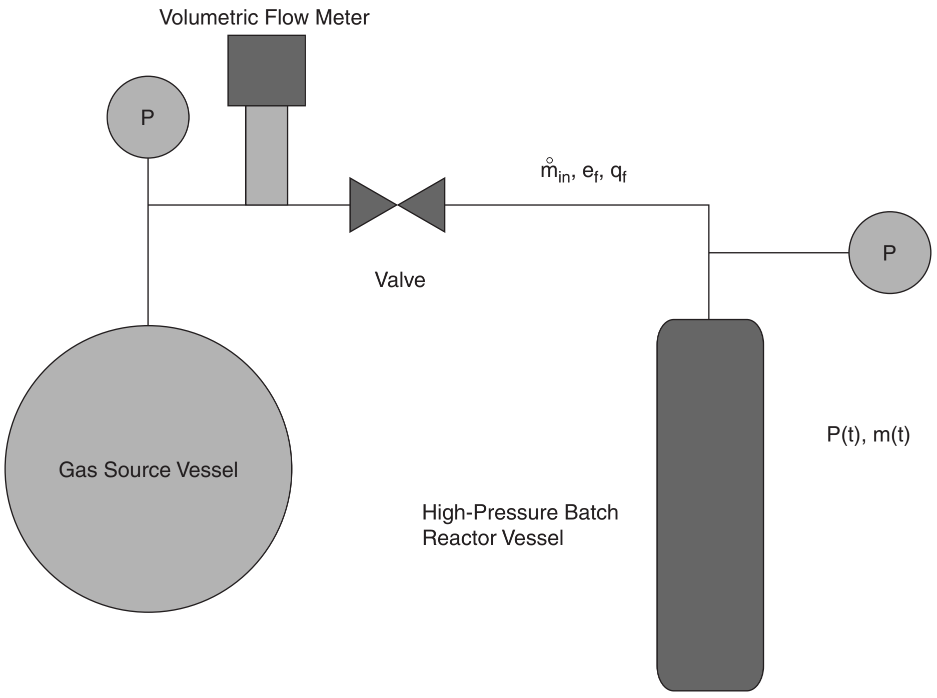

Volumetric Flow Meter
P
$\overset{\circ}{m}_{in}$, e_f, q_f
Valve
P
Gas Source Vessel
P(t), m(t)
High-Pressure Batch Reactor Vessel

Figure 3

the gaseous nature of this fluid, this problem looks like that of filling a tank with liquid phase fluid.

The source of the gas is a large-volume vessel at high pressure. We assume that the whole system remains isothermal—that is, at constant temperature throughout the procedure. This vessel must be at a pressure higher than or equal to the pressure we need to attain in the reactor. The pressure upstream of the valve is a constant and just after the valve location and into the reactor the pressure is time dependent. The reaction vessel is initially evacuated so that only the pure gas will be present in the gas phase. Opening the valve allows one to control the flow of gas into the vessel. By monitoring the pressure gauge the flow can be stopped when the proper pressure has been attained. The rise in pressure at the reactor gauge as a function of time is also a measure of the mass flow of gas into the reactor vessel. Alternatively, if we know the mass flow or the volumetric flow and the pressure upstream of the valve, then we can predict the time required to reach a set pressure in vessel give its volume and temperature. We can consider the latter situation first.

The overall material balance for the process of pressurizing the reactor vessel is:

$$\frac{dm[t]}{dt} = \overset{\circ}{m}_{in}$$

The mass accumulation of gas in the reactor is given by:

$$\frac{dm(t)}{dt} = \frac{d}{dt}\left[\frac{PMW}{RT}V\right] = \frac{VMW}{RT}\frac{dP}{dt}$$

The right-hand side is given as the product of the feed gas density and the volumetric flow rate:

$$\mathring{m}_{in} = \rho_f q_f = \frac{P_f MW}{RT} q_f$$

The overall equation becomes:

$$\frac{VMW}{RT}\frac{dP}{dt} = \frac{P_f MW}{RT} q_f$$

$$\frac{dP}{dt} = \frac{P_f}{V} q_f$$

The final equation tells us that the rate of pressure rise in the vessel will be a constant equal to the product of the ratio of the feed gas pressure to the vessel volume and the volumetric flowrate of the feed gas. (Assuming that the volume of the line leading from the valve to the vessel is negligible.) Therefore, we can see immediately that the pressure rise will be linear:

$$P[t] = \left(\frac{P_f q_f}{V}\right) t$$

The linear rise in pressure is nearly the same as the linear rise in liquid level in the filling tank. In the case of the liquid level rise in the tank we found that it would rise until it reached the ultimate level of the tank and then it would spill over. Yet, the equation we had derived did not demonstrate this. We had to analyze it in a second regime to find this out. In this case we see a similar feature of the solution: namely, it states that the pressure will rise to an infinite value with infinite time. This is just the same as the problem of the finite tank height. Here there is a finite pressure beyond which the vessel pressure may not rise. Thus the equation is only good up to that point and we might be right to suspect that as the vessel pressure rises to come close to the feed gas pressure the predictions we make using this equation may become inaccurate. A deeper level of analysis would be required to address this problem. The other way that this procedure may be done is to make measurements of volumetric flow rates for different valve settings. By measuring the pressure as a function of time in a ballast vessel we could calibrate the valve. For example, suppose we have the following set of data of pressure (psia) as a function of time for vessel that is 300 L, and an upstream pressure of 500 psia:

t/min	P/psia
0	5.12
10	84.4
20	166.
30	246.
40	341.
50	427.
60	501.

A plot of the data shows that within error it is linear and the slope can be evaluated to find that the flow rate **qf** was 5 L per min.

Time-Dependent Flows

To this point we have considered all the inlet flows to be constants. We should now consider what happens when they are functions of time. When we specify the flow rate as a function of time we have said nothing about the mechanism that gives rise to the observed functionality. It is simply a statement based on observation. There are cases in which the consideration of the mechanism can make it plain that flows will be time dependent. For example the pumping of our hearts is periodic and gives rise to periodic or pulsating flow of blood. As we open a valve or faucet the flow grows in relation to the rate at which we open it and the opposite happens when we are closing the valve. Therefore, there are ample numbers of examples in which the flow may be periodic, pulsating, or otherwise time dependent.

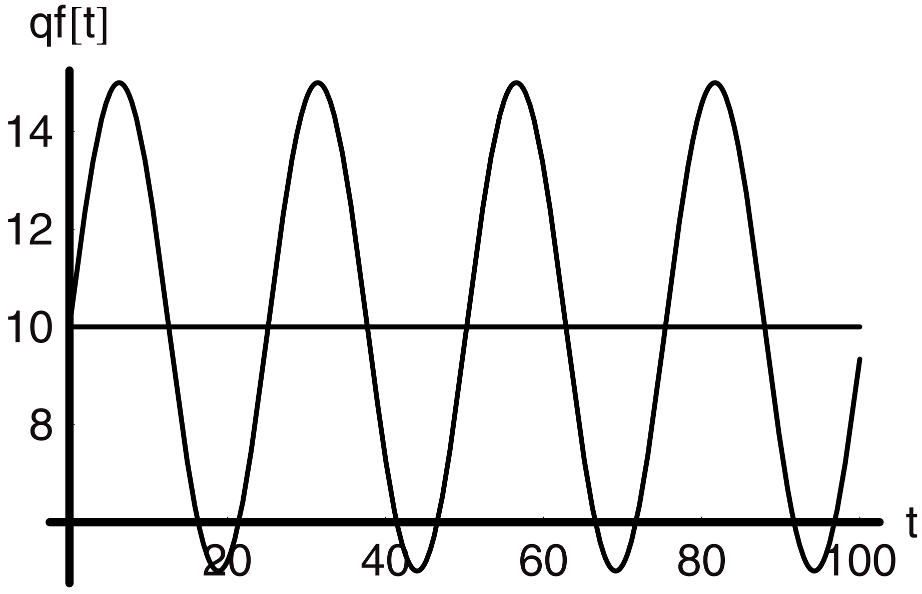
An interesting case to examine is the flow into a tank. We have already analyzed this for constant flow, but what would be different about, for example, a periodic flow? How would this affect the time dependence of the rate of mass accumulation in the tank? To begin we consider a continuous but periodic flow rate. This could be nicely described by a sinusoidal dependence upon time. The flow can always be taken to be positive, but with a superimposed periodicity. The flow rate could be described as:

$$\mathrm{qf}(t) = \mathrm{qfo}(1 + \alpha\,\mathrm{Sin}\,(\beta t))$$

To see what this would look like we can plot it for some specific values of **qfo**, α, and β:

```
In[72]:= Remove[qf, α, β, t]

         qf[t_]:= qfo(1 + α Sin[βt])
         qfo = 10;
         α = 0.5;
         β = 0.25;
```

```
Plot[{qf[t], qfo}, {t, 0, 100},
    PlotStyle → {Thickness[0.006],
    Thickness[0.006], Dashing[{0, 0}],
    Dashing[{0.025, 0.025}]},
  AxesLabel → {"t", "qf[t]"}];
```



The dimensions of the constants α and β are of interest. The constant β is an element of the argument of the sine, which is a transcendental function. As such its argument must be dimensionless, and therefore β is an inverse time constant. On the other hand, the product of α and **qfo** must have dimensions of volumetric flow rate and so α must be dimensionless. From basic physics we also know that α is the amplitude of the wave while $\frac{1}{\beta}$ is the peak-to-peak time or the period of the wave. Now if this is the input to our tank how will the level as a function of time behave?

The starting point is the overall material balance:

$$\frac{\mathrm{dm[t]}}{\mathrm{dt}} = \mathring{\mathrm{m}}_{\mathrm{in}}$$

The right-hand side is now a time-dependent function:

$$\frac{\mathrm{dh[t]}}{\mathrm{dt}} = \frac{\mathrm{q_f[t]}}{\mathrm{A_C}}$$
$$\frac{\mathrm{dh[t]}}{\mathrm{dt}} = \frac{\mathrm{q_{fo}}(1 + \alpha\ \mathrm{Sin}\,[\beta \mathrm{t}])}{\mathrm{A_C}}$$

We can either separate and integrate or use **DSolve** directly:

```
In[78]:= Remove[h, qfo, α, β, t]
```

$$\text{In[78]:= Simplify[DSolve[\{}\partial_t \text{h[t] ==} \frac{\text{qfo}}{\text{Ac}}\text{(1+}\alpha\text{Sin[}\beta\text{t]), h[0]==0\}, h[t], t]]}$$

$$\text{Out[79]= \{\{h[t]} \to \frac{\text{qfo(}\alpha + t\beta - \alpha\text{Cos[t}\beta\text{])}}{\text{Ac}\beta}\text{\}\}}$$

From this solution we can see that if the amplitude α is very small, then the result looks much like it did before:

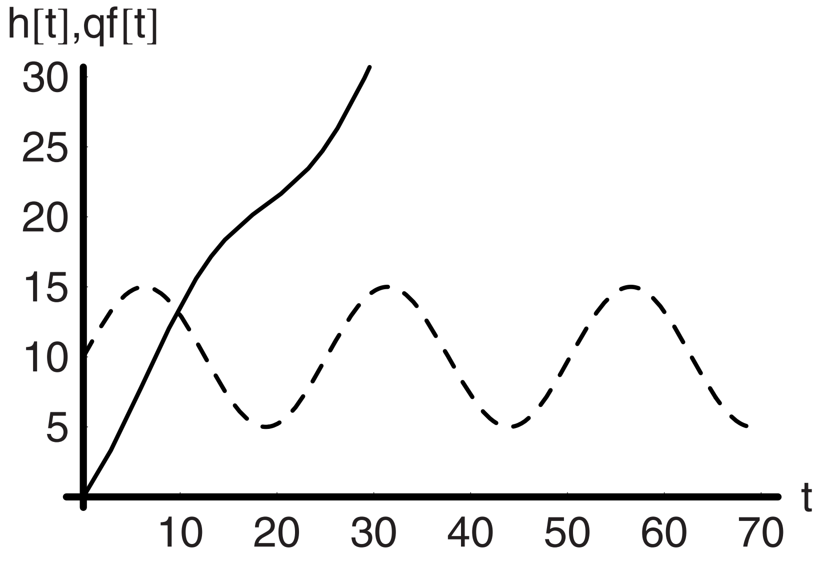
$$\text{In[80]:= Limit[}\frac{\text{qfo(}\alpha + t\beta - \alpha\text{Cos[t}\beta\text{])}}{\text{Ac}\beta}\text{, }\alpha \to 0\text{]}$$

$$\text{Out[80]= }\frac{\text{qfo t}}{\text{Ac}}$$

However, it is also clear that if the values of α and β are within certain ranges, then this will give rise to periodicity in the change in level of the tank. We can model this to see how this will look using the values that we had for α and β earlier.

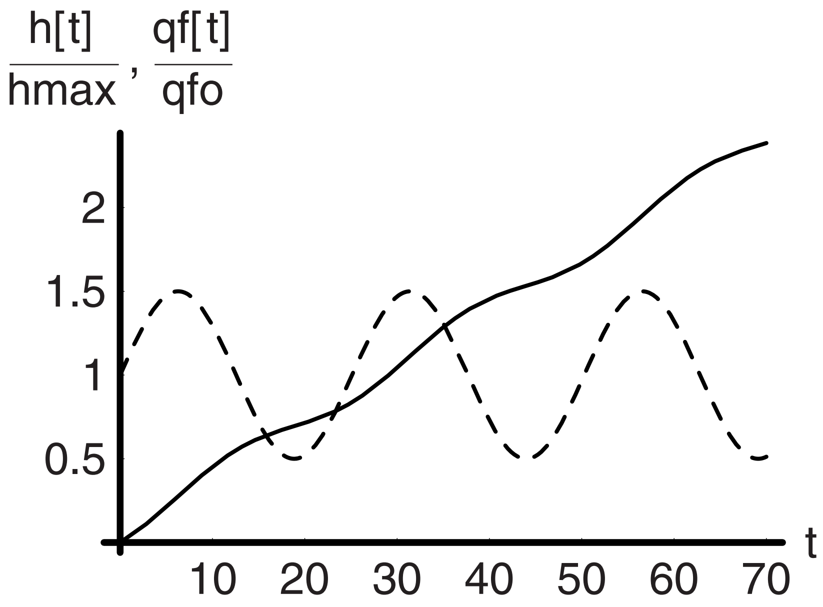
$$\text{In[81]:= h[t\_]:= }\frac{\text{qfo(}\alpha + t\beta - \alpha\text{Cos[t}\beta\text{])}}{\text{Ac}\beta}$$

```
         qf[t_]:= qfo(1 + α Sin[βt])

         qfo = 10;
         α = .5;
         β = 0.25;
         Ac = 10;

         Plot[{h[t], qf[t]}, {t, 0, 70},
           PlotStyle → {{Thickness[0.006], Dashing[{0, 0}]},
             {Thickness[0.006], Dashing[{0.025, 0.025}]}},
              AxesLabel → {"t", "h[t],qf[t]"}];
```



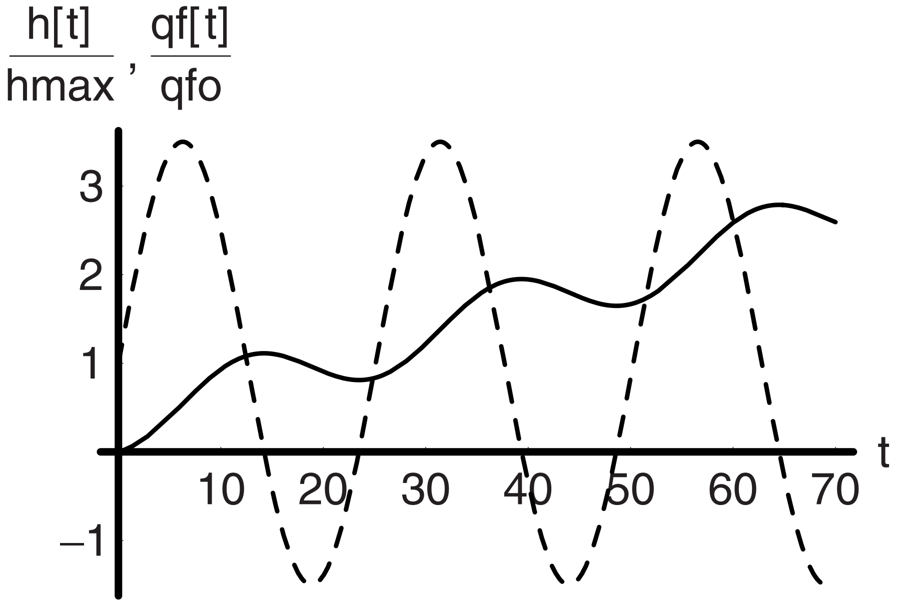
This shows us that the level will of course rise, but it will do so with varying rates depending upon the flow rate. This would actually be easier to see if we were to plot the dimensionless level and flow rates. We can obtain these by dividing **h[t]** by the maximum level in the tank and **q[t]** by **qfo**.

```
In[88]:= Plot[{h[t]/30, qf[t]/qfo}, {t, 0, 70},
           PlotStyle → {{Thickness[0.006], Dashing[{0, 0}]},
             {Thickness[0.006], Dashing[{0.025, 0.025}]}},
              AxesLabel → {"t", "h[t]/hmax, qf[t]/qfo"}];
```



What would happen if we were to bring the amplitude up, say, by a factor of five?

```
In[89]:= α = 2.5;

          Plot[{h[t]/30, qf[t]/qfo}, {t, 0, 70},
            PlotStyle → {{Thickness[0.006], Dashing[{0, 0}]},
              {Thickness[0.006], Dashing[{0.025, 0.025}]}},
               AxesLabel → {"t", "h[t]/hmax, qf[t]/qfo"}];
```



If we look closely, we see that in the time range between 15 and 20 the level is actually decreasing! But how can this happen when we have only flow into the tank according to our initial total material balance? Once again we need to be very careful with the model results that we derive. In this case, when we increased the amplitude by a factor of five we went out of the region in which the solution gave physically meaningful results. If you look carefully, in the same region where the level is decreasing, the flow rate is actually below zero (negative) and in the reverse direction of the feed, namely, out of the tank. The change in sign of the input function has given rise to this negative rate of accumulation, that is, negative slope, in this time domain. This is not the situation that we had in mind when we began the problem. It could correspond to some actual situation, but it does not correspond to the situation we are analyzing. Therefore, one needs to be very mindful of the range of application of any model and should check the resulting behavior for its correspondence to the real system.

2.2 Geometry and the Left-Hand Side of the Mass Balance Equation

The Triangular Trough

To this point all of the situations we have dealt with have involved quite simple geometry—the right cylinder. We may ask the question, How do we apply the new tool we have to other geometrical shapes? The issues that we will encounter in these kinds of analyses are handled within the differential accumulation term through **V[t]**. For this reason we can think of these as "Left-Hand Side" problems. The objective of this section is to demonstrate how to do that. We begin with an analysis of the tank that is shaped like a triangular trough as shown in Figure 4.

The flow is into the tank at a constant rate given by the density of the fluid and its volumetric flow rate. The mass in the tank at any time is the product of the density and the fluid volume. Notice that as the level of the fluid increases, so too does its width. Viewed from the top, the area of the liquid surface grows as a function of time. This is the main difference between this "tank" and that of a right cylinder standing on end. In that case the surface of the liquid viewed from above remains constant, so that the volume is only a function of the level. To summarize what we have so far:

$$\frac{dV[t]}{dt} = q$$

The differential change in volume with time **dV[t]** can be viewed as taking place by making a differential change in level **dh[t]**. Since the change in level is differential, the area of the fluid

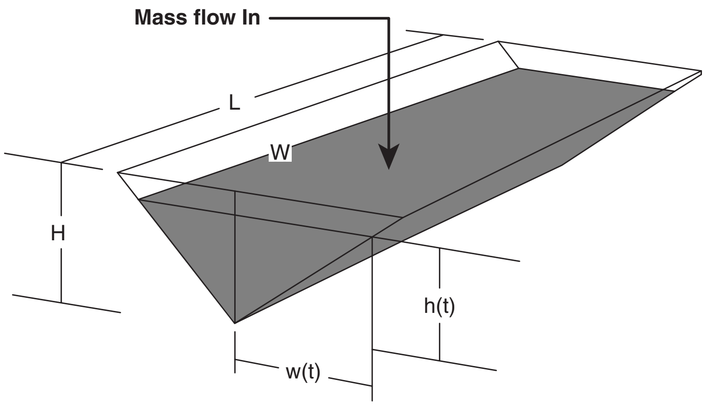


Figure 4

at the surface is virtually unchanged. This gives us:

$$dV[t] = A[t]\, dh[t]$$

However, the change in area with time is just the change in the liquid's width with time multiplied by the length of the trough, which gives us:

$$\frac{dV[t]}{dt} = 2Lw[t]\frac{dh[t]}{dt} = q$$

Now we need a relationship between **w[t]** and **h[t]**. To find this we can consider the geometry of the triangular face of the tank. From the law of similar triangles we find this:

$$\frac{W}{H} = \frac{w[t]}{h[t]}$$

The differential equation can now be rewritten in terms of only **h[t]**:

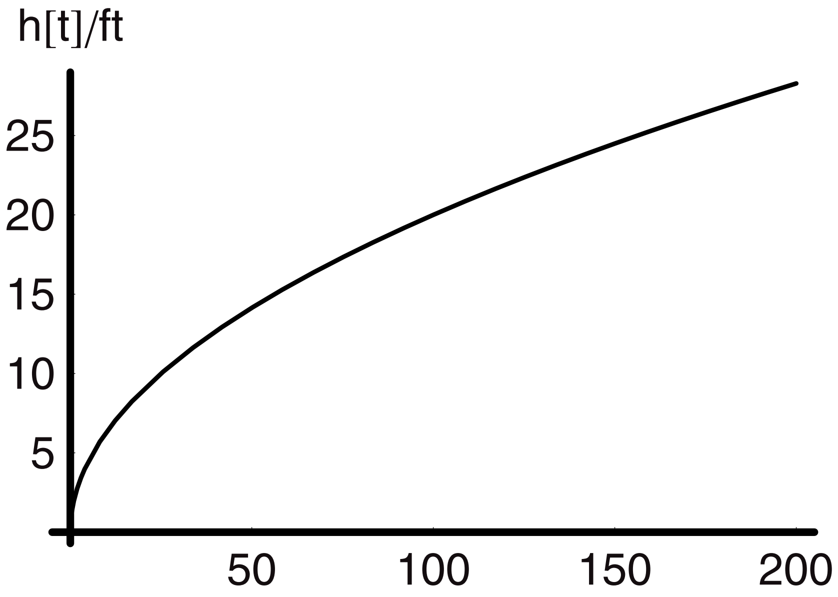
$$dV[t] = \frac{2WL}{H}h[t]\, dh[t]$$
$$h[t]\frac{dh[t]}{dt} = \frac{Hq}{2WL}$$

The equation is now readily soluble and we find that **h[t]** goes as the square root of time:

$$h[t] = \sqrt{\frac{Hq}{WL}t}$$

The level as function of time for a tank 10 ft high, 10 ft wide, and 40 ft long looks as follows, if the flow rate in is 5 $ft^3\ min^{-1}$:

```
In[91]:= Clear[h, q, H, W, L]

         htritro[t_] := Sqrt[(Hq/WL) t]

         W = 5;
         L = 10;
         q = 5;
         H = 40;

         pltritro = Plot[htritro[t], {t, 0, 200},
            AxesLabel → {"t", "h[t]/ft"}];
```



The Conical Tank

Another geometry that can be useful to consider is that of the conical tank. The analysis is similar to that of the triangular trough, except that the cone is axially symmetric. This makes some difference in the outcome. The geometry for the tank is shown here in Figure 5.

As in the case of the triangular trough, the area of the liquid surface changes with the level in the tank. We know that the mass balance will lead to the same equation for the differential

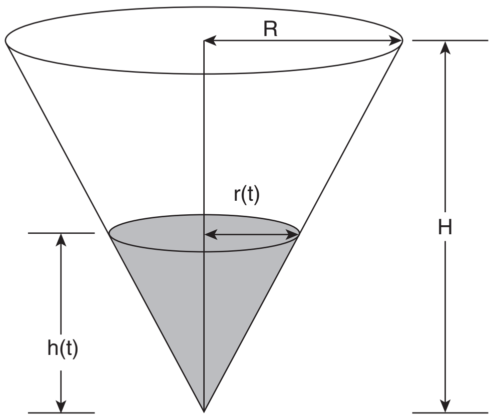


Figure 5

change in volume with time. Therefore, we must find the relationship that will render the differential volume change as a function of the level alone.

If we were to take a slice through the tank along the central axis, we would be left with a triangular face. From the similar triangles on that face we find:

$$\frac{R}{H} = \frac{\mathrm{r[t]}}{\mathrm{h[t]}}$$

If the volume were to change differentially by some differential level change we would have:

$$\begin{aligned}\mathrm{dV[t]} &= \mathrm{Adh[t]}\\ &= \pi\,\mathrm{r[t]}^2\,\mathrm{dh[t]}\\ &= \frac{\pi R^2}{H^2}\mathrm{h[t]}^2\,\mathrm{dh[t]}\end{aligned}$$

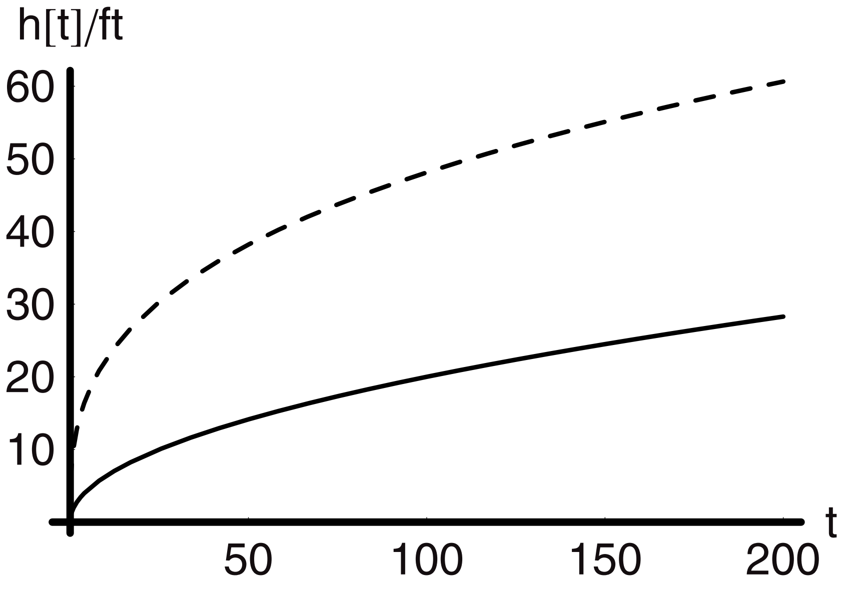
The solution to the level change as a function of time is the solution to this differential equation:

$$\begin{aligned}\frac{\pi R^2}{H^2}\mathrm{h[t]}^2\,\mathrm{dh[t]} &= \mathrm{q}\\ \mathrm{h[t]}^3 &= \frac{3qH^2}{\pi R^2}\mathrm{t}\\ \mathrm{h[t]} &= \sqrt[3]{\frac{3qH^2}{\pi R^2}t}\end{aligned}$$

For a tank of the same dimensions as the previous one and with the same flow rate the level as a function of time is shown here along with a comparison:

```
In[98]:=  Clear[W, H]

          N[Solve[2000 == 2 W^2 H, H] /. W → 5]

Out[99]=  {{H → 40.}}

In[100]:= Clear[R, H]

          N[Solve[2000 == πR^2 H/3, H] /. R → 5]

Out[101]=  {{H → 76.3944}}

In[102]:=  Clear[h, q, H, R, L]

          hcone[t_] := ∛(3qH^2/(πR^2) t)

          R = 5;
          q = 5;
          H = 76.4;
```

```
plcone = Plot[hcone[t], {t, 0, 200},
   AxesLabel → {"t", "h[t]/ft"}, PlotStyle →
    {Thickness[.006], Dashing[{0.025, 0.025}]},
     DisplayFunction → Identity];

Show[pltritro, plcone, DisplayFunction →
 $DisplayFunction];
```



The Semicylindrical Trough

A variation of the left-hand side theme and the issues of geometry is that of the semicylindrical trough lying on its side. The physical situation is quite similar to that of the triangular trough, except that the walls follow a circular curve. The physical system is shown in Figure 6.

In view of the last two analyses that we have done, geometry is the only question posed by this problem. We need the relationship between **r[t]** and **h[t]** once again. One line in the diagram holds the key to this. That line is the hypotenuse of the triangle which has **r[t]** for one leg and **R − h[t]** for the other. The Pythagorean theorem links the three:

$$R^2 = (R - h[t])^2 + r[t]^2$$

Solving for **r[t]** in terms of **h[t]**:

```
In[109]:= R =.
          Solve[R^2 == Expand[(R - h[t])^2] + r[t]^2, r[t]]
```

```
Out[110]= {{r[t] → -√(2Rh[t] - h[t]^2)}, {r[t] → √(2Rh[t] - h[t]^2)}}
```

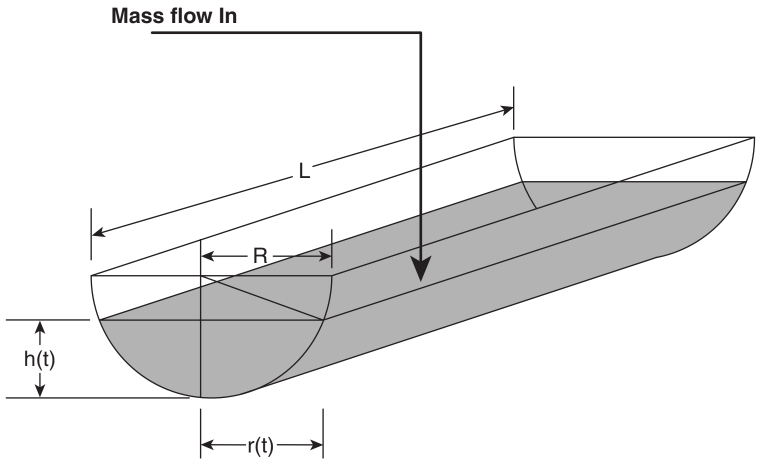


Figure 6

The first solution is unphysical, so we substitute the second into the expression for **dV[t]**:

$$\begin{aligned} \mathrm{dV[t]} &= \mathrm{Adh[t]} \\ &= \mathrm{2Lr[t]\,dh[t]} \\ &= \mathrm{2L}\sqrt{\mathrm{2Rh[t]} - \mathrm{h[t]}^2}\,\mathrm{dh[t]} \end{aligned}$$

Replacing this in the material balance (assuming all densities are constant and equal everywhere), we find:

$$\sqrt{\mathrm{2Rh[t]} - \mathrm{h[t]}^2}\,\frac{\mathrm{dh[t]}}{\mathrm{dt}} = \frac{\mathrm{q}}{2L}$$

The solution of this equation is:

$$In[111] := \mathbf{Simplify}\left[\int_0^{\mathbf{h[t]}} \sqrt{\mathbf{2\,x\,R - x^2}}\, d\mathbf{x} == \int_0^{\mathbf{t}} \frac{\mathbf{q}}{\mathbf{2L}}\, d\mathbf{s}\right]$$

$$Out[111] = \frac{1}{2}\sqrt{(2R-h[t])h[t]}\left(-R + \frac{2R^2\,\mathrm{ArcTan}\left[\frac{\sqrt{h[t]}}{\sqrt{2R-h[t]}}\right]}{\sqrt{2R-h[t]}\,\sqrt{h[t]}} + h[t]\right) == \frac{5t}{2L}$$

The solution of this equation involves an integral on the left-hand side that results in an implicit solution for **h[t]**. We can try to solve directly for **h[t]**, but *Mathematica* cannot do it:

In[112]:= $\mathbf{Solve[\frac{1}{2}\sqrt{2R - h[t])h[t]}(-R + \frac{2R^2\,ArcTan[\frac{\sqrt{h[t]}}{\sqrt{2R - h[t]}}]}{\sqrt{2R - h[t]}\ \sqrt{h[t]}} + h[t]) == \frac{qt}{2L},\ h[t]]}$

```
Solve::tdep:
 The equations appear to involve the variables to be
  solved for in an essentially non-algebraic way.
```

Out[112]= $\mathrm{Solve}[\frac{1}{2}\sqrt{(2R - h[t])h[t]}(-R + \frac{2R^2\mathrm{ArcTan}[\frac{\sqrt{h[t]}}{\sqrt{2R - h[t]}}]}{\sqrt{2R - h[t]}\ \sqrt{h[t]}} + h[t]) == \frac{5t}{2L}, h[t]]$

Therefore, the best bet for us it to use the analytical solution to solve for **h[t]** numerically or graphically. The graphical method is not used much today but it is worth illustrating because it reinforces a good "feel" for the functions and the numbers that result. We will look at the graphical solution first. As you no doubt recall, the method used finds the solutions to this equation that make the left- and right-hand sides equal. We can view each side as a statement for two different functions that intersect at certain points; these intersections are the solutions. If we graph the two functions we can find these points. This is very easy to do in *Mathematica* as follows.

Let the radius of the tank be 3 m and the length 20 m. We should find the total volume of the tank first so that we can choose a numerical value for the flow rate that does not require too long a time to make a real change in the tank level. To find the maximum volume we want to integrate the following equation from 0 to $V_{\mathbf{max}}$ on the left- and from 0 to $h_{\mathbf{max}}$ on the right-hand side:

$$dV[t] = 2L\sqrt{2Rh[t] - h[t]^2}\,dh[t]$$

In[113]:= **Clear[V, R, h, t]**
Vmax =.
$\mathbf{Simplify[\int_0^{Vmax} dV[t] == Simplify[\int_0^R 2L\sqrt{2Rh[t] - h[t]^2}\,dh[t]]]}$

Out[115]= $\mathrm{Vmax} == \frac{1}{2}L\pi R\sqrt{R^2}$

In[116]:= $\mathbf{Vmax[R\_,\ L\_] := N[\frac{1}{2}L\pi R\sqrt{R^2}]}$
Vmax[5, 20]

Out[117]= 785.398

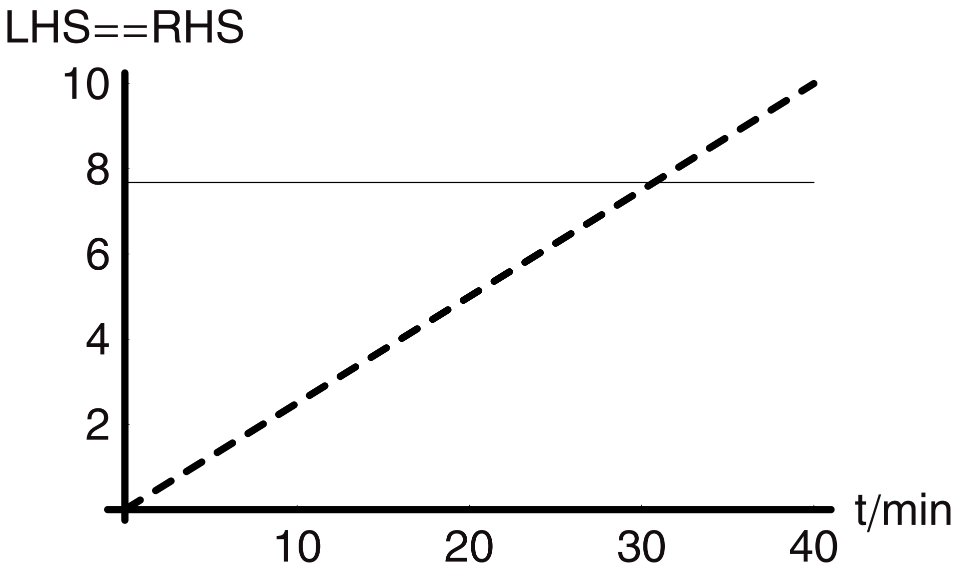
At 785 m^3 we can choose a flow rate of 10 m^3 min^{-1} as a reasonable value. The time to overflow to maximum capacity would be ~78 min. To solve the problem graphically we write the left- and right-hand sides as two separate functions. Then we choose a value of **h** and graph this as a horizontal line versus the right-hand side as a function of time. The point of intersection provides the time at which that level would be reached subject to the chosen parameter values. We would choose another value of **h** and find a new value of **t** with a new graph. Before we begin it is worth noticing that the magnitude of **h[t]** can never exceed **R** because to do so would be unphysical. Unphysical or not, it is very easy to begin plugging in values for **h** on the left-hand side that mistakenly range over the chosen value for **R**. If we were to do this, the mathematics will inform us of our error by providing an imaginary (complex) solution.

We begin by doing the graphical solution for the halfway point at **h** = 2.5. We seek to find the time at which this will happen. The *Mathematica* method for doing one point is shown here:

In[118]:= $\mathtt{lhs[h\_] := N[}\frac{1}{2}\sqrt{\mathtt{(2R-h)h}}\mathtt{(-R+}\frac{\mathtt{2R^2ArcTan[}\frac{\sqrt{\mathtt{h}}}{\sqrt{\mathtt{2R-h}}}\mathtt{]}}{\sqrt{\mathtt{2R-h}}\sqrt{\mathtt{h}}}\mathtt{+h)]}$

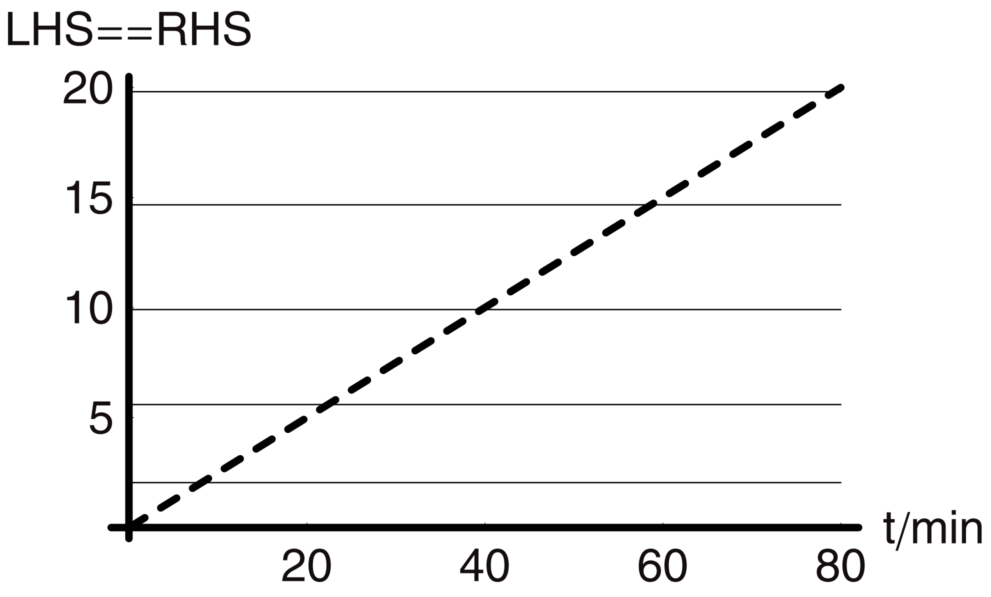
$\mathtt{rhs[t\_] := }\frac{\mathtt{q}}{\mathtt{2L}}\mathtt{t}$

```
L = 20;
R = 5;
q = 10;

Graphics[{Line[{{0, lhs[2.5]}, {40, lhs[2.5]}}]}];

Plot[{rhs[z]}, {z, 0, 40}, PlotStyle → {{Thickness[.01],
 Dashing[{0.025, 0.025}]}}, DisplayFunction → Identity];

Show[%, %%, DisplayFunction → $DisplayFunction,
 AxesLabel → {"t/min", "LHS==RHS"}];
```



From this we can see that the halfway point in terms of level will occur at just over 30 min. We could replot this in the vicinity of 30 min to obtain this more accurately, but as we shall see there are better ways to do it. To find a set of solutions beginning at a level of 1 m in increments of 1 m, we could use the following code to create the graphic we need:

```
In[126]:= Graphics[Table[{Line[{{0, lhs[h]}, {80, lhs[h]}}]},
            {h, 1, 5, 1}]];

          Plot[{rhs[z]}, {z, 0, 80}, PlotStyle → {{Thickness[.01],
            Dashing[{0.025, 0.025}]}}, DisplayFunction → Identity];

          Show[%, %%, DisplayFunction → $DisplayFunction,
            AxesLabel → {"t/min", "LHS==RHS"}];
```



As interesting as this approach is, it is not all that useful in comparison to what we can obtain by solving the problem numerically. To solve numerically, however, we really do the very same calculation: We choose a value of the level, evaluate the left-hand side, and then back solve the resultant equation for the time. An example of this procedure is given for the halfway point at **h** = 2.5 m:

```
In[129]:= h = 2.5
          L = 20;
          R = 5;
          q = 10;
```

NSolve[

$$\frac{1}{2}\sqrt{(2R-h)h}\left(-R+\frac{2R^2\mathrm{ArcTan}[\frac{\sqrt{h}}{\sqrt{2R-h}}]}{\sqrt{2R-h}\sqrt{h}}+h\right)==\frac{qt}{2L},t]$$

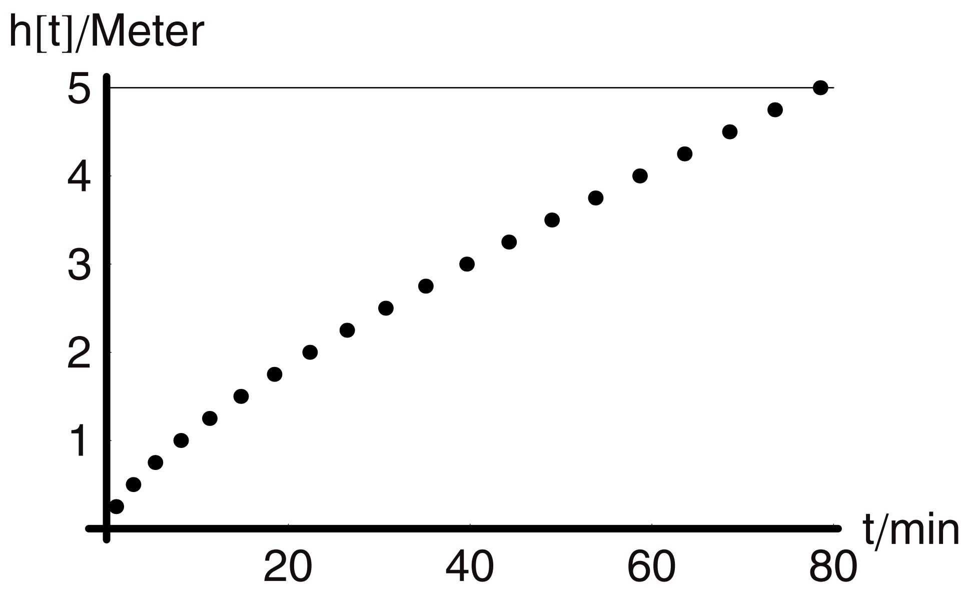
```
Out[129]= 2.5

Out[133]= {{t → 30.70924246521891}}
```

We solve this to find that the exact time is 30.7 min. (We could have *Mathematica* limit the figures by enclosing the command in **NumberForm**.) Of course what we really want to see is a plot of the level versus time for this tank. To obtain this we need to repeat this procedure for many different levels and then plot the resultant time and level pairs.

```
In[134]:= L = 20;
          R = 5;
          q = 10;
          ntimes = Table[NSolve[
```

$$\frac{1}{2}\sqrt{(2R-h)h}\left(-R+\frac{2R^2\mathrm{ArcTan}[\frac{\sqrt{h}}{\sqrt{2R-h}}]}{\sqrt{2R-h}\sqrt{h}}+h\right)==\frac{qt}{2L},t],$$

```
             {h, .25, 5, .25}];
          levels = Table[h, {h, .25, 5, .25}];
          timeleveldat =
            Transpose[Partition[Join[Flatten[t /. ntimes], levels],
             Length[levels]]];
          pllevdat = ListPlot[timeleveldat,
            AxesLabel → {"t/min", "h[t]/Meter"},
              DisplayFunction → Identity];
          Show[pllevdat, Epilog → {Line[{{0, 5}, {80, 5}}]},
            DisplayFunction → $DisplayFunction];
```

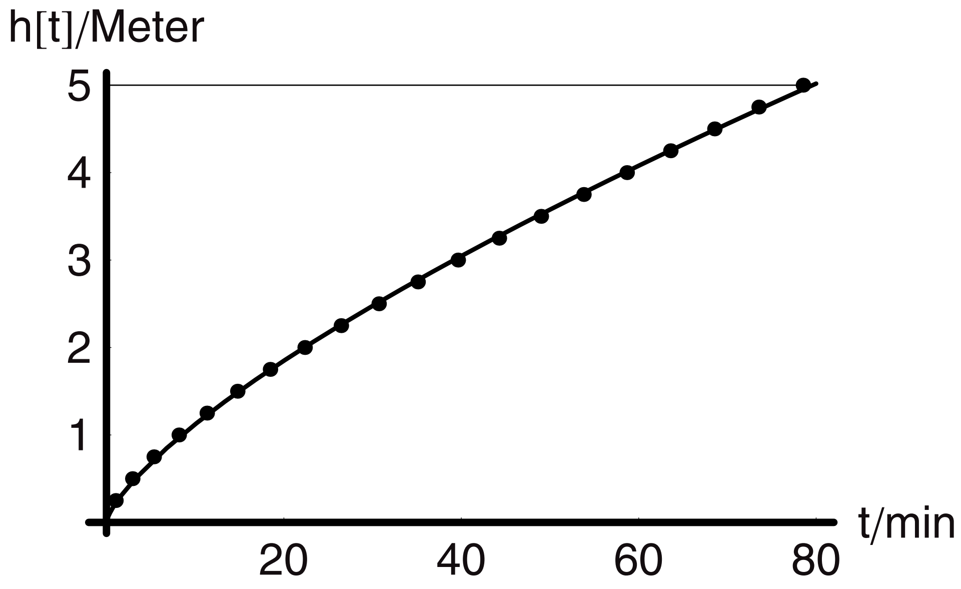


It was troublesome to create these points. We might want to provide an operator with a graph of the level versus time that could be used at any time to find the level or vice versa. The logical thing to do at this point is to fit these points to a function of time. We can easily do so using two parameters in a simple power law **h[t]** $= \mathbf{a}t^n$:

```
In[142]:= timeleveldat

Out[142]= {{1.046151218379871, 0.25}, {2.93629534388009, 0.5},
           {5.35228742419150, 0.75}, {8.17505543966421, 1.},
           {11.33279384944024, 1.25}, {14.77494200930721, 1.5},
           {18.46273275392473, 1.75}, {22.36476090008060, 2.},
           {26.45457644963383, 2.25}, {30.70924246521891, 2.5},
           {35.10840690045179, 2.75}, {39.63367125654707, 3.},
           {44.2681414116735, 3.25}, {48.9960956177208, 3.5},
           {53.8027306257427, 3.75}, {58.6739613290956, 4.},
           {63.5962577259286, 4.25}, {68.5565080961337, 4.5},
           {73.5419004550265, 4.75}, {78.5398163397448, 5.}}

In[143]:= Remove[NonlinearFit, a, n, t]

In[144]:= <<Statistics`NonlinearFit`

In[145]:= NonlinearFit[timeleveldat, at^n, t, {a, n}]
```

```
Plot[%, {t, 0, 80}, PlotStyle → Thickness[0.006],
  DisplayFunction → Identity,
   AxesLabel → {"t/min", "h[t]/Meter"}];

Show[%, pllevdat, Epilog → {Line[{{0, 5}, {80, 5}}]},
  DisplayFunction → $DisplayFunction];
```

Out[145]= $0.21373t^{0.720228}$



With the value of *a* at 0.2137 and that of *n* at 0.72 we find a very nice fit to the data. It is interesting to note that this statistically fitted function fits so well and yet offers us no insight into what is really taking place. How often do we see experimental data fitted statistically in this way and then physical mechanisms posed to explain the form of the fitted function? Do you think that anything fundamental ever comes from such an approach in the absence of an analysis? Beware!

The Spherical Tank

Figure 7 shows a simple case of filling a spherical tank. The liquid flow into the tank is again a constant and we wish to be able to predict the level as a function of time.

The total material balance statement is:

$$\frac{dV[t]}{dt} = q$$

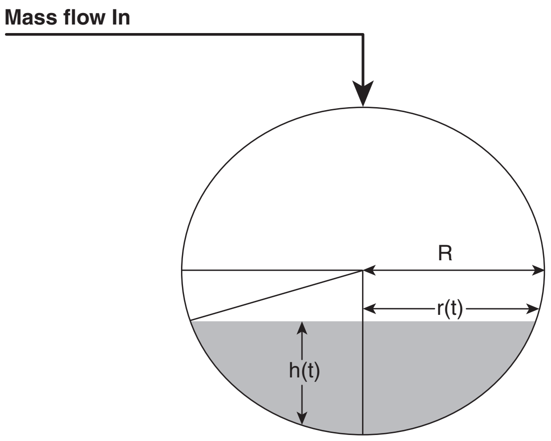


Figure 7

The differential change in volume can be thought of in terms of an area times a differential level change:

$$\begin{aligned} dV[t] &= A\, dh[t] \\ &= \pi r[t]^2\, dh[t] \\ \frac{r[t]^2 dh[t]}{dt} &= \frac{q}{\pi} \end{aligned}$$

The triangle in the left lower quadrant of the circular face of the sphere has a hypotenuse of **R**, and two legs, one of which is **r[t]** and the other is **R – h[t]**. Just as in the case of the semicylindrical trough, the Pythagorean theorem can be applied to this right triangle to give:

$$R^2 = r[t]^2 + (R - h[t])^2$$

```
In[148]:= Remove[R, r, h]

          Solve[R^2 == r[t]^2 + Expand[(R - h[t])^2], r[t]]
```

$$\text{Out[159]= } \{\{r[t] \to -\sqrt{2Rh[t] - h[t]^2}\}, \{r[t] \to \sqrt{2Rh[t] - h[t]^2}\}\}$$

Returning to the differential equation we substitute and need to find a solution for **h[t]**:

$$2Rh[t] - h[t]^2 \frac{dh[t]}{dt} = \frac{q}{\pi}$$

This is similar to the equation that we encountered in the last problem for the semicylindrical trough:

$$\sqrt{2\mathrm{Rh[t]} - \mathrm{h[t]}^2}\,\frac{\mathrm{dh[t]}}{\mathrm{dt}} = \frac{q}{2L}$$

The solution method would be the same. The main difference is that the tank is now axially symmetric (as was the conical tank) and this gives rise to a π on the right-hand side rather than **2L**. Recall that the sphere is made by rotating the semicircular face around the central axis by π radians. The semicylindrical tank is constructed by translating the same semicircular face by a distance **L** along a straight line. Symmetries of this kind are very interesting and will be useful to take advantage of in the solution of more sophisticated problems, but at this point we merely wish to point it out in passing.

We can proceed with the solution as follows. First, we show that by integration over **h** of the πr^2**dh** gives us the proper expression for the volume:

```
In[150]:= R =.
          q =.
```

$$\int_0^V dV == \pi\int_0^R (2Rh - h^2)\,dh$$

$$\mathit{Out[152]}= V == \frac{2\pi R^3}{3}$$

Next we integrate the left-hand side over h and the right-hand side over time. The left-hand side has two integrals over level:

```
In[153]:= R =.
          q =.
          h =.
```

$$\int_0^h 2Rh\,dh - \int_0^h h^2\,dh == \int_0^t \frac{q}{\pi}\,dt$$

$$\mathit{Out[156]}= -\frac{h^3}{3} + h^2R == \frac{qt}{\pi}$$

This last expression can now be solved for **h** as a function of time:

```
In[157]:= R =.

          Q =.
```

$$\texttt{h3o = Solve}\left[-\frac{h^3}{3} + h^2R - \frac{qt}{\pi} == 0,\ h\right]$$

Out[159]=

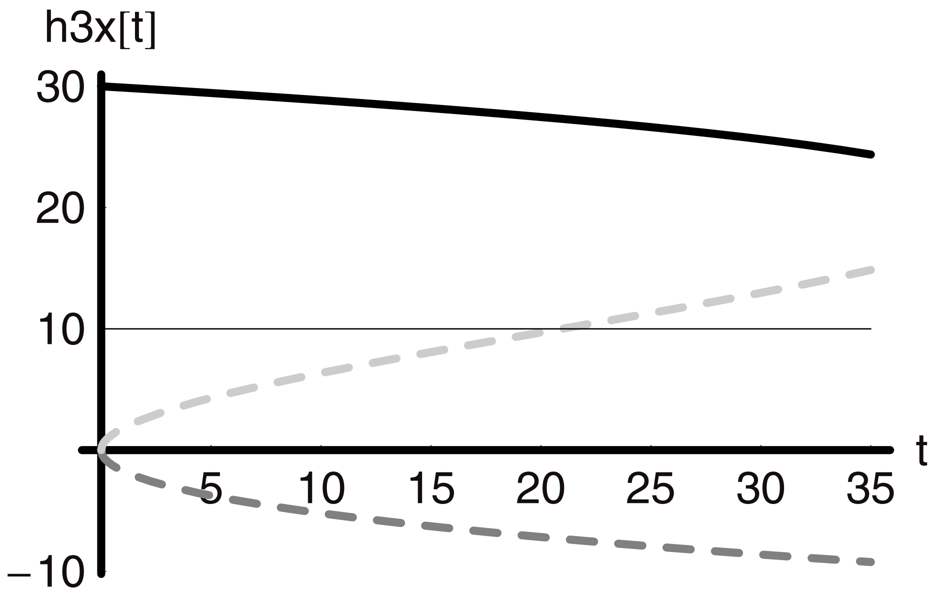
$$\Big\{\Big\{h \to R + \frac{2^{1/3}\pi R^2}{(2\pi^3R^3 - 3\pi^2qt + \sqrt{3}\sqrt{-4\pi^5qR^3t + 3\pi^4q^2t^2})^{1/3}} + \frac{(2\pi^3R^3 - 3\pi^2qt + \sqrt{3}\sqrt{-4\pi^5qR^3t + 3\pi^4q^2t^2})^{1/3}}{2^{1/3}\pi}\Big\},$$

$$\Big\{h \to R - \frac{(1 + i\sqrt{3})\pi R^2}{2^{2/3}(2\pi^3R^3 - 3\pi^2qt + \sqrt{3}\sqrt{-4\pi^5qR^3t + 3\pi^4q^2t^2})^{1/3}} - \frac{(1 - i\sqrt{3})(2\pi^3R^3 - 3\pi^2qt + \sqrt{3}\sqrt{-4\pi^5qR^3t + 3\pi^4q^2t^2})^{1/3}}{22^{1/3}\pi}\Big\},$$

$$\Big\{h \to R - \frac{(1 + i\sqrt{3})\pi R^2}{2^{2/3}(2\pi^3R^3 - 3\pi^2qt + \sqrt{3}\sqrt{-4\pi^5qR^3t + 3\pi^4q^2t^2})^{1/3}} - \frac{(1 + i\sqrt{3})(2\pi^3R^3 - 3\pi^2qt + \sqrt{3}\sqrt{-4\pi^5qR^3t + 3\pi^4q^2t^2})^{1/3}}{22^{1/3}\pi}\Big\}\Big\}$$

We obtain from this three different functions for **h[t]** because the equation we solved was cubic in the level. In order to decide which of the three is correct, we assign them to three different functions and then evaluate them over time with a given set of parameter values:

```
In[160]:= h31[t_] := Evaluate[h /. h3o[[1, 1]]]
          h32[t_] := Evaluate[h /. h3o[[2, 1]]]
          h33[t_] := Evaluate[h /. h3o[[3, 1]]]

          q = 100;
          R = 10;
          tf = 35;

          Plot[{h31[t], h32[t], h33[t]}, {t, 0, tf},
             PlotStyle → {{Thickness[0.01], GrayLevel[0]},
               {Thickness[0.01], GrayLevel[0.5],
                Dashing[{0.03, .03}]},
               {Thickness[0.01], GrayLevel[0.8],
                Dashing[{0.03, .03}]}},
             AxesLabel → {"t", "h3x[t]"},
             Epilog → Line[{{0, R}, {tf, R}}]
            ];
```



The plot shows us that the first solution in black **h31[t]**, is a decreasing function, when in fact we know we should have one that is increasing with time since we are filling a tank. The same problem and worse arises with **h32[t]**, which is not only decreasing, but has negative values of the level. This leaves us with **h33[t]** in gray as the only physically realistic solution. Another way to attack this problem without a graphical presentation of the data would be to solve the equation numerically as is shown in the cell that follows:

```
In[160]:= q = 100;
          R = 10;
```

$$\texttt{Table[NSolve[}Rh^2 - \frac{h^3}{3} == \frac{q}{\pi}t\texttt{, h],}$$

```
           {t, 0.1, 27, 3}]// TableForm
```

```
Out[162]//TableForm=
          h → -0.559006     h → 0.569623     h → 29.9894
          h → -2.9953       h → 3.33172      h → 29.6636
          h → -4.13119      h → 4.80867      h → 29.3225
          h → -4.98393      h → 6.01977      h → 28.9642
          h → -5.68991      h → 7.10392      h → 28.586
          h → -6.30241      h → 8.11758      h → 28.1848
          h → -6.84878      h → 9.09225      h → 27.7565
          h → -7.34529      h → 10.0497      h → 27.2956
          h → -7.8025       h → 11.008       h → 26.7945
```

What we see is that for each value of **t** in the range from 0.1 to 27 we have three different values of **h[t]** computed. In this case the middle column is the proper solution set. This is just a small example of the very many different ways in which one can solve a problem like this using *Mathematica*.

Depositing a Polymer Coating on a Disk

Polymers have come to play an ever more important role in our lives since the discovery of Nylon by Wallace Carruthers more than 60 years ago. From grocery bags and clothing to medical devices and implants, synthetic polymers have proven to be a boon to mankind. In Figure 8 we look at a simple schematic of a polymer coating process. Nonstick cookware, automobile finishes, and magnetic storage media all involve some polymeric coating on a substrate. We will consider a very rudimentary example from the perspective of our analyses based on a total material balance.

In the particular process shown in Figure 9, a substrate is translated under a spray nozzle at some velocity v_z. At the same time small droplets containing a 5:1 mixture of monomer and activator (polymerization initiator) are sprayed. The droplets are created by sonication of the liquid mixture in the spray gun. The action of sonication not only provides very small

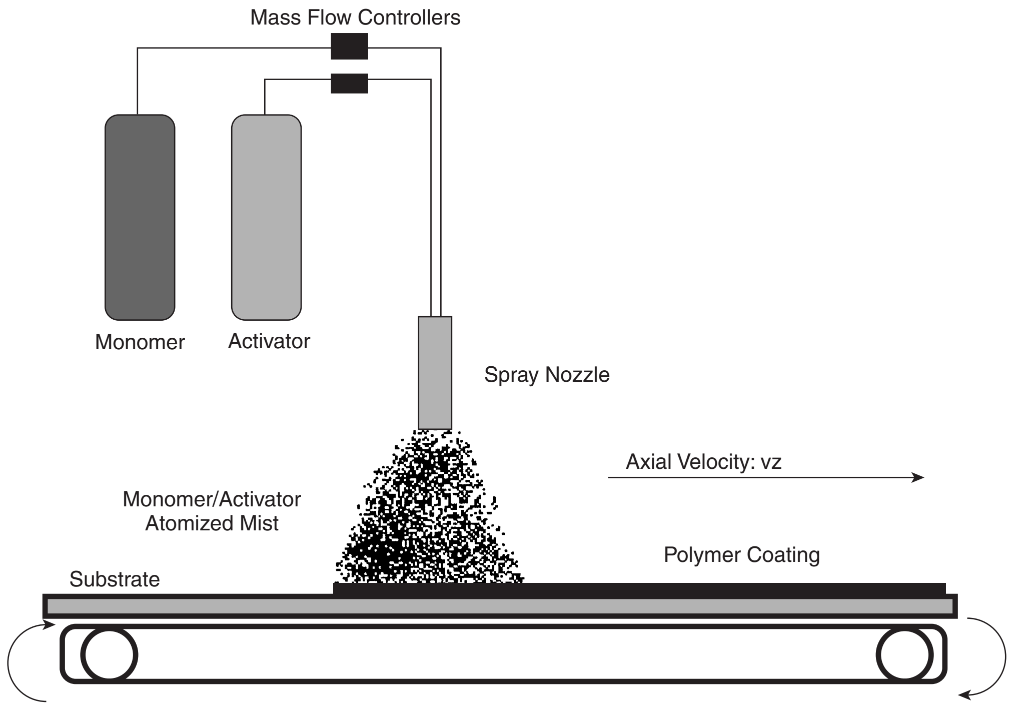


Figure 8 Polymer coating process on a substrate.

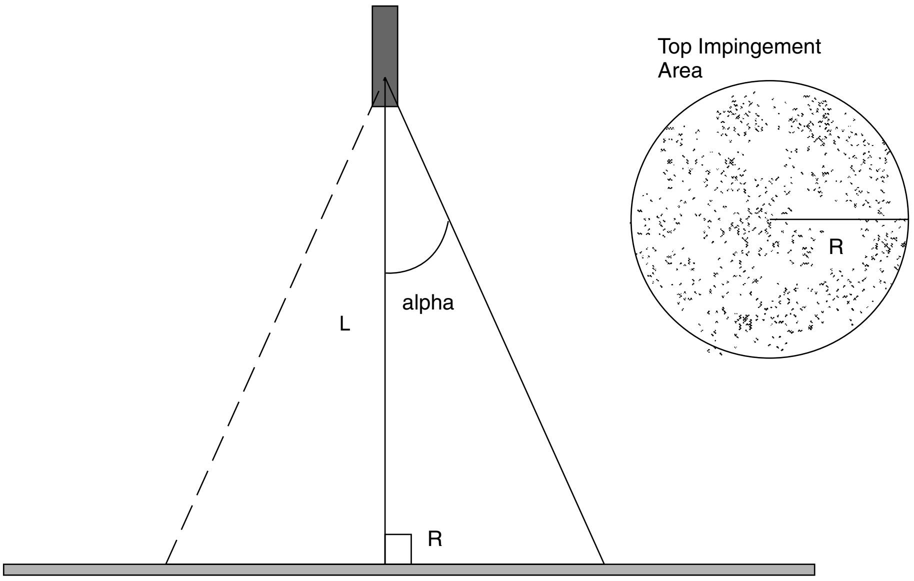


Figure 9

droplets (~0.5 μ diameter), the energy combined with exposure to oxygen in the air initiates the conversion of monomer into polymer. As the substrate moves under the spray gun, the polymerized droplets impinge and gel on the solid substrate, which creates a solid film on the surface that hardens with time.

At the end of the spray nozzle there is an orifice with a circular area $\boldsymbol{A_o}$. All of the mass of monomer plus activator must pass through this orifice area. The mass flowing per unit time per unit area through this area is referred to as *flux*. After leaving the orifice the spray area expands to form a cone-shaped region above the substrate. The spray area on a motionless substrate would be approximately circular. The radius of the circular impingement area is related to the distance **L** between the nozzle and the substrate. Experiments have shown that the mass impinging on this area is homogeneous. Because the substrate has a width equal to the diameter of the impingement area, if it is moved under the nozzle at constant velocity, it will be evenly coated (excluding end effects).

From this description we should be able to predict the thickness of the polymer coating as a function of the delivery rate and substrate velocity. The monomer and activator are taken to have equal densities. The total material balance around the spray nozzle is:

$$\frac{\mathrm{dm[t]}}{dt} = \mathring{\mathrm{m}}_{\mathrm{in}} - \mathring{\mathrm{m}}_{\mathrm{out}}$$

There should be no net accumulation in the spray nozzle, meaning that it will nearly always operate at a steady state. (In fact a design criterion for a sprayer like this is that it have nearly instantaneous rise and shut-off times.) From this then we can say that the mass flow rate in must equal the mass flow rate out:

$$\mathring{m}_{\text{in}} = \mathring{m}_{\text{out}}$$

The rate of mass flow into the spray gun is given as the sum of the mass flows of the monomer and the activator. The densities of the two liquids are nearly identical and the volumetric flow rate of the monomer is five times the flow rate of the activator. Therefore the mass out of the orifice is:

$$\begin{aligned}\mathring{m}_{\text{out}} &= \rho_{\text{monomer}}\, q_{\text{monomer}} + \rho_{\text{activator}}\, q_{\text{activator}}\\ &= \rho_{\text{monomer}}\,(q_{\text{monomer}} + q_{\text{activator}})\\ &= 1.2\ \rho_{\text{monomer}}\, q_{\text{monomer}}\end{aligned}$$

The mass flux from the orifice is the mass flow rate out divided by the orifice area. Since the spray spreads in the form of a right cone, the mass flux at the substrate is smaller than the flux from the orifice. To solve our problem we must know this. It is obvious then that the product of the flux and the area is the mass flow rate. By the conservation of mass the mass flow to the substrate must be the same as the mass flow from the orifice. Therefore we can proceed as follows:

$$\begin{aligned}\mathring{m}_{\text{impingement}} &= \mathring{m}_{\text{out}}\\ J_{\text{imp}}\, A_{\text{imp}} &= J_{\text{out}}\, A_{\text{out}}\\ J_{\text{imp}} &= \frac{J_{\text{out}}\, A_{\text{out}}}{A_{\text{imp}}}\\ J_{\text{imp}} &= \frac{\mathring{m}_{\text{out}}}{A_{\text{imp}}}\end{aligned}$$

We will review what we know. We know the mass flow out of the spray nozzle and we can calculate the area of impingement knowing the area of the orifice $\boldsymbol{A_o}$, the distance to the substrate **L**, and the angle, alpha, made by the spray leaving the nozzle. Thus the next step must be to find the area of impingement in terms of alpha, **L**, and $\boldsymbol{A_o}$.

From the geometry diagram shown in Figure 10 we see that the radius of the orifice is r, which is related to l through the tangent of alpha. Then the radius **R** of the impingement area is related to (**l** + **L**) through the tangent of alpha ($\mathbf{tan}\alpha = \frac{\mathbf{opp}}{\mathbf{adj}}$). The distance l is not known, but it corresponds to the vertex of the triangle that is within the sprayer. We can solve for it in terms of alpha and **r**, both of which are known. Then we can set the two ratios equal to one another and solve for **R** in terms of **L**.

```
In[163]:= Clear[R, L, l, r, l, α]
          Solve[{r/l == Tan[α], r/l == R/(L + l)}, {l, R}]

Out[164]= {{R → r + L Tan[α], l → r Cot[α]}}
```

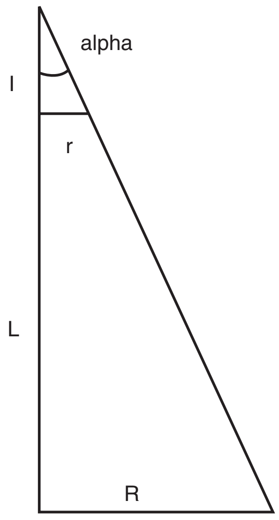


Figure 10

The area of impingement is πR^2, so the impingement flux is:

```
In[165]:=  A_imp = r  + L Tan[α]
           J_imp == m°_out / A_imp
Out[165]=  r + LTan[α]
```

$$A_{imp} = r\ + L\,\mathrm{Tan}[\alpha]$$

$$J_{imp} == \frac{\overset{\circ}{m}_{out}}{A_{imp}}$$

$$\mathrm{Out[165]=}\ r + L\mathrm{Tan}[\alpha]$$

```
        General::spell1: Possible spelling error: new symbol name
         "out" is similar to existing symbol "Out".
```

$$\mathrm{Out[166]=}\ J_{imp} == \frac{\overset{\circ}{m}_{out}}{r + L\ \mathrm{Tan}[\alpha]}$$

The next part of the analysis is to find the mass accumulation of polymer on the substrate given J_{imp} and that the substrate moves under the nozzle at a velocity equal to v_z. If the substrate did not move then the mass would accumulate on the circular impingement area. Given a rigid solid polymer, we would begin to grow a deposit of circular cross section and increasing thickness. As the deposit grew off the substrate, it would decrease **L**, and so the circular deposition area would decrease. We could imagine that we would grow a deposit that would be roughly like the frustrum of a cone and eventually a cone. If we allowed the deposition to take place for some time τ, such that the thickness grew to $\boldsymbol{\delta}$, and then moved the substrate by a distance **2R** and deposited for a period $\boldsymbol{\tau}$ again, and then continued this stepwise process, we would have a series of circular deposits of nearly uniform thickness. We would also have areas between the circular deposits that were nearly uncovered. This would be a semicontinuous process. If $\boldsymbol{\tau}$ were to become very short, then the process would begin to

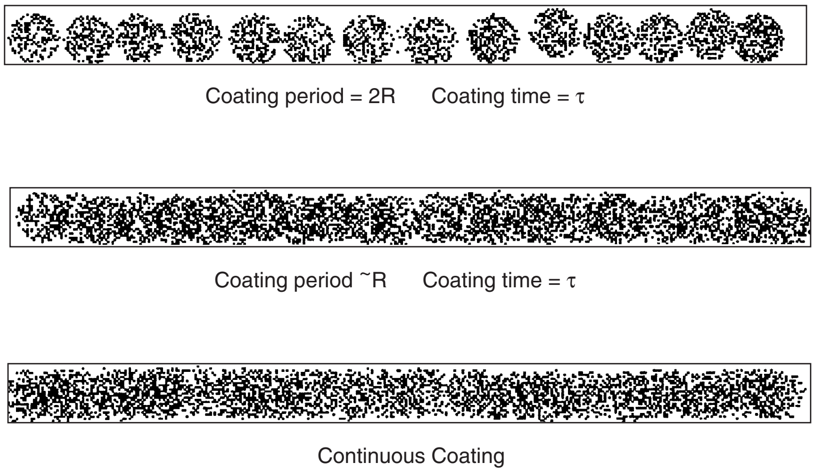

Coating period = 2R Coating time = τ
Coating period ~R Coating time = τ
Continuous Coating

Figure 11

approach a continuous one, especially if we only moved the substrate by some fraction of **2R** every time. Figure 11 shows all three deposition types discussed here.

In the continuous process we let τ shrink to small values by moving at a continuous velocity. Now the time spent under the nozzle is just $\tau = \frac{2R}{v_z}$, that is the "residence time" under the sprayer. By translating in this way the circular spray area becomes nearly rectangular, save for the edges. We can write a material balance for the problem in the usual way—that is, choosing the spray area and the growing polymer film as the control volume:

$$\frac{\mathrm{dm[t]}}{\mathrm{dt}} = \mathring{m}_{\mathrm{in}}$$

$$\frac{\mathrm{d}[\rho_{\mathrm{polymer}} V[\mathrm{t}]_{\mathrm{deposit}}]}{\mathrm{dt}} = \mathring{m}_{\mathrm{in}}$$

$$\frac{\mathrm{d}[\rho_{\mathrm{polymer}} A[\mathrm{t}]_{\mathrm{deposit}} \delta[\mathrm{t}]]}{\mathrm{dt}} = \mathring{m}_{\mathrm{in}}$$

$$\frac{\mathrm{d}[wz[\mathrm{t}]\delta[\mathrm{t}]]}{\mathrm{dt}} = \frac{\mathring{m}_{\mathrm{in}}}{\rho_{\mathrm{polymer}}}$$

At this point we can recognize that w is related to **R** and substitute it in:

$$\mathrm{w} = 2\mathrm{R}$$

$$\frac{\mathrm{d}[\mathrm{z}[\mathrm{t}]\delta[\mathrm{t}]]}{\mathrm{dt}} = \frac{\mathring{m}_{\mathrm{in}}}{2R\rho_{\mathrm{polymer}}}$$

However, R is related to distance **L**, angle α, and dimension of the orifice **r**:

$$2\mathrm{R} = 2(\mathrm{r} + \mathrm{L}\,\mathrm{Tan}[\alpha])$$

This brings us to this equation:

$$\frac{d[z[t]\delta[t]]}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{2(r + L\ \mathrm{Tan}[\alpha])\rho_{\mathrm{polymer}}}$$

Applying the chain rule to the left-hand side, we find:

$$\delta[\mathrm{t}]\frac{dz[t]}{dt} + z[\mathrm{t}]\,\frac{d\delta[t]}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{2(r + L\ \mathrm{Tan}[\alpha])\rho_{\mathrm{polymer}}}$$

If the substrate is translated at a fixed velocity then we can say that **z[t]**, the position of the substrate at any time, is merely a linear function of time:

$$z(\mathrm{t}) = v_z\mathrm{t} + \mathrm{c}$$

This leaves us with the following equation to solve for the thickness as a function of time. By rearranging it we can see that it is a nonlinear differential equation and we must treat it accordingly:

$$\delta[t]v_z + (v_zt + \mathrm{C})\frac{d\delta[t]}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{2(r + L\ \mathrm{Tan}[\alpha])\rho_{\mathrm{polymer}}}$$
$$(v_zt + \mathrm{C})\frac{d\delta[t]}{dt} = \frac{\mathring{\mathrm{m}}_{\mathrm{in}}}{2(r + L\ \mathrm{Tan}[\alpha])\rho_{\mathrm{polymer}}} - \delta[\mathrm{t}]v_z$$

```
In[167]:= Clear[δ]
          m̊in=.
          Together [m̊in/(2(r + L Tan[α]) ρpolymer) - δ[t] vz]

        (Unset::norep: Assignment on Subscript for m̊in not found.

Out[168]= $Failed
```

$$\mathit{Out[169]=}\ \frac{\mathring{\mathrm{m}}_{\mathrm{in}} - 2\mathrm{r}\mathrm{v}_{\mathrm{z}}\rho_{\mathrm{polymer}}\delta[\mathrm{t}] - 2\mathrm{L}\mathrm{v}_{\mathrm{z}}\rho_{\mathrm{polymer}}\mathrm{Tan}[\alpha]\delta[\mathrm{t}]}{2\rho_{\mathrm{polymer}}(\mathrm{r} + \mathrm{L}\,\mathrm{Tan}[\alpha])}$$

```
In[170]:= (m̊in - Collect[2rvz ρpolymer δ[t] - 2Lvz ρpolymer Tan[α] δ[t],
           {vz ρpolymer δ[t]}])/(2ρpolymer (r + L Tan[α]))
```

$$\mathit{Out[170]=}\ \frac{\mathring{\mathrm{m}}_{\mathrm{in}} - \mathrm{v}_{\mathrm{z}}\rho_{\mathrm{polymer}}(2\mathrm{r} - 2\mathrm{L}\,\mathrm{Tan}[\alpha])\delta[\mathrm{t}]}{2\rho_{\mathrm{polymer}}(\mathrm{r} + \mathrm{L}\,\mathrm{Tan}[\alpha])}$$

The last transformation shows us that the equation is separable into:

$$\frac{d\delta[t]}{\mathring{m}_{in} - 2v_z\rho_{polymer}(r - L\,\mathrm{Tan}[\alpha])\delta[t]} == \frac{dt}{2\rho_{polymer}(r + L\,\mathrm{Tan}\,[\alpha])(v_z t + C)}$$

These two sides can be integrated indefinitely to give:

In[171]:= $\mathbf{Simplify}\left[\int \frac{1}{\mathring{m}_{in}-2v_z\rho_{polymer}(r-L\,\mathrm{Tan}[\alpha])\delta[t]}d\delta[t]\right] ==$

$\int \frac{1}{2\rho_{polymer}(r+L\,\mathrm{Tan}[\alpha])(v_z t + C)}dt$

Out[171]= $-\frac{\mathrm{Cos}[\alpha]\,\mathrm{Log}[-\mathrm{Cos}[\alpha]\mathring{m}_{in} + 2(r\,\mathrm{Cos}[\alpha] - L\,\mathrm{Sin}[\alpha])v_z\rho_{polymer}\delta[t]]}{2r\mathrm{Cos}[\alpha]v_z\rho_{polymer} - 2L\,\mathrm{Sin}[\alpha]v_z\rho_{polymer}} ==$

$\frac{\mathrm{Log}[C + tv_z]}{2v_z\rho_{polymer}(r + L\,\mathrm{Tan}[\alpha])}$

Or integrated definitely to give:

In[172]:= $\mathbf{Simplify}\left[\int_0^{\delta[t]} \frac{1}{\mathring{m}_{in}-2v_z\rho_{polymer}(r-L\,\mathrm{Tan}[\alpha])y}dy\right] ==$

$\int_0^t \frac{1}{2\rho_{polymer}(r+L\ \mathrm{Tan}[\alpha])(v_z X + C)}dx$

Out[172]= $\frac{\mathrm{Cos}[\alpha](\mathrm{Log}[-\mathrm{Cos}[\alpha]\mathring{m}_{in}] - \mathrm{Log}[-\mathrm{Cos}[\alpha]\mathring{m}_{in} + 2(r\,\mathrm{Cos}[\alpha] - L\,\mathrm{Sin}[\alpha])v_z\rho_{polymer}\delta[t]])}{2(r\,\mathrm{Cos}[\alpha] - L\,\mathrm{Sin}[\alpha])v_z\rho_{polymer}}$

$== \frac{-\mathrm{Log}[C] + \mathrm{Log}[C + tv_z]}{2v_z\rho_{polymer}(r + L\,\mathrm{Tan}[\alpha])}$

As in:

In[173]:= **Clear[a, b, m, n, x, y, p, δ]**

$\int_0^\delta \frac{1}{a+by}dy == \frac{1}{p}\int_0^t \frac{1}{mx+n}dx$

Out[174]= $\frac{-\mathrm{Log}[a] + \mathrm{Log}[a + b\delta]}{b} == \frac{-\mathrm{Log}[n] + \mathrm{Log}[n + mt]}{mp}$

In[175]:= $$\mathbf{b}\left(\frac{\mathbf{-Log[a] + Log[a + b\delta]}}{\mathbf{b}}\right) \mathbf{==} \mathbf{b}\left(\frac{\mathbf{-Log[n] + Log[n + mt]}}{\mathbf{mp}}\right)$$

Out[175]= $$\mathtt{-Log[a] + Log[a + b\delta] ==} \frac{\mathtt{b(-Log[n] + Log[n + mt]}}{\mathtt{mp}}$$

In[176]:= $$\frac{\mathbf{a + b\delta}}{\mathbf{a}} \mathbf{==} \left(\frac{\mathbf{n + mt}}{\mathbf{n}}\right)^{\frac{\mathbf{b}}{\mathbf{mp}}}$$

In[176]:= $$\frac{\mathbf{a + b\delta}}{\mathbf{a}} \mathbf{==} \left(\frac{\mathbf{n + mt}}{\mathbf{n}}\right)^{\frac{\mathbf{b}}{\mathbf{mp}}}$$

In[177]:= $$\mathbf{Solve}\left[\frac{\mathbf{a + b\delta}}{\mathbf{a}} \mathbf{==} \left(\frac{\mathbf{n + mt}}{\mathbf{n}}\right)^{\frac{\mathbf{b}}{\mathbf{mp}}}, \delta\right]$$

Out[177]= $$\{\{\delta \rightarrow \frac{\mathtt{a}\left(-1 + \left(\frac{\mathtt{n + m}}{\mathtt{n}}\right)^{\frac{\mathtt{b}}{\mathtt{mp}}}\right)}{\mathtt{b}}\}\}$$

We assume here that the nozzle is held 10 cm away from the substrate (**L** = 10) and that the cone angle made by the spray is 20°. The polymer may have density of 2 g cm^{-3}. The axial velocity should be small, so we choose 0.01 cm per min ($v_z = 0.01$). The flow of polymer mass to the substrate may also be small at 10 mg min^{-1} ($\mathring{\mathbf{m}}_{\mathbf{in}} = 0.01$ mg min^{-1}). The nozzle radius will be set at 0.003 cm. Before we use this result, we should check it for dimensional consistency. The units for each of the groups in the final formula are included on the right-hand side of what follows. The left-hand side is just the units of δ in cm. If the units are consistent then the logical operation should return a True.

In[178]:= $$\mathbf{Simplify[cm ==} \frac{\frac{\mathbf{q}}{\mathbf{min}}\left(-1 + \left(\frac{\mathbf{cm} + \frac{\mathbf{cm}}{\mathbf{min}}\mathbf{min}}{\mathbf{cm}}\right)^{\frac{\left(\frac{\mathbf{cm}}{\mathbf{min}}\frac{\mathbf{q}}{\mathbf{cm}^3}\mathbf{cm}\right)}{\frac{\mathbf{cm}}{\mathbf{min}}\left(\frac{\mathbf{q}}{\mathbf{cm}^3}\mathbf{cm}\right)}}\right)}{\left(\frac{\mathbf{cm}}{\mathbf{min}}\frac{\mathbf{q}}{\mathbf{cm}^3}\mathbf{cm}\right)}\mathbf{]}$$

```
General::spell1: Possible spelling error: new symbol name
 "min" is similar to existing symbol "Min".
```

Out[178]= True

The result is True, but maybe you would be more pleased to see the actual reduction to cm on the right-hand side. If so here is that result:

In[179]:= $$\frac{\frac{\mathbf{q}}{\mathbf{min}}\left(-1 + \left(\frac{\mathbf{cm} + \frac{\mathbf{cm}}{\mathbf{min}}}{\mathbf{cm}}\right)^{\frac{\left(\frac{\mathbf{cm}}{\mathbf{min}}\frac{\mathbf{q}}{\mathbf{cm}^3}\mathbf{cm}\right)}{\frac{\mathbf{cm}}{\mathbf{min}}\left(\frac{\mathbf{q}}{\mathbf{cm}^3}\mathbf{cm}\right)}}\right)}{\left(\frac{\mathbf{cm}}{\mathbf{min}}\frac{\mathbf{q}}{\mathbf{cm}^3}\mathbf{cm}\right)}$$

Out[179]= cm

Now that we can have confidence that the formula we have derived for δ[t] is dimensionally correct, we can test it with some "real" parameter values to see what it predicts:

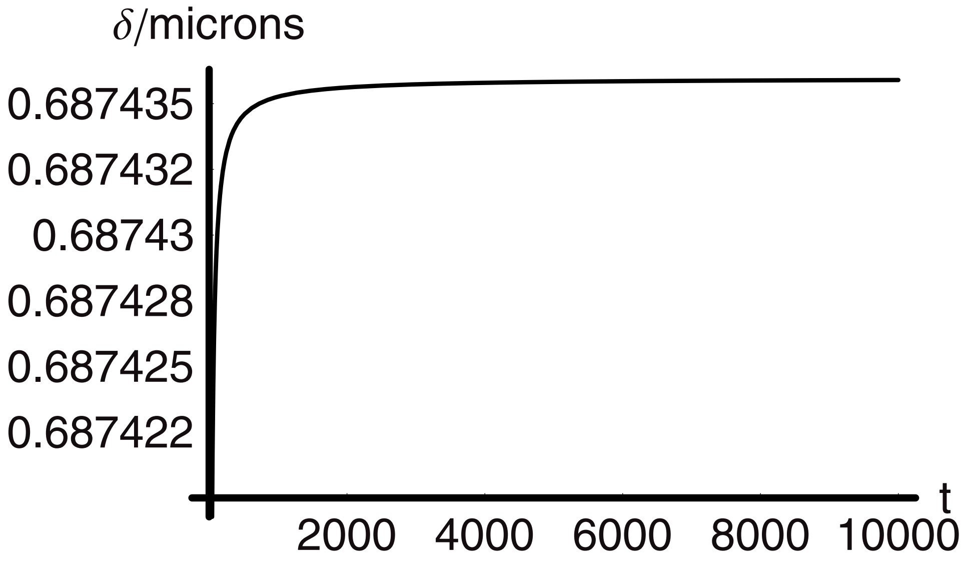
```
In[180]:= ρpolymer = 2;
          L = 10;
          α = 20/360 2π;
          m̊in = .001;
          vz = 1;
          r = 0.003;
          c = 0.001;

          a = m̊in;
          b = 2vz ρpolymer (r - L Tan[α]);
          p = 2ρpolymer (r + L Tan[α]);
          m = vz;
          n = c;
```

$$\delta[\text{t\_}] := \frac{a\left(-1+\frac{n+mt}{n}\right)^{\frac{b}{mp}}}{b}$$

```
          δ[.1]

          Plot[10^4 δ[x], {x, 0, 10000}, AxesLabel →
           {"t", "δ/microns"}];
```

$$\text{NumberForm}\left[\text{Limit}\left[\frac{a\left(-1+\frac{n+mt}{n}\right)^{\frac{b}{mp}}}{b}, t \to \infty\right] 10^4, 2\right]$$

```
Out[193]= 0.0000680578
```



```
Out[195] //NumberForm=
        0.69
```

At this delivery rate of polymer, and with these substrate dimensions and velocity, the thickness of the layer will be 0.69μ. Notice that the transient in the graph of thickness versus time is more apparent than real, as it involves changes in $\boldsymbol{\delta}$ that are on the order of one part in 10^5. Taking the limit of the expression for δ as the value of time approaches infinity gives us the same answer. Recall that we did not handle the startup of the substrate motion. We could do so if we replaced **z[t]** with a form that included quadratic time dependence and acceleration. Why don't you try it?

2.3 Summary

In this chapter we have covered quite a number of apparently different problems that are readily attacked by use of the concepts of a control volume and the conservation of mass. For each case the same equation was used as the point of departure for the analysis—namely, the differential statement of the total mass balance. It is surprising at first to find that so many different situations can be analyzed by the correct application of this equation along with some calculus and geometry along the way.

During the course of this chapter we also have learned how to use *Mathematica* interactively to do analysis. The methods we have used are general and we will use them throughout the text. The objective is to think and analyze the problem with the assistance of the computer in real time.

CHAPTER 3

The Draining Tank and Related Systems

3.1 The Right-Hand Side of the Mass Balance Equation

In Chapter 2 we developed models based on analyses of systems that had simple inputs. The right-hand side was either a constant or it was simple function of time. In those systems we did not consider the cause of the mass flow—that was literally external to both the control volume and the problem. The case of the flow was left implicit. The pump or driving device was upstream from the control volume, and all we needed to know were the magnitude of the flow the device caused and its time dependence. Given that information we could replace the right-hand side of the balance equation and integrate to the functional description of the system.

This level of simplicity is not the usual case in the systems that are of interest to chemical engineers. The complexity we will encounter will be much higher and will involve more detailed issues on the right-hand side of the equations we work with. Instead of a constant or some explicit function of time, the function will be an explicit function of one or more key characterizing variables of the system and implicit in time. The reason for this is that of cause. Time in and of itself is never a physical or chemical cause—it is simply the independent variable. When we need to deal with the analysis of more complex systems the mechanism that causes the change we are modeling becomes all important. Therefore we look for descriptions that will be dependent on the mechanism of change. In fact, we can learn about the mechanism of change by testing our ideas about physics or chemistry in the context of a model derived from an analysis based on some specific assumptions. Comparison of the model predictions with the actual behavior of the system provides a check on the analysis and its assumptions. Chemical engineers do this with almost every problem they encounter. Depending on the level of analysis and the nature of the problem, the results can be anything from a useful engineering description of the systems to new science. Regardless of the goals and objectives of the project, the chemical engineer uses the same powerful analysis paradigm to make progress and to solve the problem.

With this in mind we turn now to another problem that is seemingly naive at first glance, but which offers considerable insight into the next level of this process called analysis. The problem that we shall consider now is that of flow out of a vessel due to the force of gravity. We will apply the same principles as in Chapter 2, but the cause of the flow will be an essential part of the analysis. We also have the chance to see how this problem fits into the history and foundations of our physics. This is a 300-year-old problem that is still full of fascination for us and from which we can learn much.

3.2 Mechanism of Water Flow from Tank—*Torricelli's Law, A Constitutive Relationship*

In 1640 a young man educated at the Collegio di Sapienza in Rome published a treatise entitled "Trate del Moto," that is the Treatise on Motion. This brought the author to the attention of the preeminent natural philosopher of the day—Galileo Galilei. Impressed by the work, Galileo invited the author, Evangelista Torricelli, to join him at the Florentine Academy in 1641. Torricelli worked as Galileo's personal secretary for just one year before the great man died in 1642. The faculty at Florence immediately appointed Torricelli to succeed Galileo as professor of mathematics that same year. Torricelli is ensconced in physical science through the unit of pressure that bears his name, in honor of his experiments that led to the creation of the first partial vacuum and the barometer. But Torricelli did much more than this and it is to some of his other work that we will turn to now.

Just as Galileo studied gravitational effects by dropping small and large solid masses from towers, Torricelli followed in his footsteps by analyzing liquids "falling" out of tanks. What he did was to measure the flow rate of liquids flowing out of tanks with holes in the bottom as a function of initial fluid level at a fixed orifice area. The physical system that corresponds to this is shown in Figure 1. The vessel is a right cylinder with cross-sectional area **A**. At the bottom of the vessel is a hole with area **Ao** that has a plug in it. The initial liquid level has been set to **ho**.

When the plug is pulled out of the orifice, the liquid flows out. One can measure the flow rate or the level as a function of time to learn how the system behaves. If we did this, then we would notice from these experiments that the flow rate out is not constant, but seems to drop with time. As the level drops, so does the rate at which mass leaves the tank. Before we

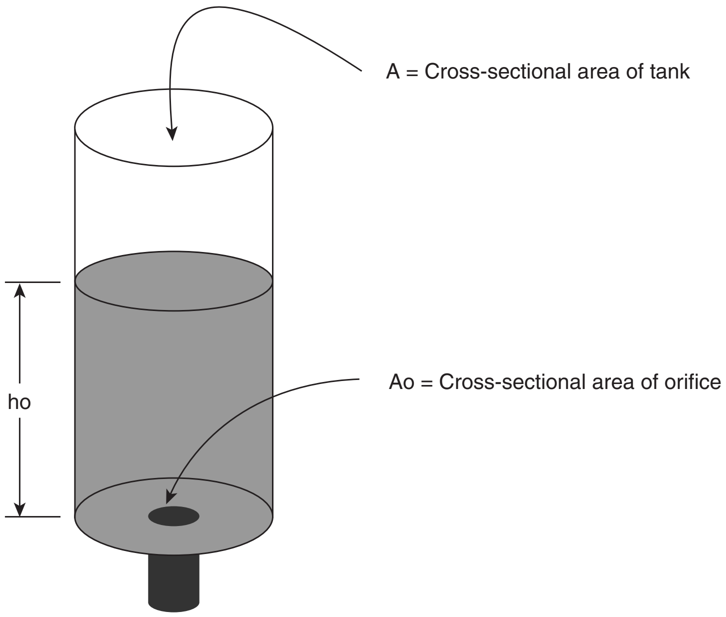

A = Cross-sectional area of tank
Ao = Cross-sectional area of orifice
ho

Figure 1

consider the data that comes from an experiment like this one, we should apply the principle of the conservation of mass to the system to see what it is that we know and do not know about it. We begin with the same overall equation as always:

$$\frac{dm[t]}{dt} = -\mathring{m}_{\text{out}}$$

The mass flow in is zero, so we are left with just one term on the right-hand side. The density is a constant inside and outside of the vessel and the cross section of the liquid is just that of the tank **A**, and it too remains constant as the level drops. This leads us to the following equation for the change in level as a function of time:

$$\frac{dh[t]}{dt} = -\frac{q[t]}{A}$$

On the right-hand side, we have noted that the flow rate changes with time by writing **q** as a function of time, but that is all that we can do at this point. We have no way of knowing how **q** varies with time or with the level in the tank. Therefore, we cannot go any further with this analysis until we have some way to express this dependence. What we are seeking for **q** is a *constitutive relationship* that will express it in terms of the level change **h[t]** so that the equation has only one dependent variable in time and not two. In other words, the analysis we have just done is incomplete, but it has shown us exactly what our experiments should be designed to do. We must find a functional relationship between the level in the tank and the flow rate out of it. Or in mathematical terms we seek:

$$q[t] \to f[h[t]]$$
$$\frac{dh[t]}{dt} = -\frac{f[h[t]]}{A}$$

The first is the constitutive relationship and the second is just a restatement of the material balance. The next section takes this into account as we trace what Torricelli is likely to have done.

3.3 Experiment and the Constitutive Equation

What causes the fluid to flow or "fall" out of the tank? Gravity of course. That is easy for us to answer because we are educated in fundamental physics concepts. But turn the clock back to Torricelli's day and try to answer the question again in the year in which Galileo died and Isaac Newton was born. That means that Torricelli was working on this problem prior to calculus and Newtonian physics! Now, Galileo and Torricelli both had strong notions and good ideas of what this force, what we now call gravity, was and Newton was indebted to them when he did his work, but all of that came later. So we will attack this problem from the "experimental" side and try to piece together the findings that led to Torricelli's Law.

In the Table are collected data from several experiments with different starting levels, but all with the same orifice area, and all done in the same cylindrical tank. Look at the data for some clues as to the behavior of this system. Note that it takes 60 sec to drain the tank from an initial level of 10 cm. When the initial level is set to 50 cm, it takes 130 sec for the tank to drain:

t/sec			**Level/cm**		
0	10	20	30	40	50
10	6.8	15.4	24.2	33.3	42.5
20	4.2	11.3	19.1	27.2	35.6
30	2.3	7.9	14.6	21.8	29.3
40	0.9	5.1	10.7	16.9	23.6
50	0.2	2.9	7.3	12.7	18.5
60	0	1.3	4.7	9.	14.1
70	0	0.4	2.6	6.	10.2
80	0	0	1.1	3.6	7.
90	0	0	0.3	1.8	4.4
100	0	0	0	0.7	2.4
110	0	0	0	0	1
120	0	0	0	0	0.2
130	0	0	0	0	0

We raised by a factor of five both the initial level and the volume that must be drained and yet it took only twice the total time to complete the process. Looking even more closely at the data, we note that in the same experiment at which the initial level was 50 cm, when the level in that experiment had fallen to 10 cm, it took another 60 sec to completely drain the tank. In other words, once the level gets to 10 cm it takes the same time to completion as if the experiment had begun at 10 cm. Therefore, the first 40 cm fell out of the tank in 70 sec, but the final 10 cm took another 60 sec to fall out. Looking across the data sets we see the same thing in each. The time required to go to completion beginning at 20 cm is the same as the time required to empty the tank after having reached 20 cm when beginning at any higher level. But the time required to go from an initial level of 40 cm to 20 cm is just 30 sec, or less than half the time required to drain the second 20 cm worth of fluid! This behavior had to have been very intriguing to Torricelli and the analysis must not have been obvious, at least at the outset of the work.

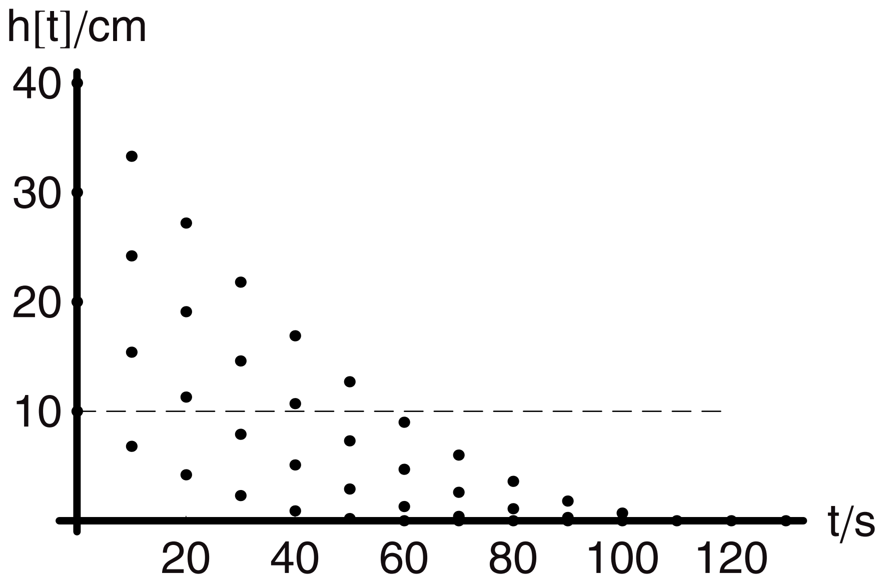
The relationships that we have just described in words are much easier to see if we plot the data as level versus time and do so all on one graph. We can use a few functions to pull this all together. The data in the table has been entered as a matrix of levels at each time called "**totdat**":

```
In[1]:= totdat = {{0, 10, 20, 30, 40, 50}, {10, 6.8, 15.4, 24.2, 33.3, 42.5},
        {20, 4.2, 11.3, 19.1, 27.2, 35.6}, {30, 2.3, 7.9, 14.6, 21.8, 29.3},
        {40, 0.9, 5.1, 10.7, 16.9, 23.6}, {50, 0.2, 2.9, 7.3, 12.7, 18.5},
        {60, 0, 1.3, 4.7, 9.0, 14.1}, {70, 0, 0.4, 2.6, 6.0, 10.2},
        {80, 0, 0, 1.1, 3.6, 7.0}, {90, 0, 0, 0.3, 1.8, 4.4},
        {100, 0, 0, 0, 0.7, 2.4}, {110, 0, 0, 0, 0, 1},
        {120, 0, 0, 0, 0, .2}, {130, 0, 0, 0, 0, 0}}

Out[1]= {{0, 10, 20, 30, 40, 50}, {10, 6.8, 15.4, 24.2, 33.3, 42.5},
        {20, 4.2, 11.3, 19.1, 27.2, 35.6}, {30, 2.3, 7.9, 14.6, 21.8, 29.3},
        {40, 0.9, 5.1, 10.7, 16.9, 23.6}, {50, 0.2, 2.9, 7.3, 12.7, 18.5},
        {60, 0, 1.3, 4.7, 9., 14.1}, {70, 0, 0.4, 2.6, 6., 10.2},
        {80, 0, 0, 1.1, 3.6, 7.}, {90, 0, 0, 0.3, 1.8, 4.4},
        {100, 0, 0, 0, 0.7, 2.4}, {110, 0, 0, 0, 0, 1},
        {120, 0, 0, 0, 0, 0.2}, {130, 0, 0, 0, 0, 0}}
```

The matrix form of the data needs to be transformed into data sets having each time and each level in pairs. We can do this in one small program as follows and the new data set will be called "**sepdat**" for the separated data:

```
In[2]:= SetOptions[{Plot, ListPlot}, AxesStyle → {Thickness[0.01]},
        PlotStyle → {PointSize[0.015], Thickness[0.006]},
        DefaultFont → {"Helvetica", 17}];

In[3]:= sepdat = Table[
        Table[{totdat[[n]][[1]], totdat[[n]][[m]]},
        {n, 1, Length[totdat]}],{m, 2, 5}];
```

```
ListPlot[Flatten[sepdat, 1],
  Epilog → {Dashing[{0.025, 0.015}], Line[{{0, 10}, {120, 10}}]},
  AxesLabel → {"t/s", "h[t]/cm"}];
```



For reference the dashed line across the data is set at the 10-cm level. With this we can see that once this level is reached, then independent of the starting point, it takes 50 sec to finish the process. The fluid moving out of the vessel then has no "memory" of the level at which the process was initiated. What we seek now is the relationship between the rate of change in level and the level in the tank, since both the material balance and the experimental data drive in this direction. We can get to this by computing the rate of change in level as a function of time for each experiment and then plotting this for comparison.

The approximate rate of change can be computed from the data by taking the slope between successive data points and plotting this versus the time at that second point. We can write a function to do this and then plot the data. The algorithm for implementing this procedure on the set "**sepdat**" is:

Take the nth set of data, from this extract the (m + 1) data pair and from this take the second number, subtract from this the second number from the mth data point of the same data set; divide this by the difference between the first number from the (m + 1) pair from the nth set and the first number in the mth data pair of the nth set. Do this for all the n datasets and all the m pairs in each set.

The function to do this is shown here:

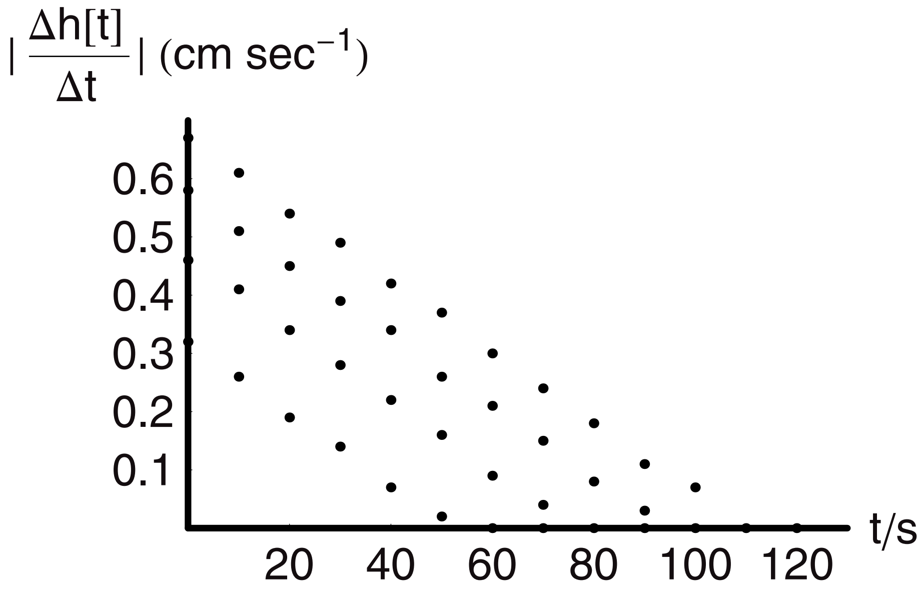
$$\text{rttot}[x\_, m\_, n\_] := \frac{x[[n, m+1, 2]] - x[[n, m, 2]]}{x[[n, m+1, 1]] - x[[n, m, 1]]}$$

It is worth understanding because it is such a useful tool when analyzing data—the relationship is written in a general way that can be implemented often in our analyses. To implement this we simply place it inside a pair of nested **Table** commands. The inside command creates a loop around the **m** data pairs in each set and the outside command loops over the **n** data sets. As we want the slopes associated with times we also include a command that takes each of the times and pairs it with a slope that looks like this:

$$\{\text{sepdat}[[1, m + 1, 1]], \text{rttot}[\text{sepdat}, m, n]\}$$

The following cell puts all of this together and makes a plot of the slopes versus time for each set of data. Each data set corresponds to a different initial level:

```
In[5]:= rttot[x__, m_, n_] := (x[[n, m+1, 2]] - x[[n, m, 2]]) / (x[[n, m+1, 1]] - x[[n, m, 1]])

        Table[Table[{sepdat[[1, m, 1]], rttot[sepdat, m, n]},
           {m, 1, Length[sepdat[[1]]] - 1}], {n, 1, 4}];

        ListPlot[Flatten[Abs[%], 1],
          AxesLabel → {"t/s", "| Δh[t]/Δt |(cm sec^-1)"},
          PlotRange → {{0, 130}, {0, 0.7}}];
```

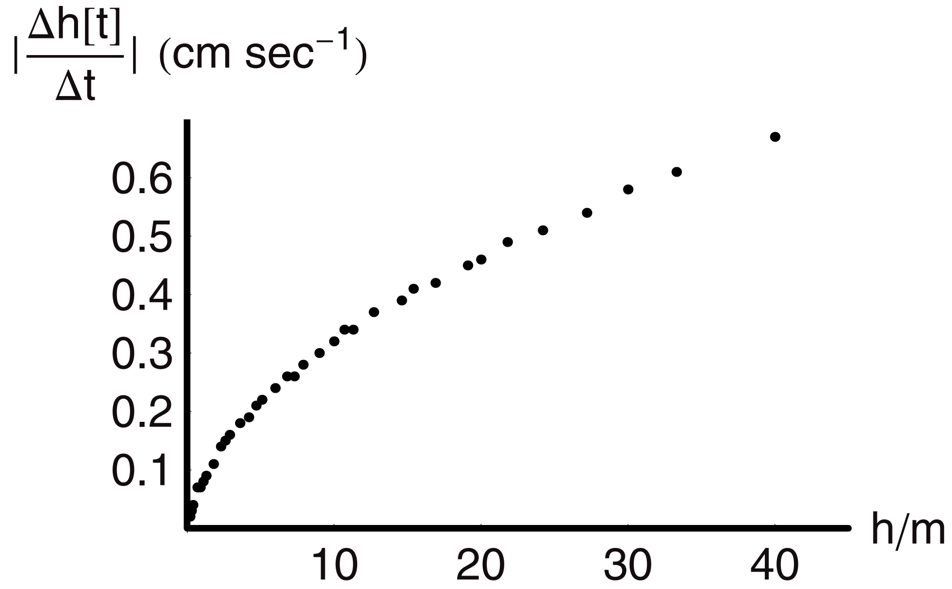


Each data set falls on a line and the slopes of each of these lines are identical! The slopes are the approximate second derivative of the change in level with time. From the graph we can see that all of the second derivatives are constant and each is equal to the same value, or stated more precisely:

$$\frac{\Delta}{\Delta t}\left(\frac{\Delta h[t]}{\Delta t}\right) = \text{constant}$$

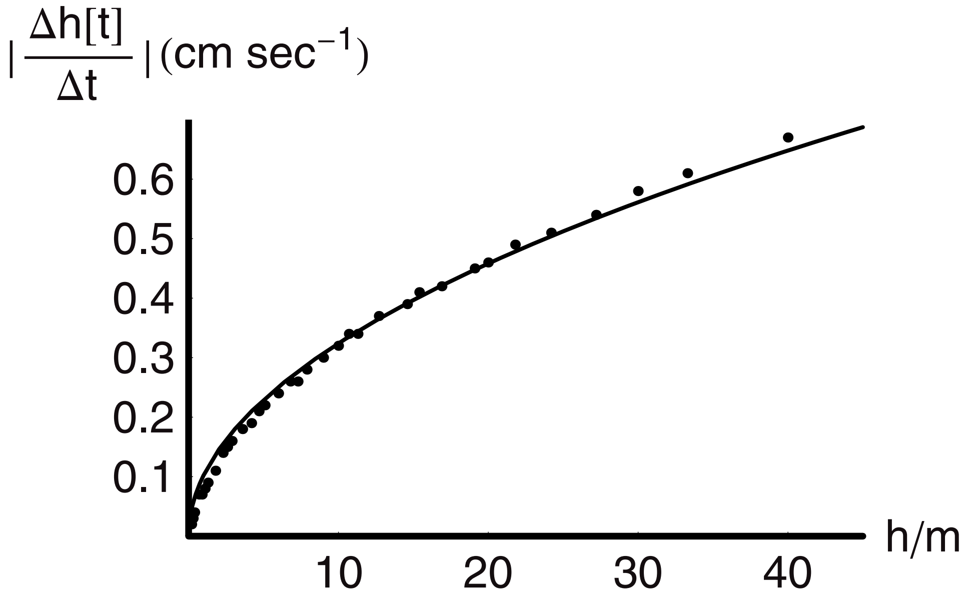
We realize now that the constant is the gravitational acceleration **g**. Even without knowing this it would be logical to replot the rate of change in level against the level rather than against the time. Out initial analysis of the system suggested this very approach, since level as a function of time is the key characterizing variable in this system. To do this we reuse the last cell but with a change in the table function to make the plot one of rate versus level rather than rate versus time. Hence the line "**sepdat[[1, m, 1]]**" becomes "**sepdat[[n, m, 2]]**."

```
                                 x[[n, m+1, 2]] - x[[n, m, 2]]
In[8]:= rttot[x_, m_, n_]:= ─────────────────────────────
                                 x[[n, m+1, 1]] - x[[n, m, 1]]

        Table[Table[{sepdat[[n, m, 2]], rttot[sepdat, m, n]},
           {m, 1, Length[sepdat[[1]]] - 1}], {n, 1, 4}];

        ListPlot[Flatten[Abs[%], 1],
                                   Δh[t]
          AxesLabel → {"h/m", "| ───── |(cm sec^-1)"},
                                    Δt
          PlotRange → {{0, 45}, {0, 0.7}}];
```



This is quite interesting—all the data from the different experiments now fall on one curve! We notice that the data have distinct functional dependence, which is neither linear nor logarithmic, but is, rather, square root. If Torricelli pursued this approach, and no doubt he did, then we can imagine that he felt quite a thrill when he made this graph and realized that he had uncovered a fundamental physical law. To be sure that the rate of change data do follow a square root dependence on level, we can fit the data to this form and evaluate the constants. We will use **Fit** for doing this. The data set will be constructed from the nested "**Table**" code in the last cell. The function we seek to fit to will be $\mathbf{k}\sqrt{h[t]}$. The procedure to do this and the comparison to the data are done in the following routine:

```
In[11]:= lsdat = Flatten[
            Table[
             Table[{sepdat[[n, m, 2]], rttot[sepdat, m, n]},
             {m, 1, Length[sepdat[[n]]] - 1}], {n, 1, 4}]
            , 1];

         Fit[
          Abs[1sdat], √h, h]

         Plot[Abs[%], {h, 0, 50}, DisplayFunction → Identity];

         ListPlot[Abs[1sdat],
            AxesLabel → {"h/m", "|Δh[t]/Δt| (cm sec^-1)"},
            PlotRange → {{0, 45}, {0, 0.7}}, DisplayFunction → Identity];

         Show[%, %%, DisplayFunction → $DisplayFunction];

Out[12]= 0.102464√h
```



The fit to the experimental data is quite good. (We can get a report on the "goodness of fit" if we need to, but it is not necessary here.) At this point we have found a function that relates the flow rate out of the tank to the level in the tank. The parameter value that fitted the data includes within it the cross-sectional area of the tank, as you may recall from the original statement:

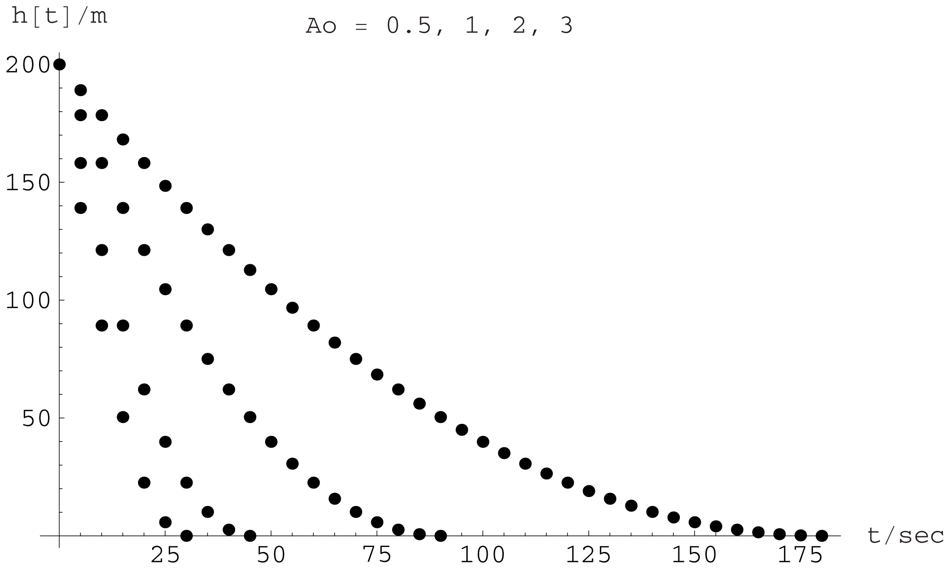
$$\frac{dh[t]}{\mathrm{dt}} = -\frac{f[h[t]]}{A} = -\frac{q[h[t]]}{A} = -\mathrm{k}\sqrt{h[t]} = -\frac{m}{A}\sqrt{h[t]}$$

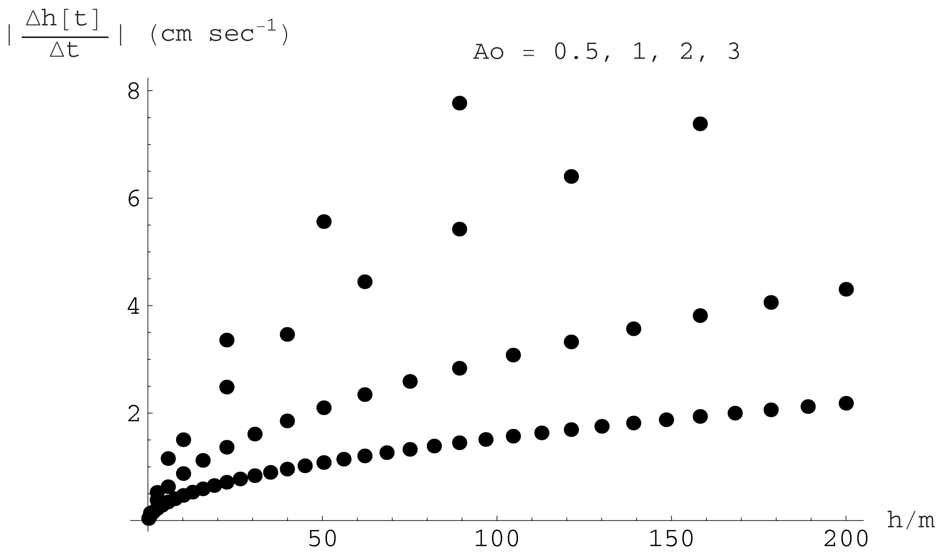
$$\Rightarrow \mathrm{k} = \frac{m}{A} = 0.10$$

At this point we would do well to analyze the dimensions of all the parameters on the right-hand side. As the left-hand side has dimensions of length of time^{-1}, so too must the right-hand side. Therefore, we can solve for the dimensions on the parameter **m**:

```
In[16] Solve[Length/time == m/Length^2 √Length, m]

Out[16] {{m → Length^(5/2)/time}}
```

Now if we rerun the experiment we just described, but vary the orifice area, then we will find data that looks as follows:





An analysis of this data shows that the flow rate out of the vessel is directly proportional to the orifice area. From experiments like these, Torricelli finally deduced that the flow rate looked like this:

$$\text{q[h]} = \text{b1 Ao}\sqrt{h[t]}$$

It turns out that **b1** is the square root of 2 **g**, where **g** is the acceleration due to gravity. Thus in full form Torricelli's law is:

$$\text{q[h]} = \text{b Ao}\sqrt{2gh[t]}$$

The parameter **b** is found empirically. It is related to the resistance to flow through the orifice. If the orifice were perfectly smooth, then **b** would have a value of unity. It is essentially a coefficient of friction. We check for dimensional consistency one more time and find:

In[17]:= **Simplify[$\frac{\text{Length}^3}{\text{time}}$ == (1) Length^2 PowerExpand[$\sqrt{\frac{\text{Length}}{\text{time}^2}\text{Length}}$]]**

Out[17]= True

Today it is clear that the dependence should be of half order in **t** because we can do a simple energy balance to determine that this is the case. If due to the force of gravity the fluid is falling out of the vessel when it flows, then it truly is a falling body. The kinetic energy during the fall can never exceed the potential energy that the body has prior to the fall. We can then state that:

$$\mathrm{mgh[t]} = \frac{1}{2}\,\mathrm{mv[t]}^2$$

$$\mathrm{v[t]} = \sqrt{2\,gh[t]}$$

In the case of the fluid, this is its velocity through the orifice. The product of this velocity and the area of the orifice is by definition the volumetric flow rate:

$$\mathrm{q[h[t]]} = \mathrm{Ao\,v[t]} = \mathrm{Ao}\sqrt{2gh[t]}$$

Although Torricelli did not know this, his work helped to point Newton in the right direction. Therefore, Torricelli's Law is the constitutive relationship that we seek to complete our model. We return to that endeavor now.

3.4 Solving for Level as a Function of Time

The equation that will yield the level as a function of time is this one:

$$\frac{dh[t]}{dt} = -\frac{b\,\mathrm{Ao}\sqrt{2gh[t]}}{A}$$

We can solve this for **h[t]**:

In[18]:= **Clear[Ao, g, A, ho, h, b]**

$$\mathbf{Simplify[DSolve[\{h'[t] == -\frac{b\,Ao\sqrt{2gh[t]}}{A},\ h[0] == ho\},\ h[t],\ t]]}$$

Out[19]=

$$\{\mathrm{h[t]} \to \mathrm{ho} - \frac{\sqrt{2}\,\mathrm{Ao\,b}\sqrt{\mathrm{g}}\sqrt{\mathrm{ho}}\,\mathrm{t}}{\mathrm{A}} + \frac{\mathrm{Ao}^2\,\mathrm{b}^2\,\mathrm{gt}^2}{2\mathrm{A}^2},$$

$$\mathrm{h[t]} \to \mathrm{ho} + \frac{\sqrt{2}\,\mathrm{Ao\,b}\sqrt{\mathrm{g}}\sqrt{\mathrm{ho}}\,\mathrm{t}}{\mathrm{A}} + \frac{\mathrm{Ao}^2\,\mathrm{b}^2\,\mathrm{gt}^2}{2\mathrm{A}^2}\}$$

The functional form is what we should expect for the position of a body in motion and we find it to be a quadratic in time. As we know the level is falling and not rising, the first solution is the appropriate one for this situation. We can simplify it further as follows:

In[20]:=

$$\mathbf{ho\ Expand[(ho - \frac{\sqrt{2}\,Ao\,b\sqrt{g}\sqrt{ho}\,t}{A} + \frac{Ao^2\,b^2\,gt^2}{2A^2})/ho]}$$

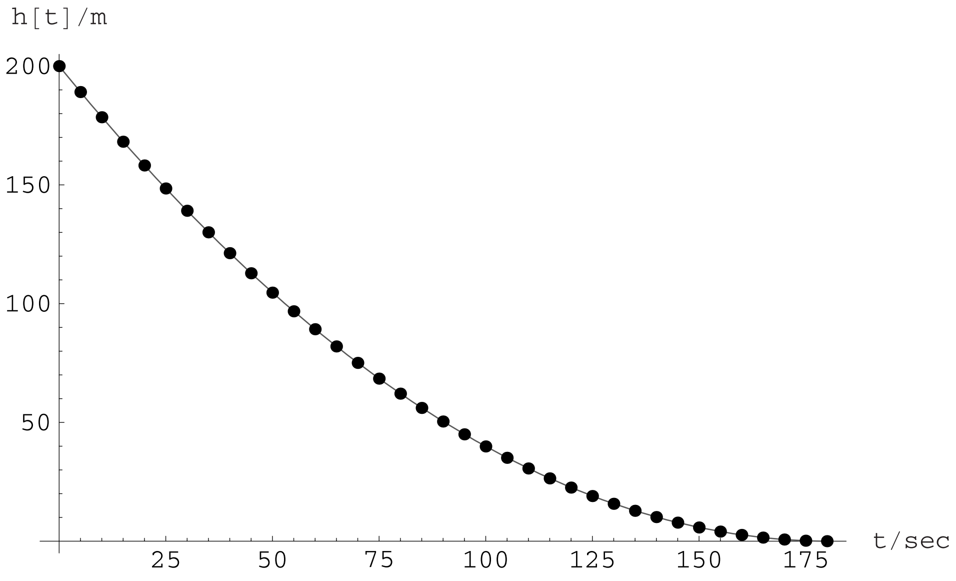
Out[20]=

$$\mathrm{ho}\,(1 - \frac{\sqrt{2}\,\mathrm{Ao\,b}\sqrt{\mathrm{g}}\mathrm{t}}{\mathrm{A}\sqrt{\mathrm{ho}}} + \frac{\mathrm{Ao}^2\,\mathrm{b}^2\,\mathrm{g\,t}^2}{2\mathrm{A}^2\,\mathrm{ho}})$$

This can be factored because the coefficient of **t** is two times the square root of the coefficient of t^2. Therefore, we find:

$$\text{h[t]} = \text{ho}\left[1 - \frac{\text{Ao}\, b\sqrt{g}\, t}{A\sqrt{2\,\text{ho}}}\right]^2$$

A comparison of the function to the data points gives an excellent fit:



3.5 Mass Input, Output, and Control

Mass Input and Output

Constant Input

The next logical problem to consider is that of a vessel with a specified mass flow in and gravity mass flow out. In a very real sense the accumulation term is the response to the interaction of these two flows. Remember also that the problem we are doing is not only a real one, but it is also easily extended and modified for other problems that are seemingly unrelated. We use this problem simply to illustrate the principles and that is its real value. The physical picture is shown in Figure 2.

The material balance equation has two terms on the right-hand side.

$$\frac{dm(t)}{dt} = \mathring{m}_{\text{in}} - \mathring{m}_{\text{out}}$$

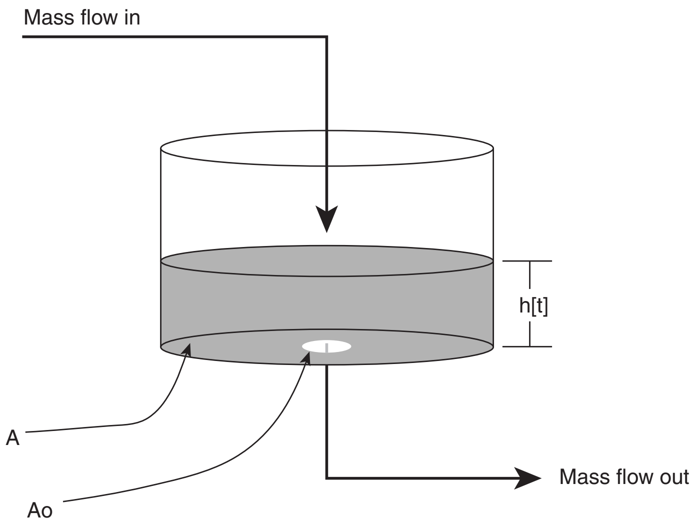


Figure 2

The mass flow term in can be taken as the product of the density of the fluid and its volume flow rate. The mass flow out can be specified by Torricelli's Law multiplied by the fluid density and the mass in the control volume is the product of the fluid density, the tank's cross-sectional area, and the level at any time **t**. This gives us the following equation:

$$\frac{d[\rho A h[t]]}{dt} = \rho \mathrm{q} - \rho \mathrm{b}\ \mathrm{Ao}\sqrt{2gh[t]}$$

Simplification provides us with:

$$\frac{dh[t]}{dt} = \frac{q - b\ \mathrm{Ao}\sqrt{2gh[t]}}{A}$$

This can be solved and we can examine the behavior of the time-dependent solution. We will begin the analysis assuming that the tank is initially empty, that is, that **h[0] = 0**, the initial condition. We will assume for now that **q** is a constant. This is also a good problem because we can approach the solution of this equation in several ways using *Mathematica*.

We can rearrange this by separation to find:

$$\frac{dh[t]}{q - b\ \mathrm{Ao}\sqrt{2gh[t]}} = \frac{dt}{A}$$

Solution by Direct Indefinite Integration after Separation

This in turn can be simplified to this form:

$$\frac{dy}{a - \mathrm{b1}\sqrt{y}} = \frac{dt}{A}$$

$$\mathrm{a} = \mathrm{q} \quad \text{and} \quad \mathrm{b1} = \mathrm{b\,Ao}\sqrt{2g}$$

Here we do the indefinite integration:

In[21]:= **Clear[a, b1, y, yo, t]**

$$\int \frac{1}{\mathbf{a-b1}\sqrt{\mathbf{y}}}\,d\mathbf{y} == \int \frac{1}{\mathbf{A}}\,d\mathbf{t}$$

Out[22]= $-\frac{2\sqrt{y}}{b1} - \frac{2a\,\mathrm{Log}[a - b1\sqrt{y}]}{b1^2} == \frac{t}{A}$

The result does not explicitly include the constant of integration; we add this and then evaluate it at $t = 0$ and $y = yo = 0$:

In[23]:= **yo = 0;**

$$\frac{-2\sqrt{\mathbf{yo}}}{\mathbf{b1}} - \frac{\mathbf{2a\,Log[a-b1}\sqrt{\mathbf{yo}}\mathbf{]}}{\mathbf{b1}^2} == \mathbf{C}$$

Out[24]= $-\frac{2a\,\mathrm{Log}[a]}{b1^2} == C$

In[25]:= $-\frac{2\sqrt{y}}{b1} - \frac{2a\,\mathrm{Log}[a - b1\sqrt{y}]}{b1^2} - \left(-\frac{2a\,\mathrm{Log}[a]}{b1^2}\right) == \frac{t}{A}$

Out[25]= $-\frac{2\sqrt{y}}{b1} + \frac{2a\,\mathrm{Log}[a]}{b1^2} - \frac{2a\,\mathrm{Log}[a - b1\sqrt{y}]}{b1^2} == \frac{t}{A}$

In[26]:= $-\frac{2\sqrt{y}}{b1} - \frac{2a\,\mathrm{Log}[a - b1\sqrt{y}]}{b1^2} - \left(-\frac{2a\,\mathrm{Log}[a]}{b1^2}\right) == \frac{t}{A}$ **// Simplify**

Out[26]= $-\frac{2(b1\sqrt{y} - a\,\mathrm{Log}[a] + a\,\mathrm{Log}[a - b1\sqrt{y}])}{b1^2} == \frac{t}{A}$

Now we can set this result up as function, apply parameter values, and plot it:

In[27]:= **s1[y_] := -A** $\frac{2(b1\sqrt{y} - a\,\mathrm{Log}[a] + a\,\mathrm{Log}[a - b1\sqrt{y}])}{b1^2}$

As we have an expression in y for t, we use this to compute the values of time corresponding to a set y-value. Thus the Table function that follows gives the value of $s1\,[t]$ first and then the set value of y.

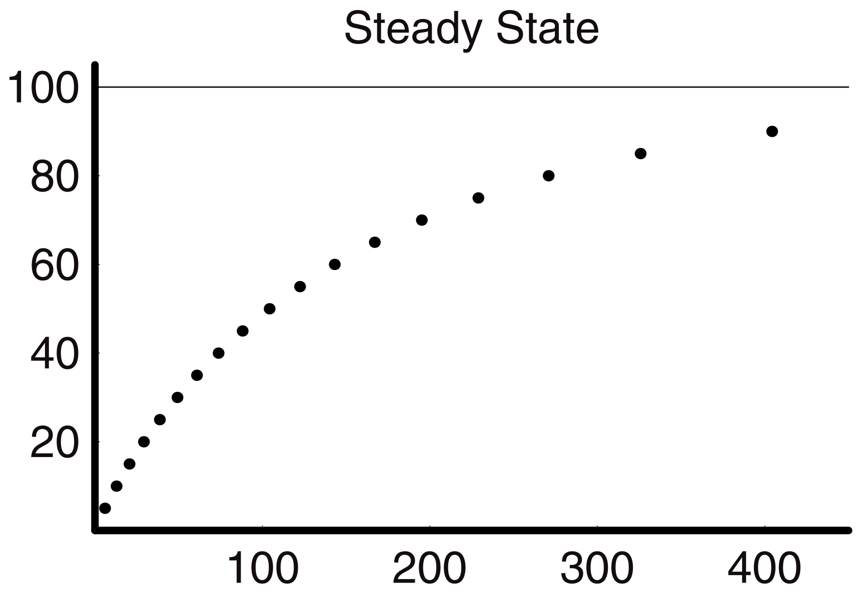
```
In[28]:= Clear[y]

In[29]:= a = 10;
         b1 = 1;
         A = 10;
         ts1 = Table[{s1[x] // N, x}, {x, 0, 200, 5}]

Out[32]= {{0., 0}, {5.89788, 5}, {12.7805, 10}, {20.5158, 15},
           {29.114, 20},{38.6294, 25}, {49.1474, 30}, {60.784, 35},
           {73.6911, 40}, {88.0663, 45}, {104.168, 50}, {122.341, 55},
           {143.053, 60}, {166.967, 65}, {195.052, 70}, {228.816, 75},
           {270.785, 80}, {325.702, 85}, {404.211, 90}, {540.292, 95},
           {∞, 100}, {535.291 - 628.319ⅈ, 105}, {394.207 - 628.319ⅈ, 110},
           {310.687 - 628.319ⅈ, 115}, {250.752 - 628.319ⅈ, 120},
           {203.75 - 628.319ⅈ, 125}, {164.937 - 628.319ⅈ, 130},
           {131.782 - 628.319ⅈ, 135}, {102.775 - 628.319ⅈ, 140},
           {76.9389 - 628.319ⅈ, 145}, {53.6089 - 628.319ⅈ, 150},
           {32.3096 - 628.319ⅈ, 155}, {12.69 - 628.319ⅈ, 160},
           {-5.5166 - 628.319ⅈ, 165}, {-22.5176 - 628.319ⅈ, 170},
           {-38.4775 - 628.319ⅈ, 175}, {-53.5291 - 628.319ⅈ, 180},
           {-67.7808 - 628.319ⅈ, 185}, {-81.3229 - 628.319ⅈ, 190},
           {-94.2306 - 628.319ⅈ, 195}, {-106.568 - 628.319ⅈ, 200}}
```

It is very interesting to note that the numerical values of time are real until we reach a value of 100 and then they depart for the complex plane. We can start to see why if we compute the steady-state value of *y* directly from the differential equation for these parameter values:

```
In[33]:= Solve[0 == (a - b1√ystst)/A, ystst]

Out[33]= {{ystst→100}}
```

Therefore we see that the integration solution we have obtained works up to the point of steady state, but not beyond. We plot the real values of this solution to see how the solution behaves:

```
In[34]:= pl1 = ListPlot[Take[ts1, 20],
             PlotLabel → "Steady State",
             Epilog → Line[{{0, 100}, {450, 100}}],
             PlotRange → {{0, 450}, {0, 105}}];
```



So this solution approaches the correct steady state value but we do not know if the time-dependence is correct. We might wonder how and where this solution was derived.

Solution by Substitution

We can try the old method of substitution to solve this problem. We know that the integral of $\frac{1}{a+bx}$ over x gives a simple logarithmic form. Therefore, we can try using the substitution of $\mathrm{u} = \sqrt{y}$:

```
In[35]:= u = √y;
         D[u, y]
```

Out[36]= $\frac{1}{2\sqrt{\mathrm{y}}}$

From this we know that $\mathrm{dy} = 2\sqrt{y}\,\mathrm{du} = 2\,\mathrm{u}\,\mathrm{du}$. We can transform the integral as follows:

$$\int \frac{1}{\mathrm{a} - \mathrm{b1}\sqrt{\mathrm{y}}}\,d\mathrm{y} == \int \frac{2u}{a - \mathrm{b1}u}\,du == \int \frac{1}{\mathrm{A}}\,d\mathrm{t}$$

```
In[37]:= Clear[u, a, b1, A]
```

In[38]:= $\int \frac{\mathbf{2u}}{\mathbf{a - b1u}}\,d\mathbf{u} == \int \frac{\mathbf{1}}{\mathbf{A}}\,d\mathbf{t}$

Out[38]= $2\left(-\frac{\mathrm{u}}{\mathrm{b1}} - \frac{\mathrm{a\,Log[a - b1\,u]}}{\mathrm{b1}^2}\right) == \frac{\mathrm{t}}{\mathrm{A}}$

To this we must add the constant of integration and then we need to evaluate this at the initial condition, which can be after substitution of u:

```
In[39]:= Clear[y]
In[40]:= u = √y;
```

In[41]:= $2\left(\frac{-u}{b1}-\frac{a\,\text{Log}[a - b1\,u]}{b1^2}\right) + C == \frac{t}{A}$

Out[41]= $C + 2\left(-\frac{\sqrt{y}}{b1}-\frac{a\,\text{Log}[a - b1\sqrt{y}]}{b1^2}\right) == \frac{t}{A}$

```
In[42]:= t = 0;
         y = yo;
```

Solve[$C + 2\left(-\frac{\sqrt{y}}{b1}-\frac{a\,\text{Log}[a - b1\sqrt{y}]}{b1^2}\right) == \frac{t}{A}$, C]

Out[44]= $\{\{C \to \frac{2a\,\text{Log}[a]}{b1^2}\}\}$

In[45]:= yo = 0;

$\frac{2(b1\sqrt{yo} + a\,\text{Log}[a - b1\sqrt{yo}])}{b1^2}$

Out[46]= $\frac{2a\,\text{Log}[a]}{b1^2}$

Adding this to the previous general solution we obtain:

$$\frac{2a\,\text{Log}[a]}{b1^2}+2\left(-\frac{\sqrt{y}}{b1}-\frac{a\,\text{Log}[a-b1\sqrt{y}]}{b1^2}\right)==\frac{t}{A}$$

We can bring this over just $b1^2$ to give:

```
In[47]:= yo =.
         y =.
```

Together[$\frac{2a\,\text{Log}[a]}{b1^2} + 2\left(-\frac{\sqrt{y}}{b1}-\frac{a\,\text{Log}[a - b1\sqrt{y}]}{b1^2}\right)$] // Simplify

Out[49]= $-\frac{2(b1\sqrt{y} - a\,\text{Log}[a] + a\,\text{Log}[a - b1\sqrt{y}])}{b1^2}$

This is exactly the same solution that we had obtained earlier from the *Mathematica* direct indefinite integration.

$$-A\frac{2(b1\sqrt{y}-a\,\text{Log}[a]+a\,\text{Log}[a-b1\sqrt{y}])}{b1^2}==\frac{t}{A}$$

Power Series Expansion

At this point we could ask the question "Why not just do the integration over definite limits, that is, from yo = 0 or even y = yo at t = 0 to y at t?" The answer for both the unsubstituted and the substituted cases is shown here:

```
In[50]:= Clear[a, b1, t, A]

In[51]:= u =.
         t =.
```

$$\int_0^y \frac{1}{a - b1\sqrt{y}}\,dy == \int_0^t \frac{1}{A}\,dt$$

$$\int_0^u \frac{2u}{a - b1\,u}\,du == \int_0^t \frac{1}{A}\,dt$$

```
         Series::vcnt : Center point -y of power series expansion
          involves the variable y.

         Series::vcnt : Center point -y of power series expansion
          involves the variable y.

         Series::vcnt : Center point -y of power series expansion
          involves the variable y.

         General::stop : Further output of Series::vcnt will be
          suppressed during this calculation.
```

Out[53]= $\int_0^y \frac{1}{a - b1\sqrt{y}}\,dy == \frac{t}{A}$

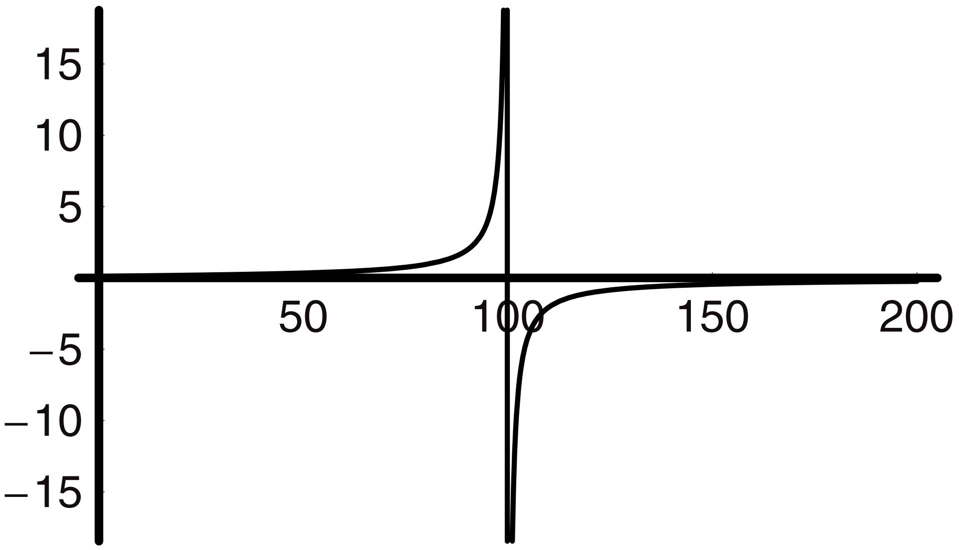
```
         Series::vcnt : Center point -u of power series expansion
          involves the variable u.

         Series::vcnt : Center point -u of power series expansion
          involves the variable u.

         Series::vcnt : Center point -u of power series expansion
          involves the variable u.

         General::stop : "Further output of Series::vcnt will be
          suppressed during this calculation.
```

Out[54]= $2\int_0^u \frac{u}{a - b1\,u}\,du == \frac{t}{A}$

In both cases we find that the error message is the same: "Center point −y of power series expansion involves the variable y." This tells us why it failed, but it also gives us a direct clue as to how *Mathematica* is solving this integral—it is using a power series expansion of the integrand. Since the indefinite integration works, it may also be using this method or the substitution method. We plot the integrand to remind ourselves what is happening and why there is a problem:

In[55]:= **f[y_] :=** $\frac{1}{a - b1\sqrt{y}}$

```
a = 10;
b1 = 1;
A = 10;

Plot[f[y], {y, 0, 200}];

f[100]
```



Power::infy : Infinite expression $\frac{1}{0}$ encountered.

Out[60]= ComplexInfinity

Right—the integrand goes to complex infinity in the vicinity of y = 100, that is, when the numerical value of b1$\sqrt{y}$ is the same as a! This causes some difficulties in the integration. We can now turn to the power series expansion of the integrand. We can do this as follows out to terms of any order *n*; in this case we choose to go out to order 3:

In[61]:= **Series[**$\frac{1}{a - b1\sqrt{y}}$**, {y, 0, 3}]**

Out[61]= $\frac{1}{10} + \frac{\sqrt{y}}{100} + \frac{y}{1000} + \frac{y^{3/2}}{10000} + \frac{y^2}{100000} + \frac{y^{5/2}}{1000000} + \frac{y^3}{10000000} + O[y]^{7/2}$

Now we can use this series to do the integration:

```
In[62]:= ∫ Series[1/(a - b1√y), {y, 0, 3}] ⅆy
```

$$\text{Out[62]=}\ \frac{y}{10}+\frac{y^{3/2}}{150}+\frac{y^2}{2000}+\frac{y^{5/2}}{25000}+\frac{y^3}{300000}+\frac{y^{7/2}}{3500000}+\frac{y^4}{40000000}+O[y]^{9/2}$$

How does this relate to the solution that we obtained via the direct integration of the original function? We know that solution, so we will test it against this new solution:

$$-\frac{2\,(\mathrm{b1}\sqrt{y}-a\,\mathrm{Log}[a]+a\,\mathrm{Log}\,[a-\mathrm{b1}\sqrt{y}])}{\mathrm{b1}^2}$$

To do so we can expand the log in a power series about zero and subtract it from the $-\frac{2\sqrt{y}}{b1}$ term:

```
In[63]:= -2 (b1√y - a Log[a] + a Series[Log[a - b1√y], {y, 0, 4}]) / b1^2
```

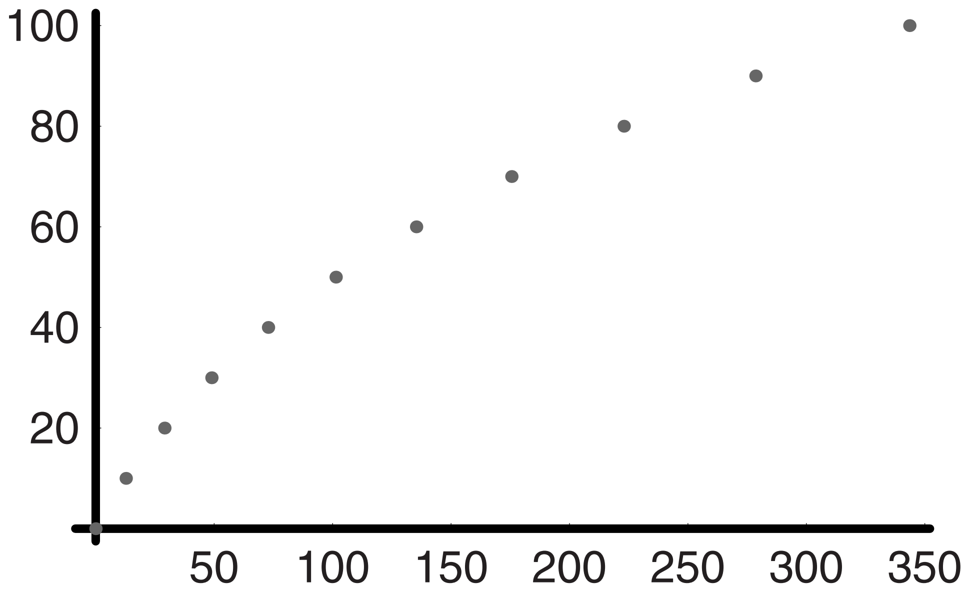
$$\text{Out[63]=}\ \frac{y}{10}+\frac{y^{3/2}}{150}+\frac{y^2}{2000}+\frac{y^{5/2}}{25000}+\frac{y^3}{300000}+\frac{y^{7/2}}{3500000}+\frac{y^4}{40000000}+O[y]^{9/2}$$

Aha! It is clear that these are the same solution. Thus, *Mathematica* found the solution in terms of the power series and then recognized that this could be written as a difference including the log function of the argument!

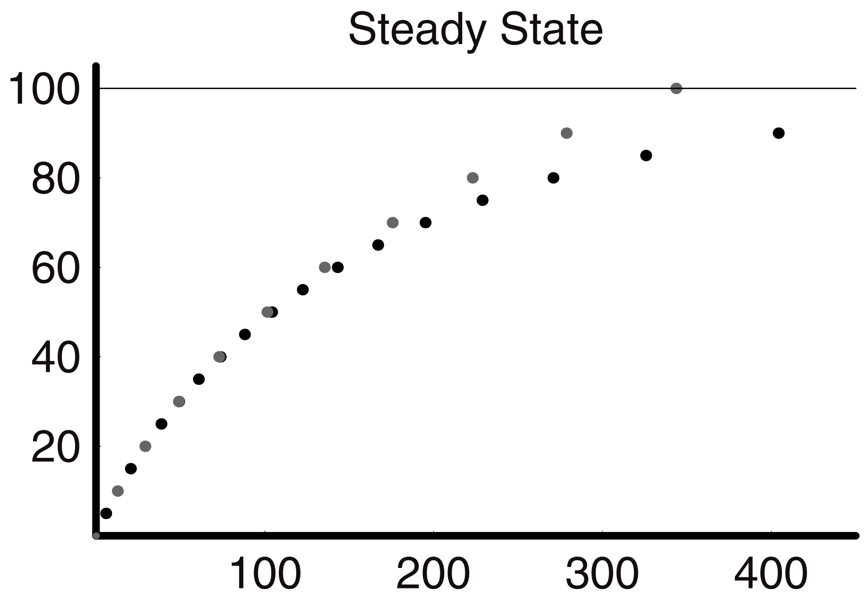
We can now evaluate this power series result and compare it to the result we obtained from the closed form solution. That is to say, if we had not recognized, as *Mathematica* did, that the power series solution could be recast as a log, then we might have simply used the solution we had. Let's compare this new solution with the previous one by making a function of it, evaluating *t* at each *y* and then plotting it against the previous results:

$$\text{In[64]:=}\ \mathrm{s2[y\_]:=A}\left(\frac{y}{a}+\frac{2\,\mathrm{b1}\,y^{3/2}}{3a^2}+\frac{\mathrm{b1}^2\,y^2}{2a^3}+\frac{2\,\mathrm{b1}^3\,y^{5/2}}{5a^4}+\frac{\mathrm{b1}^4\,y^3}{3a^5}+\frac{2\,\mathrm{b1}^5\,y^{7/2}}{7a^6}+\frac{\mathrm{b1}^6\,y^4}{4a^7}\right)$$

```
In[65]:= a = 10;
         b1 = 1;
         A = 10;

In[68]:= tst2 = Table[{s2[x] // N, x}, {x, 0, 100, 10}]

Out[68]= {{0., 0}, {12.7795, 10}, {29.0873, 20}, {48.9513, 30},
          {72.843, 40}, {101.396, 50}, {135.358, 60}, {175.578, 70},
          {222.991, 80}, {278.621, 90}, {343.571, 100}}

In[69]:= pl2 = ListPlot[tst2, PlotStyle →
           {PointSize[0.015], GrayLevel[0.4]}];
```



```
In[70]:= Show[{pl1, pl2}, DisplayFunction → $DisplayFunction];
```



The results indicate that the fourth-order approximation of the integral does follow the closed from solution rather well for about 60% of the steady-state value, and then it deviates and does so markedly. Notice also that it does not have an upper bound; its values go right through the steady state. How can we account for this? Recall that we approximated the integrand with a power series of order $n = 3$. In so doing we dropped the higher-order terms. This has to lead to numerical errors. Clearly, when *Mathematica* numerically evaluates any log function, however it actually does it, it does so in a fashion that is far more accurate than of order 3 accuracy. We could go to higher order, reintegrate, and see if the agreement is better. We do that now:

In[71]:= **Clear[a, b1, t, A]**

$$\int \mathbf{Series}\left[\frac{1}{a - b1\sqrt{y}}, \{y, 0, 10\}\right] dy$$

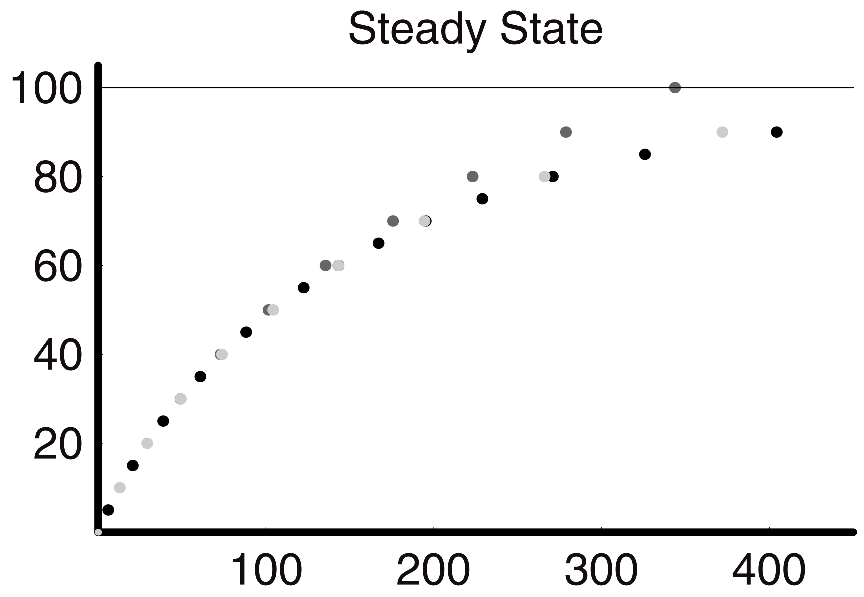
Out[72]=

$$\frac{y}{a}+\frac{2b1y^{3/2}}{3a^2}+\frac{b1^2y^2}{2a^3}+\frac{2b1^3y^{5/2}}{5a^4}+\frac{b1^4y^3}{3a^5}+\frac{2b1^5y^{7/2}}{7a^6}+\frac{b1^6y^4}{4a^7}+\frac{2b1^7y^{9/2}}{9a^8}$$
$$+\frac{b1^8y^5}{5a^9}+\frac{2b1^9y^{11/2}}{11a^{10}}+\frac{b1^{10}y^6}{6a^{11}}+\frac{2b1^{11}y^{13/2}}{13a^{12}}+\frac{b1^{12}y^7}{7a^{13}}+\frac{2b1^{13}y^{15/2}}{15a^{14}}$$
$$+\frac{b1^{14}y^8}{8a^{15}}+\frac{2b1^{15}y^{17/2}}{17a^{16}}+\frac{b1^{16}y^9}{9a^{17}}+\frac{2b1^{17}y^{19/2}}{19a^{18}}+\frac{b1^{18}y^{10}}{10a^{19}}+\frac{2b1^{19}y^{21/2}}{21a^{20}}$$
$$+\frac{b1^{20}y^{11}}{11a^{21}}+O[y]^{23/2}$$

In[73]:= **s3[y_] := A(**

$$\frac{y}{a}+\frac{2b1y^{3/2}}{3a^2}+\frac{b1^2y^2}{2a^3}+\frac{2b1^3y^{5/2}}{5a^4}+\frac{b1^4y^3}{3a^5}+\frac{2b1^5y^{7/2}}{7a^6}+\frac{b1^6y^4}{4a^7}$$
$$+\frac{2b1^7y^{9/2}}{9a^8}+\frac{b1^8y^5}{5a^9}+\frac{2b1^9y^{11/2}}{11a^{10}}+\frac{b1^{10}y^6}{6a^{11}}+\frac{2b1^{11}y^{13/2}}{13a^{12}}+\frac{b1^{12}y^7}{7a^{13}}$$
$$+\frac{2b1^{13}y^{15/2}}{15a^{14}}+\frac{b1^{14}y^8}{8a^{15}}+\frac{2b1^{15}y^{17/2}}{17a^{16}}+\frac{b1^{16}y^9}{9a^{17}}+\frac{2b1^{17}y^{19/2}}{19a^{18}}+\frac{b1^{18}y^{10}}{10a^{19}}$$
$$+\frac{2b1^{19}y^{21/2}}{21a^{20}}+\frac{b1^{20}y^{11}}{11a^{21}})$$

```
In[74]:= a = 10;
         b1 = 1;
         A = 10;

         tst3 = Table[{s3[x] // N, x}, {x, 0, 100, 10}]

Out[77]= {{0., 0}, {12.7805, 10}, {29.114, 20}, {49.1474, 30},
          {73.6905, 40}, {104.159, 50}, {142.957, 60}, {194.309, 70},
          {265.875, 80}, {371.793, 90}, {538.163, 100}}

In[78]:= p13 = ListPlot[tst3,
             PlotStyle →{PointSize[0.015],
             GrayLevel[0.8]}, DisplayFunction →Identity];

         Show[{p11, p12, p13}, DisplayFunction → $DisplayFunction];
```



Clearly, by going out to more terms, that is, to terms on the order of $n = 10$, the accuracy is much better and comes closer than that which we had for the analytical solution.

Solution with DSolve—the Differential Equation Solver

Mathematica also provides us with the differential equation solver **DSolve**, which can be employed for this problem. When we do this we do not have to work quite as much as we did using the integration methods. The solution looks as follows:

```
In[80]:= Remove[a, b1, A]

         soln = DSolve[{y'[t] == (a - b1√y[t])/A, y[0] == 0},
          y[t], t] // Simplify

         s4[t_] := Evaluate[y[t] /. soln]

         InverseFunction::ifun : Inverse functions are being used.
          Values may be lost for multivalued inverses.

         Solve::ifun : Inverse functions are being used by Solve,
          so some solutions may not be found.

         Solve::ifun : Inverse functions are being used by Solve,
          so some solutions may not be found.
```

$$\text{Out[81]= }\{y[t] \to \frac{a^2\left(1 + \text{ProductLog}\left[-\frac{\sqrt{a^2}\,e^{-1-\frac{b1^2 t}{2aA}}}{a}\right]\right)^2}{b1^2}\}$$

Now we have a solution that is different from any of the others that we have derived so far. In fact, the solution depends upon a new function that is unfamiliar in name—the ProductLog function. The **Help Browser** tells us that the **ProductLog[z]** is the principal solution to the equation $z = we^{w}$. We should test this solution to be sure that it is one that satisfies the original equation. We can do so as follows. We have specified the solution as the function s4 [t]. Taking the derivative of this function, we should obtain the same result as we acquire when we put this function into the right-hand side of the equation and simplify. Therefore, we set the derivative with respect to time equal to the right-hand side after substitution. To be efficient we **Simplify**, **PowerExpand**, and **Simplify** again using the // Command structure. If you need to see what is happening here, redo the derivative and the right-hand side without these additional commands, and then take the results and apply them sequentially to reach the same final forms.

```
In[83]:= Clear[a, A, b1]

In[84]:= lhs = ∂t s4[t] // Simplify // PowerExpand // Simplify
```

$$\texttt{rhs} = \frac{\texttt{a - b1}\sqrt{\texttt{s4[t]}}}{\texttt{A}} \texttt{ // Simplify // PowerExpand // Simplify}$$

```
         lhs == rhs
```

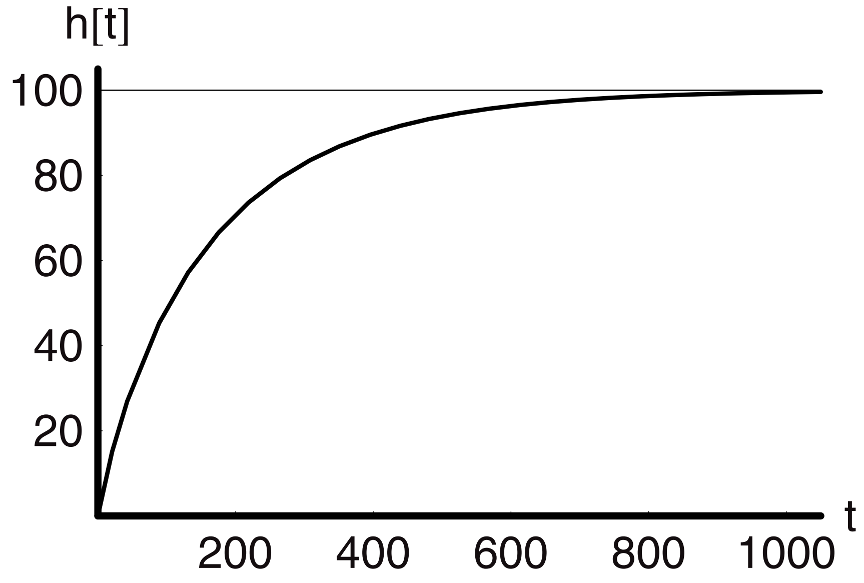
$$\texttt{Out[84]} = -\frac{a\,\text{ProductLog}[-e^{-1-\frac{b1^2 t}{2aA}}]}{A}$$

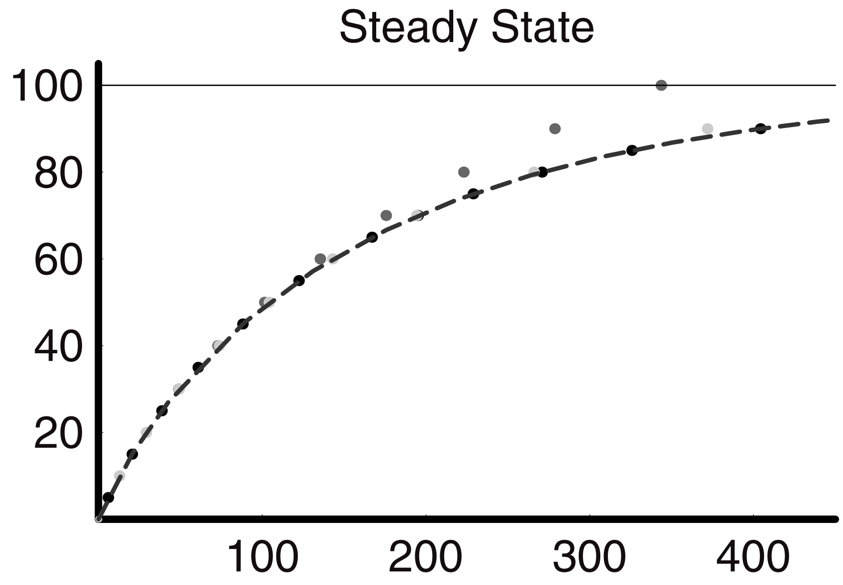
$$\texttt{Out[85]} = -\frac{a\text{ProductLog}[-e^{-1-\frac{b1^2 t}{2aA}}]}{A}$$

```
Out[86] = True
```

Therefore, we can be sure that this **ProductLog** function is a full-time solution to the equation. The next step in our analysis then should be to compare this solution's behavior in time with the previous solutions using the same parameter values. We do this as follows:

```
In[87]:=a = 10;
        b1 = 1;
        A = 10;

        Plot[s4[t], {t, 0, 1050},
          Epilog → {Line[{{0, 100}, {1050, 100}}]},
          PlotRange → {{0, 1050}, {0, 105}},
          AxesLabel → {"t", "h[t]"}];

        p14 = Plot[s4[t], {t, 0, 1050},
            PlotStyle → {Thickness[0.006], Dashing[{0.025, 0.015}],
            GrayLevel[0.2]}, DisplayFunction → Identity];

        Show[{p11, p12, p13, p14}, DisplayFunction →$DisplayFunction];
```





Here we can see that the solution provided by **DSolve** is in fact one that is a full time-dependent solution. This solution rises in time according to the same dependence of the log function we had obtained earlier, but it also approaches the limiting level asymptotically and, therefore, correctly.

If we leave all else the same but begin the process with either a higher or a lower volume flow rate, what will be the result? Will the steady-state position change? We can see the answer immediately by using the steady-state solution. Choosing a higher or lower flow rate will markedly affect the position of the steady state, as this level depends on the square of the flow rate in.

Fluxional Input

We can now ask what the effect of fluxional input would be. Suppose, for example, that the input were sinusoidal as we saw in Chapter 1: What would the output look like given a gravity-driven flow response? The virtue of *Mathematica* is that we can solve this problem with very little effort beyond what we have already done and we can compare the results with those from constant input. Here is how we do it.

First, solve the new differential equation taking **q[t]** for the input as **qo (1 + α Sin[β t]):**

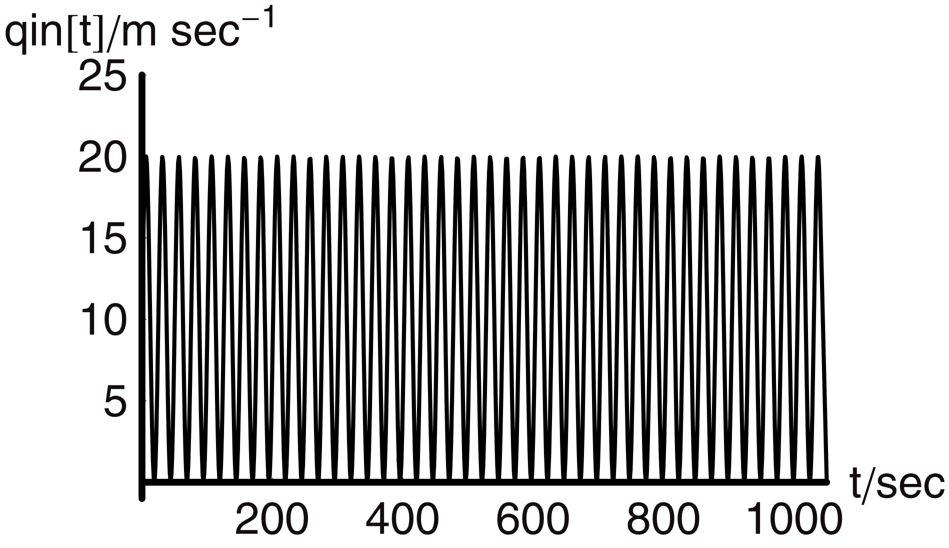
$$\frac{dh[t]}{dt} = \frac{\text{qo}(1+\alpha\,\text{Sin}[\beta t]) - b\,\text{Ao}\sqrt{2gh[t]}}{A}$$

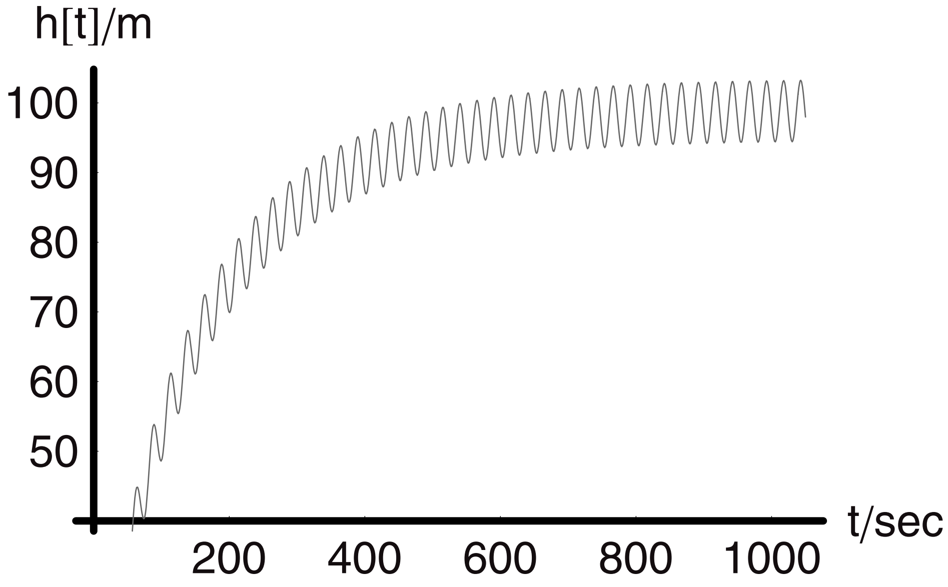
This can be solved numerically for a specific value of the parameters. We then evaluate this new definition of **h[t]** and call it **hsin[t]** to distinguish it from the earlier work we have done. We make similar changes to the plot names. The value for **qo** is taken as 20 in order to be the same as the constant flow case. The magnitudes of α and β are taken to be the same as they were in Chapter 1.

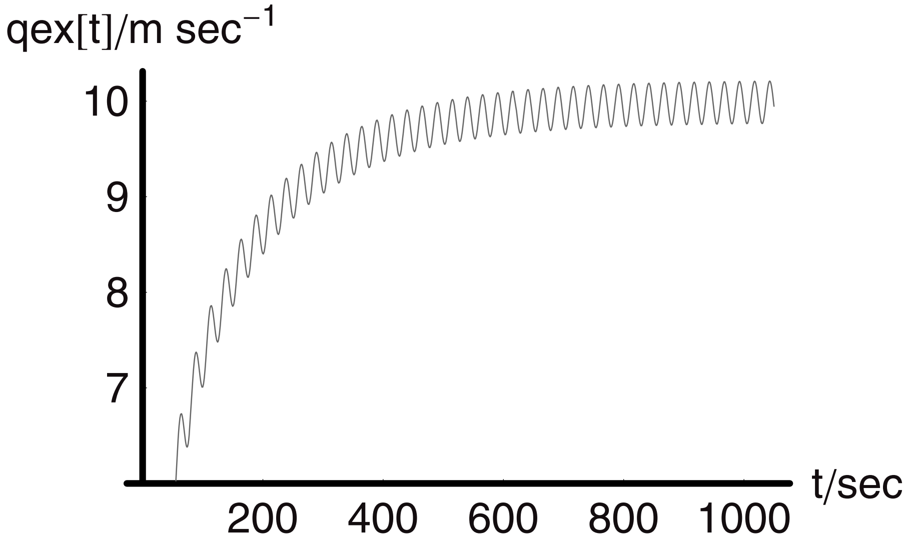
```
In[93]:= Clear[α, β, g, q, qo, b, Ao, A, h]

In[94]:= r = 1.7;
         A = N[πr^2];
         Ao = 0.1 A;
         b = .25;
         g = 9.80;
         qo = 10;
         α  = 1.;
         β  = .25;
         b Ao √(2g);
         tmax = 1050;

In[104]:=soln0 = NDSolve[
          {∂t h[t] == (qo(1 + α Sin[βt]) - b Ao √(2gh[t])) / A ,
           h[0] == 0}, h[t], {t, 0, tmax}];

         hsin[t_] := Evaluate[h[t] /. soln0]

         qin[t] := qo (1 +  α Sin[βt])

         qexsin[t_] := b Ao √(2g hsin[t])

         plqsinin = Plot[qin[t], {t, 0, tmax},
          AxesLabel → {"t/sec", "qin[t]/m sec^-1"},
          PlotRange →{{0, tmax}, {-1, 25}}];
```

```
plhsin = Plot[hsin[t], {t, 0, tmax},
 AxesLabel → {"t/sec", "h[t]/m"},
 PlotStyle → GrayLevel[0.4]];
```

```
plqexsin = Plot[qexsin[t], {t, 0, tmax},
 PlotStyle → GrayLevel[0.4],
 AxesLabel → {"t/sec", "qex[t]/m sec^-1"}];
```

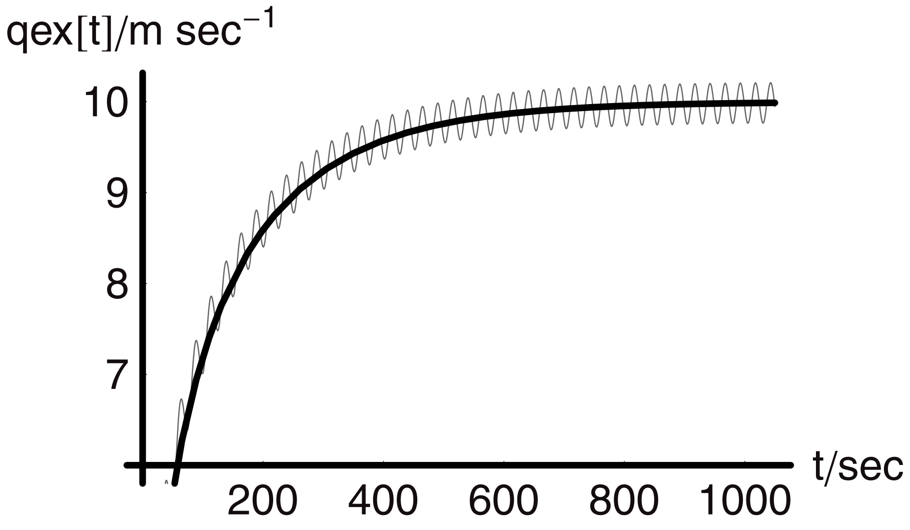


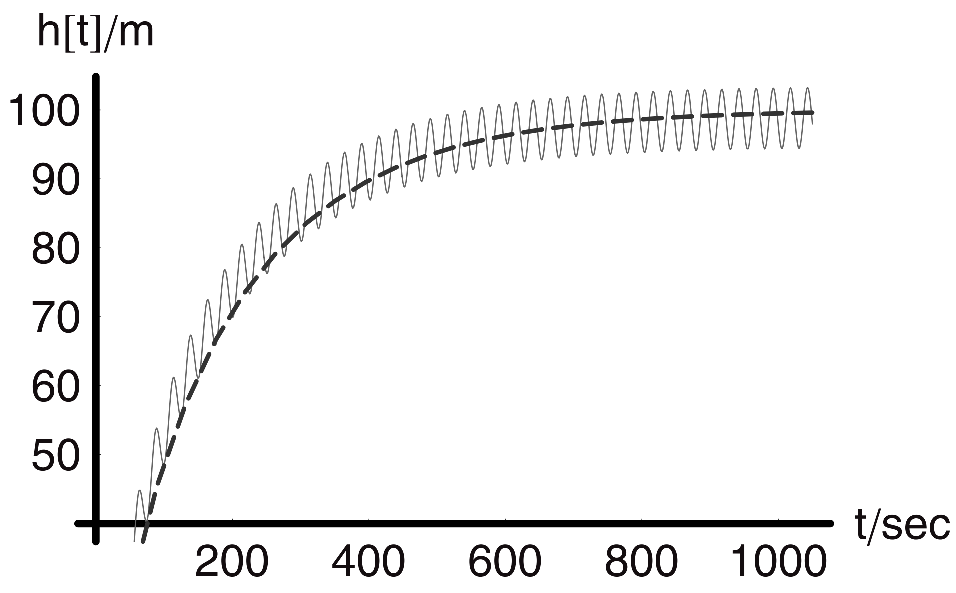




The results show that the output flow is coupled to the input flow quite tightly after an initial transient period. We can compare these responses to the sinusoidal input with those from the constant input case by plotting **qex[t]** and **h[t]** for both cases:

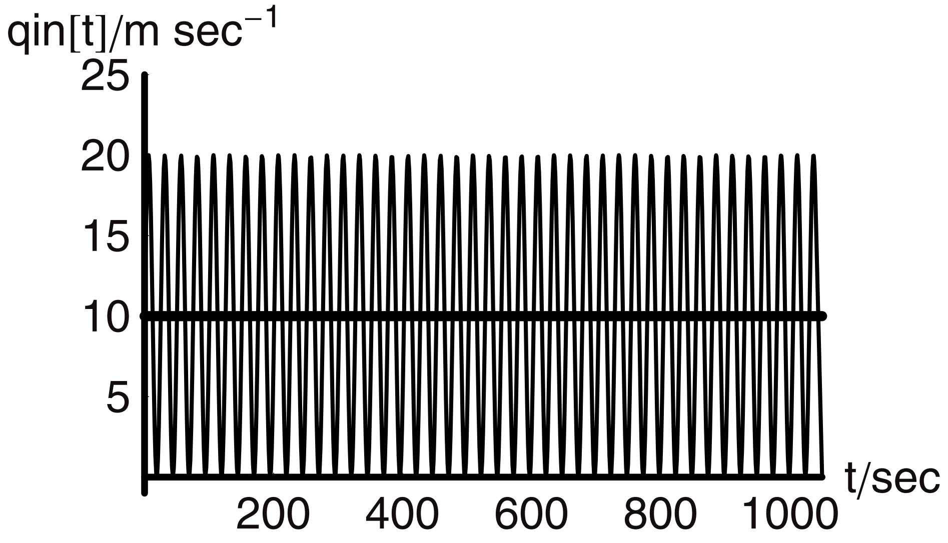
```
In[111]:= b1 = 1;
          a = 10;
          Plot[b1√s4[t], {t, 0, tmax}, PlotStyle → Thickness[0.01],
           DisplayFunction → Identity];
          Show[plqexsin, %];
          Show[plhsin, pl4];
```





Since we chose all the parameters to be the same, we note the fluxional values of **h** and **qex** for the sinusoidal input case are larger earlier than they are for the fixed input. The reason is that the input flow in the sinusoidal case rises to a value well over its average value.

```
In[116]:= Show[plqsinin, Graphics[{Thickness[0.015], Line[{{0, qo},
           {1100, qo}}]}]];
```



3.6 Control

It would be very nice to be able to dampen the input fluctuations and to smooth the output from this vessel. To do this requires a control function. One form of control would be to increase the flow rate out of the tank whenever the level in the tank rises or falls above or below a designed set point level h_d. For example, the set point level could be the steady-state level that we found from the earlier example with constant input flow, which is the also the bold black horizontal line in level graph above. To increase the flow rate in the case of gravity-driven flow, we must increase the size of the orifice. We can increase it in proportion to the difference between the actual level in the tank at any time and design level. The actual implementation would involve having a level sensor tied to an actuator, which would open the valve more or less depending on the level. The mathematical description of this control function can be given as:

$$\mathrm{Ao[h[t]] = Aoo(1 - K(hd - h[t]))}$$

In this expression **Aoo** is the nominal aperture size to deliver at the design flow rate based on the constant set input flow rate. The second term in the parenthetical expression is the product of a proportionality constant **K** and the difference between the set point level and the actual level as a function of time. We substitute this for **Ao** in Torricelli's Law and also in the equation describing a system with sinusoidally fluctuating input flow:

$$\frac{dh[t]}{dt} = \frac{\mathrm{qo}\,(1+\alpha\,\mathrm{Sin}[\beta t]) - b\,\mathrm{Aoo}\,(1 - K(\mathrm{hd} - h[t]))\sqrt{2gh[t]}}{A}$$

Notice that if **K** were set to zero, the equation would revert back to that which we have already solved for the uncontrolled system. We can operate on the right-hand side to put it into a form that is more readily understood:

```
In[117]:=Clear[α, β, g, q, qo, b, Ao, A, h, K, hd, Aoo]
```

In[118]:=Collect[Simplify[PowerExpand[
$$\frac{\mathrm{qo(1 + \alpha\,Sin[\beta t]) - }b\,\mathrm{Aoo(1 - }K\mathrm{(hd - }h\mathrm{[}t\mathrm{]))}\sqrt{2gh[t]}}{\mathrm{A}}$$
]], qo]

Out[118]=
$$-\frac{\sqrt{2}\,\mathrm{Aoo\,b}\sqrt{\mathrm{g}}\sqrt{\mathrm{h[t]}}\,\mathrm{(1 - hd\,K + Kh[t])}}{\mathrm{A}} + \frac{\mathrm{qo(1 + \alpha\,Sin[t\beta])}}{\mathrm{A}}$$

In[119]:=
$$-\frac{\sqrt{2}\,\mathrm{Aoo\ b}\sqrt{\mathrm{g}}\sqrt{\mathrm{h[t]}}\,\mathrm{Collect[(1 - hd\,K + Kh[t]),\ K]}}{\mathrm{A}}$$

Out[119]=
$$-\frac{\sqrt{2}\,\mathrm{Aoo\ b}\sqrt{\mathrm{g}}\sqrt{\mathrm{h[t]}}\,\mathrm{(1 + K(-hd + h[t]))}}{\mathrm{A}}$$

From this form of the equation we can see that we have one function of time alone and another that is a function of level h[t]:

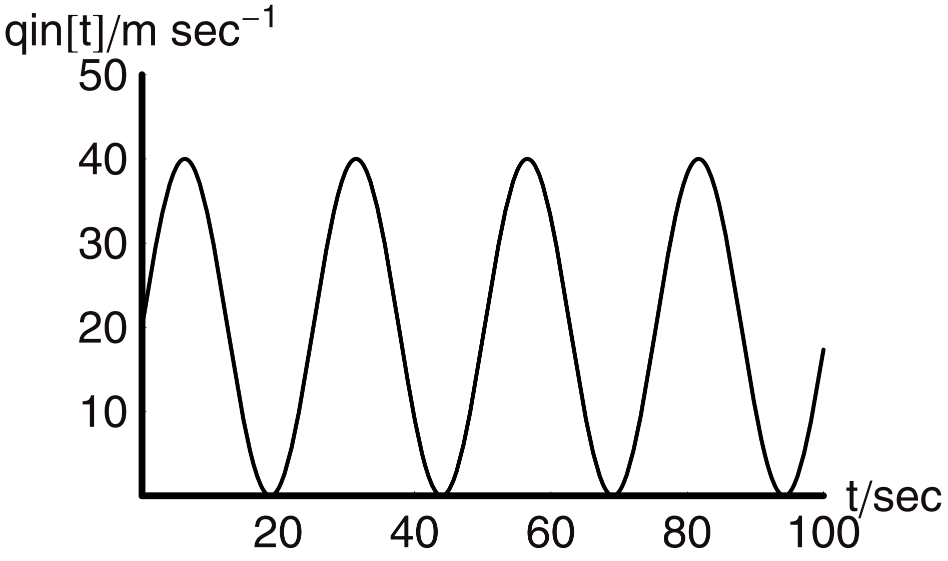
$$\frac{dh[t]}{dt} = -\frac{\sqrt{2}\,\mathrm{Aoo}\,\mathrm{b}\sqrt{\mathrm{g}}\sqrt{\mathrm{h[t]}}\,(1+\mathrm{K}(\mathrm{h[t]}-\mathrm{hd}))}{\mathrm{A}} + \frac{\mathrm{qo}(1+\alpha\,\mathrm{Sin}\,[\mathrm{t}\beta])}{\mathrm{A}}$$
$$\frac{dh[t]}{dt} + \frac{\sqrt{2}\,\mathrm{Aoo}\,\mathrm{b}\sqrt{\mathrm{g}}\sqrt{\mathrm{h[t]}}\,(1+\mathrm{K}(\mathrm{h[t]}-\mathrm{hd}))}{\mathrm{A}} = \frac{\mathrm{qo}(1+\alpha\,\mathrm{Sin}\,[\mathrm{t}\beta])}{\mathrm{A}}$$

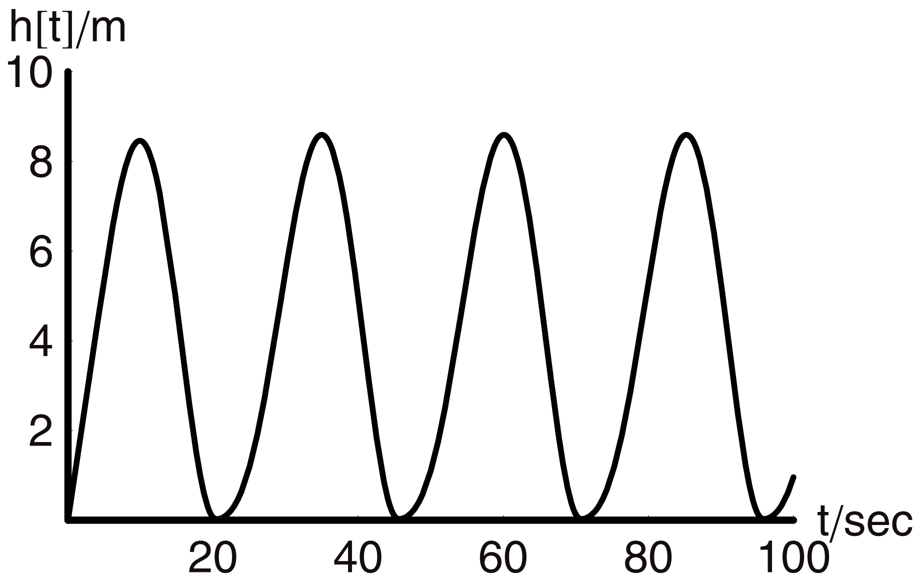
We will use all the same parameter values that we have used in the previous problem, but we will have to pick a magnitude for **K**. This is best done by solving the problem and then resetting the value to get a sense of the solution's parametric sensitivity to the magnitude of **K**. We start with a value of zero to be sure that the solution to this new equation reduces to that of the one we have solved already. See the following graphical illustrations.

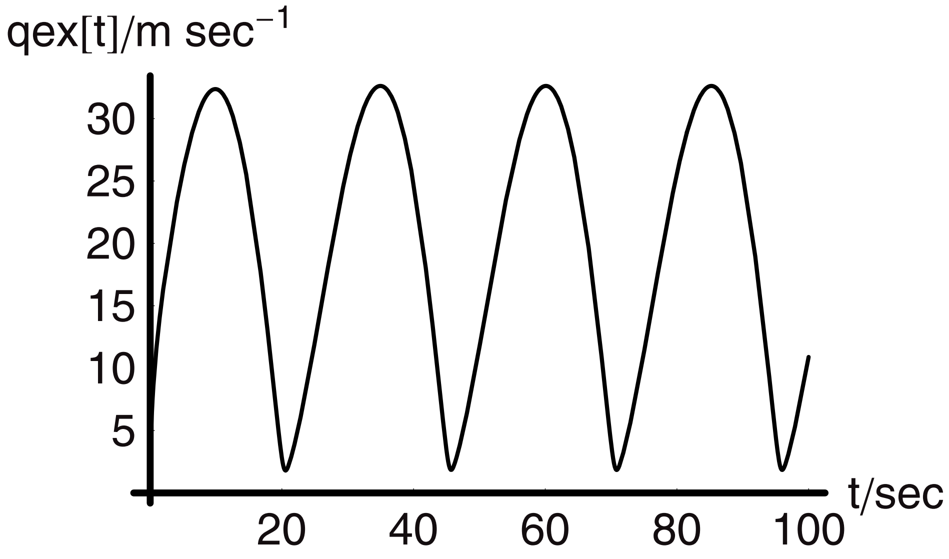
```
In[120]:= Clear[α, β, g, q, qo, b, Ao, A, h, K, hd, Aoo, t]

In[121]:= Ao =.
          r = 2;
          hd = 12;
          A = N[πr^2];
          Aoo = 0.2 A;

          b = 1;
          g = 9.80;
          qo = 20;
          α = 1;
          β = .25;
          K = 0;
          tmax = 100;
          soln1 = NDSolve[
           {hx'[t] == -(√2 Aoo b √g √hx[t] (1 + K(hx[t] - hd)))/A
            + (qo(1 + α Sin[tβ]))/A, hx[0] == 0}, hx[t], {t, 0, tmax}];

          hc[t_] := Evaluate[hx[t] /. soln1]

          qin[t] := qo(1 + αSin[βt])

          Ao[t_] := Aoo(1 + K(hd - hc[t]))

          qexc[t_] := b Ao[t] √(2ghc[t])

          plqcin = Plot[qin[t], {t, 0, tmax},
           AxesLabel → {"t/sec", "qin[t]/m sec^-1"},
           PlotRange → {{0, tmax}, {0, 50}}];
```

```
plhc0 = Plot[{hc[t]}, {t, 0, tmax},
 PlotStyle → Thickness[0.008],
 PlotRange → {{0, tmax}, {0, 10}},
 AxesLabel → {"t/sec", "h[t]/m"}];
```

```
plqexc0 = Plot[qexc[t], {t, 0, tmax},
 AxesLabel → {"t/sec", "qex[t]/m sec^-1"}];
```





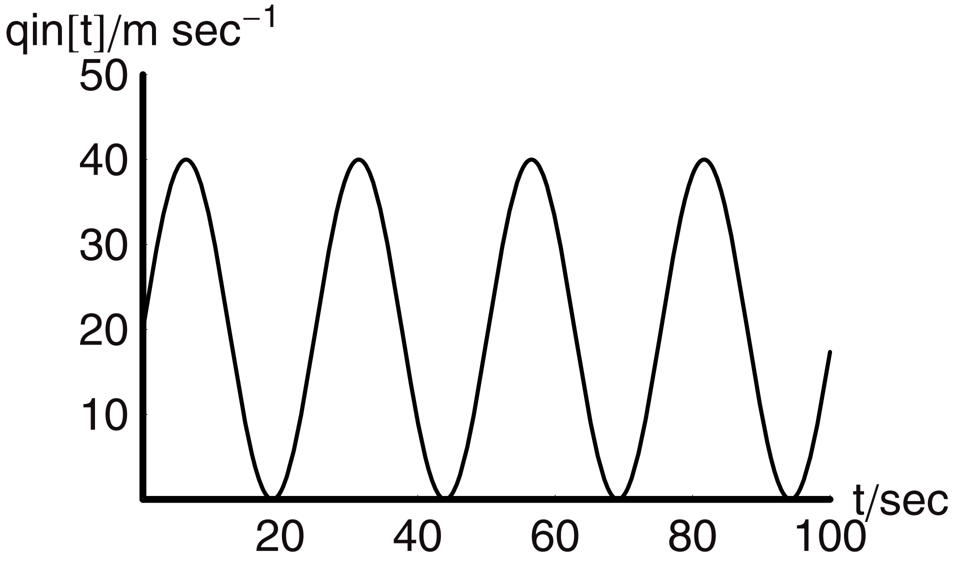


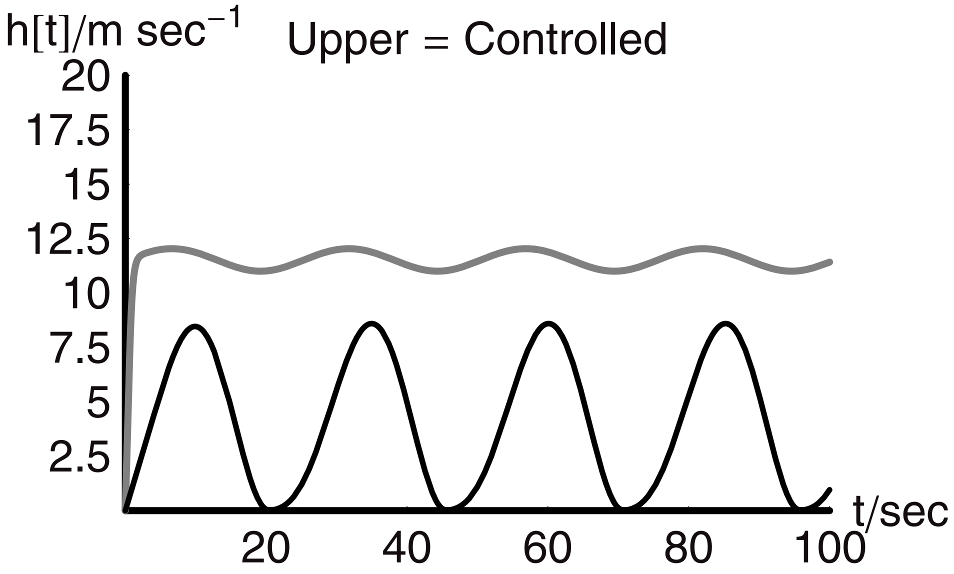
```
In[141]:= K = 1;
          soln1 = NDSolve[
```
$$\{hx'[t] == -\frac{\sqrt{2}\,Aoo\ b\sqrt{g}\sqrt{hx[t]}(1 + K(hx[t] - hd))}{A} + \frac{qo(1 + \alpha\,Sin[t\beta])}{A}, hx[0] == 0\}, hx[t], \{t, 0, tmax\}];$$

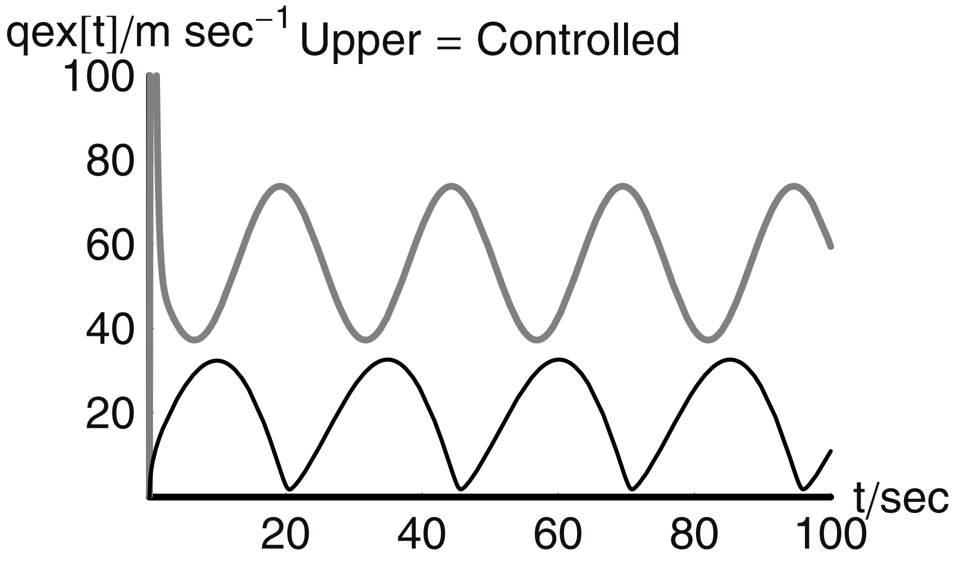
```
          hc[t_] := Evaluate[hx[t] /. soln1]

          qin[t] := qo(1 + αSin[βt])

          Ao[t_] := Aoo(1 + K(hd - hc[t]))
```
$$qexc[t\_] := b\,Ao[t]\sqrt{2 g hc[t]}$$

```
          plqcin = Plot[qin[t], {t, 0, tmax},
           AxesLabel → {"t/sec", "qin[t]/m sec^-1"},
           PlotRange → {{0, tmax}, {0, 50}}];

          plhc = Plot[hc[t], {t, 0, tmax},
           AxesLabel → {"t/sec", "h[t]/m sec^-1"},
           PlotStyle → {GrayLevel[0.5], Thickness[.01]},
           DisplayFunction → Identity];

          plqexc = Plot[qexc[t], {t, 0, tmax},
           AxesLabel → {"t/sec", "qex[t]/m sec^-1"},
           PlotStyle → {GrayLevel[0.5], Thickness[.01]},
           DisplayFunction → Identity];
```

```
Show[plhc, plhc0, PlotRange → {{0, tmax}, {0, 20}},
 DisplayFunction → $DisplayFunction,
 PlotLabel → "Upper = Controlled"];

Show[plqexc, plqexc0, DisplayFunction → $DisplayFunction,
 PlotRange → {{0, tmax}, {0, 100}},
 PlotLabel → "Upper = Controlled"];
```







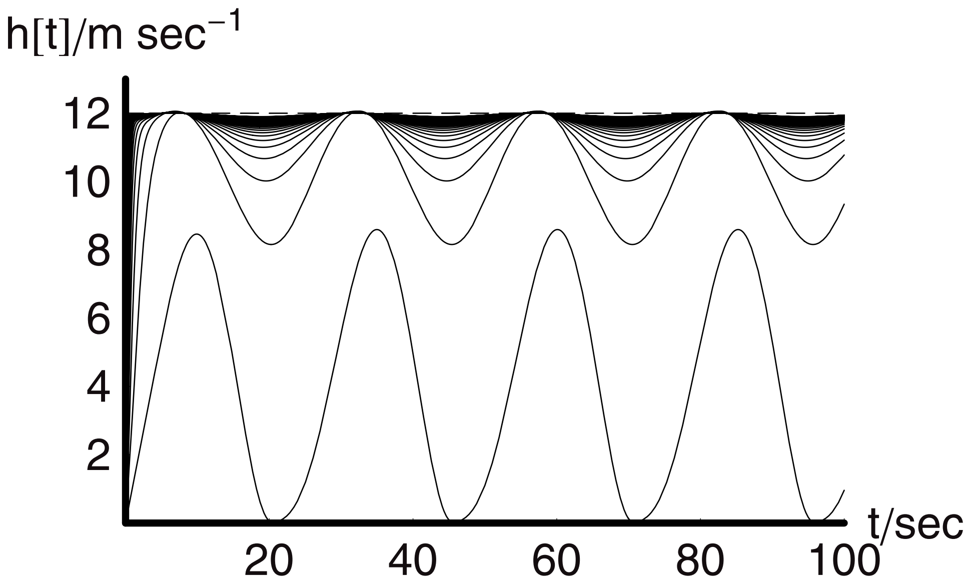
Using a bit of code and ingenuity we can have *Mathematica* compute the responses to incremental changes in **K** ranging from 0 to $2^1/_4$ in increments of a quarter unit. The code to do this is shown along with the outputs in two graphical forms—a stack plot and a graphics array:

```
In[152]:= Clear[α, β, g, q, qo, b, Ao, A, h, K, hd, Aoo, t]

          Ao =.
          r = 2;
          hd = 12;
          A = N[πr²];
          Aoo = 0.1 A;
          b = 1;
          g = 9.80;
          qo = 20;
          α = .25;
          β = .25;
          K = .1;
          tmax = 200;
          Clear[α, β, g, q, qo, b, Ao, A, h, K, hd, Aoo, t]

          Ao =.
          r = 2;
          hd = 12;
          A = N[πr²];
          Aoo = 0.2 A;
```

```
b = 1;
g = 9.80;
qo = 20;
α = 1;
β = .25;
K = 0;
tmax = 100;
solns=Table[NDSolve[{hx'[t] ==
```

$$-\frac{\sqrt{2}\,\text{Aoo b}\,\sqrt{\text{g}}\sqrt{\text{hx[t]}}\,(1 + \text{K(hx[t] - hd)})}{\text{A}}+\frac{\text{qo(1 + }\alpha\,\text{Sin[t}\beta\text{])}}{\text{A}},$$

```
 hx[0] == 0}, hx[t], {t, 0, tmax}], {K, 0, 10, .25}];

fns = Table[Evaluate[hx[t] /. solns[[n]]][[1]],
 {n, 1, Length[solns]}];

plots = Table[Plot[{fns[[n]]}, {t, 0, tmax},
 PlotRange → {{0, tmax}, {0, 13}},
 AxesLabel → {"t/sec", "h[t]/m sec^-1"},
 DisplayFunction → Identity], {n, 1, Length[solns]}];

Show[plots, DisplayFunction → $DisplayFunction,
 Epilog → {Dashing[{0.025, 0.015}],
 Line[{{0, 12.0}, {tmax, 12.0}}]}];

General::spell1 : Possible spelling error: new symbol name
 "solns" is similar to existing symbol "soln".
```



By increasing the magnitude of **K** we meet the original goal of dampening out the excursions that would take place had we not included the control function. There is nothing essential about the particular manner in which we solved this problem. Other functions could have been chosen for the control function. The essential feature of this analysis is the logical, stepwise manner in which we solved it.

3.7 Summary

In this chapter we have extended the analyses that we can do well beyond the simple systems of Chapter 1. We began with a fairly simple problem, the gravity-driven flow of fluid from a tank that led to Torricelli's Law. With this in the tool box we were able to step smartly through a series of systems with input and out fluid flows that were increasingly more complex, culminating in the proportional control of the level of a tank with sinusoidally driven input flow. As we moved through these examples, we have begun to use *Mathematica* in ever more sophisticated ways, providing us with new techniques to add to our arsenal of problem-solving weapons.

CHAPTER 4

Multiple-Component Systems

Single-component systems are not adequate for realistic chemical engineering problems. It is rare to have a single component unless it is the product of many different unit operations. If chemical engineering is the science of chemical and physical change, then it is also a science of complexity. A major source of complexity comes as a result of having to deal with real systems that are composed of many interacting components. The objective of this chapter is to set up a strong foundation for the problem of multicomponent systems of all kinds.

4.1 The Concept of the Component Balance

The masses of components can be handled in much the same way that we have handled total mass. The total mass balance is simply the sum of each of the component balances. Imagine we are playing a game tossing black and gray balls into a box on a scale (see Figure 1). Each ball has the same mass. The player tossing the gray balls is more skillful than the one tossing the black ones, and as a result she is able to throw more gray balls into the box every minute than the fellow who is tossing black balls. The scale tells us how fast the total mass of balls, both black and gray, is changing. If we want to know how fast the mass of just black balls in the box is changing, then we need to know how many are being thrown per unit time over the period of the measurement and similarly for the gray balls. The sum of the arrival rates of the black and gray balls together is the rate of mass change in total within the box.

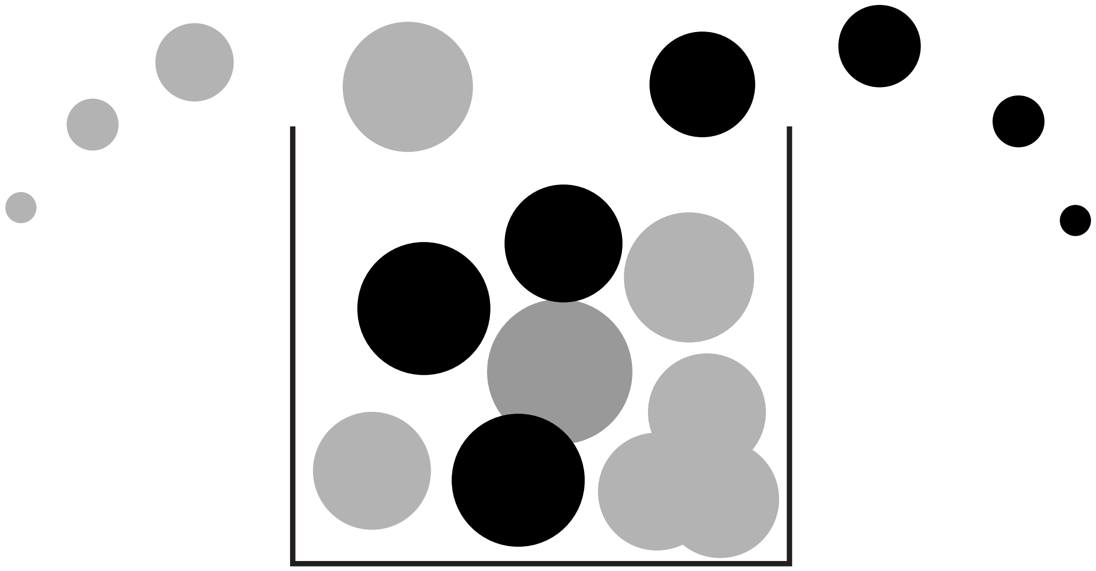


Figure 1

The total material balance for this system is:

$$\frac{dm_{\text{tot}}[t]}{dt} = \mathring{m}_{\text{black}} + \mathring{m}_{\text{gray}}$$

The component balances are:

$$\frac{dm_{\text{black}}[t]}{dt} = \mathring{m}_{\text{black}}$$
$$\frac{dm_{\text{gray}}[t]}{dt} = \mathring{m}_{\text{gray}}$$

Restating the total material balance, we have:

$$\frac{dm_{\text{tot}}[t]}{dt} = \frac{dm_{\text{black}}[t]}{dt} + \frac{dm_{\text{gray}}[t]}{dt}$$

Therefore, the sum of the component balances is the total material balance while the net rate of change of any component's mass within the control volume is the sum of the rate of mass input of that component minus the rate of mass output; these can occur by any process, including chemical reaction. This last part of the dictum is important because, as we will see in Chapter 6, chemical reactions within a control volume do not create or destroy mass, they merely redistribute it among the components. In a real sense, chemical reactions can be viewed from this vantage as merely relabeling of the mass.

4.2 Concentration versus Density

To this point we have had to deal only with the mass per unit volume in the form of density, since we were concerned only with single-component systems. Multiple components share the volume and because of this we must use concentration as well as density. The density of a single component **i** is the mass of that component per unit volume:

$$\rho_i = \frac{\text{mass}_i}{\text{vol}}$$

For a multicomponent system the total density is the sum of the masses of the components per unit volume:

$$\rho_{\text{tot}} = \frac{\sum_{i=1}^{n} \text{mass}_i}{\text{vol}}$$

The concentration of any component **i** can be either a mass concentration or a molar concentration:

$$C_{\boldsymbol{i}} = \frac{m_i}{\text{vol}} \equiv \frac{\text{mass}}{\text{vol}}$$
$$M_i = \frac{N_i}{\text{vol}} = \frac{1}{\text{Mw}_i}\frac{m_i}{\text{vol}} = \frac{C_i}{\text{Mw}_i} \equiv \frac{\text{mole}}{\text{vol}}$$

Although these definitions are straightforward, they do seem to cause problems more often than they should, especially for those who are just beginning to work with them in earnest.

The component material balance for a system with input and output, but no chemical reaction, is written as follows:

$$\frac{dm_i}{dt} = \mathring{m}_{i,\text{in}} - \mathring{m}_{i,\text{out}}$$

If the mass flows are those of liquids, then in terms of mass concentrations, this becomes:

$$\frac{V}{V}\frac{dm_i}{dt} = (\mathring{m}_{i,\text{in}} - \mathring{m}_{i,\text{out}})\frac{V}{V}$$
$$\frac{d}{dt}\left[\frac{m_i}{V}V\right] = \left(\frac{\mathring{m}_{i,\text{in}}}{V} - \frac{\mathring{m}_{i,\text{out}}}{V}\right)V$$
$$\frac{d}{dt}[C_i V] = (\mathring{C}_{i,\text{in}} - \mathring{C}_{i,\text{out}})V$$
$$\frac{d}{dt}[C_i V] = (C_{i,\text{in}}\, q_{\text{in}} - C_{i,\text{out}}\, q_{\text{out}})$$

The last statement is the typical form of a liquid-phase component mass balance. When this is divided through by the molecular weight of species **i**, this becomes a differential mole balance since the concentrations are expressed in molarity units:

$$\frac{1}{\mathrm{MW}_i}\frac{d}{dt}[C_i V] = (C_{i,\mathrm{in}}\, q_{\mathrm{in}} - C_{i,\mathrm{out}}\, q_{\mathrm{out}})\frac{1}{\mathrm{MW}_i}$$
$$\frac{d}{dt}[M_i V] = (M_{i,\mathrm{in}}\, q_{\mathrm{in}} - M_{i,\mathrm{out}}\, q_{\mathrm{out}})\frac{1}{\mathrm{MW}_i}$$

Typically, this last statement is written with the symbol C for molar concentration just as it is for mass concentration. Given that this is the case and it is not likely to change, the particular meaning of C must be understood from context. Fortunately, this is usually easy to do.

4.3 The Well-Mixed System

Once we move away from single component systems there is the real possibility that the components will partition themselves in different parts of the vessel due to different densities, solubilities, or miscibilities. Partitioned systems are also referred to as "distributed." That means that the properties are not everywhere the same over macroscopic length scales. To handle distributed systems we typically have to choose a differential control volume, that is, an infinitesimal volume within the macroscopic system. We will see this when we consider plug flow down a tube.

Although partitioning is often encountered, and even though it may be advantageous in many cases, it is also true that many systems are either naturally homogeneous or are forced to be by the action of vigorous mixing. When a system is homogeneous, it means that the density and concentration are everywhere the same throughout the control volume. This is referred to as the condition of being "well-mixed." From the purely mathematical vantage, it refers to any system that can be described solely in terms of time as the independent variable. We turn now to problems of systems with multiple components and which are well mixed.

4.4 Multicomponent Systems

Liquid and an Insoluble Solid

Mixtures are combinations of two or more components that share the same volume but retain their identity—liquid plus an insoluble solid, for example. Preparing such a mixture may be done in a mixing tank, such as that which is used to make cement. Often done in batch mode, it can be done continuously as well in a system such as this one shown in Figure 2.

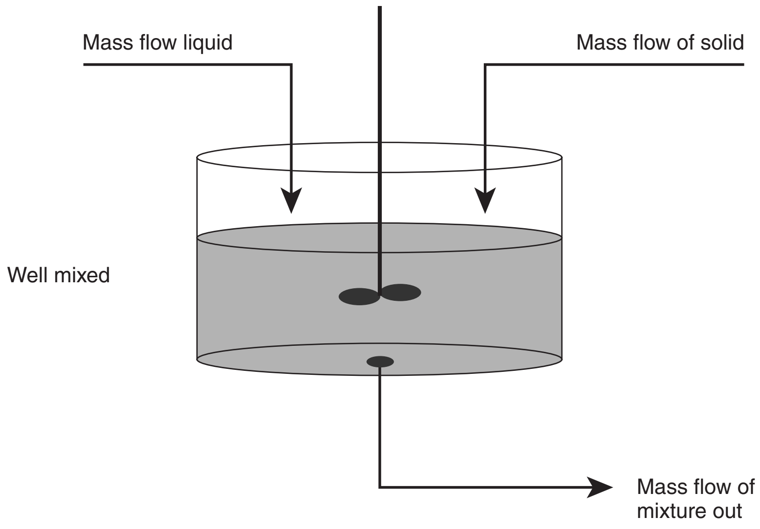


Figure 2

There is a mass flow of both liquid and solid into the tank; the two are mixed well and then flow out of the tank to their application. The total and component mass balances for the system are:

$$\text{Total:}\quad \frac{d\rho_{\text{mix}}V}{dt} = \rho_l q_l + \mathring{m}_s - \rho_{\text{mix}}\,q_{\text{mix}}$$
$$\text{Liquid:}\quad \frac{dC_{l,\text{mix}}V}{dt} = \rho_l q_l - C_{l,\text{mix}}\,q_{\text{mix}}$$
$$\text{Solid:}\quad \frac{dC_{s,\text{mix}}V}{dt} = \mathring{m}_s - C_{s,\text{mix}}\,q_{\text{mix}}$$

We have said that the total material balance is the sum of the component balances. Is that the case here? If so, what can it teach us? We will check it here:

$$\frac{dC_{l,\text{mix}}V}{dt} + \frac{dC_{s,\text{mix}}V}{dt} = \rho_l q_l - C_{l,\text{mix}}q_{\text{mix}} + \mathring{m}_s - C_{s,\text{mix}}q_{\text{mix}}$$
$$\frac{d[C_{l,\text{mix}} + C_{s,\text{mix}}]V}{dt} = \rho_l q_l + \mathring{m}_s - (C_{l,\text{mix}} + C_{s,\text{mix}})q_{\text{mix}}$$

For this to be equal to the total material balance, it must be true that the sum of the mass concentrations of solid and liquid are equal to the density of the mixture:

$$\frac{d\rho_{\text{mix}}V}{dt} = \frac{d[C_{l,\text{mix}} + C_{s,\text{mix}}]V}{dt};\quad \text{iff } \rho_{\text{mix}} = [C_{l,\text{mix}} + C_{s,\text{mix}}]$$

This is of course the case. If we remember that density is the sum of the masses occupying the same unit of volume, then we can see that:

$$\rho_{\text{mix}} = \frac{m_l + m_s}{V_{\text{mix}}} = \frac{m_l}{V_{\text{mix}}} + \frac{m_s}{V_{\text{mix}}} = C_{l,\text{mix}} + C_{s,\text{mix}}$$

This also means that there are really only two independent equations describing this system and that, given any two, the other can be derived. Now we can now proceed to solve these. But before we do, it is important to inspect the component balances and the total balance for their other details—namely, the consequence of the system being "well-mixed."

Notice that in the total material balance the argument of the derivative involves this mixture density. This as we have seen is the sum of the two concentrations of the two components, not the density of the solid alone, nor that of the liquid alone. On the right-hand side of the same equation, we note that the two input terms do involve the densities of the solid and the liquid in their pure states. This is because they are being delivered to the system as pure "feeds." The outflow term, however, includes the mixture density, the same density that appears in the argument of the differential. This is critical to understand. It says that everywhere in the control volume the density is the same at any time and that the material exiting the control volume also has the same density as the material in the tank. This is the consequence of assuming the system is well-mixed. The same analysis can be made for the two component balances. They show the well-mixed assumption because they include the corresponding mixture concentrations in the differential and the out-flow term.

To solve these equations we need to have a set of initial conditions for the system. We must decide, or know, whether the tank is initially empty and both the solid and the liquid are added simultaneously, if the tank is initially loaded with pure liquid (or pure solid), or if the tank contains a product mix from some previous production run. For the sake of this example we will assume that we must start up the tank and the process from scratch, that is, from an initially empty condition to production of the target mix. To do this we will follow three time intervals:

1. Fill the tank with liquid to a predetermined level **hmax**.
2. Feed only solid with good mixing until a target density for the mixture $\boldsymbol{\rho}^*_{\mathbf{mix}}$ is reached.
3. Feed liquid with solid to maintain this density while product mix flows from the tank continuously.

We will want to know how long it will take to reach each stage for a given set of inputs such as the feed rates and the target mixture density. To obtain that information we need to solve the balance equations in each interval. We do that now.

Interval 1: No solid flow and only liquid flow implies a single-component balance problem. The total material balance becomes:

$$\frac{d\rho_{\text{mix}}V}{dt} = \rho_l q_l + \mathring{m}_s - \rho_{\text{mix}} q_{\text{mix}}$$

$$\frac{d\rho_l V}{dt} = \rho_l q_l$$

$$V(t) = q_l t \quad \text{and} \quad h(t) = \frac{q_l}{A} t \Longrightarrow t_o = \frac{h_{\text{liq},o} A}{q_l}$$

Interval 2: Solid flow and no liquid flow to reach the critical density $\boldsymbol{\rho}^*_{\mathbf{mix}}$ corresponding to $\boldsymbol{C}^*_{\boldsymbol{l},\mathbf{mix}}$ and $\boldsymbol{C}^*_{\boldsymbol{s},\mathbf{mix}}$. The density of the mixture will be changing with time as the solid is fed to the system. We need to know how long it will take to reach the target density:

$$\frac{d\rho_{\text{mix}}V}{dt} = \mathring{m}_s$$

$$\text{V}\,\frac{d\rho_{\text{mix}}}{dt} + \rho_{\text{mix}}\frac{dV}{dt} = \mathring{m}_s$$

$$\text{V}\,\frac{d[C_{l,\text{mix}} + C_{s,\text{mix}}]}{dt} + [C_{l,\text{mix}} + C_{s,\text{mix}}]\,\frac{dV}{dt} = \mathring{m}_s$$

$$\left[\text{V}\,\frac{dC_{l,\text{mix}}}{dt} + C_{l,\text{mix}}\frac{dV}{dt}\right] + \left[\text{V}\frac{dC_{s,\text{mix}}}{dt} + C_{s,\text{mix}}\frac{dV}{dt}\right] = \mathring{m}_s$$

$$\frac{dC_{l,\text{mix}}V}{dt} + \frac{dC_{s,\text{mix}}V}{dt} = \mathring{m}_s$$

$$\frac{dC_{l,\text{mix}}V}{dt} = 0$$

$$\frac{dC_{s,\text{mix}}V}{dt} = \mathring{m}_s$$

If we expand the total material balance equation, we obtain the sum of these two back and so we are unable to solve the problem. This is because we have the unknown and only two independent equations. The unknowns are the concentrations of each component and the change in volume. Each is a function of time. We need another independent equation. To obtain this we need to think about what is happening.

When we add an insoluble solid to a liquid, the volume of the mixture must increase. In fact the volume must grow by the volume of solid that has been added according to Archimedes' Law.

$$V_{\text{mix}} = V_{\text{solid}} + V_{\text{liquid}} = \frac{\text{mass}_{\text{solid}}}{\rho_{\text{solid}}} + \frac{\text{mass}_{\text{liquid}}}{\rho_{\text{liquid}}}$$

During interval 2, the volume of the liquid is a constant but the volume of the solid in the mixture changes with time:

$$V_{\text{mix}}[\text{t}] = \frac{\mathring{m}_s t}{\rho_{\text{solid}}} + V_{\text{liquid}}$$

$$\frac{dV_{\text{mix}}[t]}{dt} = \frac{\mathring{m}_s}{\rho_{\text{solid}}}$$

Returning to the overall material balance equation, we find:

$$V_{\text{mix}}[\text{t}]\frac{d\rho_{\text{mix}}[t]}{dt} + \rho_{\text{mix}}[t]\frac{dV_{\text{mix}}[t]}{dt} = \mathring{m}_s$$

$$\left(\frac{\mathring{m}_s t}{\rho_{\text{solid}}} + V_{\text{liquid}}\right)\frac{d\rho_{\text{mix}}[t]}{dt} + \rho_{\text{mix}}[t]\frac{\mathring{m}_s}{\rho_{\text{solid}}} = \mathring{m}_s$$

Now this equation is soluble because it involves just one independent variable (function)—that of $\rho_{\mathbf{mix}}[t]$. At the start of interval 2 the density of the "mixture" in the control volume is just that of the liquid. This provides the essential initial condition that we need for solving the problem:

In[1]:= **Clear[ms, V]**

$$\texttt{Simplify[DSolve[\{(}\frac{\mathring{m}s}{\rho_s}\texttt{t + V}_1\texttt{)}\rho'_{\mathbf{mix}}\texttt{[t] == }\mathring{m}s\texttt{(1 - }\frac{\rho_{\mathbf{mix}}[t]}{\rho_s}\texttt{),}$$
$$\rho_{\mathbf{mix}}\texttt{[0] == }\rho_1\texttt{\}, }\rho_{\mathbf{mix}}\texttt{[t], t]]}$$

Out[2]= $\{\{\rho_{\mathrm{mix}}[t] \rightarrow \frac{(t\mathring{m}s + V_1\rho_1)\,\rho_s}{t\mathring{m}s + V_1\rho_s}\}\}$

We can also solve for the change in the concentration of the solid in the mixture as a function of time:

$$\frac{dC_{s,\mathrm{mix}}[t]V_{\mathrm{mix}}[t]}{dt} = \mathring{m}_s$$

$$V_{\mathrm{mix}}[t]\frac{dC_{s,\mathrm{mix}}[t]}{dt} + C_{s,\mathrm{mix}}[t]\frac{dV_{\mathrm{mix}}[t]}{dt} = \mathring{m}_s$$

$$\left(\frac{\mathring{m}_s t}{\rho_{\mathrm{solid}}} + V_{\mathrm{liquid}}\right)\frac{dC_{s,\mathrm{mix}}[t]}{dt} + C_{s,\mathrm{mix}}[t]\frac{\mathring{m}_s}{\rho_{\mathrm{solid}}} = \mathring{m}_s$$

In[3]:=
$$\texttt{Simplify[DSolve[\{(}\frac{\mathring{m}s}{\rho_s}\texttt{t + V}_1\texttt{)c}'_{\mathbf{s,mix}}\texttt{[t] == }\mathring{m}s\texttt{(1 - }\frac{c_{\mathbf{s,mix}}[t]}{\rho_s}\texttt{),}$$
$$c_{\mathbf{s,mix}}\texttt{[0] == 0\}, }c_{\mathbf{s,mix}}\texttt{[t],t]]}$$

Out[3]= $\{\{C_{s,\mathrm{mix}}[t] \rightarrow \frac{t\mathring{m}s\rho_s}{t\mathring{m}s + V_1\rho_s}\}\}$

and for the change in concentration of the liquid in the mixture:

In[4]:=
$$\texttt{DSolve[\{(}\frac{\mathring{m}s}{\rho_s}\texttt{t + V}_1\texttt{)c}'_{\mathbf{1,mix}}\texttt{[t] + }\mathring{m}s\ c_{\mathbf{1,mix}}\texttt{[t] == 0,}$$
$$c_{\mathbf{1,mix}}\texttt{[0] == }\rho_1\texttt{\}, }c_{\mathbf{1,mix}}\texttt{[t], t]}$$

Out[4]= $\{\{C_{1,\mathrm{mix}}[t] \rightarrow \rho_1(V_1\rho_s)^{\rho_s}(t\,\mathring{m}s + V_1\rho_s)^{-\rho_s}\}\}$

Our objective at this stage in the analysis was to solve for the time required to reach the target product mixture density of ($\rho^*_{\mathbf{mix}}$):

In[5]:= $\texttt{Simplify[Solve[}\rho_{\mathbf{mix,p}}\texttt{ == }\frac{(t\,\mathring{m}s + V_1\rho_1)\rho_s}{t\,\mathring{m}s + V_1\rho_s}\texttt{, t]]}$

Out[5]= $\{\{t \rightarrow \frac{V_1\rho_s(-\rho_1 + \rho_{\mathrm{mix,p}})}{\mathring{m}s(\rho_s - \rho_{\mathrm{mix,p}})}\}\}$

Thus at a given delivery rate of solid, fixed initial volume of liquid and solid density, we find that the time is:

$$t^* = \frac{V_1\rho_s(\rho^*_{mix} - \rho_1)}{\overset{\circ}{m}s(\rho_s - \rho^*_{mix})}$$

We should also like to know the level in the tank of given volume and cross section **A** when this product density is reached. It is critical that we check this. In other words the initial level of water in the tank must also be fixed by the maximum level of the target density of the mixture that can be held in the tank and mixed very well. We have an expression for the volume of the mixture in the tank as a function of time during this interval. We should divide it by **A** and then substitute in the time we just solved for to find the level that will be called for to achieve target density:

$$V_{mix}[t] = \frac{\overset{\circ}{m}_s t}{\rho_{solid}} + V_{liquid}$$
$$h_{mix}[t] = \frac{\overset{\circ}{m}_s t}{A\rho_{solid}} + h_{liquid,o}$$

```
In[6]:= Clear[A]
        Solve[h_mix == m̊s t/(A ρ_solid) + h_liquid,o, h_mix]

Out[7]= {{h_mix → t m̊s/(A ρ_solid) + h_liquid,o}}
```

The new level in the tank at the end of this second interval will then be:

$$h^*_{mix} = \frac{V_1(-\rho_1 + \rho^*_{mix})\rho_s}{A(-\rho^*_{mix} + \rho_s)\rho_s} + h_{liquid,o}$$

but we know that V_l is just **A** $\mathbf{h_{liquid,o}}$ so the overall expression becomes:

$$h^*_{mix} = h_{liquid,o}\left[\frac{(-\rho_1 + \rho^*_{mix})\rho_s}{(-\rho^*_{mix} + \rho_s)\rho_s} + 1\right]$$

Interval 3: This interval begins when the product mixture reaches its target density; then the flow out of the tank is turned on. At the same time the flow of the liquid feed must also be turned on in order to maintain the system in a steady state. We can solve for this:

$$\frac{d\rho_{mix}V}{dt} = \rho_l q_l + \overset{\circ}{m}_s - \rho_{mix}q_{mix}$$
$$0 = \rho_l q_l + \overset{\circ}{m}_s - \rho_{mix}q_{mix}$$
$$\rho_l q_l + \overset{\circ}{m}_s = \rho_{mix}q_{mix}$$
$$q_l = \frac{\rho_{mix}q_{mix} - \overset{*}{m}_s}{\rho_l}$$

As the solid is insoluble in the liquid, we know that the volume flow rate of the mixture must be the sum of the volume flow rates of the two components. The volume flow rate of the solid is just the mass flow of the solid divided by its density, assuming that the latter is nonporous. Therefore we can solve for **ql** as follows:

```
In[8]:= Clear[ql, p]
                                  ρmix,p(ql + qs) - m̊s             m̊s
        Simplify[Solve[{ql == ─────────────────────── , qs == ── }, ql]]
                                         ρl                       ρs

                   qs (-ρs + ρmix,p)
Out[9]= {{ql  →  ───────────────── }}
                     ρl - ρmix,p
```

Now, we will use these solutions. The mixing will be done in a pilot-scale unit. The tank that is available is 5 m high and has an aspect ratio of 3:1:h:d. Its diameter, area, and volume can be immediately computed:

```
In[10]:= hmax = 5;
         d = N[hmax/3]
                  d
         A = N[π(─)²]
                  2
         Vtank = Ad

Out[11]= 1.66667

Out[12]= 2.18166

Out[13]= 3.6361
```

This makes the volume 3.6 m^3, the cross-sectional area is 2.18 m^2, and the diameter is 1.67 m. The density of the solid is 2 kg L^{-1}, the liquid is 1 kg L^{-1}, and the target density $\rho^*_{\mathbf{mix}}$ is 1.5 kg L^{-1}. Experience indicates that the level of the mixture when settled should never rise to more than half the maximum level of the tank, to ensure that no mass leaves the tank during vigorous mixing. This means that we can base our calculations on a mixture level of $\frac{\mathbf{hmax}}{2}$. From this information and the parameters we can solve for the initial level of water in the tank at the end of interval 1:

```
In[14]:= ρmix = 1.5;
         ρsolid = 2;
         ρliq = 1;
         hmixt = 2.5;
         hliqmax = Solve[hmixt == hliqo
                          (-ρliq + ρmix)ρsolid
                        (─────────────────────── + 1), hliqo]
                         (-ρmix + ρsolid)ρsolid
         hliqmax[[1, 1, 2]] A
```

```
Out[18]= {{hliqo  → 1.25}}

Out[19]= 2.72708
```

The volume of water that must be added during interval 1 is 2.727 m^3, which is the product of the area of the tank and the liquid level. The water can be fed at .25 m^3 min^{-1} so the time required to add the water is 10.9 min. The mixing of the solid with the water is best done slowly to ensure homogeneity. Therefore, 30 min is to be allowed for interval 2. Given this, we can compute the mass flow of solid required from either of the following equations. We use both to verify the result. We note that the densities are in units of kg/L, whereas the volumes and levels are in units of meters. There are (as shown in what follows) 1000 L per m^3. Therefore, each of the densities must be multiplied by this factor to convert them to kg per m^3:

In[20]:= $\frac{1\,L}{1000\,cm^3}\left(\frac{100cm}{1\,m}\right)^3$ // N

Out[20]= $\frac{1000.L}{m^3}$

In[21]:= **t = 30;**
Vl = 2.727;
Solve[$1000\,\rho mix == \frac{(t\,\overset{\circ}{m}s + Vl\,\rho liq\,1000)\,\rho solid\,1000}{t\,\overset{\circ}{m}s + Vl\,\rho solid\,1000}$, $\overset{\circ}{m}s$**]**
Solve[$t == \frac{Vl\,1000\,\rho solid\,1000\,(\rho mix - \rho liq)}{\overset{\circ}{m}s\,1000\,(\rho solid - \rho mix)}$, $\overset{\circ}{m}s$**]**

Out[23]= {{$\overset{\circ}{m}s$ → 181.8}}

Out[24]= {{$\overset{\circ}{m}s$ → 181.8}}

The mass flow of solid must be 181.8 kg min^{-1} during interval 2. We can check our work to this point to be sure that at the end of the second interval we have a slurry with the target density of 1.5 kg/L:

In[25]:= $\frac{181.8\frac{kg}{min}30\,min}{2\frac{kg}{L}1000\frac{L}{m^3}}$ **(*Volume of the Solid Added During Interval 2*)**

```
General::spell1: Possible spelling error: new symbol name
"min" is similar to existing symbol "Min".
```

Out[25]= 2.727 m^3

In[26]:= **2.727 m^3 (*Volume of the Liquid Added During Interval 1*)**

Out[26]= 2.727 m^3

```
In[27]:= 2(2.727 m^3) (*Total Volume of the Solid and Liquid at
          End of Interval 2*)

Out[27]= 5.454 m^3
```

$$\texttt{In[28]:=}\ \frac{\left(181.8\frac{\text{kg}}{\text{min}}30\,\text{min} + 2.727\,\text{m}^3\frac{1000\,\text{k}}{\text{m}^3}\right)1\,\text{m}^3}{2\,(2.727\,\text{m}^3)\,1000\,\text{L}}$$

```
          (*Mass Solid + Liquid/Total Volume*)
```

$$\texttt{Out[28]=}\ \frac{1.5\,\text{kg}}{\text{L}}$$

This number checks and confirms that we have the right quantities, flows, and times to this point.

Our attention now turns to interval 3, which will be the steady-state production of slurry. At steady state we would like to be producing 1000 kg min^{-1} $= \rho^*_{\text{mix}}q^*_{\text{mix}}$ of slurry at the density of 1.5 kg/L. This would correspond to a volume flow rate equal to 666.67 L/min or 0.667 m^3/min slurry. The steady-state material balance can be used to find the required solids mass flow rate needed to achieve this production rate:

$$\rho_l q_l + \mathring{m}_s = \rho_{\text{mix}} q_{mix}$$

$$q_l = \frac{\rho_{\text{mix}} q_{\text{mix}} - \mathring{m}_s}{\rho_l}$$

As the maximum water flow rate is 0.25 m^3 min^{-1}, we can compute the solids flow rate:

```
In[29]:= ρliq = 1;
         ql = 0.25;

         Solve[1000 ρliq ql + m̊s == 1000, m̊s]

Out[31]= {{m̊s → 750.}}
```

To make the production rate we seek the solids must be flowed in at a rate of 0.75 kg min^{-1}. A good question to ask at this point would be how would we transition from the end of interval 2 into the steady state? How would we program the increase in mass flows that would have to take place in order to maintain the product density and to maintain the steady level in that tank? We can check the steady-state quantities by recomputing the volume flow rate of slurry from the mass flow rate of solids, the volume flow rate of water, and the target density of the product:

$$\texttt{In[32]:=}\ \frac{750\frac{\text{kg}}{\text{min}} + .25\frac{\text{m}^3}{\text{min}}\frac{1000\,\text{kg}}{\text{m}^3}}{1500\frac{\text{kg}}{\text{m}^3}}\ \texttt{(*Volume Flow of Slurry at St.St.*)}$$

$$\texttt{Out[32]=}\ \frac{0.666667\,\text{m}^3}{\text{min}}$$

This checks perfectly with the previously computed value.

4.5 Liquid and Soluble Solid

A more complex problem is that of a soluble solid and a liquid. The physical situation is the same as in the previous problem. The initial equations are also the same except that now we are dealing with a solution rather than with a mixture:

$$\begin{aligned}
\text{Total:}\quad & \frac{d\rho_{\text{soln}}V}{dt} = \rho_l q_l + \overset{\circ}{m}_s - \rho_{\text{soln}} q_{\text{soln}} \\
\text{Liquid:}\quad & \frac{dC_l,V}{dt} = \rho_l q_l - C_l, q_{\text{soln}} \\
\text{Solid:}\quad & \frac{dC_{s,\text{soln}}V}{dt} = \overset{\circ}{m}_s - C_{s,\text{soln}} q_{\text{soln}}
\end{aligned}$$

Case 1: Constant densities

If the concentration of the soluble solid does not reach a high level, then it is reasonable to assume that the densities of the pure solvent and the solution are similar enough to treat as equal and constant. Doing this transforms the total material balance into:

$$\frac{dV}{dt} = q_l + \frac{\overset{\circ}{m}_s}{\rho_{\text{soln}}} - q_{\text{soln}}$$

This can be integrated immediately if the exit flow rate is a constant:

$$\text{V[t]} = \text{Vo} + \left[q_l + \frac{\overset{\circ}{m}_s}{\rho_{\text{soln}}} - q_{\text{soln}}\right]\text{t}$$

The component balances can be integrated in the same way. The initial condition is that the volume in the tank is **Vo** of pure solvent and the concentration of the solid is zero. To find the analytical solutions to these equations, we specify **V[t]** and then we use **DSolve** to simultaneously solve for the concentrations, calling the set of two solutions "a." Two functions are named and then extracted from the solution set and assigned to these names. Finally, the two new functions are placed back into the original differential equations and tested for validity.

```
In[33]:= Remove[a, V, cs, cl, ρ, ql, qsoln, t, Vo, cs1, cl1, ms];
         V[t_] := Vo + (ql + m̊s/ρ - qsoln) t
         a = Simplify[DSolve[{
              ∂t(cs[t] V[t]) == ρql - cs[t] qsoln, cs[0] == 0,
              ∂t(cl[t] V[t]) == ρql - cl[t] qsoln, cl[0] == ρ
            },
            {cs[t], cl[t]}
                  t]]
```

```
          cs1[t_] := Evaluate[cs[t] /. a]
          cl1[t_] := Evaluate[cl[t] /. a]
```

Out[35]=

$$\left\{\mathrm{cl[t]} \to \frac{\mathrm{ql}\rho^2}{\mathrm{ql}\rho+\mathring{m}\mathrm{s}} + \frac{\rho\,(\mathrm{Vo}\,\rho)^{\frac{\mathrm{ql}\rho+\mathring{m}\mathrm{s}}{\mathrm{ql}\rho-\mathrm{qsoln}\rho+\mathring{m}\mathrm{s}}}\,\mathring{m}\mathrm{s}\,((\mathrm{ql\,t}-\mathrm{qsoln\,t}+\mathrm{Vo})\,\rho+\mathrm{t}\,\mathring{m}\mathrm{s})^{-\frac{\mathrm{ql}\rho+\mathring{m}\mathrm{s}}{\mathrm{ql}\rho-\mathrm{qsoln}\rho+\mathring{m}\mathrm{s}}}}{\mathrm{ql}\rho+\mathring{m}\mathrm{s}},\right.$$

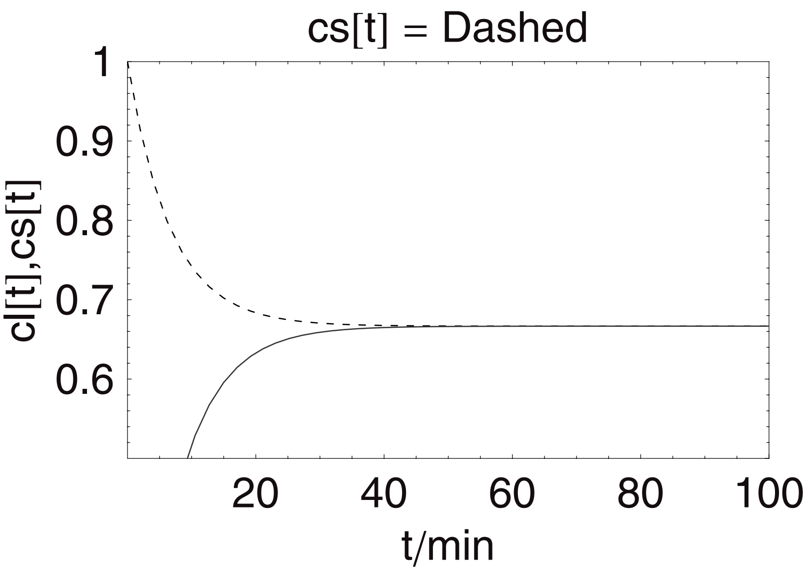
$$\left.\mathrm{cs[t]} \to \frac{\mathrm{ql}\rho^2}{\mathrm{ql}\rho+\mathring{m}\mathrm{s}} - \frac{\mathrm{ql}\rho^2\,(\mathrm{Vo}\,\rho)^{\frac{\mathrm{ql}\rho+\mathring{m}\mathrm{s}}{\mathrm{ql}\rho-\mathrm{qsoln}\rho+\mathring{m}\mathrm{s}}}\,((\mathrm{ql\,t}-\mathrm{qsoln\,t}+\mathrm{Vo})\,\rho+\mathrm{t}\mathring{m}\mathrm{s})^{-\frac{\mathrm{ql}\rho+\mathring{m}\mathrm{s}}{\mathrm{ql}\rho-\mathrm{qsoln}\rho+\mathring{m}\mathrm{s}}}}{\mathrm{ql}\rho+\mathring{m}\mathrm{s}}\right\}$$

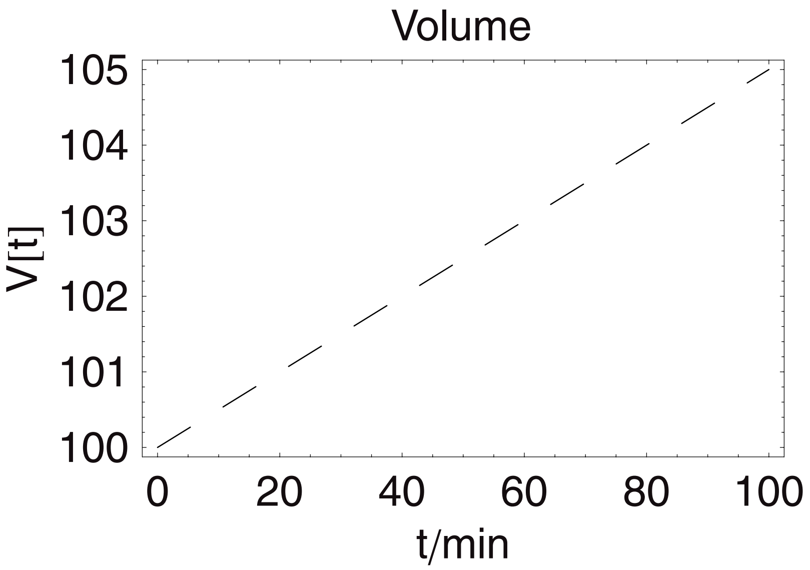
```
In[38]:= Simplify[∂t(cs1[t] V[t]) == ρql - cs1[t] qsoln]
          Simplify[∂t(cl1[t] V[t]) == ρql - cl1[t] qsoln]

Out[38]= True

Out[39]= True
```

Parameter values are applied and the functions are plotted in time:

```
In[40]:= SetOptions[{Plot, ListPlot}, AxesStyle → {Thickness[0.01]},
            PlotStyle  → {PointSize[0.015], Thickness[0.006]},
            DefaultFont  → {"Helvetica", 17}];

In[41]:= Clear[ρ, ql, qsoln, t, Vo]
          m̊s = .
          ρ = 1;
          Vo = 100;
          m̊s = 5;
          ql = 10;
          tmax = 100;
          qsoln = 14.95;

          Plot[{cl1[t], cs1[t]}, {t, 0, tmax},
            PlotStyle  → {{GrayLevel[0], Dashing[{0.01, 0.015}]},
            GrayLevel[0.2]}, FrameLabel  → {"t/min", "cl[t],cs[t]"},
            PlotRange  → {{0, tmax}, {.5, 1.0}},
            Frame  → True, PlotLabel  → "cs[t] = Dashed"];

          Plot[V[t], {t, 0, tmax}, PlotStyle → Dashing[{0.06, 0.06}],
            FrameLabel  → {"t/min", "V[t]"}, Frame  → True,
            PlotLabel  → "Volume"];

         Unset::norep : Assignment on Overscript for m̊s not found.
```

```
Out[42]= $Failed

                General::spell1: Possible spelling error: new symbol
                name "tmax" is similar to existing symbol "hmax".
```

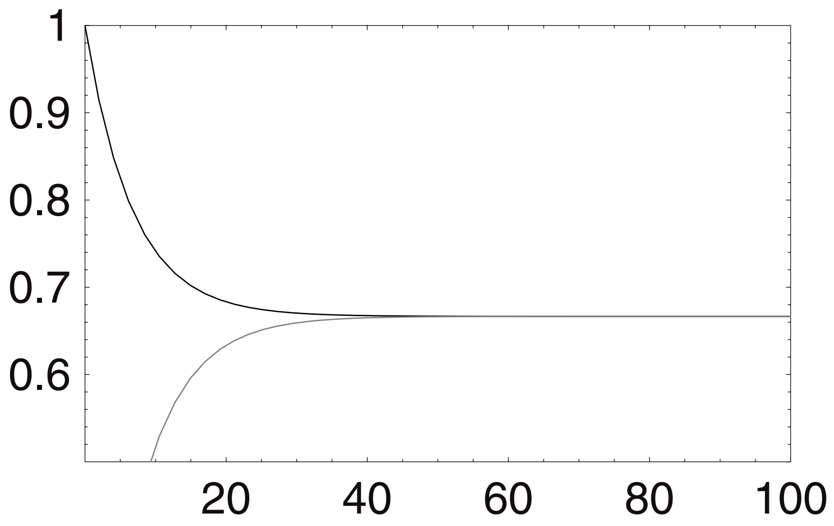


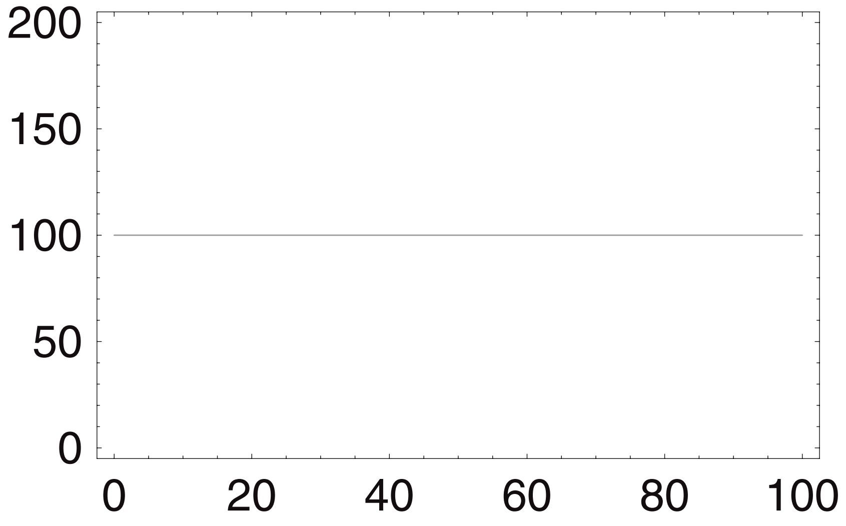


The solutions show two important aspects of this model as written: The concentrations come to a constant and equal value and the volume continues to rise indefinitely. The reason for this is that we took the outlet concentration to be a constant. Although this may have made for a simple model to solve, it is also one that is not very realistic. The tank would be overflowing. A more realisitc model would be one in which the exit flowrate was either set to match the inlet flow rate, which would make $\frac{dV}{dt}$ zero and the volume a constant at its initial level, or we could assume the flow rate out was gravity driven and would respond to the level in the tank, that is, Torricelli's Law.

The first case in which an instantaneous achievement of steady state is assumed follows:

```
In[51]:= Remove[a1, V, Vo, cs, cs1, cl, cl1, ρ, ql, qsoln];

In[52]:= a1 = Simplify[DSolve[{
               ∂t(cs[t] Vo) == ρql - cs[t] qsoln, cs[0] == 0,
               ∂t(cl[t] Vo) == ρql - cl[t] qsoln, cl[0] == ρ
              }, {cs[t], cl[t]},
                     t]];
         cs1[t_]:= Evaluate[cs[t] /. a1]
         cl1[t_]:= Evaluate[cl[t] /. a1]
         Simplify[∂t(cs1[t] Vo) == ρql - cs1[t] qsoln]
         Simplify[∂t(cl1[t] Vo) == ρql - cl1[t] qsoln]
         ρ = 1;
         Vo = 100;
         m̊s = 5;
         ql = 10;
         tmax = 100;
                      m̊s
         qsoln = ql + ──;
                      ρ
         Plot[{cl1[t], cs1[t]}, {t, 0, tmax},
           PlotStyle → {GrayLevel[0], GrayLevel[0.5]},
           AxesLabel → {"t/min", "cl[t],cs[t]"},
           PlotRange → {{0, tmax}, {.5, 1.0}}, Frame → True];
         Plot[Vo, {t, 0, tmax}, PlotStyle → GrayLevel[0.6],
           AxesLabel → {"t/min", "V[t]"}, Frame → True];

Out[55]= True

Out[56]= True
```





The asymptotic concentrations that we compute are the same as those that we had in the previous case, but the volume within the system is a constant.

In the second case the outlet flow rate is given by Torricelli's Law, that is, **qsoln=** $\mathbf{bAo}\sqrt{2gh[t]}$:

$$\frac{dV}{dt} = q_l + \frac{\mathring{m}_s}{\rho} - \mathrm{bAo}\sqrt{2gh[t]}$$

$$\frac{dh[t]}{dt} = \frac{q_l}{A} + \frac{\mathring{m}_s}{\rho A} - \mathrm{b}\frac{\mathrm{Ao}}{A}\sqrt{2gh[t]}$$

```
In[65]:= Remove[a2, ρ, ql, A, ms, ρ, b, Ao, g, ho, tmax, Vo, V, t];
         V =.
         ql = 10;
         A = 10;
         m̊s = 5;
         ρ = 1;
         b = 1;
         Ao = 0.1 A;
         g = 9.8;
         ho = 10;
         tmax = 100;
         Vo = 100;
```

```
a2 = NDSolve[{
```

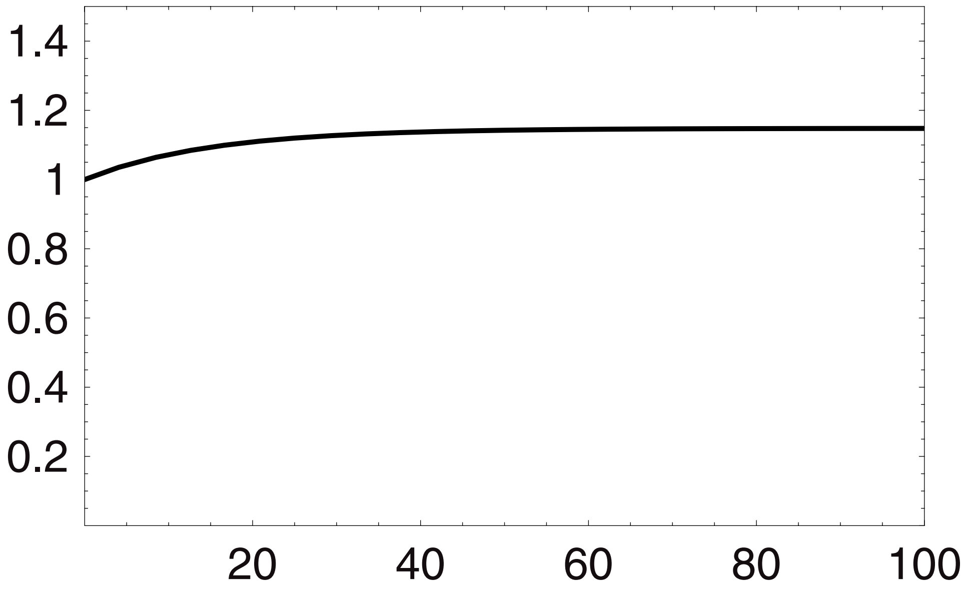
$$\partial_t h[t] == \frac{1}{A}\left(ql + \frac{\mathring{m}s}{\rho} - bAo\sqrt{2gh[t]}\right),\ h[0] == ho,$$

$$\partial_t (A\ cs[t]\ h[t]) == \frac{\mathring{m}s}{\rho} - cs[t]bAo\sqrt{2gh[t]},\ cs[0] == 0,$$

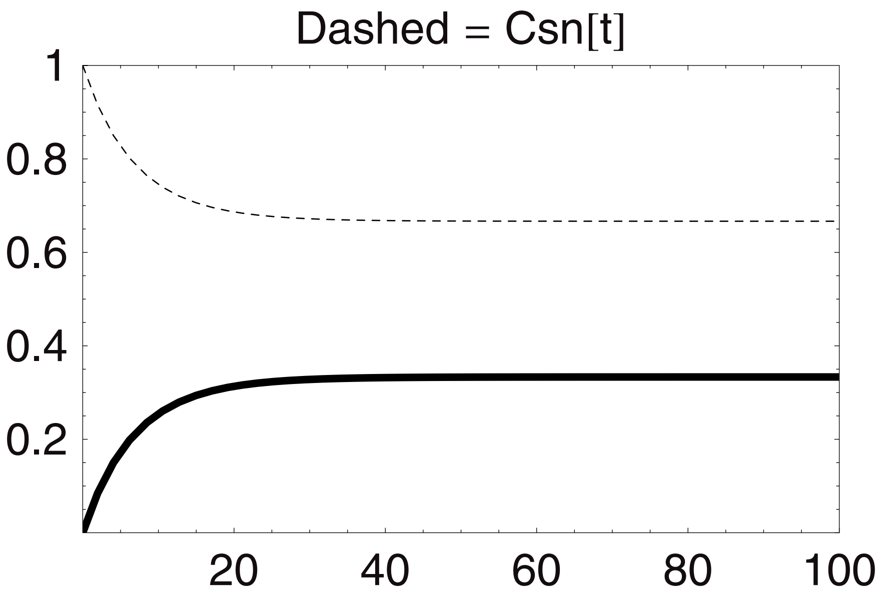
$$\partial_t (A\ cl[t]\ h[t]) == \rho ql - cl[t]\ bAo\sqrt{2gh[t]},\ cl[0] == \rho$$

```
  },
  {h[t], cs[t], cl[t]},
  {t, 0, tmax}];

hn[t_]:= Evaluate[h[t] /. a2]
Vn[t_]:= A hn[t]

pla2V = Plot[{hn[t]/ho, Vn[t]/Vo},
   {t, 0, tmax}, AxesLabel → {"t/min", "hn[t]/ho,Vn[t]/Vo"},
   PlotRange → {{0, tmax}, {0, 1.5}}, Frame → True];

csn[t_]:= Evaluate[cs[t] /. a2]
cln[t_]:= Evaluate[cl[t] /. a2]

pla2C = Plot[{cln[t], csn[t]}, {t, 0, tmax},
    AxesLabel → {"t/min", "csn[t],cln[t]"},
    PlotRange → {{0, tmax}, {0, 1.0}},
    PlotStyle → {Dashing[{0.01, 0.01}], Thickness[0.01]},
    PlotLabel → "Dashed = Csn[t]", Frame → True];
```

```
Remove::remal : Symbol Removed[ρ] already removed.

General::spell1 : Possible spelling error: new symbol name
"tmax" is similar to existing symbol "hmax".
```



```
General::spell1 : Possible spelling error: new symbol name
"pla2C" is similar to existing symbol "pla2V".
```



Now we have solved the full problem with gravity-driven flow. We see that the concentrations transition smoothly once again to steady-state values, but now the level and volume of liquid in the tank do so also. The assumption that the density of solution does not change very much with concentration is quite restrictive. Therefore, we deal with this problem explicitly in the next section.

Case 2: Variable Densities

The 'how and why' of variable density. Assuming that the densities were all similar in magnitude was a restriction on the solution we derived. We can rederive the solution without this assumption; but we do need a constitutive relationship to functionally couple the density and concentration. A suitable expression can be found by consulting either the *CRC Handbook of Chemistry and Physics*, or *Perry's Handbook* for data relating the concentration of various solutions of salts to their densities. From an analysis of these data we would find that the density of a solution is linearly related to the concentration of that salt over a wide range of concentrations. This relationship can be expressed as follows:

$$\rho = \mathrm{a} + \gamma \mathrm{C}$$

With some salts the volume of the solution expands as their concentration increases; this leads to a value of the constant γ. This tells us mathematically that as the salt dissolves into the solvent, it causes volume expansion and density diminution. In other words, if metal ions and their counterions are low in mass and if they tend to repel the water molecules, then the overall salt plus water structure occupies more space. For some salts (and neutral solutes) the opposite occurs and the density increases; thus the value of γ is > 0. Here the masses of the ions are high and their charges may also be high. Thus they tend to draw the water molecules into a more densely packed configuration, so that more mass is packed into a smaller volume when compared to water (or the solvent). If γ were 0, then that would indicate that the solute was close in mass to the solvent, occupied a similar volume when dissolved in a given volume of solvent, and left the solvent structure unchanged. If, for example, we were to add deuterated water D_2O to normal untreated water H_2O, the changes in density would be small and γ would be very small.

These observations, and the linear relationship they lead to, can be rationalized by considering the definition of the density of a solution. The density of a solution is the sum of the mass of the solute and the mass of the solvent divided by the total volume of the solution:

$$\rho[m_{\mathrm{salt}}] = \frac{m_{\mathrm{solvent}} + m_{\mathrm{solute}}}{V_{\mathrm{solution}}}$$

$$\rho[0] = \frac{m_{\mathrm{solvent}}}{V_{\mathrm{solvent}}}(m_{\mathrm{salt}} \to 0; V_{\mathrm{solution}} \to V_{\mathrm{solvent}}) = \rho_{\mathrm{solvent}}$$

If the mass of the solute in solution were 0, then the density is just that of the solvent. What if the solute were very special in its interactions with the solvent—suppose it neatly occupied those spaces between the solvent molecules that were open (interstices) but caused no net increase or decrease in the volume of the solvent? If the mass of dissolved solute causes no change in volume, then the solution volume would be the same as the original solvent volume and the density becomes:

$$\rho = \frac{m_{\text{solvent}} + m_{\text{solute}}}{V_{\text{solution}}} = \frac{m_{\text{solvent}}}{V_{\text{solvent}}} + \frac{m_{\text{solute}}}{V_{\text{solvent}}} = \rho_{\text{solvent}} + \frac{m_{\text{solute}}}{V_{\text{solvent}}} = \rho_{\text{solvent}} + C_{\text{solute}};$$
$$\text{iff } V_{\text{solution}} = V_{\text{solvent}}$$

The implication is that the constant γ has the value of unity (1). This can only be true if the salt simply adds to the solution and occupies no more or less space than the solvent molecules, as we can see from the equation. If we now reintroduce the linear relationship for density in terms of solute concentration, and substitute in for the terms, then we see that the case we have just considered is a special case of the general one in which $\gamma = 1$. To take this analysis one step further, we can solve for γ in terms of the measurables of the solution:

```
In[84]:= Clear[γ]
         Simplify[
          Solve[msalt/Vsolution + msolvent/Vsolution == γ msalt/Vsolution + msolvent/Vsolvent, γ]]

Out[85]= {{γ → (msalt Vsolvent + msolvent (- Vsolution + Vsolvent))/(msalt Vsolvent)}}
```

We can rearrange this expression taking $\boldsymbol{\delta}\mathbf{V} = \mathbf{Vsolution} - \mathbf{Vsolvent}$—in other words, the extent to which the solute either expands or contracts the solvent volume by its presence, and we find:

$$\gamma = 1 - \left[\frac{\text{msolvent}}{\text{msalt}}\right]\left[\frac{\delta\text{V}}{\text{Vsolvent}}\right]$$

Then in the case where $\gamma = 1$, by this expression we see that $\boldsymbol{\delta}\mathbf{V} = \mathbf{0}$, which is a nice consistency check for what we have done to this point. This expression for $\boldsymbol{\gamma}$ is a dimensionless grouping that offers some insight into what this constant really means physically. If the solute causes restructuring of the solvent by drawing solvent molecules to itself in ensembles that have higher (or lower) numbers of molecules per unit volume, then that implies a "nonideality," and $\boldsymbol{\delta}\mathbf{V} \neq \mathbf{0}$, which implies that $\gamma \neq 1$.

We can now return to the problem of the feeding a salt and water to a mixing vessel that initially contains water, and from which the flow is governed by gravity without the assumption of ideality:

$$\text{Total:}\quad \frac{d\rho_{\text{soln}}V}{dt} = \rho_l q_l + \mathring{m}_s - \rho_{\text{soln}}q_{\text{soln}}$$

$$\text{Liquid:}\quad \frac{dC_{l},V}{dt} = \rho_l q_l - C_{l},q_{\text{soln}}$$

$$\text{Solid:}\quad \frac{dC_{s,\text{soln}}V}{dt} = \mathring{m}_s - C_{s,\text{soln}}q_{\text{soln}}$$

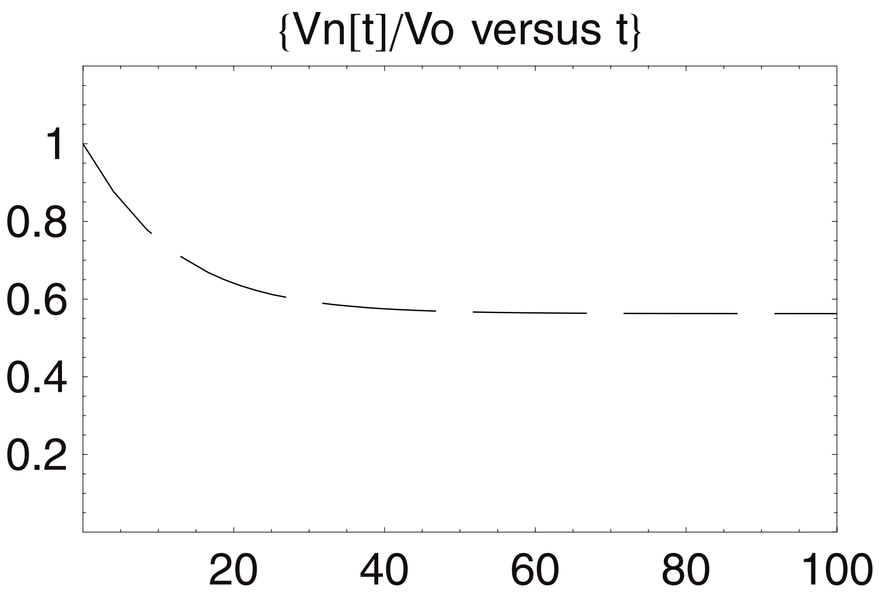
The total material balance now becomes:

$$\text{Total:}\quad \frac{d[\rho_l + \gamma C_{s,\text{soln}}]V}{dt} = \rho_l q_l + \mathring{m}_s - [\rho_l + \gamma C_{s,\text{soln}}]b\text{Ao}\sqrt{\frac{2g}{A}V}$$

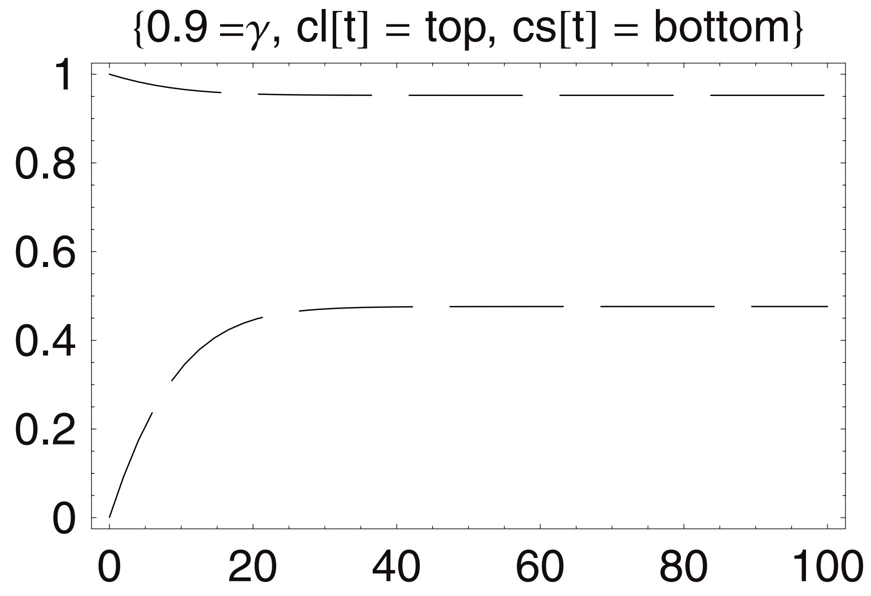
Recall that $\boldsymbol{\rho_l}$ is just the density of the pure liquid solvent, and that **h[t] =** $\frac{V[t]}{A}$ can be replaced into Torricelli's Law. These equations have become complicated enough that we shall define the density upfront and let **NDSolve** handle the work of solving the simultaneous equations (see the In statement that follows):

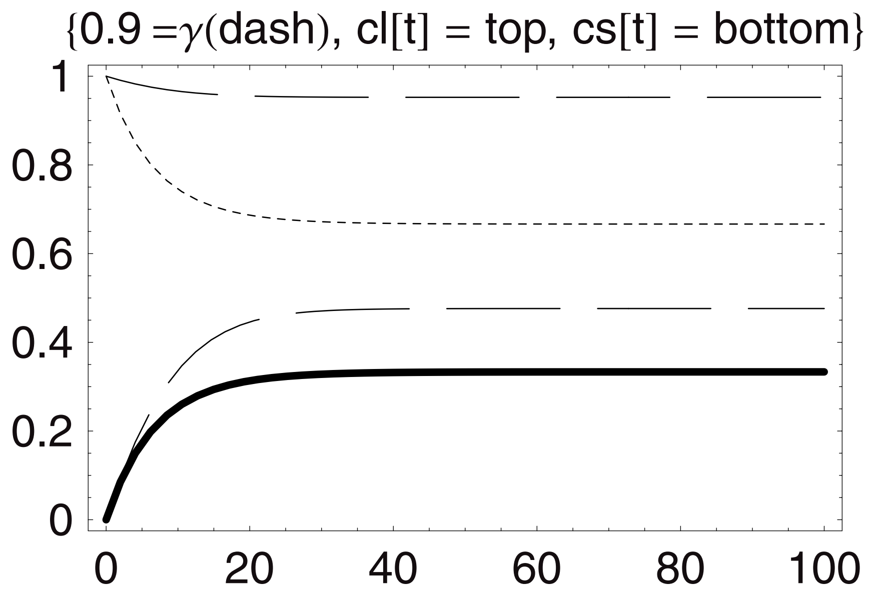
```
In[86]:= Remove[a3, ql, A, ms, ρ, b, Ao, g, ho, tmax, Vo, V,
          t, cs, cl, γ];
         V =.
         ql = 10;
         A = 10 ;
         m̊s = 5;
         ρo = 1;
         b = 1;
         Ao = 0.1 A;
         g = 9.8;
         ho = 10;
         tmax = 100;
         Vo = A ho;
         γ = .9;
         ρ[t_] := ρo + γ cs[t]
         a3 = NDSolve[{cs[0] == 0.001,
            ∂t(V[t]ρ[t]) == ρoql + m̊s - ρ[t]bAo√(2g/A V[t]), V[0] == vo,
            ∂t(cs[t]V[t]) == m̊s - cs[t]bAo√(2g/A V[t]),
            ∂t(cl[t]V[t]) == ρoql - cl[t]bAo√(2g/A V[t]), cl[0] == ρo
            },
   {V[t], cs[t], cl[t]},
   {t, 0, tmax}];

Vn[t_] := Evaluate[V[t] /. a3]

pla3V = Plot[{Vn[t]/Vo}, {t, 0, tmax},
   PlotLabel → {"Vn[t]/Vo versus t"},
   PlotStyle → {{Dashing[{0.15, 0.05}]}},
   PlotRange → {{0, tmax}, {0, 1.2}}, Frame → True];

csn[t_]:= Evaluate[cs[t] /. a3]
cln[t_]:= Evaluate[cl[t] /. a3]

pla3C = Plot[{cln[t], csn[t]}, {t, 0, tmax},
   PlotStyle → Dashing[{0.15, 0.05}],
   PlotLabel → {γ "=γ", "cl[t] = top, cs[t] = bottom"},
   Frame → True];

Show[pla3C, pla2C, PlotLabel → {γ"=γ(dash)",
   "cl[t] = top, cs[t] = bottom"}];
```

```
General::spell1 : Possible spelling error: new symbol name
 "tmax" is similar to existing symbol "hmax".
```



```
General::spell1 : Possible spelling error: new symbol name
 "pla3C" is similar to existing symbol "pla3V".
```





The last graph compares the two solutions of the problem—with and without the inclusion of the variation in density ($\gamma = 0.9$) with concentration of the salt. We can see that the salt concentration rises to a higher steady-state level (dashed curve at bottom) when the variation in density is included.

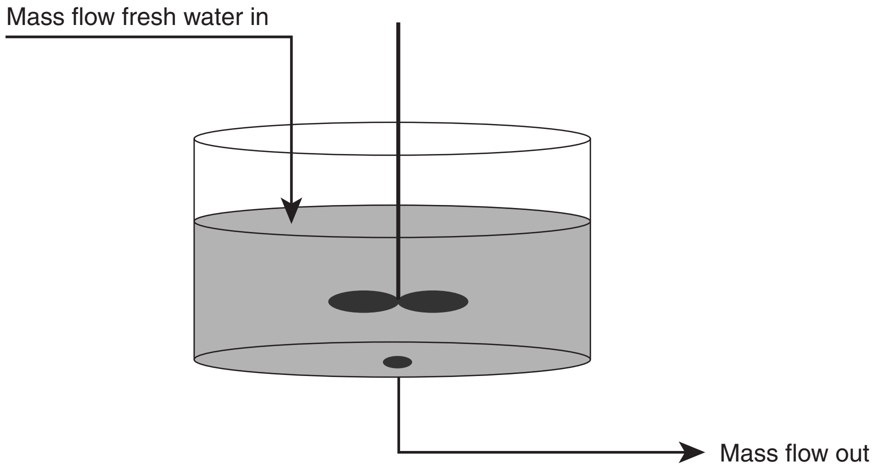


Figure 3

4.6 Washing a Salt Solution from a Vessel

In some respects, a simpler problem is the reverse of the one we have been solving. What if at the end of the process of preparing, feeding, and using a solution we have to wash the unit with fresh water in order to prevent it from corroding the vessel and to decontaminate it? How long will it take? How much water should we use? Will the flow rate matter? These are all very relevant chemical engineering questions that we can answer and do so fairly easily. Figure 3 shows that we have a feed of fresh water into a tank containing the salt solution as the initial condition.

The total and component material balances for this systems are as follows:

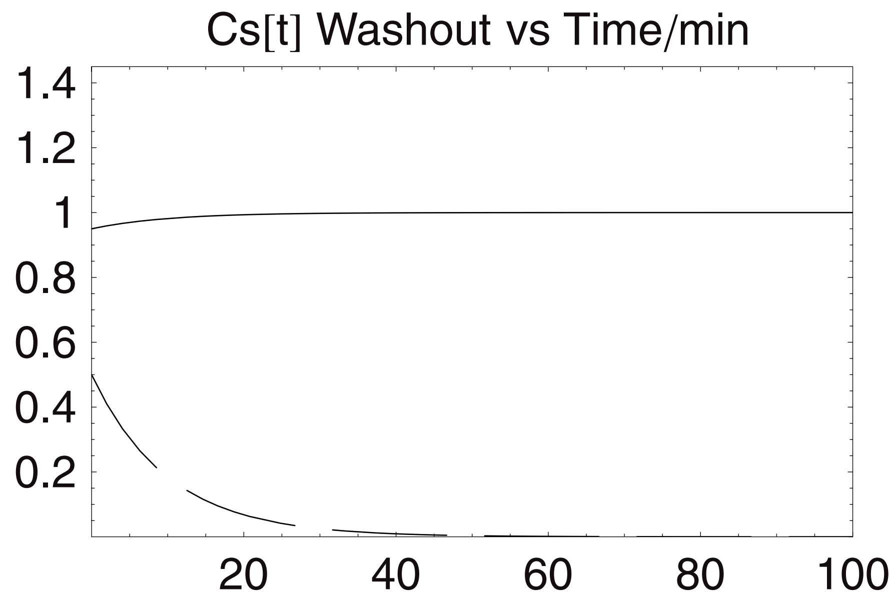
$$\text{Total:}\quad \frac{d\rho_{\text{soln}}V}{dt} = \rho_l q_l - \rho_{\text{soln}} q_{\text{soln}}$$

$$\text{Liquid:}\quad \frac{dC_l,\, V}{dt} = \rho_l q_l - C_l,\, q_{\text{soln}}$$

$$\text{Solid:}\quad \frac{dC_{s,\text{soln}}V}{dt} = -C_{s,\text{soln}} q_{\text{soln}}$$

It is easy to see that if the wash out is done in such a way that the volume in the tank does not change, the concentration of salt is low enough to ignore the density effect, and the total differential material balance is zero, then the volume flow rate in is equal to that out and so the only equation we need to solve is the last one for the salt concentration.

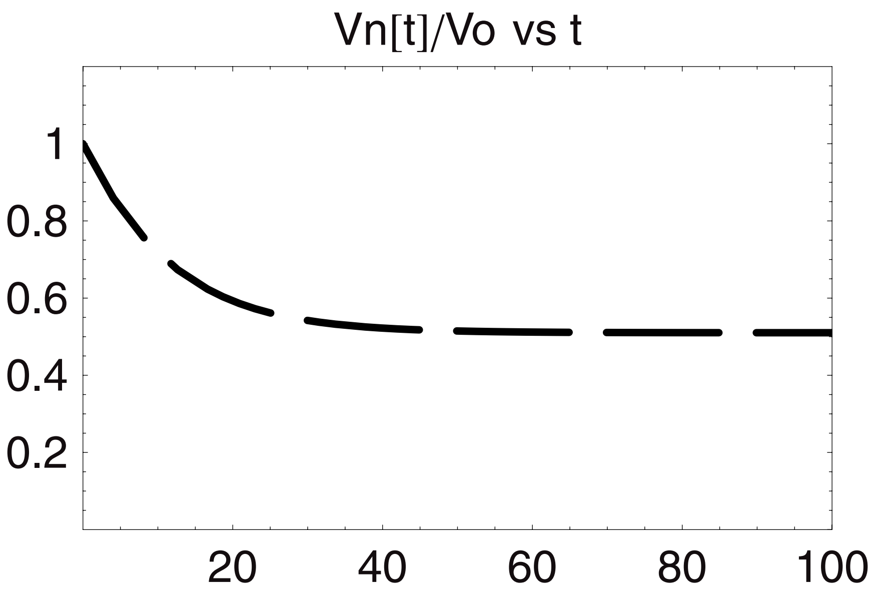
```
In[107]:= Remove[a4, qsoln, Vo, t, cs, cso, cl];
          a4 = DSolve[
                                (ρ1 - c1[t]) qsoln
            {cl'[t] ==  ──────────────────,
                                      Vo
            cl[0] == clo,
                            cs[t] qsoln
            cs'[t], == -───────────,
                                Vo
            cs[0] == cso},
            {cl[t], cs[t]},
           t];

          cl4[t_]:= Evaluate[cl[t] /. a4[[1]]];
          cs4[t_]:= Evaluate[cs[t] /. a4[[1]]];

         General::spell: Possible spelling error: new symbol name
         "ρl" is similar to existing symbols {ρ,ρo}.

                                        (ρl - cl4[t]) qsoln
In[111]:= Simplify[∂t cl4[t] == ────────────────────]
                                              Vo
                                       cs4[t] qsoln
          Simplify[∂t cs4[t] == -─────────────]
                                            Vo
Out[111]= True

Out[112]= True

In[113]:= ql = 10;
          A = 10;
          qsoln = ql;
          ho = 10;
          tmax = 100;
          Vo = A ho;
          cso = 0.5;
          ρl = 1;
          clo = 1 + 0.9 cso - cso;

          pla4C = Plot[{cl4[t], cs4[t]}, {t, 0, tmax},
            PlotStyle → {Dashing[{0, 0}], Dashing[{0.15, 0.05}]},
            PlotLabel → "Cs[t] Washout vs Time/min",
            Frame → True, PlotRange → {{0, tmax}, {0, cso + clo}}];
```



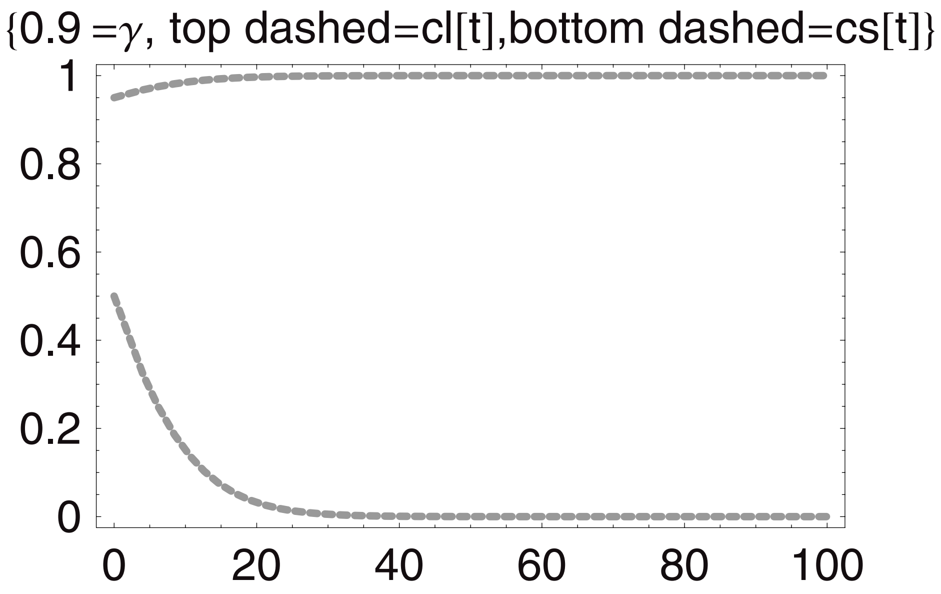
However, we will not make any of these simplifying assumptions. Instead, we will take the flow out as given by Torricelli's Law and the density of the solution to be a linear function of the salt concentration. Then all we need to do is to modify the numerical routine we had for the previous problem by eliminating the term for the salt feed:

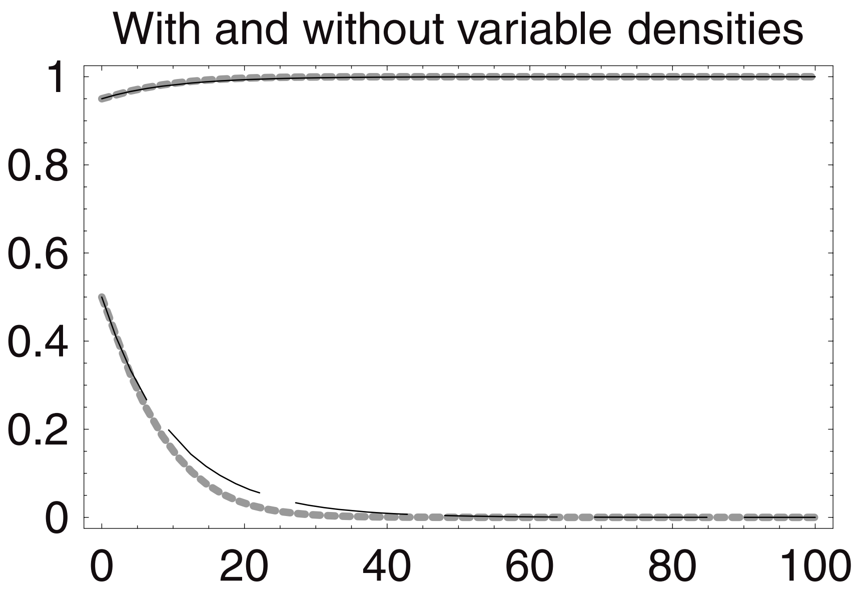
```
In[123]:= Remove[a5, ql, A, ρ, b, Ao, g, ho, Vo, V, t, cs, cl, γ];
          SetOptions[{Plot,ListPlot},AxesStyle → {Thickness[0.01]},
            PlotStyle → {PointSize[0.015], Thickness[0.006]},
             DefaultFont → {"Helvetica", 17}];
          V =.
          ql = 10;
          A = 10;
          m̊s = 5;
          ρo = 1;
          b = 1;
          Ao = 0.1 A;
          g = 9.8;
          ho = 10;
          tmax = 100;
```

```
Vo = A ho;
γ = .9;
cso = 0.5;

ρ[t_] := ρo + γcs[t]

a5 = NDSolve[{
    ∂t(V[t]ρ[t]) == ρo q1 - ρ[t] b Ao √(2g/A V[t]), V[0] == Vo,
    ∂t(cs[t]V[t]) == -cs[t] b Ao √(2g/A V[t]), cs[0] == cso,
    ∂t(cl[t]V[t]) == ρo ql - cl[t] b Ao √(2g/A V[t]),
     cl[0] == ρ[0] - cs[0]
  },
  {V[t], cs[t], cl[t]},
  {t, 0, tmax}];
Vn[t_] := Evaluate[V[t] /. a5]

pla5V = Plot[{Vn[t]/Vo}, {t, 0, tmax},
  PlotLabel → "Vn[t]/Vo vs t",
  PlotStyle → {{Dashing[{0.15, 0.05}], Thickness[0.01]}},
  PlotRange → {{0, tmax}, {0, 1.2}}, Frame → True];

csn[t_]:= Evaluate[cs[t] /. a5]
cln[t_]:= Evaluate[cl[t] /. a5]

pla5C = Plot[{cln[t], csn[t]}, {t, 0, tmax},
  PlotStyle → {{Dashing[{0.01, 0.01}], GrayLevel[0.6],
   Thickness[0.01]}, {Dashing[{0.01, 0.01}],
   GrayLevel[0.6], Thickness[0.01]}},
  AxesLabel → {"t/min", "c[t]"},
   PlotLabel → {γ"=γ", "top dashed=cl[t],
    bottom dashed=cs[t]"}, Frame → True];
Show[pla5C, pla4C, PlotLabel → "With and without
 variable densities"];
```



```
General::spell : Possible spelling error: new symbol name
"pla5C" is similar to existing symbol "pla5V".
```





We can see from the last graph, which compares the solution accounting for the change of density with concentration versus the simple solution without this taken into account, that for this value of $\gamma = 0.9$, the error made in the approximation is in fact quite small.

4.7 The Pulse Input Tracer Experiment and Analysis

The key assumption we have made throughout this chapter is that the solutions within the control volume are indeed either homogeneous or well mixed. Questions of the degree to which mixing occurs in a system arise in sciences as seemingly diverse as medicine and environmental engineering. If a system is well mixed, then when we inject a pulse of tracer, we should see a characteristic decay of the concentration in the system as a function of time. Cardiologists use this method to measuring the pumping speed of a heart by inserting a catheter and injecting a tracer of known volume into the heart. The rate of decay of the concentration within the chambers of the heart provides the flow rate away from this organ. Similarly, an environmental engineer may need to know the flow rate and mixing dynamics in a river or stream. By injecting a water-soluble and harmless dye into the flowing water, the diminution of the dye concentration at the point of "injection" can be used to visualize and then model the dynamics of the river's flow.

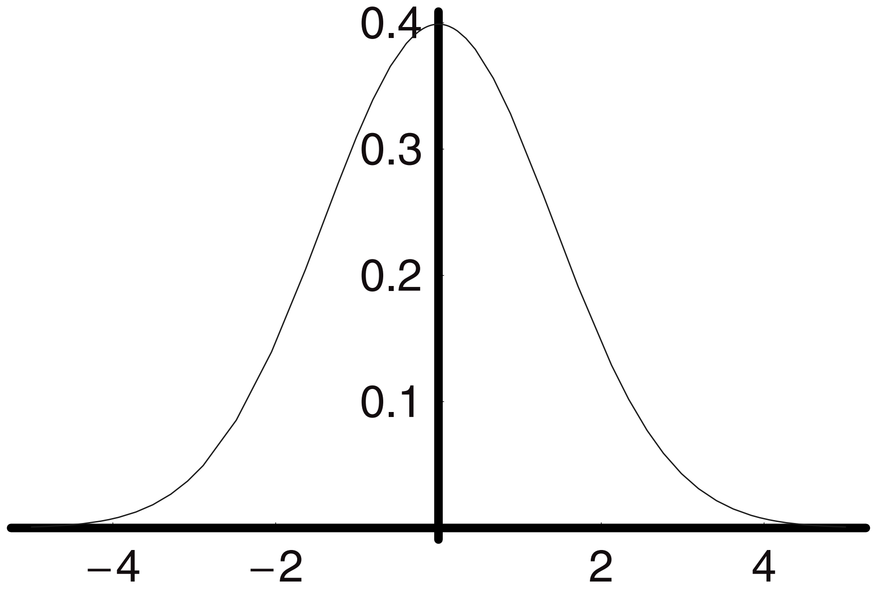
Chemical engineers also use this kind of experiment. It can be utilized to great advantage in chemical reactors to find the "residence time distribution" of the reactor, a crucial piece of information which links microscopic flow behavior, that is, fluid dynamics, to measurables of the system, such as chemical conversion and selectivity. For vessels that are not used for reaction processes, but are used for other operations that are also critically dependent upon mixing, this tracer experiment provides a great deal of insight into how the system behaves. We can analyze how a pulse of injected tracer would behave in the well-stirred vessel we have been analyzing here.

Imagine that an injection is made as a pulse of tracer, the concentration of which can be measured in the tank and in the exit stream as a function of time. For a laboratory vessel, the injection may be done by hand with a syringe full of tracer such as a dye or a radioactively tagged molecule. For larger vessels at pilot and production scale ingenious methods have been invented for putting a "pulse" of tracer into the unit. Ideally, the pulse should be added instantaneously, which means in as short a time period as possible. In other words, the time to add the tracer must be much shorter than the time required to "wash" it out of the unit.

For the case of the unit we have been considering, water would be flowing to the system continuously with stirring and the whole system would be at a steady state with respect to level and volume. The injection would be made at the top of the vessel with a very small volume of highly concentrated dye; nothing else would or should be done. The high concentration is critical to making the measurements accurate and precise. It also makes it possible to use only a small volume of the dye, which is important so that the steady state is maintained with respect to volume. Finally, small volume, high concentration injections can be done fast.

How can we model such a problem? To do the analysis we need to introduce and become comfortable with two new functions: the Dirac-Delta function and the UnitStep (or Heaviside) function. The Dirac-Delta function is infinitely intense and infinitesimally narrow—like a pulse of laser light. We can imagine it arising in the following way. We begin by considering a pulse that is quite broad, such as the function that is plotted here:

```
In[146]:= θ = 2
          Plot[Sqrt[1/(θ Pi)]Exp[-(t/θ)^2],
            {t,-5, 5}, PlotRange → All,
             Plotstyle → GrayLevel[0.1]];

Out[146]= 2
```

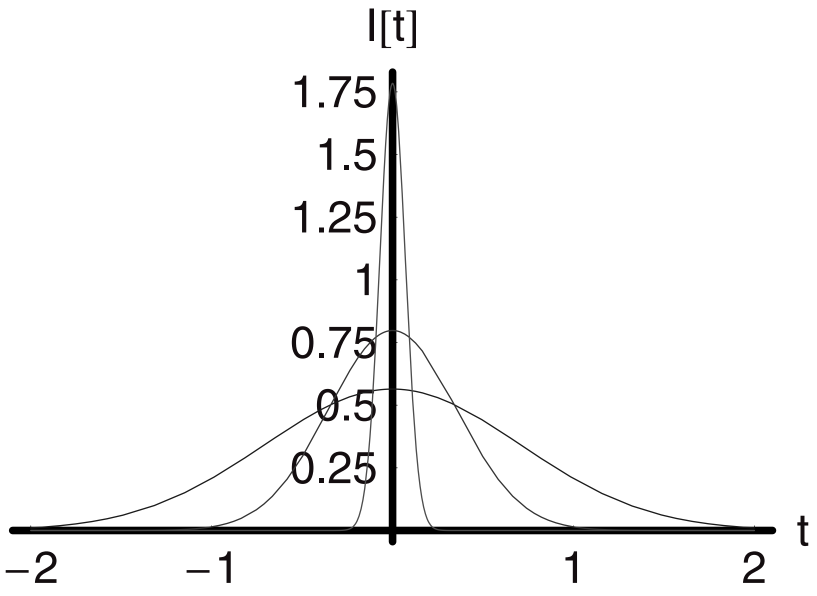


We can sharpen this function in time by decreasing the value of the "time constant" θ, as follows:

```
In[148]:= Clear[θ, f, a]
          f[θ_, t_] := N[Sqrt[1/(θ Pi)]Exp[-(t/θ)^2]]
          f[x, y]

          θ = {1, .5, .1};

          a = Table[
              Plot[f[θ, t][[n]], {t, -2, 2},
               PlotRange → All,
               DisplayFunction → Identity,
               PlotStyle → GrayLevel[.1 n],
               AxesLabel → {"t", "I[t]"}
              ],
              {n, 1, Length[θ]}];
```

Out[150]= $0.564189\,2.71828^{-\frac{1.y^2}{x^2}}\sqrt{\frac{1}{x}}$

```
In[153]:= Show[a, DisplayFunction → $DisplayFunction];
```



As we decrease the time constant the function becomes more intense in and around the $t = 0$. Doing this in the limit of $\theta \to 0$ transforms this into the infinitely intense pulse of infinitely short time duration. We can use this Dirac-Delta function, once we know more about its properties and how it is implemented in *Mathematica.*

If we begin with a simple Table function, we see that if we ask for "t" in the interval from −5 to 5, we get back a simple vector of those integers:

```
    In[154]:= Clear[t, x]
              Table[t, {t, -2, 2}]

    Out[155]= {-2, -1, 0, 1, 2}
```

Taking the product (xf[t]) over the same interval leads to a vector of elements, each one of which is the product of x and f[t] evaluated at the integer:

```
    In[156]:= Clear[t, x]
              Table[x f[t] t, {t, -2, 2}]

    Out[157]= {-2xf[-2], -xf[-1], 0, xf[1], 2xf[2]}
```

However, look at what happens when we take the product (x DiracDelta [t]) over the same range of t values:

```
In[158]:= Table[N[x DiracDelta[t]], {t, -5, 5}]

Out[158]= {0., 0., 0., 0., 0., x DiracDelta[0.], 0., 0., 0., 0., 0.}
```

The only element that is nonzero is that which falls at the point t = 0. This is because at every other point the Dirac-Delta function is identically zero by definition. This is the consequence of being infinitely intense and infinitely short in duration. We can display the position of this pulse by placing another integer in the argument of the Dirac-Delta as follows:

```
In[159]:= Table[N[x DiracDelta[t-5]], {t,-5, 5}]

Out[159]= {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., x DiracDelta[0.]}
```

In this case the nonzero value of the function has been pushed to the positive extremum of this interval on t. If we integrate the product of the dye mass m_{dye} and the Dirac-Delta, we obtain this:

```
In[160]:= Clear[t, m]

In[161]:= Integrate[DiracDelta[t] mdye, {t, -5, 5}]

Out[161]= mdye
```

The integration returns just m_{dye} integrated over time.

Now we can use this technique to determine how the concentration of dye changes as a function of time in a well-stirred vessel. We need only write and integrate the component balance on the dye to have the answer because the volume of the tank is assumed not to change with time. We will use the following equation for the rate of change of the dye mass:

$$\frac{d\, c_{dye}[t]}{dt} == \frac{m_{dye}\,\mathrm{DiracDelta}[t] - c_{dye}[t]\mathrm{qex}}{V}$$

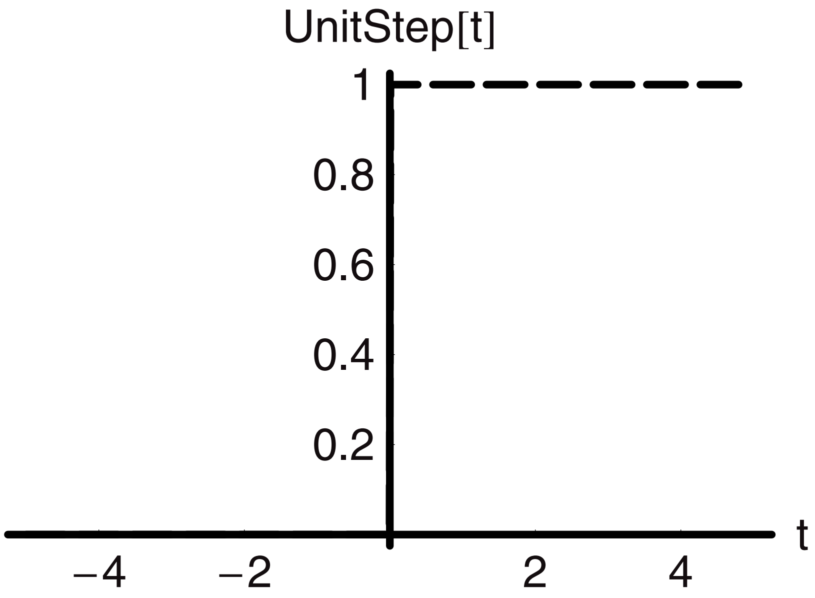
Notice how we have used the Dirac-Delta in the balance. At time t = 0 all the mass of the dye is injected instantaneously. At all other times the term for dye input is identically zero. We can integrate this analytically:

```
In[162]:= Clear[qex, t, to, V, cdye]
        General::spell1 : Possible spelling error: new symbol name
        "cdye" is similar to existing symbol "dye".

In[163]:= Simplify[
            DSolve[
              {cdye'[t] == (mdye DiracDelta[t] - cdye[t] qex)/V,
               cdye[to] == 0}, cdye[t], t]
                 ]
```

$$\text{Out[163]= }\{\{\text{cdye}[t] \to \frac{e^{-\frac{\text{qex}\,t}{V}}\, m_{dye}(\text{UnitStep}[t] - \text{UnitStep}[to])}{V}\}\}$$

We find that the integration looks like an exponential decay except that now a new function has appeared—the UnitStep function. To see how the UnitStep function behaves in time we can plot it as shown here:

```
In[164]:= Plot[UnitStep[t], {t, -5, 5},
           AxesLabel → {"t", "UnitStep[t]"},
           PlotStyle → {Thickness[0.01], Dashing[{0.05, 0.02}]}];
```

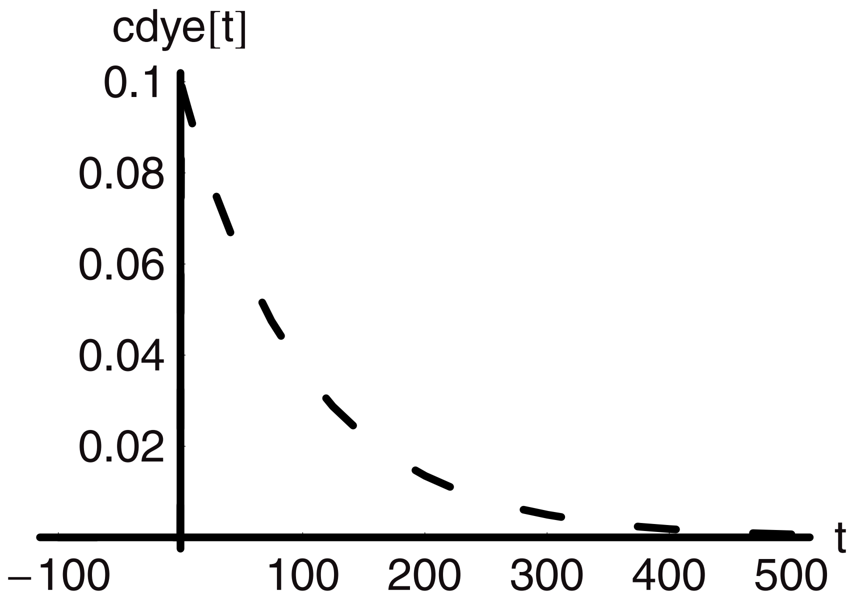


The UnitStep function is everywhere zero until it comes to t = 0 and then it goes to a value of unity, which maintains *ad infinitum*. We can rewrite the solution as a function of time:

```
In[165]:= Clear[qex, t, to, V, cdye, m]
```

In[166]:= cdye[t_] := $\dfrac{e^{-\frac{qex\,t}{V}}\, m_{dye}\,(\text{UnitStep[t] - UnitStep[to]})}{V}$

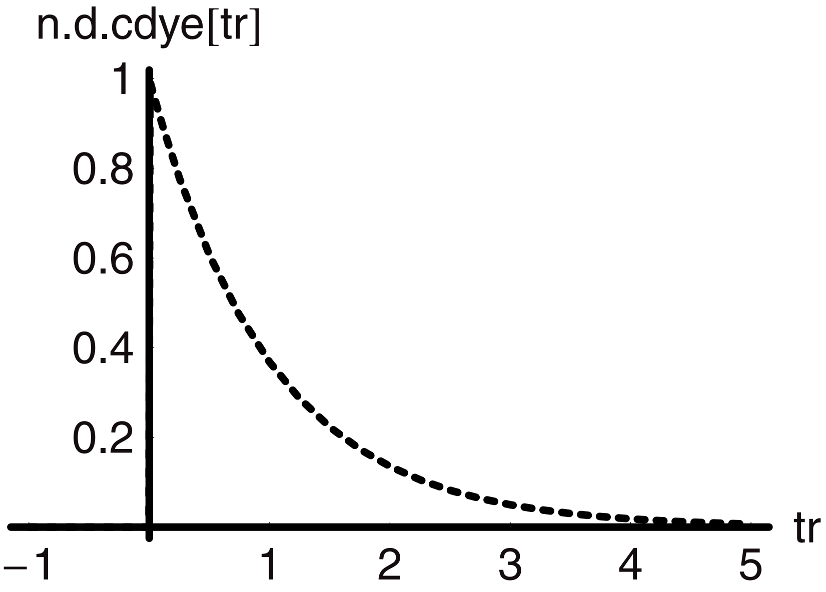
```
          mdye = 10;
          qex = 1;
          V = 100;
          to = -5;
          pl1 = Plot[cdye[t], {t, -100, 500},
             AxesLabel → {"t", "cdye[t]"},
             PlotStyle → {{Thickness[0.01], Dashing[{0.05, 0.1}]}}];
```



Here we see that the pulse is injected at time "zero," the concentration instantaneously jumps discontinuously to a magnitude of 0.1, and then begins to descend exponentially as a function of time. This is the characteristic curve we should see if the unit is well mixed.

To make this result more general we can nondimensionalize both axes. The concentration of dye can be referenced to the maximum concentration at time zero $c_{dye}[0]$. But what of the time axis? How shall we nondimensionalize this? We will use the "holding time" as the reference time. The holding time is the time required for a volume of liquid equal to the volume of the unit to pass entirely through the unit. This is the ratio of the volume to the flow rate, that is, $\tau = \frac{V}{\text{qex}}$. We can remake the graph in nondimensional form:

```
In[172]:= ṁdye = 10;
          qex = 1;
          V = 100;
          tro = -1;
                V
          τ = ───;
               qex
          cdye[0];

                          e^-tr ṁdye (UnitStep[tr] - UnitStep[tro])
          ndcdye[tr_] := ──────────────────────────────────────────
                                         cdye[0] V

          Plot[
            ndcdye[t],
            {t, -1, 5},
            AxesLabel → {"tr", "n.d.cdye[tr]"},
            PlotStyle → {Dashing[{0.01, 0.015}], Thickness[0.01]}
              ];
```



The time axis is the reduced time, which is the ratio of real time to the holding time tr $= \frac{t}{\tau}$ and the y-axis is the nondimensionalized dye concentration. If we were to plot the experimental change in nondimensionalized concentration versus reduced time, it should fall very near to this curve. The extent to which the real system deviates is a measure of the degree to which the system veers from the ideally "well-mixed" limit.

4.8 Mixing

Consider the following case, illustrated in Figure 4. In this experiment everything is the same as in the last one, except that there are two well-mixed tanks rather than one. The same overall flow qex is diverted through the two units with a 50:50 split and the same mass of tracer or dye is added instantaneously to the inlet flow. The volumes of the lines to and from the tanks are considered to be negligible. Each tank has exactly 50% of the volume of the previous tank, and therefore their residence times are half that of the one large tank with the same overall volume. The lines from the two tanks come together prior to the analysis and are assumed to be perfectly mixed when they do. The question is: Will the time distribution of the tracer concentration look the same or different from that of one well-mixed tank?

We do not need to redo the analysis. Instead we will take the solution for one well-mixed tank and apply it to the two tanks. We have to be careful about handling the splits at the input side where the stream divides into two and again when the two separate tank streams come back as one. The overall input is divided into the two flows q1 and q2 by the fraction a and b. The mass of the dye trace also will be split in the same way. This is done because the mass of dye into the first splitter must be the same as the total mass of dye out, which is just the sum

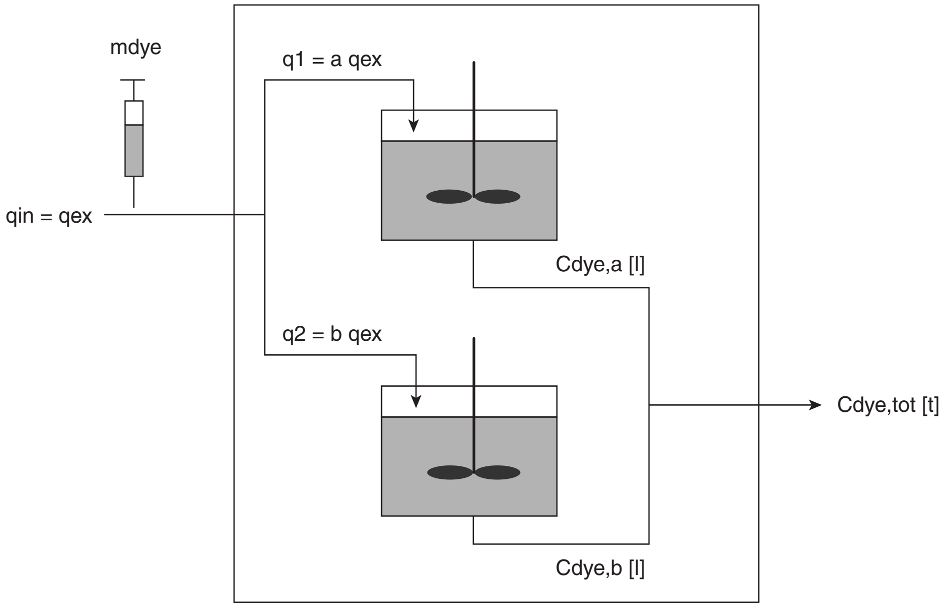


Figure 4

of the masses in the two streams:

$$\begin{aligned} m_{\text{dye,in}} &= C_{\text{dye,in}}\text{qin}\,\delta\text{t} \\ &= C_{\text{dye,1}}\text{q1}\delta\text{t} + C_{\text{dye,2}}\text{q2}\,\delta\text{t} \\ &= \text{a}\,C_{\text{dye,1}}\text{qex}\,\delta\text{t} + \text{b}\,C_{\text{dye,2}}\text{qex}\,\delta\text{t} \\ &= (\text{a}\,C_{\text{dye,1}} + \text{b}\,C_{\text{dye,2}})\,\text{qex}\,\delta\text{t} \end{aligned}$$

where δt is the instantaneous or infinitesimally short duration of the Dirac-Delta function pulse. This is also the time for the mass of the dye to hit each tank, that is, zero time.

The concentration of dye in each tank is exactly the same solution we derived already and it is now applied individually to each vessel:

$$\text{c dye 2a[t\_]} := \frac{\text{a}\,e^{-\frac{\text{a qex t}}{\text{V1}}}\,\text{m}_{\text{dye}}(\text{Unit Step[t]} - \text{Unit Step[to]})}{\text{V1}}$$

$$\text{c dye 2b[t\_]} := \frac{\text{b}\,e^{-\frac{\text{b qex t}}{\text{V2}}}\,\text{m}_{\text{dye}}(\text{Unit Step[t]} - \text{Unit Step[to]})}{\text{V2}}$$

Notice that we have rewritten q1 and q2 in the arguments of the exponentials as “a qex” and “b qex,” where “a” and “b” are the splits fractions that set the stream flows. We can compute these two concentrations and plot them as functions of time, but what we really want is the time dependence of the dye concentration after the two streams are recombined. Our goal is to compare the overall effect of two tanks on the tracer with that of one tank given that the sum of the volumes of the two are the same as the one. We also want to be able to split the flows between these two tanks in different ratios and with different tank volumes to see how this affects overall time dependence. In other words, what if we did not know there were two tanks? What if all we knew were the inputs and the outputs at the dotted box around the two tanks? Would we be able to detect a difference for this system versus the one tank system on the basis of the tracer experiment? To find out we continue our analysis.

To find the concentration at the point where the streams come back together we again apply the conservation of mass. The mass of dye in the two lines coming into this mixing point per unit time must be equal to the total mass going out of it per unit time. The rate of mass flow in is just the sum of the products of the concentrations and flow rates of the two streams exiting the tanks, while the rate of mass flow out is the concentration of the dye times the total flow rate. The mathematical statement is much more succinct:

$$(\mathrm{a\ Cdye2a\ [t]} + \mathrm{b\ Cdye2b\ [t]})\mathrm{qex} = \mathrm{Cdye2tot\ [t]\ (a + b)\ qex}$$

$$\mathrm{Cdye2tot\ [t]} = \frac{(a\,\mathrm{Cdye2a}[t] + b\,\mathrm{Cdye2b}[t])}{(a + b)}$$

This last concentration is what we can measure if the tanks inside the outer box are hidden from view, so this is the computation we want to make and to compare to the first case of one unit in plain view. We should also do one more calculation to be sure we are not making any errors. The mass of dye into the units must eventually come back out. Therefore, we should integrate the product of the exit concentrations and flows from each tank and sum these to be sure it is equal to the dye input mass. Yet another application of the conservation of mass.

The code for doing all of this is shorter than the description of it. Once it is written we can use it over and over again. We could also put it into nondimensional form if we chose to, but instead we will make our comparisons in real time and concentration. Recall that "a" and "b" are the splits—these are entered as fractions, but they must sum to unity!

To check ourselves, in the first case we set the splits to one-half each and the volumes are equal. We compare the concentration versus time curve for this case versus that for the one tank. We do all the calculations for the two-tank case and then end with a graph comparing it to the one-tank case (see the following graphs):

```
In[180]:= Clear[a, b]

In[181]:= Remove[cdye2b]

In[182]:= a = 1/2;
          b = 1/2;
```

$$\mathtt{cdye2a[t\_]\ :=} \frac{\mathtt{a}\, e^{-\frac{\mathtt{a\ qex\ t}}{\mathtt{V1}}}\, \mathtt{m_{dye}\,(UnitStep[t]\ -\ UnitStep[to])}}{\mathtt{V1}}$$

$$\mathtt{cdye2b[t\_]\ :=} \frac{\mathtt{b}\, e^{-\frac{\mathtt{b\ qex\ t}}{\mathtt{V2}}}\, \mathtt{m_{dye}\,(UnitStep[t]\ -\ UnitStep[to])}}{\mathtt{V2}}$$

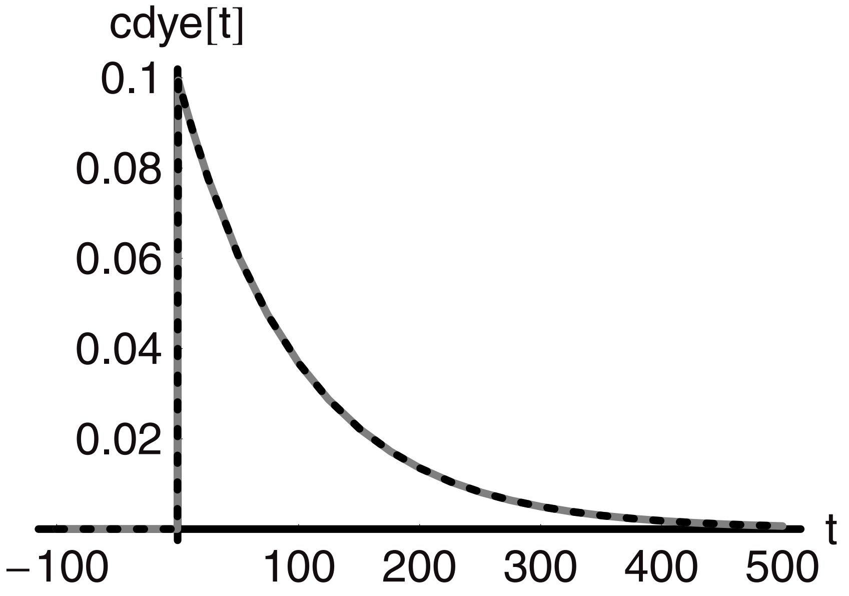
$\mathtt{m_{dye}\ =\ 10;}$

```
qex = 1;
V1 = 50;
V2 = 50;

to = -5;
pl2 = Plot[{cdye2a[t], cdye2b[t]}, {t, -100, 500},
   AxesLabel → {"t", "cdye[t]"},
    PlotStyle → {{Thickness[0.01], GrayLevel[0.5]},
     {Thickness[0.01], Dashing[{0.01, 0.025}]}},
      PlotRange → All];
```

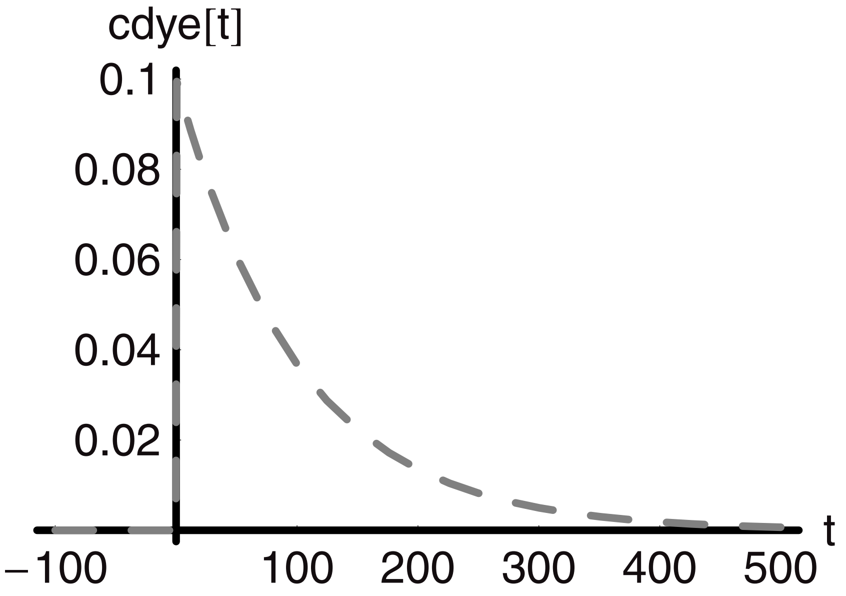
$$\texttt{pl2b = Plot[\{}\frac{\texttt{a cdye2a[t] + b cdye2b[t]}}{\texttt{a + b}}\texttt{\},\{t,-100,500\},}$$

```
   AxesLabel → {"t", "cdye[t]"},
    PlotStyle → {{Thickness[0.01], GrayLevel[0.5],
    Dashing[{0.05, 0.05}]}}, PlotRange → All]
NIntegrate[cdye2a[t] a qex, {t, -100, 400}];
NIntegrate[cdye2b[t] b qex, {t, -100, 400}];

% + %%
NIntegrate[cdye[t] qex, {t, -100, 400}]
Show[pl2b, pl1];
```

```
General::spell1: Possible spelling error: new symbol name
"cdye2b" is similar to existing symbol "cdye2a".
```

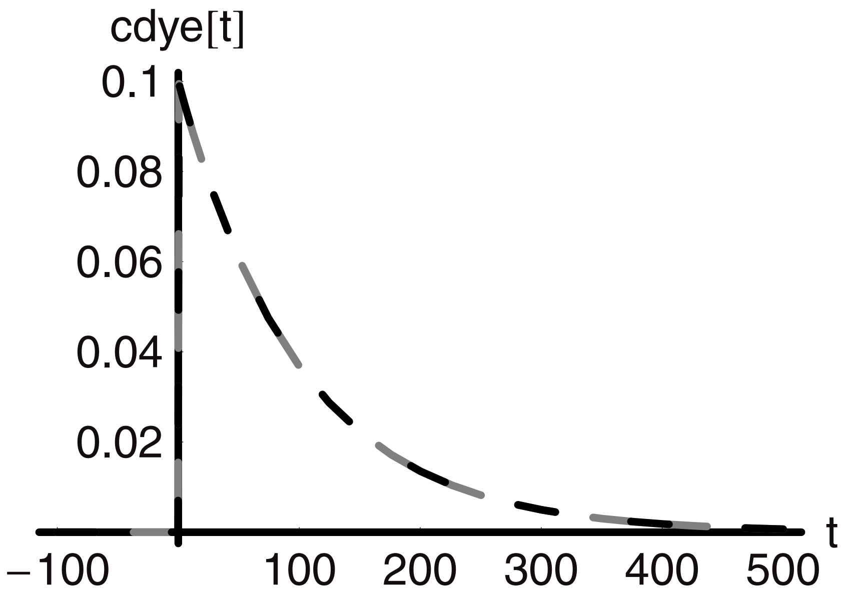


```
General::spell1: Possible spelling error: new symbol name
"pl2b" is similar to existing symbol "pl2".
```



```
Out[192]= -Graphics-

      NIntegrate::slwcon :
       Numerical integration converging too slowly; suspect one
         of the following: singularity, value of the integration
         being 0, oscillatory integrand, or insufficient
         WorkingPrecision. If your integrand is oscillatory try
         using the option Method→Oscillatory in NIntegrate.

      NIntegrate::ncvb :
       NIntegrate failed to converge to prescribed accuracy
         after 7 recursive bisections in t near t = -0.390625.

      NIntegrate::slwcon :
       Numerical integration converging too slowly; suspect one
         of the following: singularity, value of the integration
         being 0, oscillatory integrand, or insufficient
         WorkingPrecision. If your integrand is oscillatory
         try using the option Method→Oscillatory in NIntegrate.

      NIntegrate::ncvb :
       NIntegrate failed to converge to prescribed accuracy
         after 7 recursive bisections in t near t = -0.390625.
```

```
Out[195]= 9.82826

          NIntegrate::slwcon :
           Numerical integration converging too slowly; suspect
            one of the following: singularity, value of the
            integration being 0, oscillatory integrand, or
            insufficient WorkingPrecision. If your integrand
            is oscillatory try using the option
            Method → Oscillatory in NIntegrate.

          NIntegrate::ncvb :
           NIntegrate failed to converge to prescribed accuracy
            after 7 recursive bisections in t near t = -0.390625.

Out[196]= 9.82826
```



We see that the tracer curves all overlap perfectly and that the integrals are all approaching 0 after 400 time units. Therefore, the code is working and our derivations are verified. Now we can turn to a more relevant case. We will assume that the flows are not evenly split, but a is 2/3 and b is 1/3. Take the volumes to be different: We will set V1 to 20 and V2 to 80. (Recall that V1 and V2 have to sum to the same value as that of V in the one-vessel case if we are to make valid comparisons.) To demonstrate, follow the In and Out statements and the following graphs:

```
In[198]:= Clear[a, b]

In[199]:= Remove[cdye2b]

In[200]:= a = 2/3;
          b = 1/3;
```

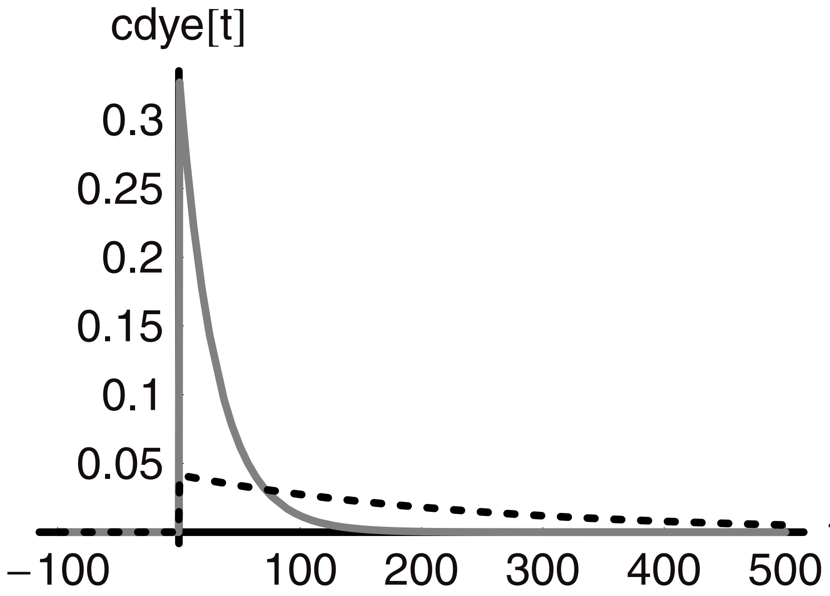
$$\texttt{cdye2a[t\_] :=} \frac{a\, e^{-\frac{a\,qex\,t}{V1}} m_{dye}\,(\texttt{UnitStep[t] - UnitStep[to]})}{V1}$$

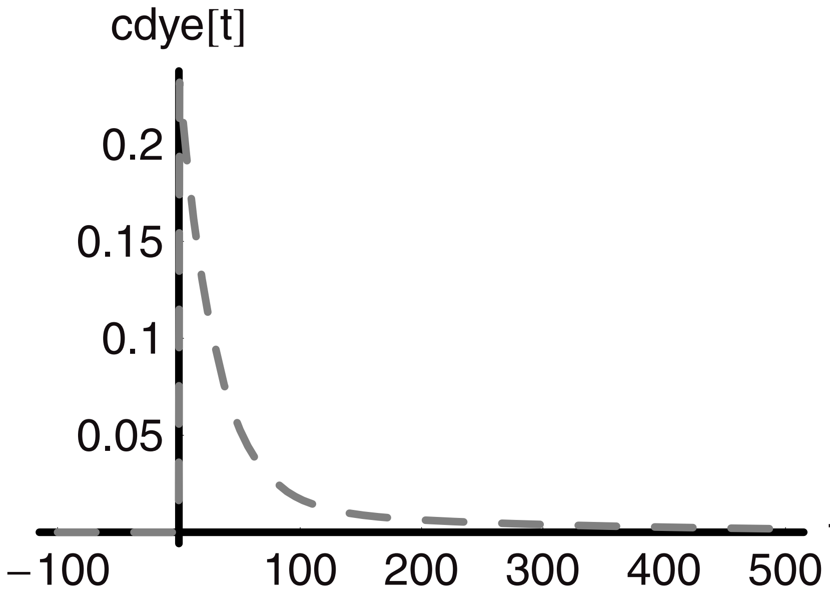
$$\texttt{cdye2b[t\_] :=} \frac{b\, e^{-\frac{b\,qex\,t}{V2}} m_{dye}\,(\texttt{UnitStep[t] - UnitStep[to]})}{V2}$$

```
          mdye = 10;
          qex = 1;
          V1 = 20;
          V2 = 80;

          to = -5;
          pl2 = Plot[{cdye2a[t], cdye2b[t]}, {t, -100, 500},
             AxesLabel → {"t", "cdye[t]"},
             PlotStyle → {{Thickness[0.01], GrayLevel[0.5]},
             {Thickness[0.01], Dashing[{0.01, 0.025}]}},
              PlotRange → All];
```

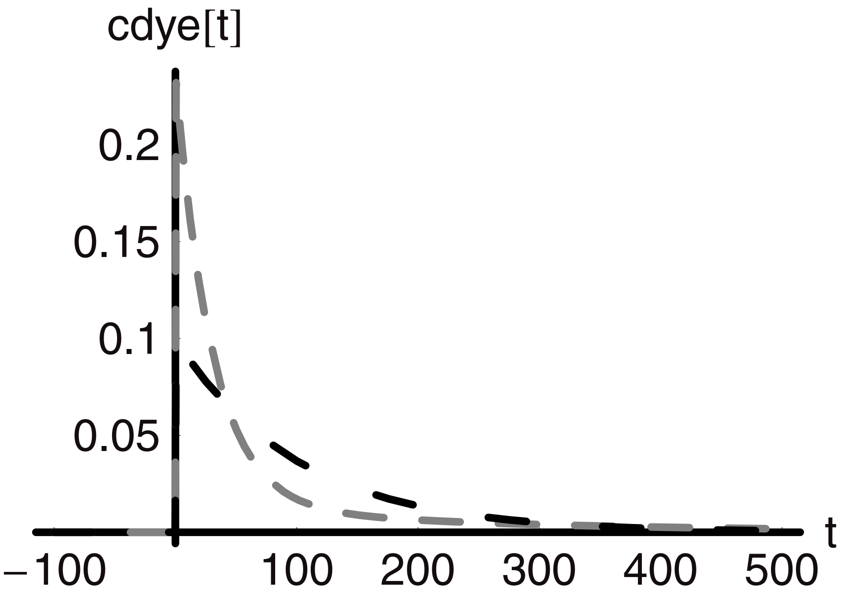
$$\texttt{pl2b = Plot[\{}\frac{\texttt{a cdye2a[t] + b cdye2b[t]}}{\texttt{a + b}}\texttt{\}, \{t, -100, 500\},}$$

```
            AxesLabel → {"t", "cdye[t]"},
            PlotStyle → {{Thickness[0.01], GrayLevel[0.5],
             Dashing[{0.05, 0.05}]}}, PlotRange → All]
          NIntegrate[cdye2a[t] a qex, {t, -100, 400}];
          NIntegrate[cdye2b[t] b qex, {t, -100, 400}];

          % + %%
          NIntegrate[cdye[t] qex, {t, -100, 400}]
          Show[pl2b, pl1];

         General::spell1 : Possible spelling error: new symbol
          name "cdye2b" is similar to existing symbol "cdye2a".
```


cdye[t]
0.3
0.25
0.2
0.15
0.1
0.05
t
−100
100
200
300
400
500


cdye[t]
0.2
0.15
0.1
0.05
t
−100
100
200
300
400
500

```
Out[210]= -Graphics-

        NIntegrate::slwcon :
         Numerical integration converging too slowly; suspect one
           of the following: singularity, value of the integration
           being 0, oscillatory integrand, or insufficient
           WorkingPrecision. If your integrand is oscillatory try
           using the option Method →Oscillatory in NIntegrate.

        NIntegrate::ncvb :
         NIntegrate failed to converge to prescribed accuracy
           after 7 recursive bisections in t near t = -0.390625.

        NIntegrate::slwcon :
         Numerical integration converging too slowly; suspect one
           of the following: singularity, value of the integration
           being 0, oscillatory integrand, or insufficient
           WorkingPrecision. If your integrand is oscillatory
           try using the option Method →Oscillatory in NIntegrate.

        NIntegrate::ncvb :
         NIntegrate failed to converge to prescribed accuracy
           after 7 recursive bisections in t near t = -0.390625.

Out[213]= 9.39734

        NIntegrate::slwcon :
         Numerical integration converging too slowly; suspect
           one of the following: singularity, value of the
           integration being 0, oscillatory integrand, or
           insufficient WorkingPrecision. If your integrand is
           oscillatory try using the option Method →Oscillatory
           in NIntegrate.

        NIntegrate::ncvb :
         NIntegrate failed to converge to prescribed accuracy
           after 7 recursive bisections in t near t = -0.390625.

Out[214]= 9.82826
```



We see that the time dependence of the tracer in this case is markedly different, sharper and narrower overall. Why? Because 2/3 of the flow is shunted through the small vessel with a much shorter holding time. Therefore, even though the total flows through the two systems with equal volumes are the same, the behavior is quite different. Therefore, even if we could not see the two tanks, we would have to know that there was a very different flow mechanism in this second case versus the case of one well-mixed tank of equal volume. Try other values of a and b as well as V1 and V2 to see what happens.

To extend this model to three tanks would be straightforward, but so too would it be to extend it to *n* tanks where *n* was large. One of the points to note about the equations is that the argument of the exponential term is a ratio of the actual time to the holding time in each unit because the holding time in the *n*th unit θ_n is $\frac{V_n}{q_n}$, that is, the ratio of the flow volume of the unit to the flow rate through it;

$$\begin{aligned}\text{cdye n[t]} &= \frac{\text{an}\, e^{-\frac{\text{t qn}}{\text{Vn}}} m_{\text{dye}}(\text{UnitStep[t]} - \text{UnitStep[to]})\text{qtot}}{\text{Vn}}\\ &= \frac{\text{an}\, e^{-\frac{\text{t}}{\theta\text{n}}} m_{\text{dye}}(\text{UnitStep[t]} - \text{UnitStep[to]})}{\theta\text{n}}\end{aligned}$$

Now if we kept the total volume and total flow rate through these *n* different units the same (as shown in Figure 5) as for the one-unit case, then we could have a very different tracer concentration-time curve, depending on the distribution of the flows and the volumes within the green box. There would be *n* different holding times in this overall unit, but the average

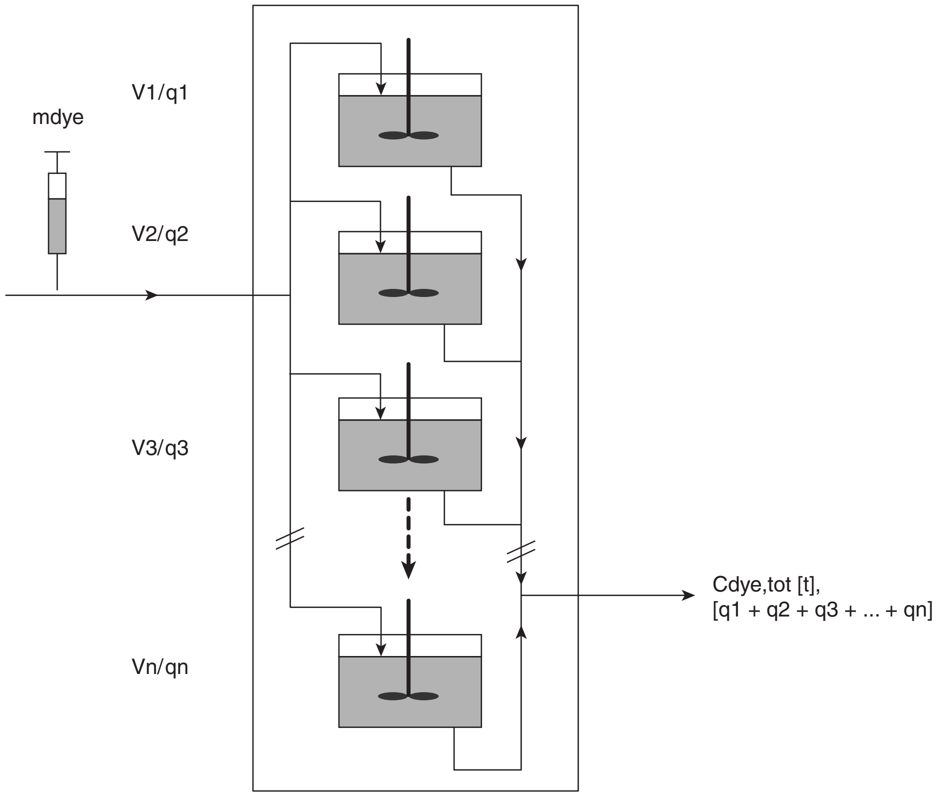


Figure 5

would be the same as the single vessel. If *n* were a discrete number of units, then there would be a discrete distribution of holding times. But as *n* grew larger, say toward infinity, the volume in any one unit would be infinitely small and so too would be the holding time in each. At this point the discrete distribution could be described nicely by one that was continuous in the holding time. The key would be to know how those discrete volumes, and hence residence times, making up the total volume were distributed.

The model we constructed in Figure 5 can be thought of as a metaphor for one unit with incomplete mixing. Rather imagine that in this poorly mixed unit some of the fluid goes through faster than the average holding time and some slower. Thus depending on the path taken, the fluid may spend more or less time in the unit than we would predict from the calculation of the holding time. These times are the residence times of the fluid elements in the unit. Such residence times come about due to the coupling of the fluid's mechanical properties with the geometry of the vessel and the type and energy of mixing. It is not uncommon to find that even in an apparently well-mixed unit the fluid moves through some regions of longer residence time due to recirculation cells, and other regions of shorter times due to bypassing (see Figure 6).

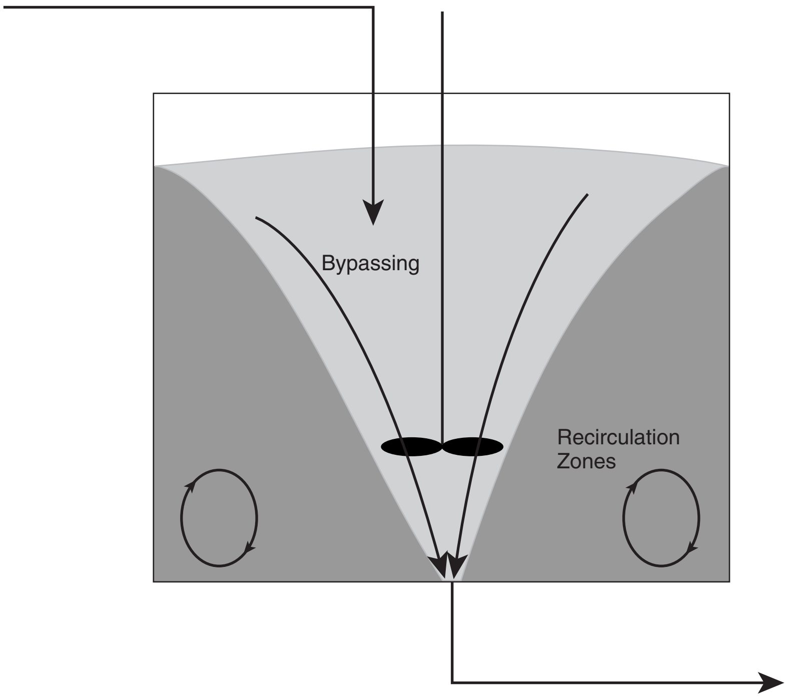


Figure 6

As we have said, the key to the analysis of a system like this one is to have a function that approximates to the actual residence time distribution. The tracer experiment is used to find that distribution function, but we will work from an assumed function to the tracer concentration-time curve to see what the experimental outcome might look like.

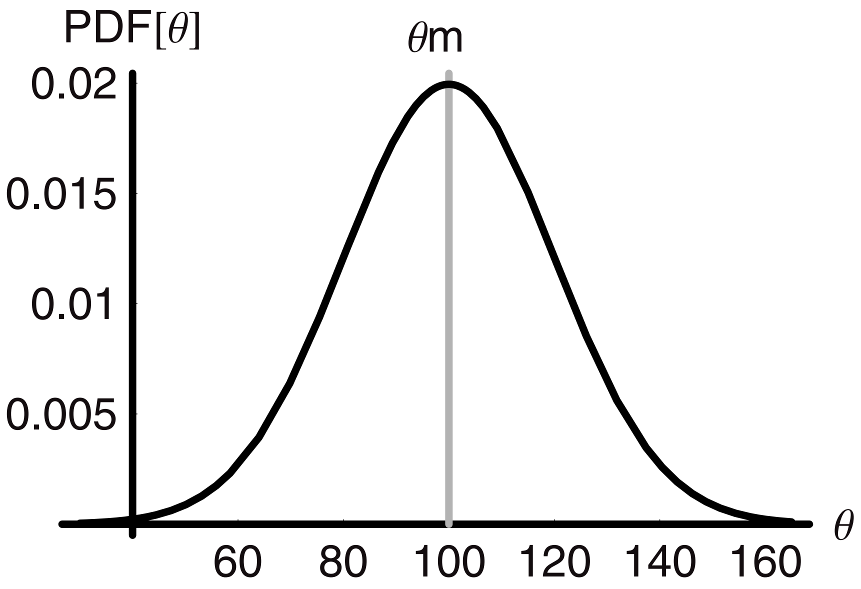
A good distribution function to examine in this context is the Normal or Gaussian distribution. Using this function, we would take the residence times θ to be normal distributed around some mean value θm and with a standard deviation or spread of $\delta\theta$:

$$\text{NormalDistribution}[\theta\text{m}, \delta\theta] = \frac{e^{-\frac{(\theta-\theta\text{m})^2}{2\,\delta\theta^2}}}{\sqrt{2\pi}\,\delta\theta}$$

Mathematica has this function and many others built into its set of "add-on" packages that are standard with the software. To use them we load the package "Statistics‘NormalDistribution‘. The syntax for these functions is straightforward: we specify the mean and the standard deviation in the normal distribution, and then we use this in the probability distribution function (PDF) along with the variable to be so distributed. The rest of the code is self-evident.

```
In[216]:= <<Statistics`NormalDistribution`

In[217]:= Remove[θ, θmin, θmax]

In[218]:= θm = 100;
          δθ = 20;
          ndist = NormalDistribution[θm, δθ];
          pd1 = PDF[ndist, θ];
          pd1
          Plot[pd1, {θ, 30, 165},
            AxesLabel → {"θ", "PDF[θ]"},
            PlotStyle → Thickness[0.01],
            Epilog → {GrayLevel[0.7], Thickness[0.01],
             Line[{{100, 0}, {100, 0.022}}]},
             PlotLabel → "θm"
              ];
```

Out[222]= $\frac{e^{-\frac{1}{800}(-100+\theta)^2}}{20\sqrt{2\pi}}$



With the mean value 100 and the deviation 20 time units the distribution has a familiar look (see the preceding graph). The function tells us that most of the fluid elements (63%) go through the unit with residence times that are between 60 and 140 time units. There are, however, 18.5% of the fluid elements that bypass with very short residence times and 18.5% that take very long times to emerge due to recirculation cells. Some of these never emerge!

Now that we have a model for the residence-time distribution, how shall we use this in the analysis of the unit? We need weighting factors for each residence time. These come from the PDF itself. For example, if we integrate the PDF between any two residence times, we obtain the probability density for that range of times:

```
In[224]:= Remove[θ, θmin, θmax]
In[225]:= θm = 100;
          δθ = 20;
          ndist = NormalDistribution[θm, δθ];
          pd2 = PDF[ndist, θ];
          NIntegrate[pd2, {θ, 50, 80}]
Out[229]= 0.152446
```

This result states that the fraction of residence times between 50 and 80 time units is just over 0.15. That would be the weighting factor for the flows with that range of residence times. If there are *n* residence times, then as we have seen there are *n* weighting factors. If the number of residence times is large then *n* tends toward infinity and the distribution of residence times is continuous. We can obtain the weighting factor for the whole of the distribution by integrating the probability density function over the range of residence times. In fact, we can see from the form of the equations, which will actually be the PDF over the residence time, that we must integrate since the form of the equation becomes:

$$\partial_t \, \text{cndis}[t] = \frac{\text{pdf}}{\theta}(\text{m}_{\text{dye}}\text{DiracDelta}[\text{t}] - \text{cndis}[\text{t}])$$

We will integrate over θ and then over t to solve the problem. This is done in what follows in two steps below for clarity and with specific values for the mean residence time and its deviation about the mean.

```
In[230]:= Remove[cndis, t, θ, θmin, θmax]
In[231]:= ndist = NormalDistribution[θm, δθ];
          pd3 = PDF[ndist, θ];
          Integrate[pd3/θ, {θ, θmin, θmax}]
          θm = 100;
          δθ = 22;
          θmin = 0;
          θmax = 5θm;
          ndist = NormalDistribution[θm, δθ];
          pd4 = PDF[ndist, θ];
```

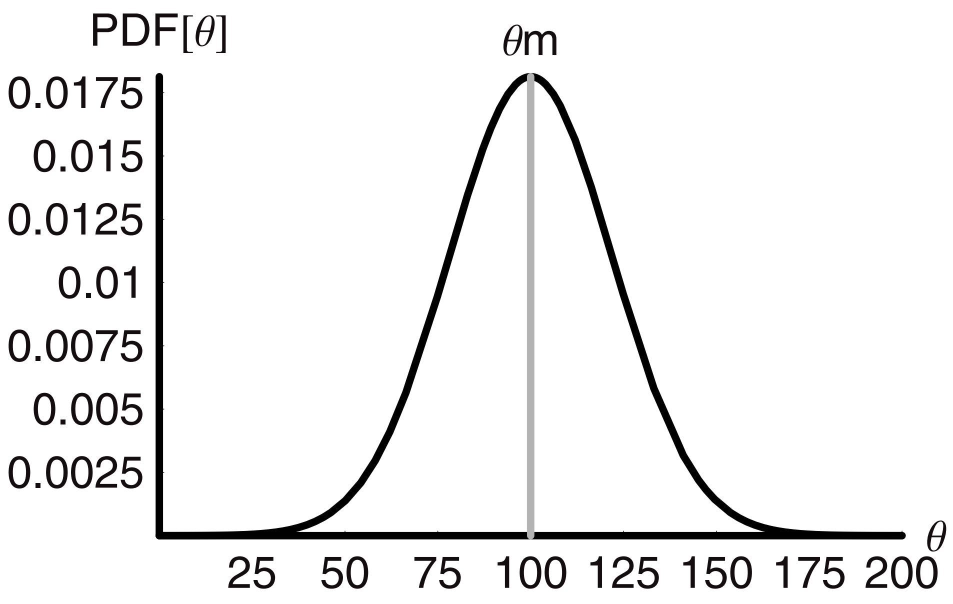
$$\texttt{wf = N[}\frac{\int_{\theta\text{min}}^{\theta\text{max}} \frac{e^{-\frac{(\theta-\theta\text{m})^2}{2\delta\theta^2}}}{\theta}\, d\theta}{\sqrt{2\pi}\delta\theta}\texttt{];}$$

```
          pl1 = Plot[pd4,
              {θ, .0001θm, 2θm},
            AxesLabel  → {"θ", "PDF[θ]"},
            PlotStyle  → Thickness[0.01],
            PlotRange  → {{0, 2θm}, {0, Max[Table[N[pd4],
             {θ, θmin, θmax}]]}},
            Epilog  → {GrayLevel[0.7], Thickness[0.01],
               Line[{{θm, 0}, {θm, Max[Table[N[pd4],
               {θ, θmin, θmax}]]}}]},
        PlotLabel  → "θm", DisplayFunction  → Identity];

        Show[pl1, DisplayFunction  → $DisplayFunction];
```

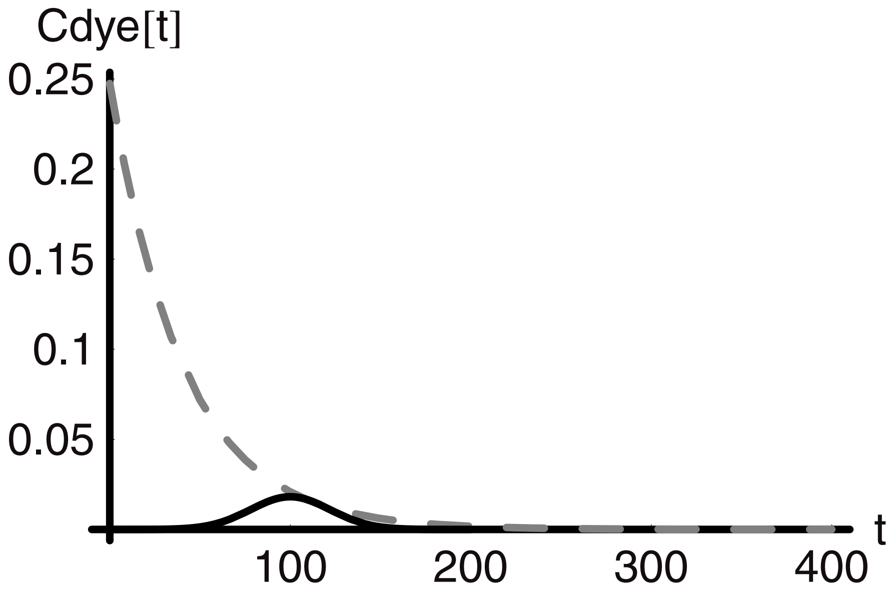
Out[233]= $\dfrac{\int_{\theta\,\min}^{\theta\,\max} \frac{e^{-\frac{1}{800}(-100+\theta)^2}}{\theta}d\theta}{20\sqrt{2\pi}\,\delta}$

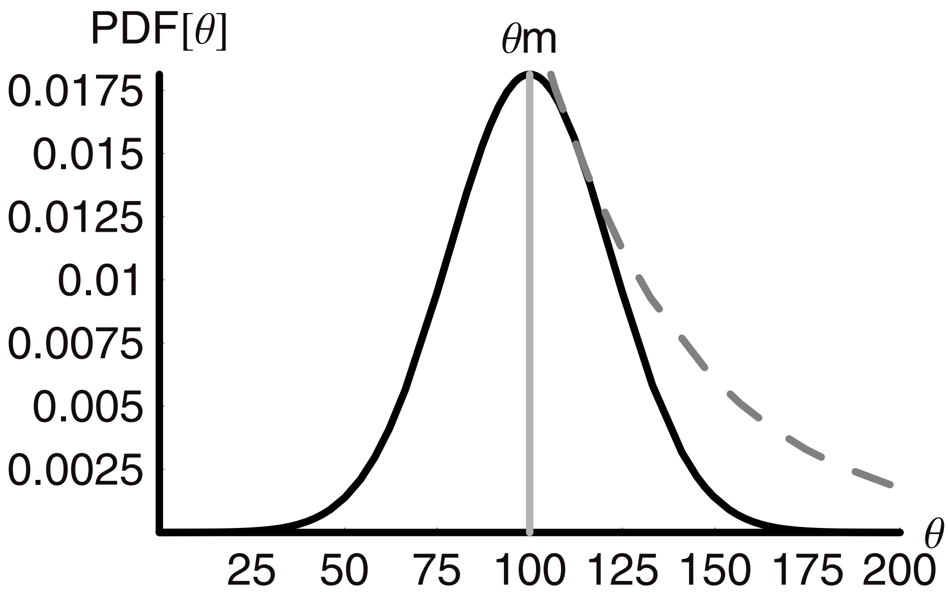
Integrate::idiv : Integral of $\frac{e^{-\frac{1}{968}(-100+\theta)^2}}{\theta}$ does not converge on {0, 500}.

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect
   one of the following: singularity, value of the
   integration being 0, oscillatory integrand, or
   insufficient WorkingPrecision. If your integrand
   is oscillatory try using the option
   Method →Oscillatory in NIntegrate.

NIntegrate::ncvb: NIntegrate failed to converge to
 prescribed accuracy after 7 recursive bisections
 in θ near θ = 2.1849968739518537‘ * ^ − 54.
```



```
In[243]:= Simplify[
            a = DSolve[
               {cndis'[t] == (wf)(mdye DiracDelta[t] - cndis[t]),
                cndis[to] == 0}, cndis[t], t]
                       ]
           cnd[t_, to_] := Evaluate[cndis[t] /. a]
           cnd[x, y]

           mdye = 10;
           to = -100;

Out[243]= {{cndis[t] → 0.247489 e^-0.0247489t UnitStep[t]}}
Out[245]= {0.247489 e^-0.0247489x UnitStep[x]}

In[248]:= plndis
             = Plot[
               cnd[t, to], {t, 0, 4θm},
               PlotRange → All,
               PlotStyle → {{Thickness[0.01], GrayLevel[0.5],
                Dashing[{0.05, 0.05}]}},
               AxesLabel → {"t", "Cdye[t]"},
               DisplayFunction → Identity];

        Show[plndis, pl1, DisplayFunction → $DisplayFunction];

        Show[pl1, plndis, DisplayFunction → $DisplayFunction];
```





The results are quite dramatic! We see that the normal distribution of residence times gives rise to a much sharper change in the dye concentration transient than does the single value. In fact, as we make the distribution broader by increasing only $\delta\theta$ while keeping the mean θm constant, we find that the transient response becomes sharper and tends toward a Delta function close to zero. Therefore, as the distribution becomes broader, we have much less perfect mixing, but the response becomes sharper! To experiment with this effect simply change the value of $\delta\theta$; the most pleasing values are in the range of 20–25; below this range the curves are too similar and above it they are too different.

4.9 Summary

We have now fully integrated the concept of a component and the rate of change of a component's mass into our analysis toolkit. Along the way we have taken some time to understand the concepts and meaning of density and how it relates to the concentration of the solute or salt and of the solvent. This included the notion of nonideality when we realized that for most solutes the volume either expands or contracts with their dissolution compared to that which it would have had if the solute added simply was more solvent, but of different mass per molecule. In going from a set of simplifying assumptions to a fuller analysis including these density changes with solute concentration, we had to introduce more computing methods, but we were able to move seamlessly from analytical solutions to numerical ones in order to compare the results from increasingly complex cases. The last section of the chapter was devoted to some new ways of looking at the idea of mixing. In this analysis we learned to use the Dirac-Delta function. We also defined the holding time and used this to construct a general nondimensionalized solution for the tracer injection problem. These are all tools that we will see again.

CHAPTER 5

Multiple Phases—Mass Transfer

A topic of utmost importance in chemical engineering is that of mass transfer. We are often faced with processes that require moving molecules between different phases in order for the outcome we desire to take place (see Figure 1). For example, a "simple" catalytic hydrogenation of a liquid-phase unsaturated molecule, such as benzene, is not really so simple in that it requires many mass transfer steps to occur prior to reaction. The hydrogen molecule must move from the gas phase to the liquid phase. Once there it must diffuse through the liquid and to the catalyst particle's outer surface. From the surface it must now move from outside to inside the particle. Next it needs to adsorb onto the internal surface and then diffuse to the active site and react with a benzene molecule, which also has undergone all the same liquid-phase steps of mass transfer and diffusion! All of this must occur before the reaction can take place. Then the product must leave the active site and the catalyst in a reversal of these steps. We can imagine that the rate at which these molecular transfers between and within phases take place will affect the rates that we observe. If the molecules transfer quickly compared to the pace at which they are reacted, then the reaction rate, that is, the chemistry will control the rate of disappearance of benzene. If, however, the rates of benzene or hydrogen transport are slow, then one or both of these may limit the rate of conversion to that of the rate of arrival of the reactants at the active site. In other words, if the chemistry is "fast," which it should be with an effective catalyst, then it "waits" on the physical transport processes.

This chapter sets out to provide a means of handling these types of interphase mass transfer problems taking into consideration their fundamental characterizing variables, the conservation of mass, and appropriate constitutive relationships.

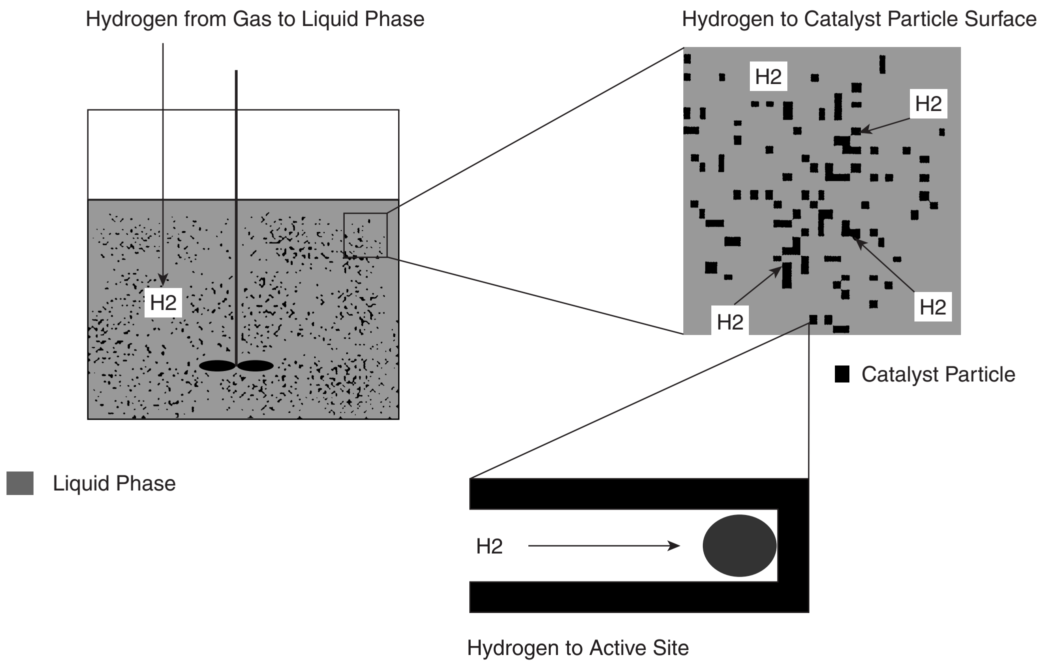

Hydrogen from Gas to Liquid Phase
H2
Liquid Phase
Hydrogen to Catalyst Particle Surface
H2
H2
H2
H2
Catalyst Particle
H2
Hydrogen to Active Site

Figure 1

5.1 Mass Transfer versus Diffusion

The concept of diffusion is one that is familiar to us. If a bottle of fragrance is opened in a room full of fresh but still air, that fragrance will slowly reach all corners of the room. Our sensation of the fragrance will be highest closest to the bottle and lowest in the corners of the room farthest away from it. Eventually, we may find that our sensation of the fragrance is about the same everywhere in the room. The process that takes the fragrance molecules from the vicinity of the uncapped bottle and throughout the room, raising their concentration as a function of time, is diffusion. Random molecular motions are all that are necessary for the fragrance molecules to migrate from regions of higher concentration to those regions that are lower.

Diffusion need not occur only in the gas phase. If a drop of dye is placed carefully into a solvent, then initially the color is very intense within the region of the droplet. With time the droplet of dye molecules becomes more "diffuse," by that we mean larger in volume and less intense in color. This process continues with time until, to the naked eye, the whole solution looks to be colored to the same intensity. Again the mechanism behind this process is diffusion, the random motion of molecules following a gradient in concentration from regions of higher to lower concentration.

Diffusion is a process that also occurs in solids. The manufacture of solid-state transistors involves the diffusion of dopants, such as boron or phosphorus, into silicon in order to create *n*- and *p*-type semiconductors. Since solids are dense, there is a high resistance to diffusion and this makes for very low diffusivities versus those measured in gases, on the order of 10 orders of magnitude lower!

In each case we have spoken about the transfer of mass along a concentration gradient (that is the differential change in concentration over the differential change in position) within one phase. Yet, there are many situations when the mass is moving between phases. For example, the phosphorus delivered to a semiconductor solid for doping typically is transferred to the solid from the gas phase. Thus, before diffusion within the solid can occur there must be gas-to-solid mass transfer of the phosphorus. Here too we can wonder which will be faster—the rate of phosphorus transport to the solid or the rate of diffusion taking phosphorus away from the gas-solid interface and into the bulk solid? In this case, because the rate of diffusion is so low within the solid, it is a good bet that this will be the slower process. When mass transfer is from the gas phase into the liquid, then it may be that the rate processes are limited by the transfer between the phases, rather than the diffusion within the liquid. However, generalizations should not be made hastily because each case needs to be analyzed separately.

We will not be concerned here with diffusion per se; instead we will concentrate on the issue of mass transfer between phases and how that is handled in the context of our analysis tools. The examples begin with an analysis of the dissolution of salt in water and move to more complex systems including the permeation of hydrogen through a palladium membrane.

5.2 Salt Dissolution

The dissolution of a solid particle of salt is a good place to begin because we already know quite a bit about this process. The solid, say sodium chloride, consists of cations and anions that make up the solid lattice in some fixed ratio. The Coulombic forces of attraction—the Madelung energy—keep the lattice together in the solid state. These forces are strong enough to make the crystalline lattice an energetically favorable configuration for the ions (see Figure 2).

When the lattice of ions held together in this way is placed in liquid hydrocarbon such as hexane, nothing happens. The lattice might just as well be standing in air. It remains stable; the hexane does not affect it. We say that the hexane is not a solvent for the salt. Why? We know the hexane is a nonpolar hydrocarbon, whereas the salt is made up of charged ions that are at the limits of polarity—one is a cation and the other is an anion! If the lattice were to fall apart into ions in hexane, it would do so only if the ions were more stable in solution than they were in the lattice. This is not the case with hexane because it lacks polarity to interact with the ions in order to stabilize them.

Experience shows, however, that water will dissolve the salt and will do so very well. The reason is that water is polar; the oxygen is electronegative and carries a more negative partial charge than the hydrogens, which are partially positively charged. These charges make all the difference in the process, because the hydrogens will coordinate with the anion of the salt to

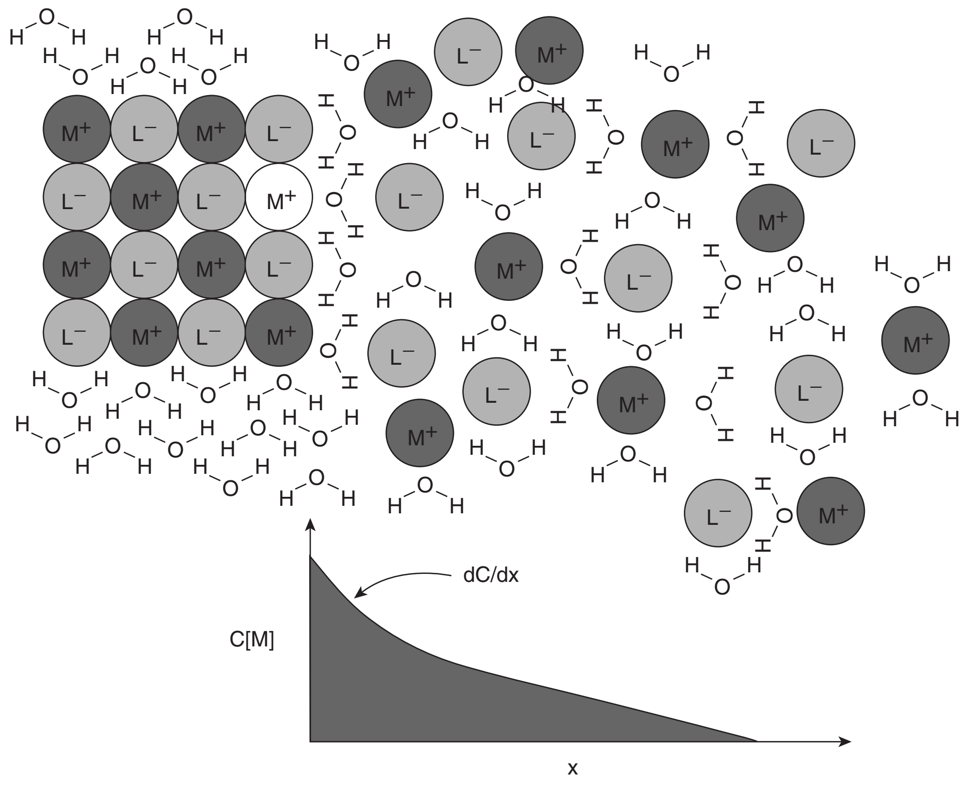


Figure 2

partially dissipate its charge, while the oxygen will coordinate to the cation to do the same. Nature finds bare charges to be unfavorable, so this coordination by water is highly favorable. For obvious reasons this interaction is called solvation.

Quantitatively, we also know that the concentration of the ions in solution is given by their solubility product or K_{sp}. This is nothing more than the equilibrium constant for the salt in water, rearranged to take up the activities of the pure water and the pure solid salt:

$$K_{\text{solvation}} = \frac{a_{\text{cation}} a_{\text{anion}}}{a_{\text{salt}} a_{\text{H}_2\text{O}}}$$

$$K_{\text{sp}} = K_{\text{solvation}} a_{\text{salt}} a_{\text{H}_2\text{O}} = a_{\text{cation}} a_{\text{anion}}$$

$$K_{\text{sp}} = \gamma_{\text{cation}} C_{\text{cation}} \gamma_{\text{anion}} C_{\text{anion}}$$

At low concentrations the activity coefficients are close to unity and we have:

$$K_{\text{sp}} \to C_{\text{cation}} C_{\text{anion}}$$

This is just a review of what we already know about cation solvation in water, based on general chemistry. The information is purely thermodynamic, however, and does nothing to tell us how long a dissolution process may take. Even if a salt is soluble, we do not have a means to get at its rate of dissolution. Furthermore, how do salts with smaller K_{sp} values compare with those with larger K_{sp} values? Will they dissolve faster, slower, or is the rate independent of this factor? What role does the form of the salt play in the rate of dissolution? Does it matter at all, only at the early stages of dissolution, or throughout the process? How does the ratio of solvent mass to solute mass figure into this? These are the kinds of questions we want to be able to handle quantitatively.

5.3 Batch

Background. The dissolution of a salt into a surrounding solution is easiest to think of as taking place in a closed vessel, that is, in a "batch" with no flows in or out of the vessel. But remember there are "flows" between the solid and the liquid phases. We take a particle of the solid as one control volume and the volume of solvent as the other. We can solve this one particle problem and then handle many particles. The process is taken to occur at constant temperature. The physical situation looks like that shown in Figure 3.

The dissolution process will continue until either all the salt has dissolved, or the saturation limit of the solvent has been reached. Therefore, the ratio of the volume of the solvent to the mass of salt will be critical. If we were to do this experiment several times with the same volume

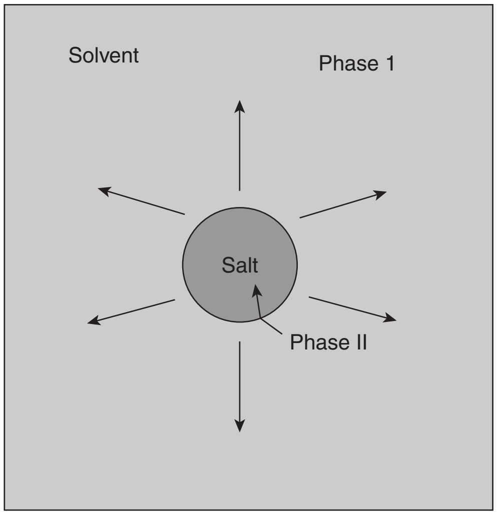


Figure 3

of solvent and with the same mass of salt, but with different numbers of particles of the salt, we would find that the experiments done using more, smaller particles would require less time to fully dissolve the salt than would those that use fewer, larger particles. Our common experience of dissolving sugar in coffee or in tea is that stirring makes the dissolution process go faster. Therefore, if we were to do a series of salt dissolution experiments, keeping all else the same, but varying the rate of mixing, we would find that faster and better mixing would lead to more rapid dissolution. This is obvious, but we will still write it mathematically before we go on:

$$\text{rate}_{\text{dissolution}} \alpha A_{\text{interface}}$$

$$\text{rate}_{\text{dissolution}} \uparrow \text{ as mixing } \uparrow$$

Rate of Dissolution. Experiments also would show that the rate of dissolution must stop when the solution reaches the saturation limit and, furthermore, the rate will be fastest early in the process, when the concentration of salt in the solvent is low. All of this behavior can be apprehended in a simple rate for the dissolution process:

$$\text{rate}_{\text{dissolution}} = K_m A_{\text{interface}}\left(C_{\text{salt}}^{\text{sat}'d} - C_{\text{salt}}^{I}[t]\right)$$

Km is the mass transfer coefficient, $\frac{\text{Length}}{\text{time}}$, $\boldsymbol{A}_{\textbf{interface}}$ is the area of the solid in contact with the liquid (either for one particle or for *n*-particles), $\boldsymbol{C}_{\textbf{salt}}^{\textbf{sat}'d}$ is the concentration of the salt in the solvent phase I at the saturation limit, and $\boldsymbol{C}_{\textbf{salt}}^{\boldsymbol{I}}[\textbf{t}]$ is the concentration of the salt in the solvent at any time **t**. This rate law includes all the phenomena we just said would be observed in experiment. The mass transfer coefficient will be larger if the mixing is larger, otherwise smaller. The interfacial area is linearly related to the rate—the more area the better. When the concentration of salt in solution hits the saturation limit, the dissolution stops. The term in brackets is the so-called "linear driving force": linear because the concentration dependence is first order, that is, power unity, and driving force because the rate is proportional to the difference between the maximum and the actual concentrations. Hence the rate is maximum at an instant after time zero when the difference is just $\boldsymbol{C}_{\textbf{salt}}^{\textbf{sat}'d}$.

Conservation of Mass across Phases. The next step is to apply the conservation of mass principle to this problem. We need to write a material balance on salt for both phases. Any mass that leaves one phase must end up in the other phase. Then we can say the following regarding the rate of salt mass accumulation in the two phases:

$$\frac{dm_{\text{salt}}^{I}}{dt} = \frac{dC_{\text{salt}}^{I}V^{I}}{dt} = +A_i r_d$$

$$\frac{dm_{\text{salt}}^{\text{II}}}{dt} = \frac{dC_{\text{salt}}^{\text{II}}V^{\text{II}}}{dt} = -A_i r_d$$

where $A_i = A_{\mathbf{interface}}$ and $r_d = \mathbf{rate}_{\mathbf{dissolution}}$. We can substitute in the constitutive expression for the rate of mass transfer between the two phases to obtain:

$$\frac{dm^I_{\text{salt}}}{dt} = \frac{dC^I_{\text{salt}}V^I}{dt} = +K_m A_i\left(C^{\text{sat}'d}_{\text{salt}} - C^I_{\text{salt}}[t]\right)$$
$$\frac{dm^{\text{II}}_{\text{salt}}}{dt} = \frac{dC^{\text{II}}_{\text{salt}}V^{\text{II}}}{dt} = -K_m A_i\left(C^{\text{sat}'d}_{\text{salt}} - C^I_{\text{salt}}[t]\right)$$

The material balance for the solid phase includes in the differential term $\boldsymbol{C}^{\mathbf{II}}_{\mathbf{salt}}$. Because salt is a pure solid, this is the same as the density of the solid, which remains constant throughout the process. This means that the water is assumed not to disrupt the solid lattice by penetrating into it and slowly expanding it to result in dissolution. Instead, it is only the first few layers that are involved in the process and the interior of the particle is left unperturbed until it becomes surface. The process is like one of layer-by-layer lift-off and dissolution. If the mass transferred between the phases is in total small, then we can ignore the change in solution volume that comes with the density change as the salt concentration rises. If this is too restrictive, then we can relax it later, but for now it makes good sense to ignore it and concentrate on the mass transfer problem. The equations can be rewritten as:

$$\frac{dm^I_{\text{salt}}}{dt} = V^I\frac{dC^I_{\text{salt}}[t]}{dt} = +K_m A_i\left(C^{\text{sat}'d}_{\text{salt}} - C^I_{\text{salt}}[t]\right)$$
$$\frac{dm^{\text{II}}_{\text{salt}}}{dt} = \rho_{\text{salt}}\frac{\text{dV}^{\text{II}}[t]}{dt} = -K_m A_i\left(C^{\text{sat}'d}_{\text{salt}} - C^I_{\text{salt}}[t]\right)$$
$$\therefore V^I\frac{dC^I_{\text{salt}}}{dt} = -\rho_{\text{salt}}\frac{\text{dV}^{\text{II}}}{dt}$$

This can be integrated assuming that at $\mathbf{t = 0}$, $\boldsymbol{C}^I_{\mathbf{salt}}[t] = \mathbf{0}$ and $\mathbf{V}^{\mathbf{II}}[t] = \boldsymbol{V}^{\mathbf{II}}_o$ and then rearranged to give:

$$\rho_{\text{salt}}\left(V^{\text{II}}_o - V^{\text{II}}[t]\right) = V^I C^I_{\text{salt}}[t]$$
$$V^{\text{II}}[t] = V^{\text{II}}_o - \frac{V^I C^I_{\text{salt}}[t]}{\rho_{\text{salt}}}$$
$$C^I_{\text{salt}}[t] = \frac{\rho_{\text{salt}}\left(V^{\text{II}}_o - V^{\text{II}}[t]\right)}{V^I}$$

The last equation relates the concentration of the salt in the liquid phase at any time to the volume of the salt remaining in the solid at the same time. This solution and the one for the concentration of salt in solvent are implicit. To use these equations we would need to measure the actual volume of the solid salt as a function of time—not an easy measurement to make in practice! What we really need then is an explicit solution in time. To obtain this we must return to the statements of the material balance between the phases.

The salt leaving the solid follows this equation:

$$\rho_{\text{salt}}\frac{\mathrm{d}\mathrm{V}^{\mathrm{II}}[t]}{dt} = -K_m A_i\left(C_{\text{salt}}^{\text{sat}'d} - C_{\text{salt}}^{I}[t]\right)$$

If we are concerned only with the case in which the total mass of salt transferred is small relative to the volume of solvent and the saturation limit, then the equation becomes:

$$\rho_{\text{salt}}\frac{\mathrm{d}\mathrm{V}^{\mathrm{II}}[t]}{dt} = -K_m A_i C_{\text{salt}}^{\text{sat}'d}$$

This looks as though we should be able to integrate it immediately, but look again! The area between the two phases is the area of the salt particle; it must be changing with time, and quite considerably at that. Therefore we cannot integrate this as of yet. We need a relationship between the volume of the solid at any time and the surface area it projects. Thankfully we can find this easily. If each particle of solid is the same size then their interfacial areas are the same and we can write:

$$A_i = \mathrm{N}A_{i,N} = \mathrm{N}\gamma\left(\frac{V^{\mathrm{II}}}{N}\right)^{\frac{2}{3}} = N^{\frac{1}{3}}\gamma V^{\mathrm{II}\frac{2}{3}}$$

Here **N** is the number of identical particles of solid, and γ is the surface area to volume ratio, or the shape factor which accounts for the geometry of the solid, assuming that it is a regular polytope; the subscript i refers to the number of any individual particle. The total volume of the solid phase divided by N is the volume of any individual particle and when we raise this to the 2/3 power we approach to within a constant γ, the surface area of that same particle. This allows us to rewrite the rate of change in solid volume (dropping the notation for t-dependence) :

$$\frac{\mathrm{d}\mathrm{V}^{\mathrm{II}}}{dt} = -\frac{K_m N^{\frac{1}{3}}\gamma V^{\mathrm{II}\frac{2}{3}} C_{\text{salt}}^{\text{sat}'d}}{\rho_{\text{salt}}}$$

We can separate, integrate, and rearrange to obtain:

$$V^{\mathrm{II}}[\mathrm{t}] = \left(\sqrt[3]{V_0^{\mathrm{II}}} - \frac{K_m N^{\frac{1}{3}} \gamma C_{\mathrm{salt}}^{\mathrm{sat}'\,d}}{\rho_{\mathrm{salt}}} t\right)^3$$

We have already stated that the measurement of the solid volume would be a difficult experiment to conduct. The measurement of salt concentration as a function of time is easy to do and so we want an explicit equation for the concentration. To obtain this we use this equation for volume change with time to obtain the concentration change with time.

$$C_{\mathrm{salt}}^{I}[t] = \frac{\rho_{\mathrm{salt}}\left(V_o^{\mathrm{II}} - \left(\sqrt[3]{V_o^{\mathrm{II}}} - \frac{K_m N^{\frac{1}{3}} \gamma C_{\mathrm{salt}}^{\mathrm{sat}'\,d}}{\rho_{\mathrm{salt}}} t\right)^3\right)}{V^I}$$

If we made a series of experiments in which we sought the mass transfer coefficient, then we would rearrange this so that we could plot a function of the salt concentration against the time:

$$\left(1 - \frac{C_{\mathrm{salt}}^{I}[t] V^I}{\rho_{\mathrm{salt}} V_o^{\mathrm{II}}}\right)^{\frac{1}{3}} = \left(1 - \frac{K_m N^{\frac{1}{3}} \gamma C_{\mathrm{salt}}^{\mathrm{sat}'\,d}}{\rho_{\mathrm{salt}} \sqrt[3]{V_o^{\mathrm{II}}}} t\right)$$

A plot of this left-hand side versus the time gives a graph whose slope is the coefficient of **t**. Everything in this group with inverse time as its dimensions should be known before the experiments are even conducted. If we knew the mass transfer coefficient, then the inverse of this group would provide the time required to dissolve all **N** particles of the salt. We can see this because when the time **t** is equal in magnitude to the reciprocal of this group the right-hand side goes to zero identically.

$$\left(1 - \frac{C_{\mathrm{salt}}^{I}[t] V^I}{\rho_{\mathrm{salt}} V_o^{\mathrm{II}}}\right)^{\frac{1}{3}} = \left(1 - \frac{t}{\theta}\right)$$

$$\theta = \frac{K_m N^{\frac{1}{3}} \gamma C_{\mathrm{salt}}^{\mathrm{sat}'\,d}}{\rho_{\mathrm{salt}} \sqrt[3]{V_0^{\mathrm{II}}}}$$

The left-hand side must also be zero. Therefore, the concentration of the salt at that time will be the reciprocal of the product of the density of the salt and its initial volume divided by the volume of the liquid.

5.4 Fit to the Batch Data

Table 1 gives data for an experiment in which 100 salt cubes ($\gamma = 6$) were dissolved in 100 cm^3 of water. The total volume of the salt was 1 cm^3, its density was 2 g cm^3, and the saturation limit of the salt was 0.05 g cm^{-3}. Data points were logged every 50 sec for a total of 2500 sec.

t/sec	Csalt [t]	t/sec	Csalt [t]
50	0.00171084	1300	0.0196429
100	0.00279533	1350	0.0192367
150	0.00424841	1400	0.0193228
200	0.00494699	1450	0.0193325
250	0.00635887	1500	0.0192889
300	0.00763428	1550	0.0201392
350	0.00835931	1600	0.0197141
400	0.010246	1650	0.0197589
450	0.0105354	1700	0.0203605
500	0.0116536	1750	0.0202748
550	0.0120036	1800	0.0200916
600	0.0135537	1850	0.0201311
650	0.0137191	1900	0.02012
700	0.0142037	1950	0.0200507
750	0.0150545	2000	0.0195248
800	0.0153611	2050	0.0202914
850	0.0165474	2100	0.0199335
900	0.0169347	2150	0.0204394
950	0.0167272	2200	0.0204604
1000	0.0177943	2250	0.0203341
1050	0.018098	2300	0.0200852
1100	0.0184004	2350	0.0196325
1150	0.0188067	2400	0.0205709
1200	0.0190497	2450	0.0198959
1250	0.019132	2500	0.020606

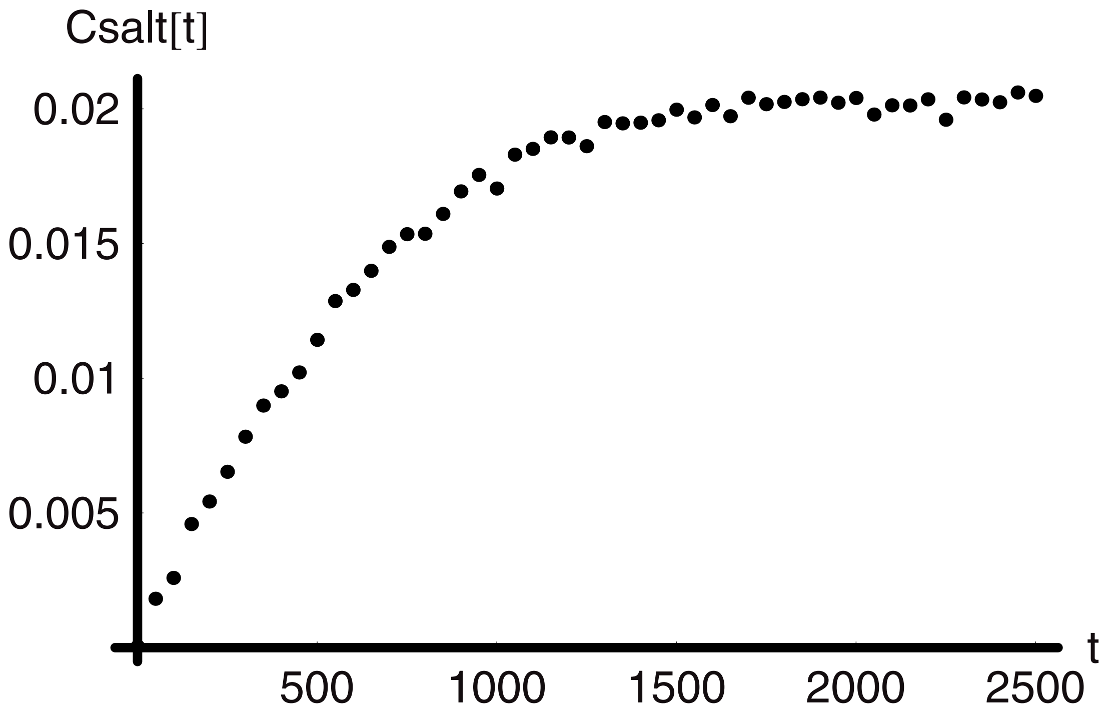
Table 1

Putting the data into vector notation for manipulation we have:

```
In[1]:= csaltdata =
        {{"0", "0.00004044"}, {"50", "0.001815"}, {"100", "0.002589"},
         {"150", "0.004589"}, {"200", "0.005424"}, {"250", "0.006528"},
         {"300", "0.00783"}, {"350", "0.008987"}, {"400", "0.009514"},
         {"450", "0.01022"}, {"500", "0.01143"}, {"550", "0.01287"},
         {"600", "0.01328"}, {"650", "0.01399"}, {"700", "0.01488"},
         {"750", "0.01535"}, {"800", "0.01536"}, {"850", "0.0161"},
         {"900", "0.01694"}, {"950", "0.01755"}, {"1000", "0.01704"},
         {"1050", "0.0183"}, {"1100", "0.01851"}, {"1150", "0.01894"},
         {"1200", "0.01894"}, {"1250", "0.01861"}, {"1300", "0.01951"},
         {"1350", "0.01946"}, {"1400", "0.01949"}, {"1450", "0.01958"},
         {"1500", "0.01997"}, {"1550", "0.01968"}, {"1600", "0.02014"},
         {"1650", "0.01973"}, {"1700", "0.02041"}, {"1750", "0.02017"},
         {"1800", "0.02026"}, {"1850", "0.02036"}, {"1900", "0.02042"},
         {"1950", "0.02023"}, {"2000", "0.0204"}, {"2050", "0.01979"},
         {"2100", "0.02013"}, {"2150", "0.02013"}, {"2200", "0.02035"},
         {"2250", "0.0196"}, {"2300", "0.02043"}, {"2350", "0.02035"},
         {"2400", "0.02025"}, {"2450", "0.02061"}, {"2500", "0.02048"}};
```

As we plan to do considerable graphing we set the options for the style of the graphs to make them most visible and then ListPlot the data.

```
In[2]:= SetOptions[{Plot, ListPlot},
          AxesStyle → {Thickness[0.01]},
          PlotStyle → {PointSize[0.015],
           Thickness[0.006]},
          DefaultFont → {"Helvetica", 17}];

In[3]:= datpl = ListPlot[csaltdata,
           AxesLabel → {"t", "Csalt[t]"},
           PlotStyle → PointSize[.015]];
```



We shall want to fit this data "**csaltdata**" as seen in the preceding graph to the expression that we have derived because, as we can see from the data, the final concentration is less than 50% of that at saturation. To be safe we will fit just the early time data out to 1500 sec. One way to do this is to do a one parameter, nonlinear fit to the expression after we have simplified it by evaluating all the parameters. The first step is to obtain the fitted expression, evaluate it, and then compare it to the data.

Here are the parameters relevant to the problem and their values followed by the function definition:

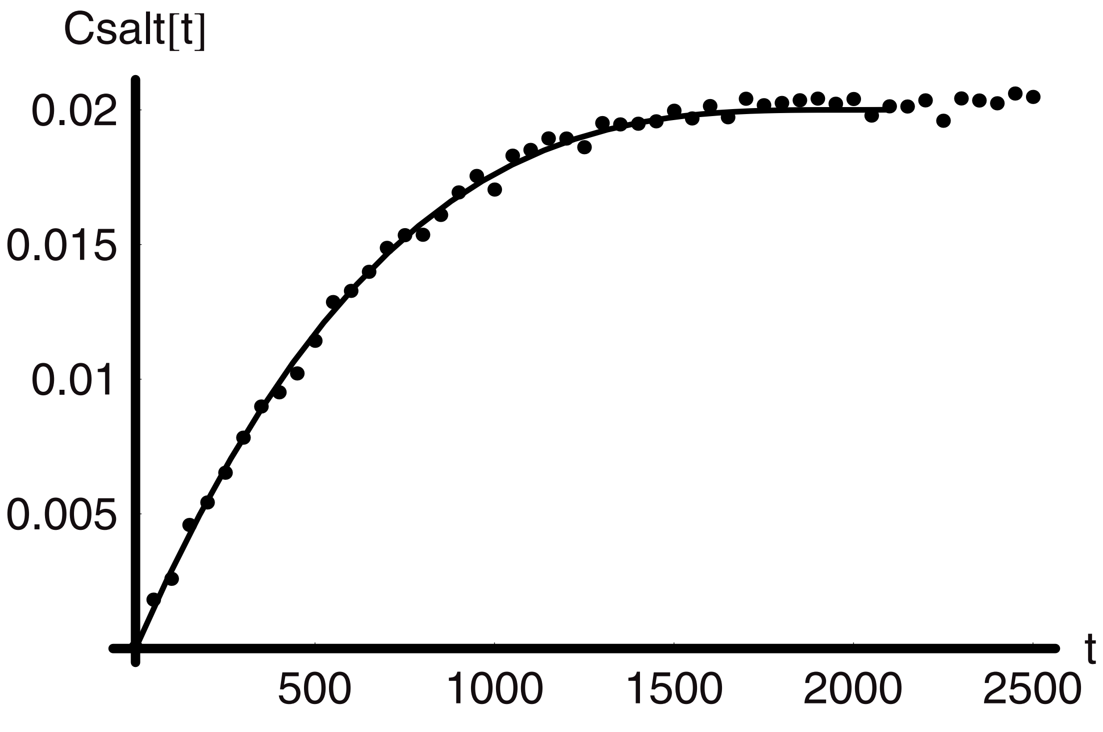
```
In[4]:= ρsalt = 2;
        VoII = 1;
        γ = 6;
        Km =.
        Csaltsatd = .05;
        VI = 100;
        n = 100;
        tmax = 2500;
```

$$\texttt{csalt[t\_] :=} \frac{\rho\texttt{salt}\left(\texttt{VoII} - \left(\sqrt[3]{\texttt{VoII}} - \frac{\texttt{Km}\, \texttt{n}^{\frac{1}{3}} \gamma\, \texttt{Csaltsatd}}{\rho\texttt{salt}}\texttt{t}\right)^{3}\right)}{\texttt{VI}}$$

```
        f[t_] := Simplify[csalt[t]]
```

Calling "Statistics'NonlinearFit'" will allow us to fit the data with the command "NonlinearFit," which we can then call "g" with the command g = % and, finally, we can Plot and Show g versus the data set:

```
In[14]:= << Statistics`NonlinearFit`

In[15]:= NonlinearFit[csaltdata, f[t], t, Km];
         g = %;

         plfit = Plot[g,
            {t, 0, 2100},
            PlotRange → {{0, 2500}, {0.0.02}},
            DisplayFunction → Identity];
         Show[datpl, plfit,
            DisplayFunction → $DisplayFunction];
```



The coefficient of **t** is used to evaluate **Km** and that is:

```
In[19]:= Solve[0.696238 Km == 0.00051, Km]

Out[19]= {{Km → 0.000732508}}
```

The fitted value is 7.3×10^{-2} cm sec^{-1}, which is a reasonable, although small, value for this constant.

5.5 Semicontinuous: Pseudo Steady State

We can imagine a situation where our goal is to dissolve a sparingly soluble salt out of a unit in which it has precipitated. An example of such salts are the alkaline salts that deposit in boilers and heat exchangers as "scale." Nothing more than the accumulation of precipitate on the inner walls of the vessel over time, these salts can present a real hazard in that they reduce the heat conduction through the wall, because they are good insulators. As a result of this, boilers can develop hot spots and explode, and heat exchangers can become much less efficient over time with similarly deleterious results. At the same time these salts may be sparingly soluble except in acidic solution, which, if the pH is too low, will etch away the vessel wall along with the salts over time. Therefore, one may be forced to accept the low solubility in the less acidic pH range, and be willing to pump large volumes of solvent through for longer periods of time. This is an optimal solution to the problem.

The essence of this problem, and others like it, is that the transfer of mass from the solid to the liquid occurs slowly over time, but now there is a continuous flow of solvent over a slowly diminishing mass of solid. The flow of solvent does two things—it provides a large volume of solvent when the flow is integrated over time and, if it is at relatively high rates, it provides much better mass transfer rates than if the same large volume were merely standing in contact with the solid without flow. Flow gives mixing and mixing gives higher mass transfer coefficients, which means it will take less time to dissolve than it would with less or zero mixing. The physical situation is as shown in Figure 4.

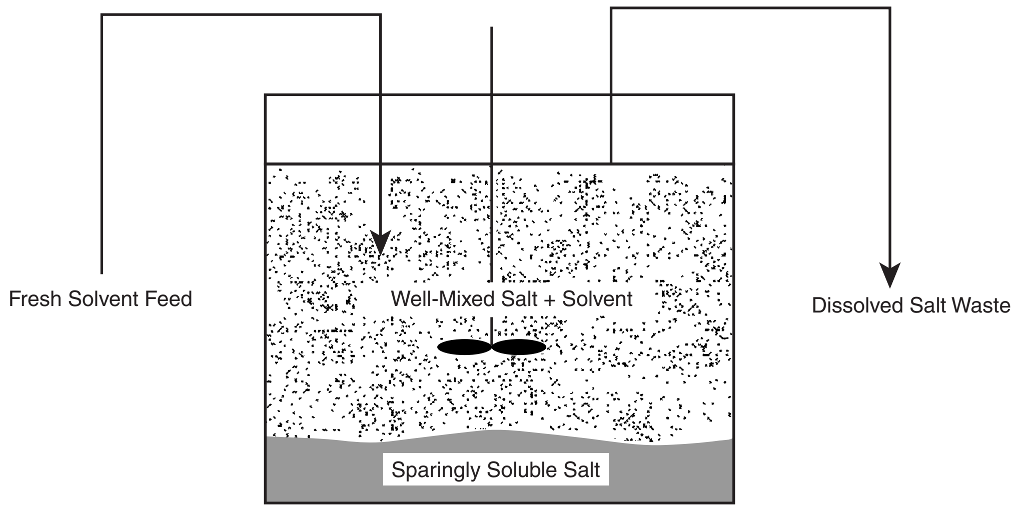


Figure 4

The equations look largely the same, except that the solution phase balance on the salt has a convective flow term for the mass of salt leaving the unit by this process:

$$\frac{dm_{\text{salt}}^{I}}{dt} = V^{I}\frac{dC_{\text{salt}}^{I}[t]}{dt} = +K_m A_i\left(C_{\text{salt}}^{\text{sat}'d} - C_{\text{salt}}^{I}[t]\right) - C_{\text{salt}}^{I}[t]\text{qex}$$

$$\frac{dm_{\text{salt}}^{\text{II}}}{dt} = \rho_{\text{salt}}\frac{\text{dV}^{\text{II}}[t]}{dt} = -K_m A_i\left(C_{\text{salt}}^{\text{sat}'d} - C_{\text{salt}}^{I}[t]\right)$$

The interfacial area in this case will be taken to be a constant well approximated by the surface area of the unit. In the diagram this would be the cross-sectional area of the tank $\boldsymbol{\pi r^2}$. This means then that the salt is removed by a process that removes layers, making the change in salt volume a one-dimensional problem of computing the salt thickness at any time.

If the salt is sparingly soluble, then $\boldsymbol{C_{\text{salt}}^{\text{sat}'d}}$ is small in magnitude, and if the product $\boldsymbol{K_m A_i}$ is relatively large due to gross mixing, then the salt concentration is likely to be a constant and close to but not as large as $\boldsymbol{C_{\text{salt}}^{\text{sat}'d}}$, depending on the magnitudes of the parameters. Given that salt concentration is a constant, then its rate of change is zero, that is, the salt in solution is at steady state. This is the case even though the salt mass is changing steadily and constantly as a function of time. Because of this mixed condition the two-phase system as a whole is said to be in a pseudo-steady state. This is the case because if we could measure only the concentration of salt exiting the reactor, we would find it to be a constant at constant conditions. However, we know the salt is coming from inside the control volume because we are not feeding it. That means that according to the principle of conservation of mass, the salt is emerging from a dissolving source within the control volume, and this mass must be decreasing with time.

The equations work out as follows for the pseudo-steady state:

$$0 = +K_m A_i\left(C_{\text{salt}}^{\text{sat}'d} - C_{\text{salt}}^{I}\text{stst}\right) - C_{\text{salt}}^{I}\text{stst qex}$$

$$\therefore K_m A_i\left(C_{\text{salt}}^{\text{sat}'d} - C_{\text{salt}}^{I}\text{stst}\right) = C_{\text{salt}}^{I}\text{stst qex}$$

and

$$\rho_{\text{salt}}\frac{\text{dV}^{\text{II}}[t]}{dt} = -C_{\text{salt}}^{I}\text{stst qex}$$

$$V^{\text{II}}[t] = V_o^{\text{II}} - C_{\text{salt}}^{I}\text{stst qex t}$$

This very simple solution comes about as a result of the fact that at the steady state the concentration of salt is a constant and the exit mass flow has to be equal to the rate of salt mass transfer into the solvent.

5.6 Full Solution

By solving the equation in the way we have just described, we make the mathematics much simpler, but we also place severe constraints on the solution. Instead of doing that, we now solve the equations without these assumptions, in this way they are then appropriate for the most general case—from short time to long, and for sparingly soluble to very soluble salts.

$$\begin{aligned}
\frac{d\rho^I[t]V^I[t]}{dt} &= (\rho_{\text{solvent}} - \rho^I[t])\,\text{qex} \\
\rho^I[\text{t}] &= \rho_{\text{solvent}} + \text{a}C^I_{\text{salt}}[t] \\
\frac{dm^I_{\text{salt}}}{dt} &= \frac{dC^I_{\text{salt}}[t]V^I[t]}{dt} = +K_m A_i\left(C^{\text{sat}'d}_{\text{salt}} - C^I_{\text{salt}}[t]\right) - C^I_{\text{salt}}[t]\,\text{qex} \\
\frac{dm^{\text{II}}_{\text{salt}}}{dt} &= \rho_{\text{salt}}\frac{\text{dV}^{\text{II}}[t]}{dt} = -K_m A_i\left(C^{\text{sat}'d}_{\text{salt}} - C^I_{\text{salt}}[t]\right)
\end{aligned}$$

Here we have added one equation—the total mass balance—which includes the density of phase one as it flows out of the system. Recall also that for a double salt $\boldsymbol{M_a L_b}$ we have the following:

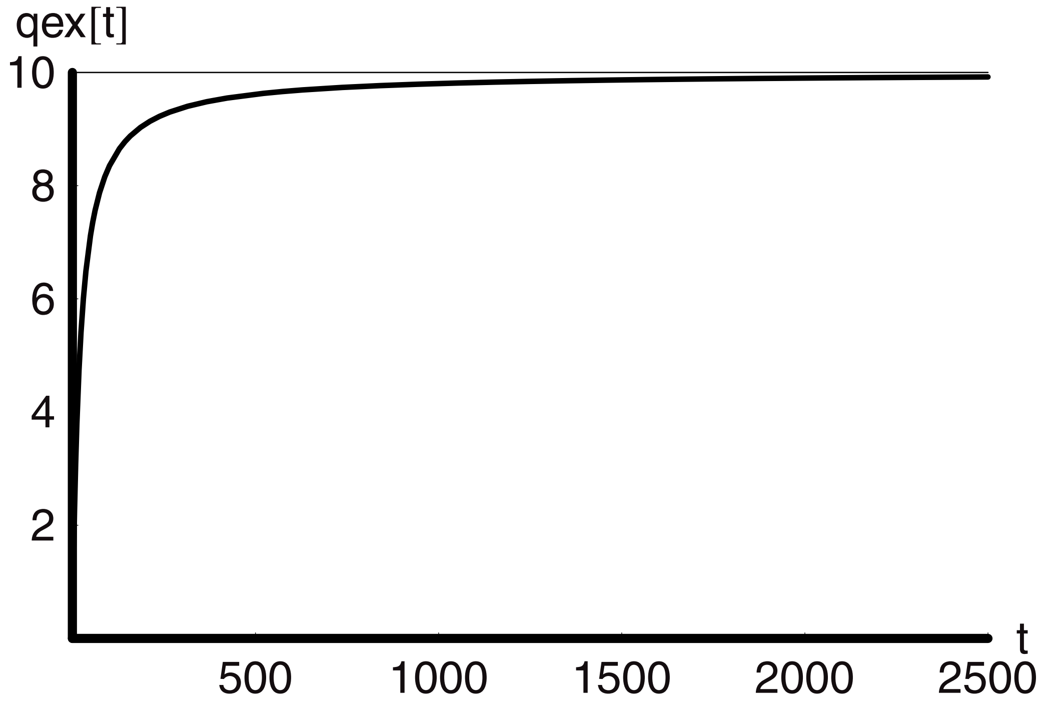
$$K_{\text{sp}} = C_{M^+}{}^a C_{L^-}{}^b = \left(\text{a}C^{\text{sat}'d}_{\text{salt}}\right)^a \left(\text{b}C^{\text{sat}'d}_{\text{salt}}\right)^b$$

We can solve for the saturation concentration of the salt in terms of its **Ksp** and the stoichiometric numbers:

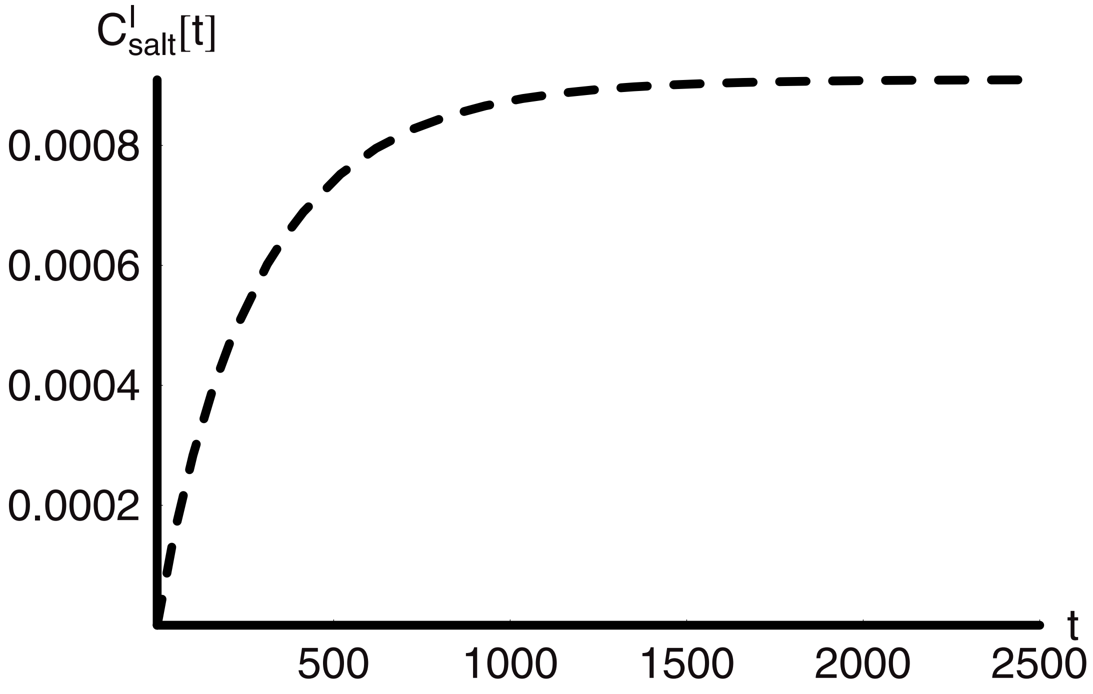
```
In[20]:= Solve[Ksp == PowerExpand[(a Csatd)^a (b Csatd)^b], Csatd]

         Solve::ifun : Inverse functions are being used by
          Solve, so some solutions may not be found.

Out[20]= {{Csatd → (a^-a b^-b Ksp)^(1/(a+b))}}
```

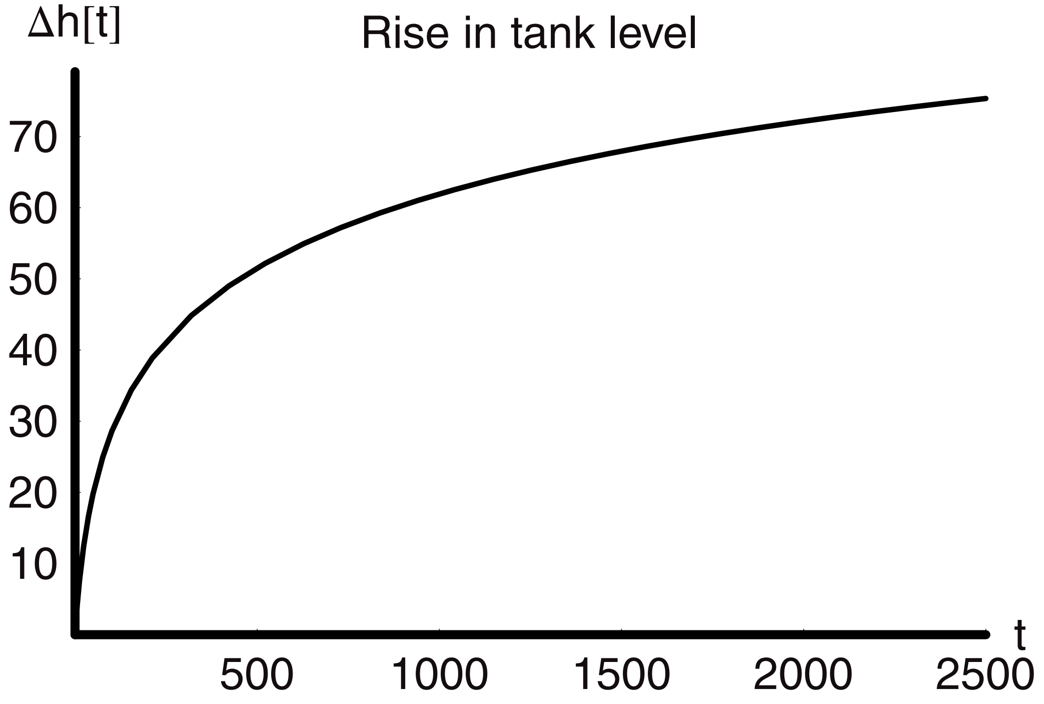
Turning once more to the equations, we will derive code that will solve these numerically and simultaneously by using this expression for the saturation concentration of the salt and the linear dependence of density upon concentration. The code that follows does just this. The tank parameters are specified along with the volumes of the solution and salt phases at time zero (**VIo** and **VIIo**), the salt parameters, the mass transfer and flow rates, the maximum time for the integration to be done, the function calls for the exit flow rate in terms of the inlet flow rate, density of the solution and the saturation concentration of the salt, the material balance equations, the implementation of the numerical solution of the equations and the assignment of the interpolation functions to function names, and finally the graphical output routines.

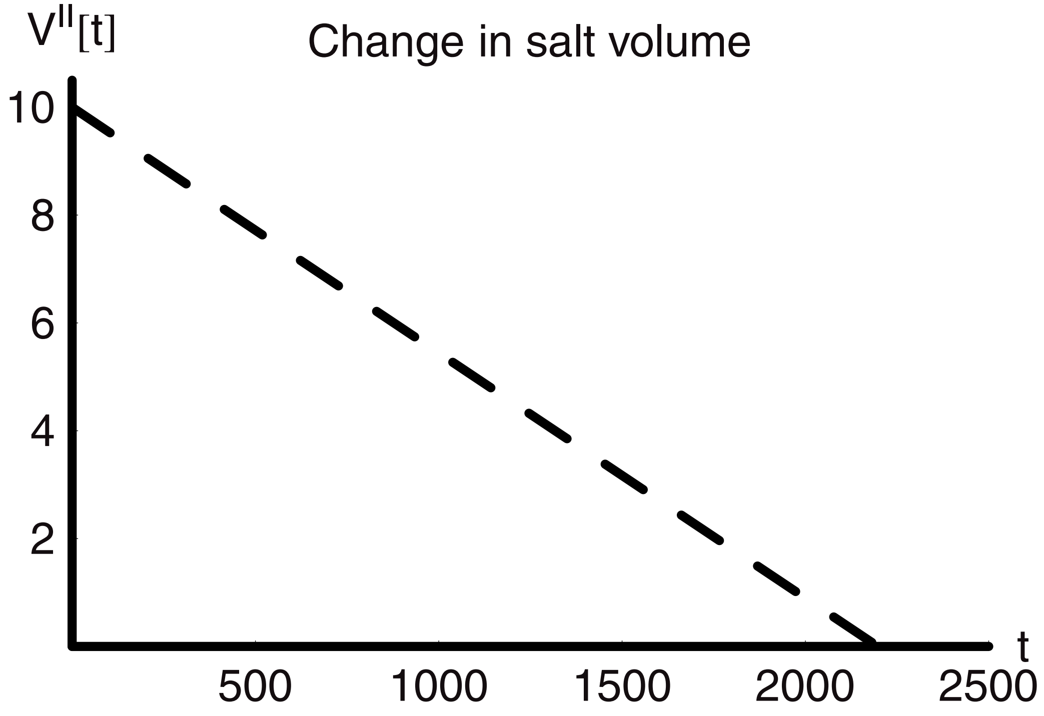
```
In[21]:= "The tank parameters are:";
         r = 2;
         Ai = N[πr^2];
         VIo = 100 Ai r;
         VIIo = 10;
         VI =.
         "These are the salt parameters";
         ρsolvent = 1;
         ρsalt = 2;
         a = 1;
         b = 1;
         Ksp = 110^-2;
         γ = 0.9;
         cIo = 10^-10;

         "The mass transfer coefficient and flow rates";
         Km = 7.3 10^-3;
         qo = 10;
         f = .05;

         "This is the maximum time for the integration";
         tmax = N[2.5 10^3];

         "These specify the exit flow the density in solution
          and the saturation concentration or solubility";
         qex[t_] := qo (ft/(1 + ft))
         ρI[t_] := N[ρsolvent + γcI[t]]

         csatd[a_, b_, Ksp_] := N[(a^-a b^-b Ksp)^(1/(a+b))]
         "Set of equations to be solved";
         eqns = {
            ∂t(ρI[t] VI[t]) == ρsolvent qo - ρI[t] qex[t],
              VI[0] == VIo,
            ∂t(cI[t] VI[t]) == Km Ai(csatd[a, b, Ksp] - cI[t])
              - cI[t] qex[t], cI[0] == cIo,
            ∂t (VII[t]) == -Km Ai(csatd[a, b, Ksp] - cI[t]) / ρsalt,
              VII[0] == VIIo};

         "Numerical solutions and assignments";
         soln = NDSolve[
            eqns,
            {VI[t], VII[t], cI[t]},
            {t, 0, tmax}];
         cOne[t_] := Evaluate[cI[t] /. soln[[1]]]
         vOne[t_] := Evaluate[VI[t] /. soln[[1]]]
         vTwo[t_] := Evaluate[VII[t] /. soln[[1]]]

        General::spell1 : Possible spelling error: new symbol
         name "VIIo"is similar to existing symbol "VIo".

        General::spell1 : Possible spelling error: new symbol
         name "csatd"is similar to existing symbol "Csatd".

        General::spell1 : Possible spelling error: new symbol
         name "vOne"is similar to existing symbol "cOne".

In[52]:= "The graphical routines";
         θ =.
         θ == (qo/VIo)^-1;
         Plot[{qo, qex[t]}, {t, 0, tmax},
           PlotRange → {{0, tmax}, {0, qo}},
           AxesLabel → {"t", "qex[t]"}
          ];
```



```
In[56]:= csatd == csatd[a, b, Ksp];
         Plot[N[cOne[t]], {t, 0, tmax},
           PlotRange → {{0, tmax},
             {0, Max[Table[cOne[t],
                 {t, 0, tmax}]]}},
                AxesLabel → {"t", "C^I_salt[t]"},
                PlotStyle → {Thickness[0.01], Dashing[{.03, .03}]}];

         VIo/Ai;

         Plot[{VIo/Ai, ((vOne[t]/Ai) - (vOne[0]/Ai))}, {t, 0, tmax},
               PlotRange → {{0, tmax},
             {0, Max[Table[(1 + .05) ((vOne[t]/Ai) - (vOne[0]/Ai)),
                 {t, 0, tmax}]]}},
                AxesLabel → {"t", "Δh[t]"},
                PlotLabel → "Rise in tank level"
             ];

         Plot[vTwo[t], {t, 0, tmax},
               PlotRange → {{0, tmax},
               {0, Max[Table[(1 + .05) vTwo[t], {t, 0, tmax}]]}},
               AxesLabel → {"t", "V^II[t]"},
           PlotStyle → {{Thickness[0.01], Dashing[{0.05, 0.05}]}},
           Pl
                PlotLabel → "Change in salt volume"
             ];
```




Δh[t]
Rise in tank level
70
60
50
40
30
20
10
t
500
1000
1500
2000
2500


V^II[t]
Change in salt volume
10
8
6
4
2
t
500
1000
1500
2000
2500

This can now be used interactively to experiment with parameter values in order to learn how they affect the observed behavior of this system. There are several points that must be noted about this code that bear explanation. First, the initial concentration of the salt in the solution **cIo** is not taken as zero; it is set to a very low value to simulate zero at time zero. If we set this identically to zero, the numerical routine will come back with a complex infinity error because it will have divided by zero at the start of the calculation. Second, the exit flow rate has been made a function of the inlet concentration. This is one way to handle the problem of the exit flow rate. By doing it this way, the exit flow rises to the inlet flow over some period of time, which is parametrically dependent upon the magnitude of **f**. One could envision that a controller could be used at the exit to produce this effect. If one wishes to see what the solutions would look like if this were not included and if the exit flow rate instantaneously equaled the inlet flow rate, this is easily accomplished by letting **f** be large in magnitude, say 10^3. The solution is remarkably stable, but this is not to say that with the right (or wrong) choices of parameters, it will not become numerically unstable. It certainly will, especially if the parameters begin to imply nonphysical conditions. The simulation has been run to times that are two orders of magnitude larger than the current tmax value, with **Ksp** at 10^{-1}, and $\mathbf{f} = \mathbf{1}$, and the only limit to going longer in time was patience. Some small instability is noted in the concentration of salt as a function of time when the integration is done for long times, at high initial volumes of solvent, and large **Ksp** values. The reader should experiment with the parameters to find cases where this type of behavior is displayed.

5.7 Liquid-Liquid System

Fully Continuous

Steady State: Equilibrium Stage. Liquid-liquid extractions are used in many different applications from chemical production to environmental clean-up. It is possible to extract organics from water by contacting the water with a better solvent for the impurities, which is also immiscible with the water. When there are two liquid phases involved we have new equilibrium considerations to take into account, whereas in the case of the salt we had only one, the solubility, since the second phase was the pure salt. The phases will not change in volume within the unit that is used for contacting them. There are now two "solubilities" of the transferred component—one for each phase. These are better termed the equilibrium concentrations of the component dissolved separately in each phase. Many, if not most of us, have some experience with this sort of process done at the bench by organic chemists. The solution to be extracted is typically aqueous and contains the desired compound. This is added to a separatory funnel first. Then a less dense, immiscible solvent with a higher affinity for the target compound is added as a layer on top. This solvent's "higher affinity" for the target, means that the target is more soluble in it than in water. Often, diethyl ether is used as this second solvent. After capping the funnel, inverting it, and opening the petcock to allow

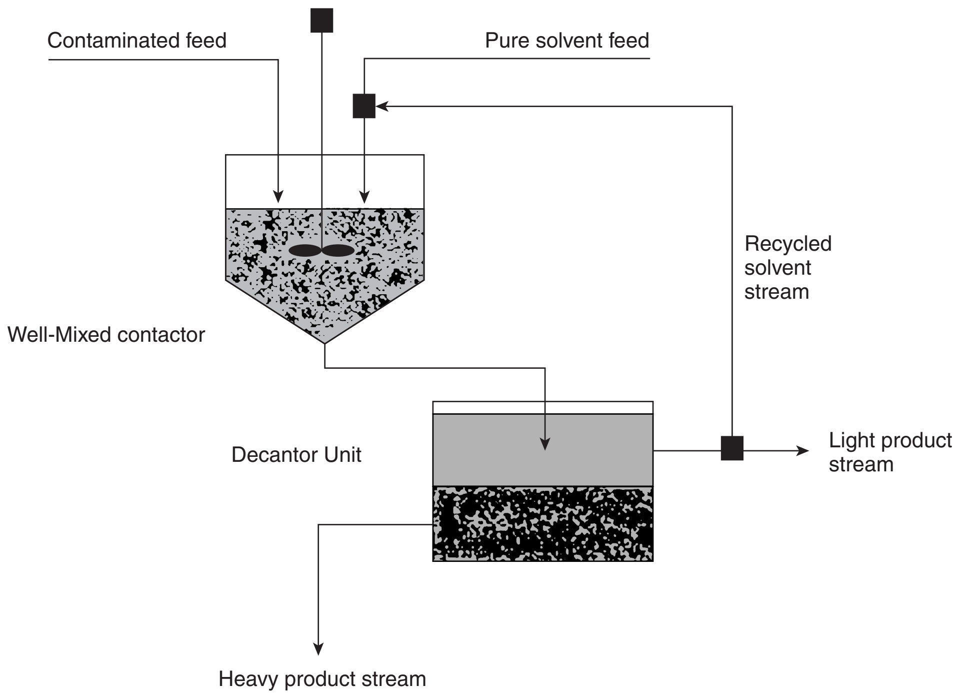


Figure 5

the ether vapor to escape, the mixture is shaken vigorously for some time. Then the funnel is returned to a stand and the two solvents are allowed to separate. The lighter solvent, now containing the target molecule, is decanted or siphoned off the top. The process is typically repeated three times. Then the second solvent is evaporated or reduced in volume.

Interestingly, at the scale of a process all the same things are done, but typically continuously for large scale production. Batch processes, however, can be scaled up to larger volumes, and this is done in processes that yield specialty chemicals or pharmaceuticals with high added value. We will consider the continuous process run at steady state. The physical situation is as shown in Figure 5.

The denser contaminated feed is mixed with the less dense pure solvent in a contactor. The well-mixed stream emerges from this unit and flows into the decanter unit where the two phases are given enough time to fully separate. This is done continuously, so at the entrance of the unit the two liquids are well mixed, but by the end, they are well separated, as shown in the schematic. We will not worry about the internal configuration of this unit. The top layer is the solvent, which leaves the unit with the impurity within it. Some of this solvent is removed from the unit continuously, but the balance is sent back to the contactor for further use. The heavier stream emerges from the decanter unit with a much reduced concentration of impurity. The analysis of this unit calls for a detailed analysis of the subunits that make it up.

We begin at the top of the unit with the pure solvent. The stream of solvent coming into the unit comes in with a density $\boldsymbol{\rho s}$ and a flow rate **qs**. This is mixed with the recycled stream that has a density $\boldsymbol{\rho}$**rs**, a flow rate **qrs**, and an impurity concentration of **Cirs**. The two streams are mixed combined into one with a flow rate **qsf**, a density of $\boldsymbol{\rho}$**sf**, and an impurity concentration of **Cisf**. The steady-state mass balances at the mixing tee are:

$$\rho\text{s qs} + \rho\text{rs qrs} = \rho\text{sf qsf}$$
$$\text{Cirs qrs} = \text{Cisf qsf}$$

At the contactor we have the impure heavy stream and the solvent stream being fed, and at the outlet the two have been mixed. The material balances for this unit are:

$$\rho\text{hf qhf} + \rho\text{sf qsf} = \rho\text{hc qhc} + \rho\text{sc qsc}$$
$$\text{Cihf qhf} + \text{Cisf qsf} = \text{Cihc qhc} + \text{Cisc qsc}$$

The decanter unit has these equations associated with it. Remember that we are not concerned with its internals but only with the mass flows into and out of it. The mass flow in is that of the mixed feed from the contactor. The flows out are those of the impure solvent and the purified heavy stream:

$$\rho\text{hc qhc} + \rho\text{sc qsc} = \rho\text{ds qds} + \rho\text{dh qdh}$$
$$\text{Cihc qhc} + \text{Cisc qsc} = \text{Cisd qds} + \text{Cihd qdh}$$

Finally, the solvent stream is split with one flow back to the inlet solvent tee and the other flow out of the unit. The equations that describe this are:

$$\rho\text{ds qds} = \rho\text{rs qrs} + \rho\text{sp qsp}$$
$$\text{Cisd qds} = \text{Cirs qrs} + \text{Cisp qsp}$$

The schematic of the flow sheet is shown once again (Figure 6) with all the streams labeled and with an imaginary box around the unit, which cuts all the streams that either enter or leave this unit.

The box is an imaginary *control surface* for the unit as a whole. Despite all the details that we have just considered, there is one overall set of mass balances for the unit as a whole. This treats the unit as a so-called "black-box," which means that even if the internal workings were hidden from view, we would be able to do an overall balance on the system, as shown in Figure 7.

As this greatly simplifies the initial stages of this problem, it is a logical place to begin. From Figure 7 and the conservation of mass, we can write that:

$$\rho\text{ihf qhf} + \rho\text{s qs} = \rho\text{dh qdh} + \rho\text{sp qsp}$$
$$\text{Cihf qhf} = \text{Cihd qdh} + \text{Cisp qsp}$$

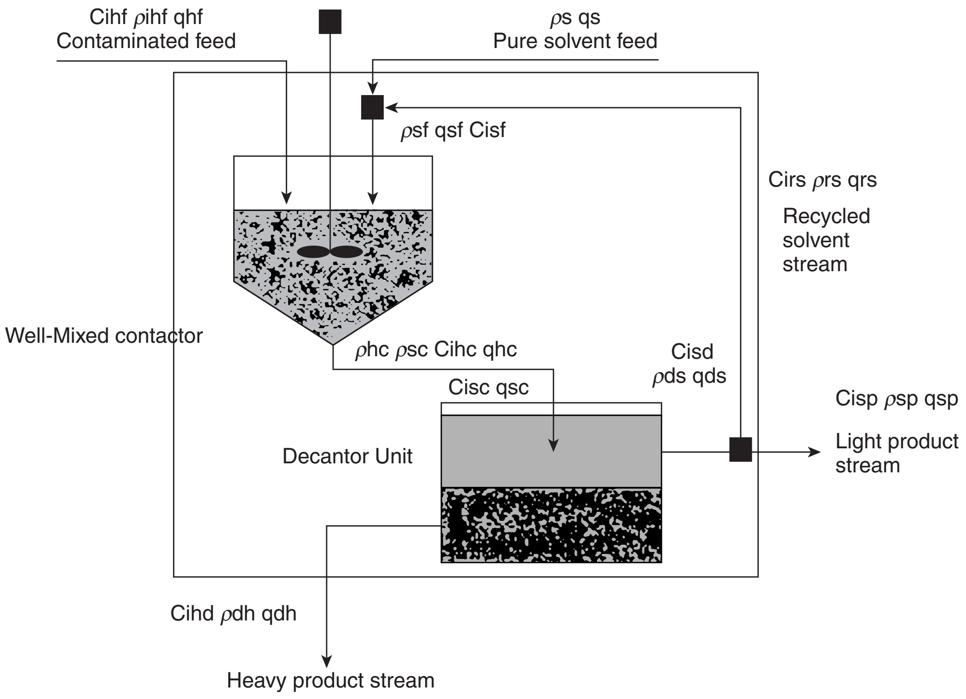


Figure 6

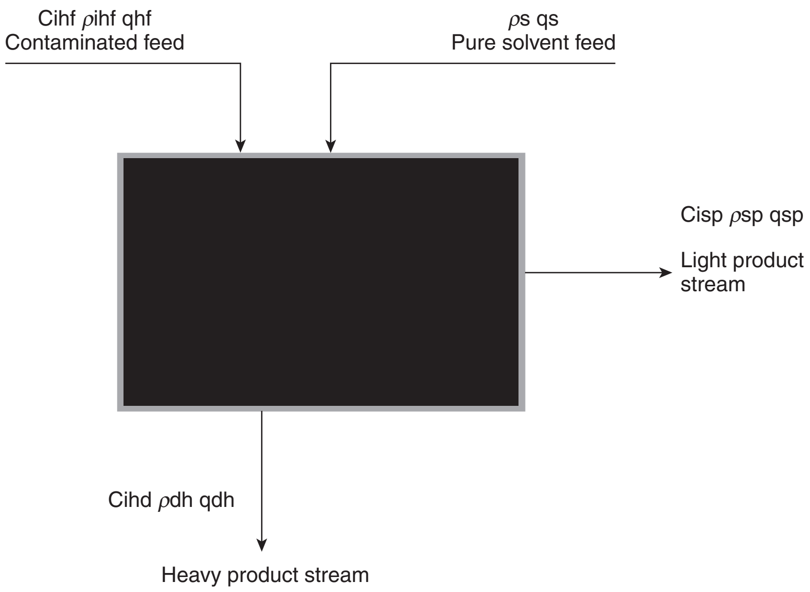


Figure 7

In these two equations we have a total of 14 parameters and variables. We need to reduce this number. How can we do this? The most crucial assumption we can make is that the unit runs at equilibrium. This means that the concentrations of the impurity in both liquid phases emerging from the unit are at equilibrium. To understand this we will pretend that the separation was done stepwise rather than continuously. The volume of light solvent added would be **qs Δt**. The volume of the impure stream would be **qihf Δt**. The initial concentration of the impurity in the heavy phase is **Cihf**. If the two phases are in contact, then the impurity will transfer spontaneously to the light phase where its affinity is higher. This transfer will occur until the concentrations of the impurity in the two phases are no longer changing—in other words, until the impurity comes to equilibrium between the two solvents. For this reason, a unit assumed to operate at the limit of equilibrium is referred to as an *equilibrium stage*, and this level of analysis is the *equilibrium stage analysis*. If the ratio of these two concentrations at equilibrium is a constant over a range of different concentrations, then the constant is referred to as the partition coefficient **Kd**:

$$\mathrm{Kd} = \frac{\mathrm{Cih},\, e}{\mathrm{Cis},\, e}$$

If the system we are examining also comes to an equilibrium condition, then the concentrations of impurity in the two outlet streams are coupled:

$$\mathrm{Kd} = \frac{\mathrm{Cihd}}{\mathrm{Cisp}}$$

It is also reasonable to expect that the densities of the contaminated streams are not too different from their pure densities, since the contaminant is usually at low concentrations:

$$\rho\mathrm{h}\ \mathrm{qhf} + \rho\mathrm{s}\ \mathrm{qs} = \rho\mathrm{h}\ \mathrm{qdh} + \rho\mathrm{s}\ \mathrm{qsp}$$
$$\mathrm{Cihf}\ \mathrm{qhf} = \mathrm{Cihd}\left(\mathrm{qdh} + \frac{1}{\mathrm{Kd}}\mathrm{qsp}\right)$$

```
In[61]:= Simplify[Solve[{ρhqhf + ρsqs == ρhqdh + ρsqsp,
                                      1
           Cihfqhf == Cihd(qdh + ——— qsp)}, {qdh, Cihd}]]
                                     Kd
         General::spell1 : Possible spelling error: new symbol
          name "ρs"is similar to existing symbol "ρh".

         General::spell1 : Possible spelling error: new symbol
          name "Cihd"is similar to existing symbol "Cihf".

                    qhf ρh + qs ρs - qsp ρs
Out[61]= {{qdh → ———————————————————————————,
                              ρh

                               Cihf Kd qhf ρh
          Cihd → ——————————————————————————————————————————}}
                  Kd qhf ρh + qspρh + Kd qs ρs - Kd qsp ρs
```

Thus we can show that the flow rate of heavy liquid from the unit is equal to its flow rate in plus a factor related to the ratios of the densities of the light and heavy liquids:

$$\text{qdh} = \frac{\text{qhf}\ \rho\text{h} + (\text{qs} - \text{qsp})\rho\text{s}}{\rho\text{h}} = \text{qhf} + \frac{\rho\text{s}}{\rho\text{h}}(\text{qs} - \text{qsp})$$

If the flow rate of the solvent is the same going in as coming out, then the flow rate of the heavy is the same in and out. Therefore, the concentration of the impurity in exit the flow is:

$$\text{Cihd} = \frac{\text{Cihf Kd qhf}\ \rho\text{h}}{(\text{Kd qhf} + \text{qsp})\rho\text{h} + \text{Kd}\ \ \rho\text{s}(\text{qs} - \text{qsp})} = \frac{\text{Cihf Kd qhf}\ \rho\text{h}}{(\text{Kd qhf} + \text{qsp})\rho\text{h}} = \text{Cihf}\left(\frac{1}{1 + \frac{\text{qsp}}{\text{Kd qhf}}}\right)$$

One way to use this result would be to compute the flow rate of the solvent that we would need in order to achieve a certain exit impurity concentration **Cihd** in the heavy stream, given **Kd**, the flow rate of the impure heavy feed and its impurity level **Cihd**.

```
In[62]:= Cihd = 10^-9;
         Cihf = 10^-4;
         Kd = .01;
         qhf = 100;
```

$$\texttt{NSolve[Cihd == Cihf}\ \frac{1}{1+\frac{\text{qsp}}{\text{Kd qhf}}}\texttt{), qsp]}$$

```
Out[66]= {{qsp → 99999.}}
```

We find that at these conditions, given that the impurity is 100× more soluble in the light solvent than in the heavy liquid, to reduce the concentration from 10^{-4} to 10^{-9} would require a solvent flow of 10^5 for a contaminated feed stream flow of 10^2. A calculation like this makes clear how costly cleanup can be.

Mass Transfer Analysis: Nonequilibrium. The previous calculation was helpful from a global perspective, but it assumes that the two streams really do come to equilibrium with respect to their impurity concentrations. Will they? How can we know this? What does it depend upon? The equilibrium stage analysis does not involve time but is simply based on thermodynamics. Yet, we know that thermodynamics can, in some cases, be misleading because we can compute the equilibrium position correctly, but for a real process it may take literally eons to move to that state. In other words, for design we need to have the time and the rate process uppermost in our minds. Equilibrium can tell us only how well we can do in the limit of everything going to its fullest extent of mass transfer. We must return then to the analysis of the units and focus our attention on the contactor, for this is the unit where the mixing and interphase mass transfer must take place. To assess how well this unit is doing, that is, how close the concentrations of the impurity in the exiting solvents are to equilibrium, we need to analyze the mass transfer rate explicitly, and especially if we are to do even a first-order design of this unit.

If we recall the material balances that we wrote around the contactor, then you may well be wondering where the rates of mass transfer come in:

$$\rho\text{hf qhf} + \rho\text{sf qsf} = \rho\text{hc qhc} + \rho\text{sc qsc}$$

$$\text{Cihf qhf} + \text{Cisf qsf} = \text{Cihc qhc} + \text{Cisc qsc}$$

The way we can answer this is to go back to our usual approach to this kind of problem and write the time-dependent mass balances for component **i** in each of the phases:

$$\text{Heavy phase: } \frac{d\ \text{Cihc Vhc}}{dt} = \text{Cihf qhf} - \text{Cihc qhc} - \text{Km Ai(Cihc} - \text{Kd Cisc)}$$

$$\text{Solvent phase: } \frac{d\ \text{Cisc Vsc}}{dt} = \text{Cisf qsf} - \text{Cisc qsc} + \text{Km Ai(Cihc} - \text{Kd Cisc)}$$

Components

i ⇒ impurity

h ⇒ heavy phase

s ⇒ lighter solvent phase

f ⇒ feed

c ⇒ contactor

If the contactor is at steady state, the left-hand side of each equation is identically zero. Adding the two equations and placing the terms for the heavy phase and the light solvent phase on opposite sides of the equation lead to the "steady-state" material balance we had before! Now we can see where the rates of mass transfer come in.

The rate of mass transfer that we introduced in this analysis requires some explanation. The constant **Kd** is the distribution coefficient for **i** between the two phases. **Km** and **Ai** are the mass transfer coefficient and the interfacial area. But what about the driving force term? Why is it written as the difference between the actual concentration of **i** in the first phase minus the actual concentration of **i** in the second phase multiplied by **Kd**?

$$\text{Driving force term} = (\text{Cihc} - \text{Kd Cisc})$$

This happens because the driving force to transfer species **i** from the first phase to the second is dependent upon the concentration of **i** in the second phase. Remember the reason an impurity transfers at all is that it is more soluble in the second phase. Thus, the concentration of **i** in the heavy phase may be well below its solubility limit, but it will still transfer to the second solvent phase because it is even farther below its solubility limit in that phase. We will put into words the rate of transfer of impurity from the heavy phase to the light phase:

> **The rate of mass transfer of i from phase h to s is proportional to the difference between the actual concentration of i in phase h and the concentration of i that *would be in equilibrium* with the actual concentration of i in phase s.**

If we try to put this into a mathematical sentence, it would look something like this:

$$r_{i,h\to s} = \text{Km Ai}(\text{Cihc} - \text{Cihc,e[Cisc]})$$

where **Cihc,e[Cisc]** means the concentration of **i** that ***would be in equilibrium*** with the actual concentration of **i** in phase **s**, that is, the theoretical concentration of **i** is a function of the concentration of **i** in the light solvent phase. However, that concentration is calculable from the partition constant:

$$\text{Cihc,e[Cisc]} = \text{Kd Cisc}$$
$$\therefore\ r_{i,h\to s} = \text{Km Ai}(\text{Cihc} - \text{Kd Cisc})$$

We could repeat the same arguments for the rate of transfer of **i** from the light solvent phase **s** to the heavy phase **h** and we would get the same expression, except that it would be the negative of the first:

$$r_{i,s\to h} = -\text{Km Ai}(\text{Cihc} - \text{Kd Cisc})$$

This is because any mass that appears in the second phase had to leave the first phase and it must appear in the second phase at the same absolute rate that it disappears from the first phase.

We can simplify these two equations by recognizing that the mass transferred between the two phases does not significantly affect the density of either phase nor its volume flow rate:

$$\textbf{h, Heavy phase}: \frac{d\ \text{Cihc}}{dt} = (\text{Cihf} - \text{Cihc})\frac{\text{qhc}}{\text{Vhc}} - \frac{\text{Km Ai}}{\text{Vhc}}(\text{Cihc} - \text{Kd Cisc})$$
$$\textbf{s, Solvent phase}: \frac{d\ \text{Cisc}}{dt} = (\text{Cisf} - \text{Cisc})\frac{\text{qsc}}{\text{Vsc}} + \frac{\text{Km Ai}}{\text{Vsc}}(\text{Cihc} - \text{Kd Cisc})$$

These two equations are nicely soluble, but before we solve them we should discuss them further. Notice that the convective flow rates are divided by the volumes of each phase.

These two terms are the reciprocal holding times for the two phases in the contactor, $\boldsymbol{\theta}_{\mathbf{hc}}^{-1}$ and $\boldsymbol{\theta}_{\mathbf{sc}}^{-1}$. The coefficients of the two driving force terms are the ratios of the product of the mass transfer coefficient and the interfacial area to the volume of the phase. Recalling that **Km** has dimensions of $\frac{\mathbf{Length}}{\mathbf{time}}$, we can see that this group is also an inverse time constant, but now this is a reciprocal characteristic time for mass transfer $\boldsymbol{\tau}^{-1}$. If we multiply through on both sides by the holding time we obtain:

$$\textbf{h, Heavy phase}: \theta_{\mathrm{hc}}\frac{d\,\mathrm{Cihc}}{dt} = (\mathrm{Cihf}-\mathrm{Cihc}) - \frac{\theta_{\mathrm{hc}}}{\tau\mathrm{hc}}(\mathrm{Cihc}-\mathrm{Kd\ Cisc})$$
$$\textbf{s, Solvent phase}: \theta_{\mathrm{sc}}\frac{d\,\mathrm{Cisc}}{dt} = (\mathrm{Cisf}-\mathrm{Cisc}) + \frac{\theta_{\mathrm{sc}}}{\tau\mathrm{sc}}(\mathrm{Cihc}-\mathrm{Kd\ Cisc})$$

We could go one more step and refer all the concentrations to the inlet concentration of the impurity in the heavy feed **Cihf**, which is a constant. If we do this we would be dividing both sides of both equations by this quantity to give the nondimensionalized concentrations **X**:

$$\textbf{h, Heavy phase}: \theta_{\mathrm{hc}}\frac{d\,\mathrm{Xihc}}{dt} = (1-\mathrm{Xihc}) - \frac{\theta_{\mathrm{hc}}}{\tau\mathrm{hc}}(\mathrm{Xihc}-\mathrm{Kd\ Xisc})$$
$$\textbf{s, Solvent phase}: \theta_{\mathrm{sc}}\frac{d\mathrm{Xisc}}{dt} = (\mathrm{Xisf}-\mathrm{Xisc}) + \frac{\theta_{\mathrm{sc}}}{\tau\mathrm{sc}}(\mathrm{Xihc}-\mathrm{Kd\ Xisc})$$

Finally, we can see that the time constants can also be used in the same way; we can multiply the second equation on both sides by $\frac{\boldsymbol{\theta}_{\mathbf{hc}}}{\boldsymbol{\theta}_{\mathbf{hc}}}$ and then reexpress both time derivatives in terms of the reduced time, that is, the ratio of real time to holding time:

$$\textbf{h, Heavy phase}: \theta_{\mathrm{hc}}\frac{d\,\mathrm{Xihc}}{dt} = (1-\mathrm{Xihc}) - \frac{\theta_{\mathrm{hc}}}{\tau\mathrm{hc}}(\mathrm{Xihc}-\mathrm{Kd\ Xisc})$$
$$\textbf{s, Solvent phase}: \theta_{\mathrm{sc}}\frac{\theta_{\mathrm{hc}}}{\theta_{\mathrm{hc}}}\frac{d\,\mathrm{Xisc}}{dt} = (\mathrm{Xisf}-\mathrm{Xisc}) + \frac{\theta_{\mathrm{sc}}}{\tau\mathrm{sc}}(\mathrm{Xihc}-\mathrm{Kd\ Xisc})$$
$$\textbf{h, Heavy phase}: \frac{d\,\mathrm{Xihc}}{d\theta} = (1-\mathrm{Xihc}) - \frac{\theta_{\mathrm{hc}}}{\tau\mathrm{hc}}(\mathrm{Xihc}-\mathrm{Kd\ Xisc})$$
$$\textbf{s, Solvent phase}: \frac{\theta_{\mathrm{sc}}}{\theta_{\mathrm{hc}}}\frac{d\,\mathrm{Xisc}}{d\theta} = (\mathrm{Xisf}-\mathrm{Xisc}) + \frac{\theta_{\mathrm{sc}}}{\tau\mathrm{sc}}(\mathrm{Xihc}-\mathrm{Kd\ Xisc})$$
$$\frac{d\,\mathrm{Xisc}}{d\theta} = (\mathrm{Xisf}-\mathrm{Xisc}) + \frac{\theta_{\mathrm{hc}}}{\tau\mathrm{sc}}(\mathrm{Xihc}-\mathrm{Kd\ Xisc})$$

Now we can obtain a general solution for this prototypical case, which can be used for specific cases simply by computing the time constants from the parameters or vice versa. The code for solving these analytically is shown here:

```
In[67]:= Remove[τhc, τsc, Xisf, Kd, θhc]

         sol = Flatten[
                 Simplify[
                  DSolve[
                 {∂θ Xihc[θ] == (1-Xihc[θ]) - θhc/τhc (Xihc[θ] - Kd Xisc[θ]),
                  ∂θ Xisc[θ] == (Xisf-Xisc[θ]) + θhc/τsc (Xihc[θ] - Kd Xisc[θ]),
                  Xihc[0] == 0, Xisc[0] == 0},
                 {Xihc[θ], Xisc[θ]},
                 θ]
                           ]
                          ];
         xihc[θ_] := sol[[1, 2]]
         xisc[θ_] := sol[[2, 2]]

         xihc[θ] // FullSimplify
         xisc[θ] // FullSimplify
```

```
        General::spell1 : Possible spelling error: new symbol
         name "τsc"is similar to existing symbol "τhc".

        General::spell1 : Possible spelling error: new symbol
         name "θhc is similar to existing symbol "τhc".

        General::spell1 : Possible spelling error: new symbol
         name "τhc"is similar to existing symbol "θhc".

        General::spell1 : Possible spelling error: new symbol
         name "Xisc"is similar to existing symbol "Xihc".

        General::spell1 : Possible spelling error: new symbol
         name "Xisf"is similar to existing symbol "Xisc".

        General::stop : Further output of General::spell1
         will be suppressed during this calculation.

        General::spell1 : Possible spelling error: new symbol
         name "xihc"is similar to existing symbol "Xihc".

        General::spell : Possible spelling error: new symbol
         name "xisc"is similar to existing symbols {xihc, Xisc}.
```

Out[71]=

$$\Big(e^{-\theta\left(1+\theta hc\left(\frac{1}{\tau hc}+\frac{Kd}{\tau sc}\right)\right)}(-1+Kd\,Xisf)\,\tau hc\,\tau sc^{2} + e^{-\theta}\Big(e^{\theta}\tau hc\,\tau sc^{2}+(-1+e^{\theta})\,Kd^{2}\,\theta hc\,\tau hc\,(\tau hc+Xisf\,\tau sc) + \frac{1}{2}(-1+e^{\theta})\,Kd\,\tau sc\,\big(2\theta hc\,(\tau hc+Xisf\,\tau sc) + \tau hc\,(2\,\tau hc+(1+e^{\theta})\,Xisf\,\tau sc)\big)\Big) - Kd\,Xisf\,\tau hc\,\tau sc^{2}\,\mathrm{Cosh}[\theta]\Big)\Big/ \Big((Kd\,\tau hc+\tau sc)(Kd\,\theta hc\,\tau hc+(\theta hc+\tau hc)\,\tau sc)\Big)$$

```
Out[72]= (- e^{-θ(1+θhc(1/τhc + Kd/τsc))} (-1 + Kd Xisf) τhc^2 τsc
           - e^{-θ} (τhc + Xisf τsc)(Kd θhc τhc + (θhc + τhc) τsc)
           + (Kd τhc + τsc)(θhc τhc + Xisf (θhc + τhc) τsc))/
           ((Kd τhc + τsc)(Kd θhc τhc + (θhc + τhc) τsc))
```

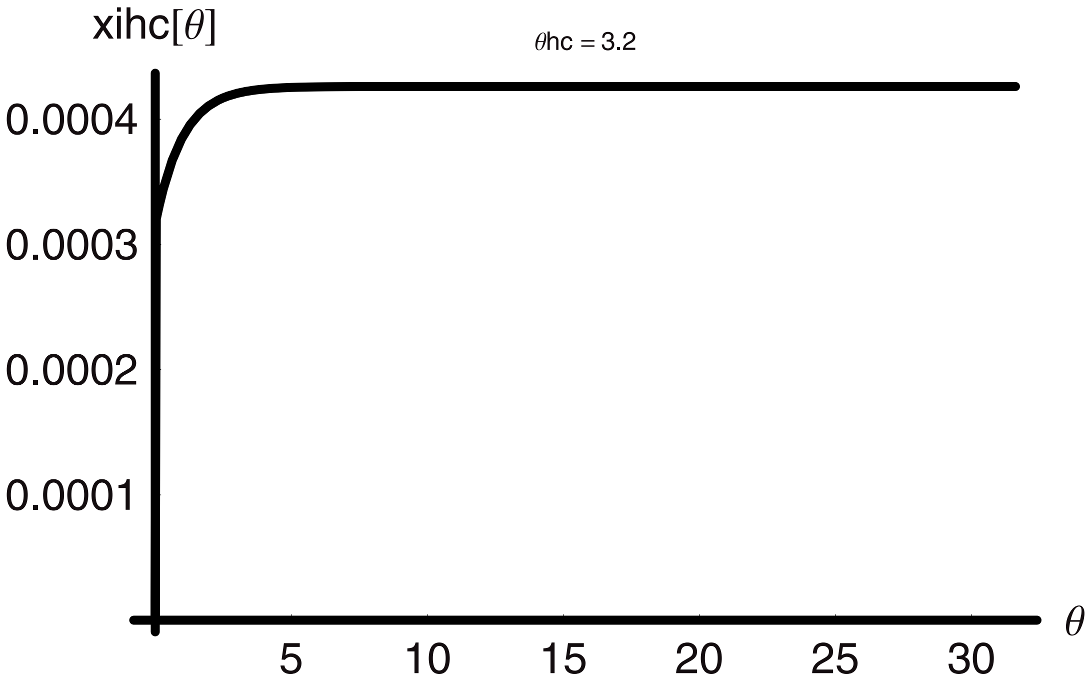
We can get a feel for these solutions by making some guesses as to the parameters. Let **τsc** be unity and **τhc** be 10^{-3} assuming based on the batch calculation we did earlier that we need about three orders of magnitude more solvent than feed. The magnitude of **Xisf** should be <1, and we can say that it may be as small as 10^{-2} or two orders of magnitude below the concentration of the impurity in the feed. It is also necessary to include the magnitude of **Kd**. The value we used earlier was 10^{-2}; we can use this again. The really difficult parameter to estimate is $\boldsymbol{\theta_{hc}}$, the holding time in the unit. We can test different values for this parameter to see its effect. The way we do it is to vary it by orders of magnitude, that is 10^n.

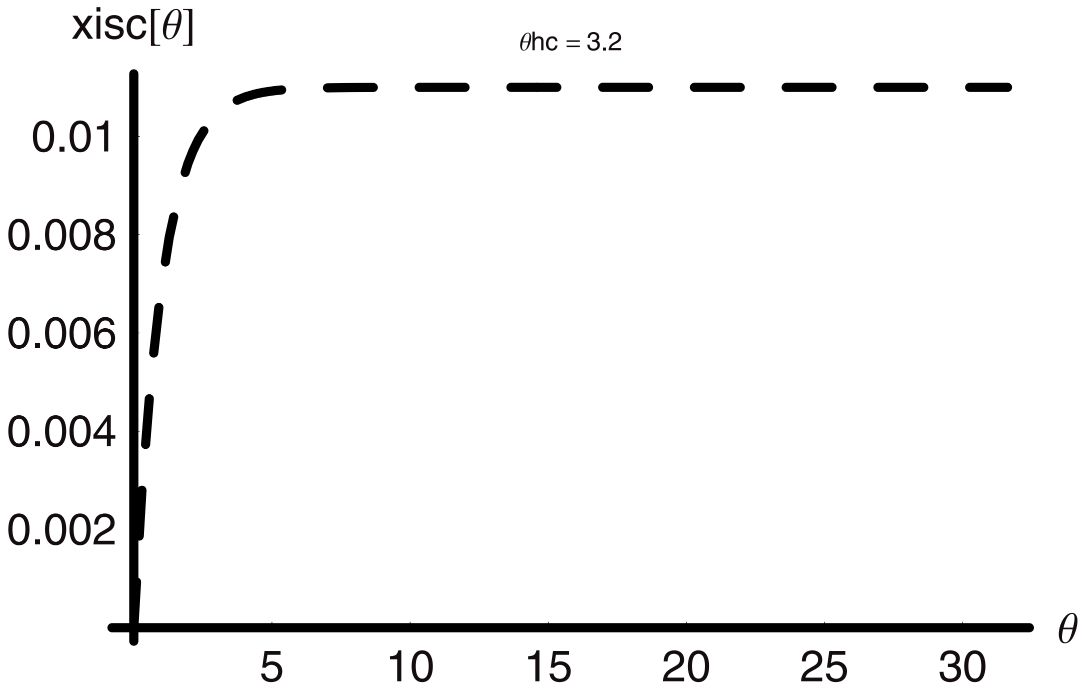
If we do this directly, it gets kind of sloppy after a few cases and we get annoying error messages about the choice of variable names that are somewhat too similar for *Mathematica*'s checker to be silent. A better way then to do this sort of calculation repetitively is to write a function call using "Module." The only variable we care about varying at this point is **n**, the exponent on 10 that sets the order of magnitude for the heavy liquid holding time in the unit. Therefore, we write one Module for each of the dimensionless concentrations. The first, **ifromh[n]**, is for **xihc**, the fraction of **i** left in **h** after contacting with **s**. The second, **itos[n]**, is **xisc**, which is the ratio of the concentration of **i** in **s** to the original concentration of **i** in **h**:

```
In[73]:= ?? NumberForm

         NumberForm[expr, n] prints with approximate real numbers
          in expr given to n-digit precision.

         Attributes[NumberForm] = {Protected}

         Options[NumberForm] = {DigitBlock → ∞,
          ExponentFunction → Automatic, ExponentStep → 1,
          NumberFormat → Automatic, NumberMultiplier → ×,
          NumberPadding → {,}, NumberPoint → .,
          NumberSeparator → ,, NumberSigns → {-,},
          SignPadding → False}

In[74]:= NumberForm[3.12256, 3]

Out[74]//NumberForm=
         3.12

In[75]:= ifromh[n_] := Module[
               {τhc = 10^-3, τsc = 10^0, Xisf = 10^-2, Kd = 10^-2,
                pl1, pl2},
               θhc = N[10^n];
          xihc[θ_] := (ⅇ^(-θ(1+θhc(1/τhc + Kd/τsc))) (-1 + Kd Xisf) τhc τsc^2
               + ⅇ^-θ (ⅇ^θ τhc τsc^2 + (-1 + ⅇ^θ) Kd^2 θhc τhc (τhc + Xisf τsc)

               + 1/2 (-1 + ⅇ^θ) Kd τsc (2 θhc(τhc + Xisf τsc)
               + τhc (2 τhc + (1 + ⅇ^θ) Xisf τsc)))
               - Kd Xisf τhc τsc^2 Cosh[θ])/
               ((Kd τhc + τsc)(Kd θhc τhc + (θhc + τhc) τsc));

               pl1 = Plot[{xihc[θ]}, {θ, 0, 10 θhc},
                 PlotStyle → Thickness[0.01],
                     AxesLabel → {"θ", "xihc[θ]"},
                         PlotLabel → StyleForm["θhc =",
                             NumberForm[θhc, 2], FontSize → 10],
                                PlotRange → All]
                  ]

In[76]:= itos[n_] := Module[
               {τhc = 10^-3, τsc = 10^0, Xisf = 10^-2, Kd = 10^-2,
                pl1, pl2},

               θhc = N[10^n];

          xisc[θ_] := (- ⅇ^(-θ(1+θhc(1/τhc + Kd/τsc))) (-1 + Kd Xisf) τhc^2 τsc
                 - ⅇ^-θ (τhc + Xisf τsc)(Kd θhc τhc + (θhc + τhc) τsc)
                 +(Kd τhc + τsc)(θhc τhc + Xisf (θhc + τhc) τsc))/
                 ((Kd τhc + τsc)(Kd θhc τhc + (θhc + τhc) τsc));

               pl2 = Plot[{xisc[θ]}, {θ, 0, 10θhc},
                 PlotStyle → {{Thickness[0.01],
                              Dashing[{0.05, 0.05}]}},
                      AxesLabel → {"θ", "xisc[θ]"},
                        PlotLabel →
                         StyleForm["θhc ="NumberForm[θhc, 2],
                         FontSize → 10], PlotRange → All]
                  ]
```

We can see how the two **Module** functions work by choosing a value of **n**, say, unity, and testing them:

```
In[77]:= ifromh[.5];
         itos[.5];
```

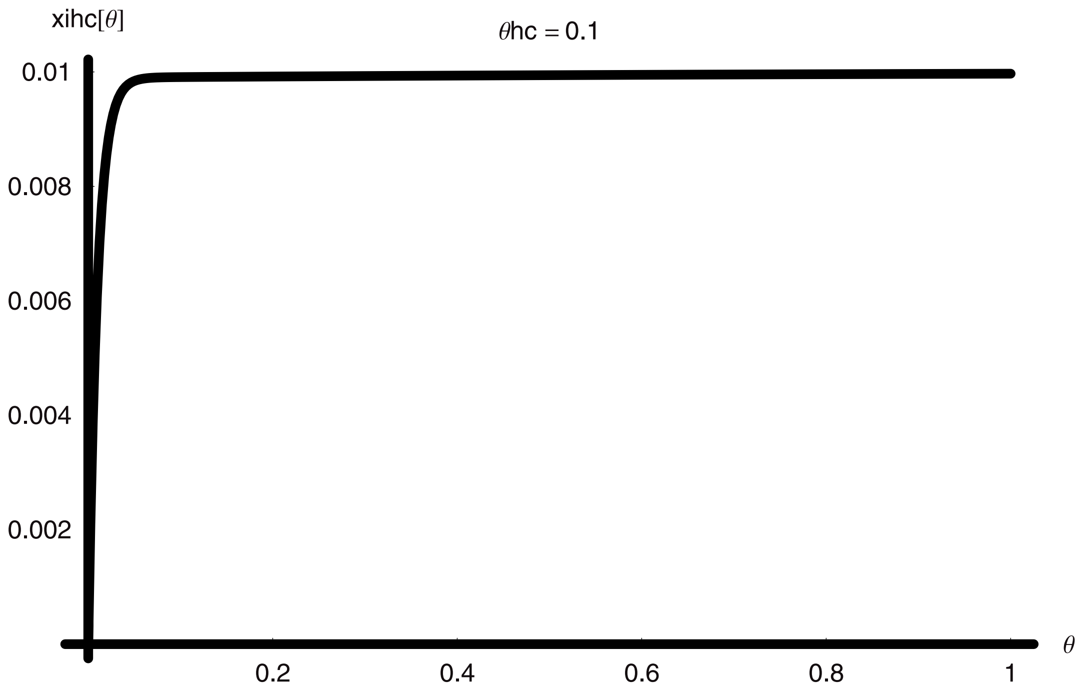

xihc[θ]
θhc = 3.2
0.0004
0.0003
0.0002
0.0001
5
10
15
20
25
30
θ

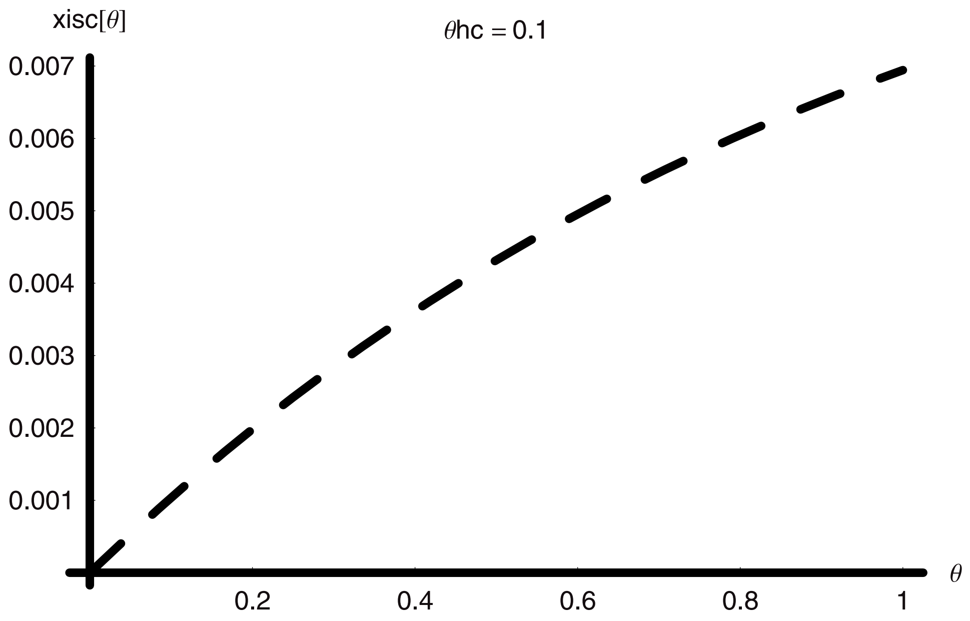

xisc[θ]
θhc = 3.2
0.01
0.008
0.006
0.004
0.002
5
10
15
20
25
30
θ

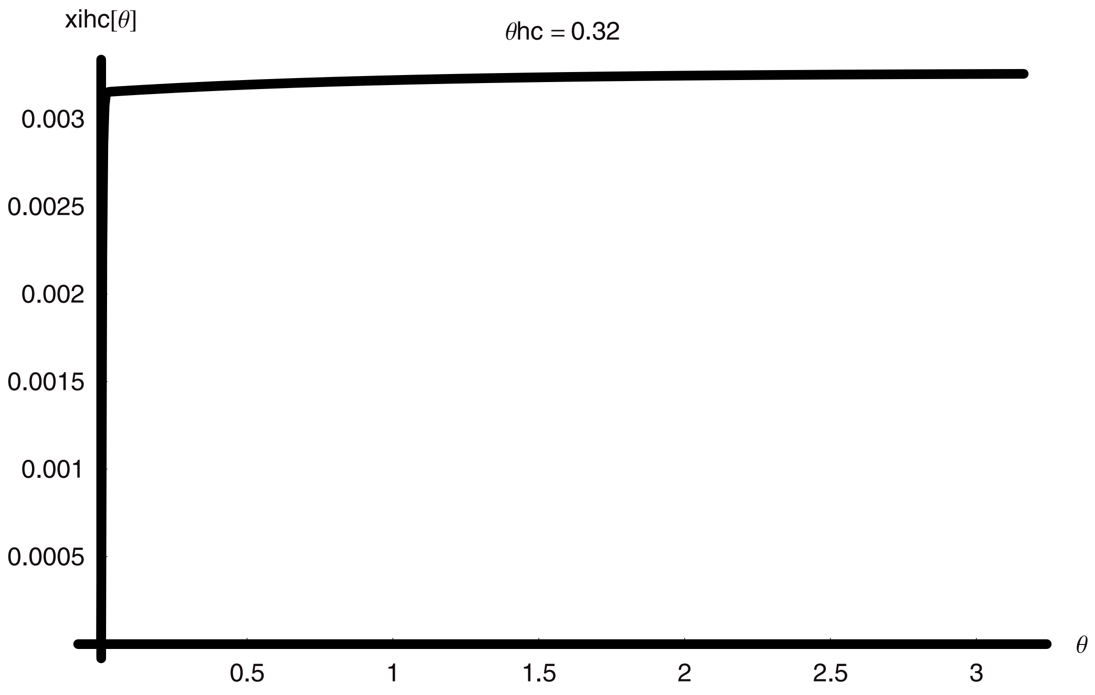
Within the **Module** functions we could have placed a semicolon ";" after the **Plot** commands. This would have allowed the graphs to be rendered, but the output "**Graphics**" would have been lost. We need the "**Graphics**" in order to plot arrays of these two functions with varying values of **n**. Therefore, we have left the semicolon out of the **Modules**. We can now use these in arrays and stacks.

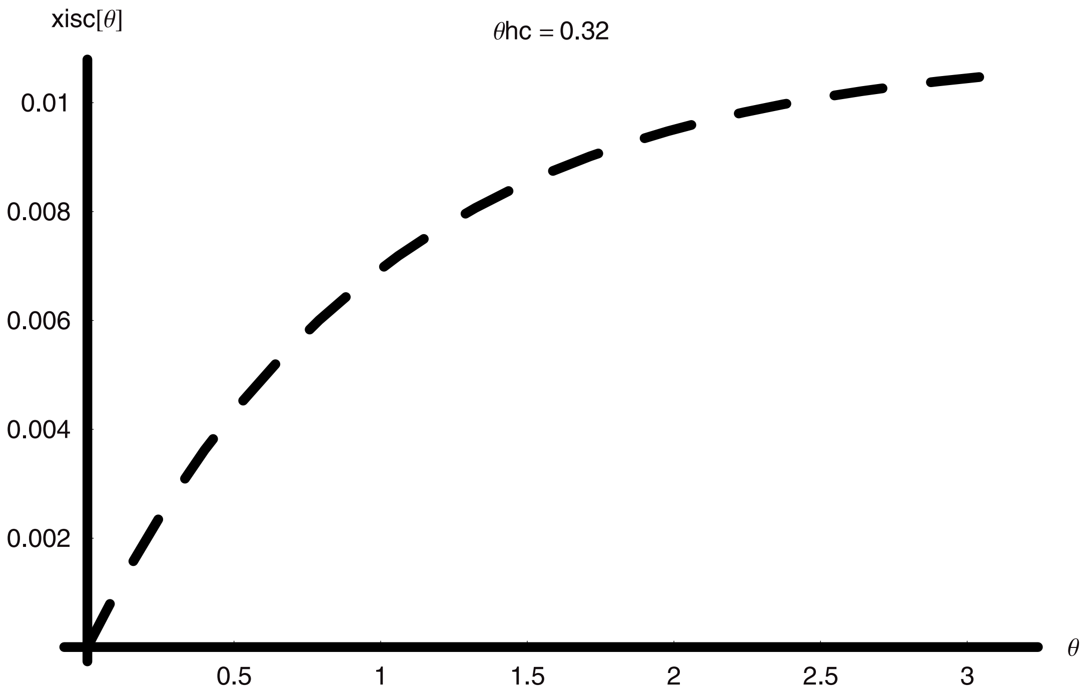
We can place the two new functions we have written in a **Table** and let **n** vary from −1 to 1 in order to see how the two ratioed concentrations vary with decade increases in the holding time of the heavy stream:

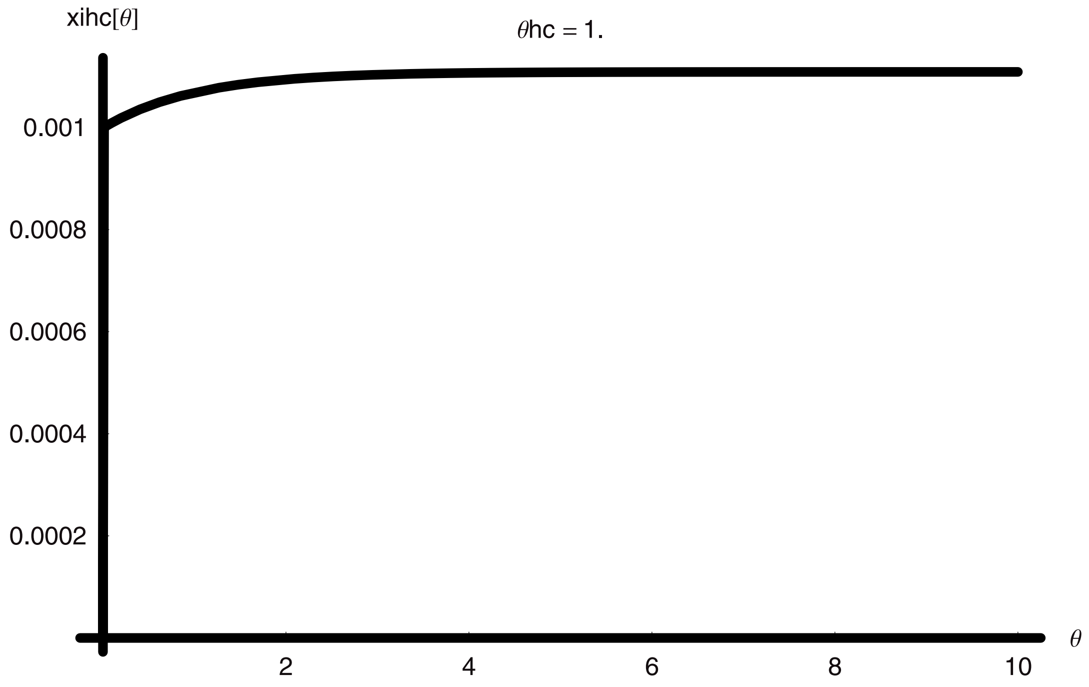
```
In[79]:= SetOptions[{Plot, ListPlot}, AxesStyle → {Thickness[0.01]},
           PlotStyle → {PointSize[0.015], Thickness[0.006]},
           DefaultFont → {"Helvetica", 10}];

In[80]:= Table[{ifromh[n], itos[n]}, {n, -1, .5, .5}];
```

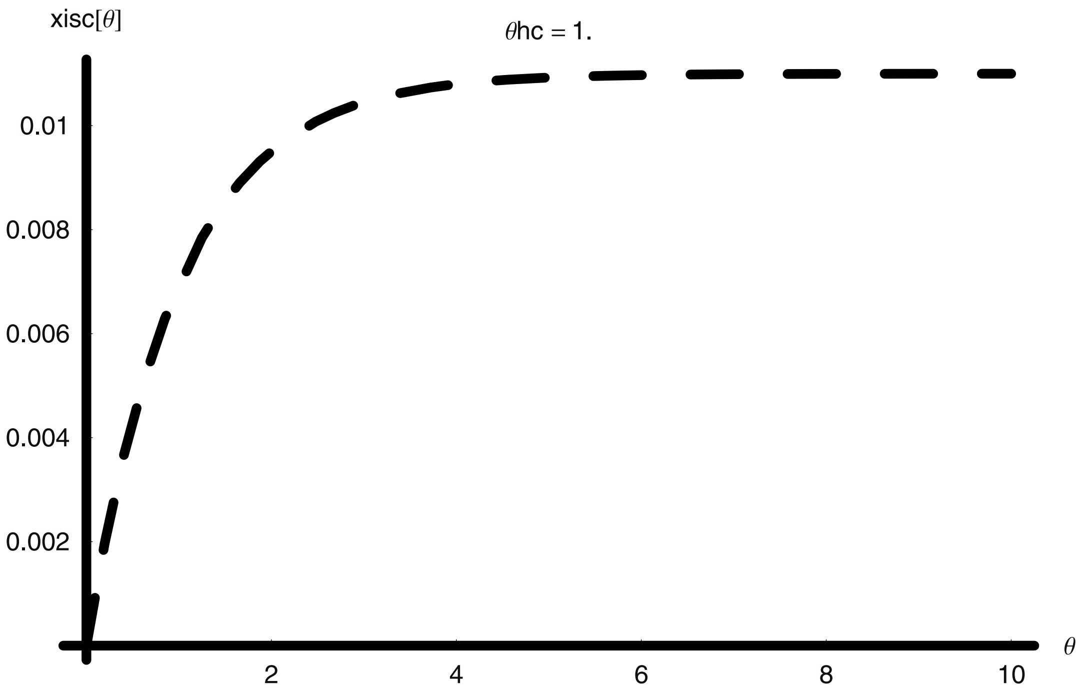


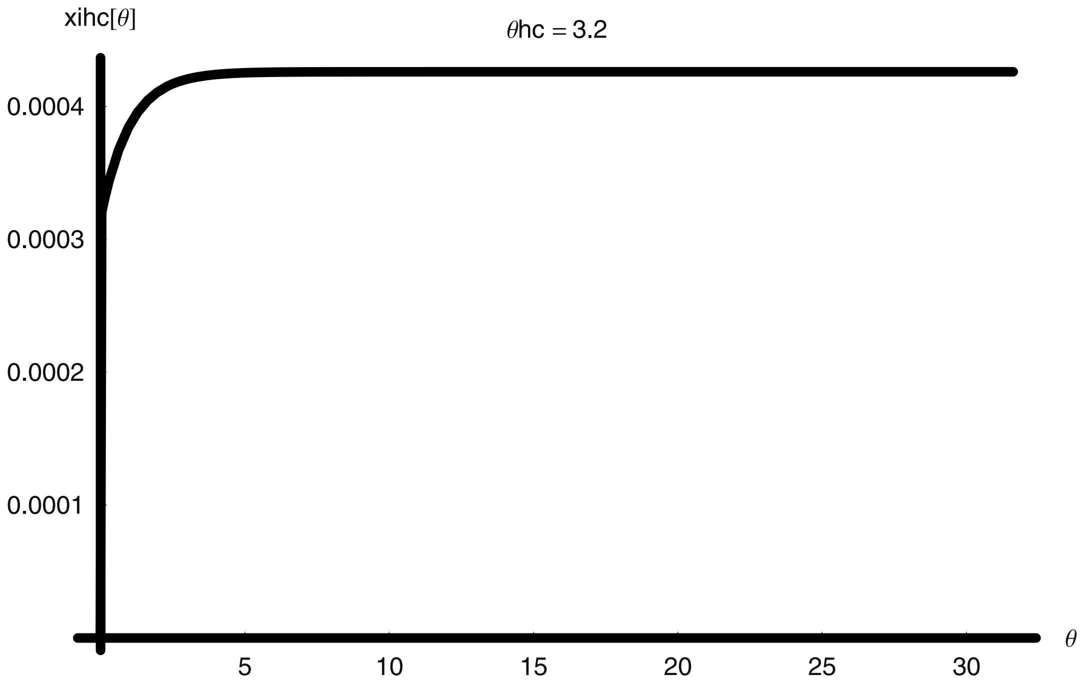

xisc[θ]
θhc = 0.1
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.2
0.4
0.6
0.8
1
θ

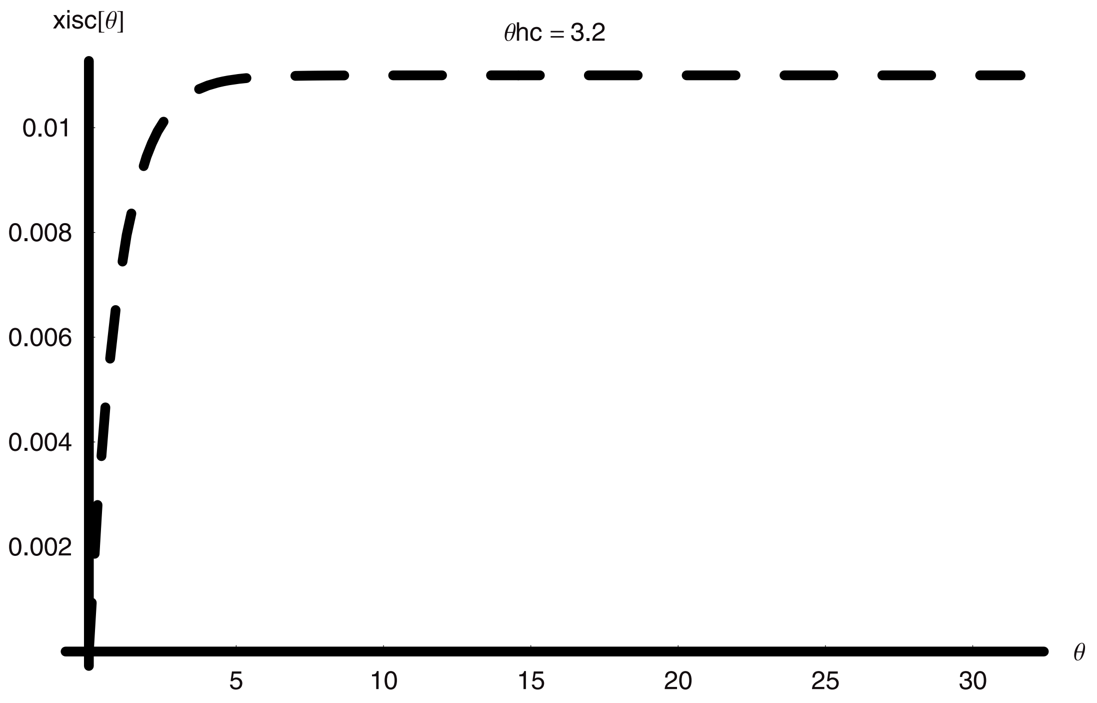

xihc[θ]
θhc = 0.32
0.003
0.0025
0.002
0.0015
0.001
0.0005
0.5
1
1.5
2
2.5
3
θ


xisc[θ]
θhc = 0.32
0.01
0.008
0.006
0.004
0.002
0.5
1
1.5
2
2.5
3
θ


xihc[θ]
θhc = 1.
0.001
0.0008
0.0006
0.0004
0.0002
2
4
6
8
10
θ

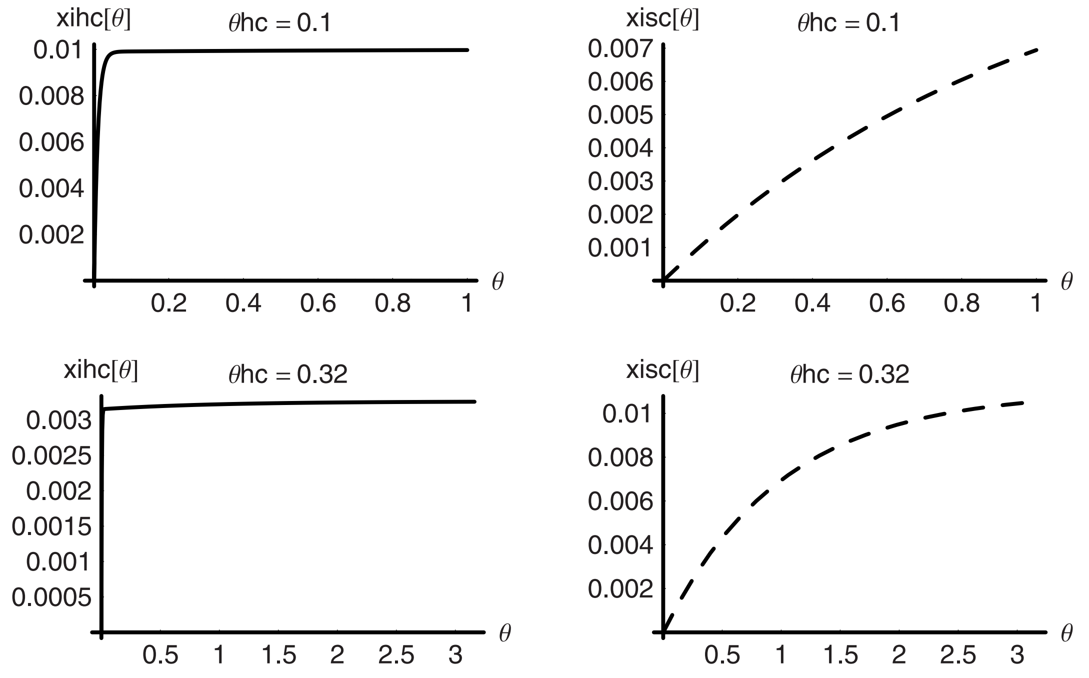

xisc[θ]
θhc = 1.
0.01
0.008
0.006
0.004
0.002
θ
2
4
6
8
10

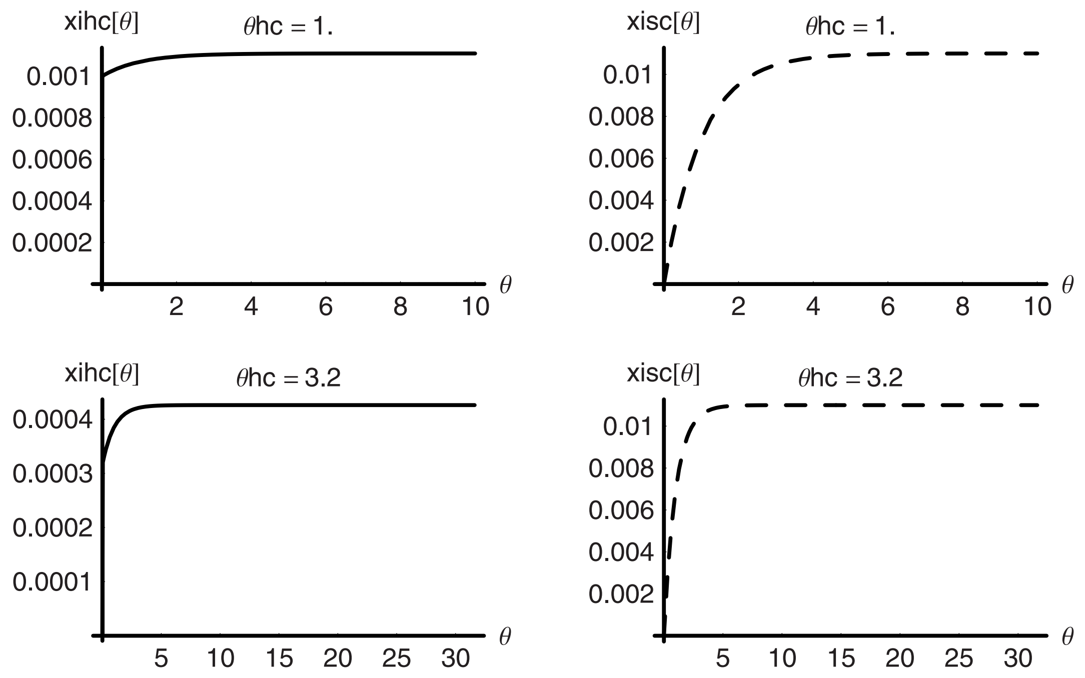

xihc[θ]
θhc = 3.2
0.0004
0.0003
0.0002
0.0001
θ
5
10
15
20
25
30



Now these can be assembled into a **GraphicsArray** within the **Show** command for a more pleasing presentation of the changes:

```
In[81]:= Show[GraphicsArray[%]];
```





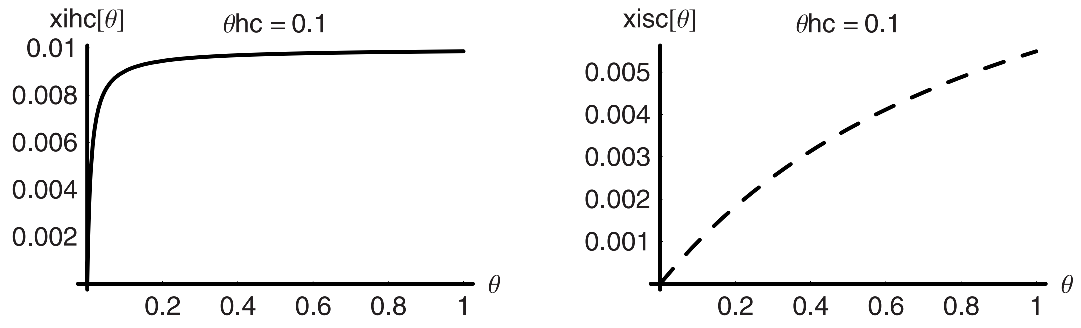
If after seeing this **GraphicsArray**, we realize that we prefer this type of presentation, then we might be tempted to place the **Table** command that generates the **Graphics** directly into the **Show[GraphicsArray[ ]]** statement, but this would not do what we want. It would give us the standard output of the individual **Graphics** and then the array. If we want only the array we need to go back and modify the **Module** functions so that the rendering of the graphs is delayed until we call for them. We do this with **DisplayGraphics** → **Identity** in the **Plot** commands:

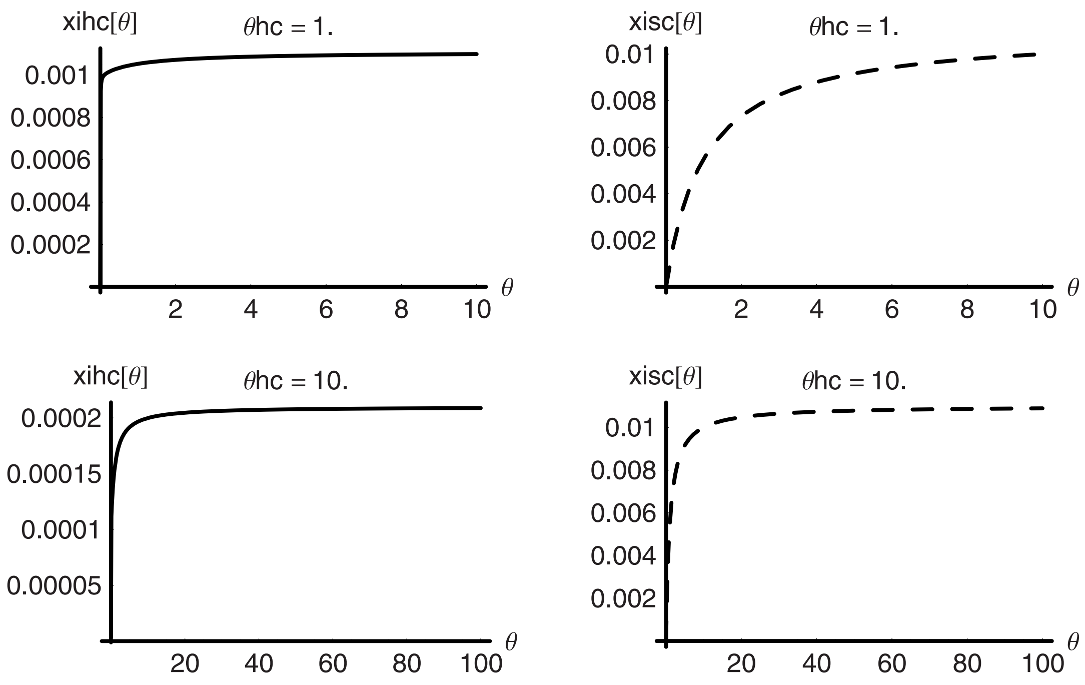
```
In[82]:= ifromh[n_] := Module[
                {τhc = 10^-3, τsc = 10^0, Xisf = 10^-2, Kd = 10^-2,
                  pl1, pl2},
                θhc = N[10^n];
                         θ((1 + θ)τhc τsc + Kd θ θhc(τhc + Xisf τsc))
          xihc[θ_]:= ─────────────────────────────────────────────────;
                     (1 + θ)(Kd θ θhc τhc + (τhc + θ(θhc + τhc)) τsc)
                pl1 = Plot[{xihc[θ]}, {θ, 0, 10θhc},
                        PlotRange → All,
                        PlotStyle → Thickness[0.01],
                        AxesLabel → {"θ", "xihc[θ]"},
                        PlotLabel → StyleForm["θhc ="
                         NumberForm[θhc, 2], FontSize → 10],
                        DisplayFunction → Identity]
          ]
```

```
In[83]:= itos[n_] := Module[
               {τhc = 10^-3, τsc = 10^0, Xisf = 10^-2, Kd = 10^-2,
                pl1, pl2},
               θhc = N[10^n];
                    θ(Xisf τhc τsc + θ(Xisf τhc τsc + θhc(τhc + Xisf τsc)))
         xisc[θ_] := ------------------------------------------------------ ;
                      (1 + θ)(Kd θ θhc τhc + (τhc + θ(θhc + τhc))τsc)
            pl2 = Plot[{xisc[θ]}, {θ, 0, 10θhc},
                       PlotRange → All,
                       PlotStyle → {{Thickness[0.01],
                        Dashing[{0.05, 0.05}]}},
                       AxesLabel → {"θ", "xisc[θ]"},
                       PlotLabel → StyleForm["θhc ="
                        NumberForm[θhc, 2], FontSize → 10],
                       DisplayFunction → Identity]
               ]
```

When we run the **Module** functions in this form, we obtain just the **Graphics** output without the rendering. This can now be placed directly inside the **Show[GraphicsArray[ ]]** command:

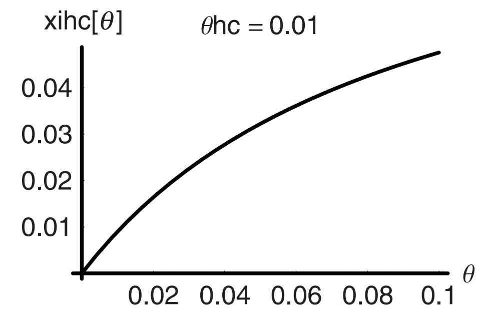
```
In[84]:= Table[{ifromh[n], itos[n]}, {n, -1, 1}]

Out[84]= {{-Graphics-, -Graphics-}, {-Graphics-, -Graphics-},
          {-Graphics-, -Graphics-}}

In[85]:= SetOptions[{Plot, ListPlot}, AxesStyle → {Thickness[0.01]},
           PlotStyle → {PointSize[0.015], Thickness[0.006]},
           DefaultFont → {"Helvetica", 10}];

In[86]:= Show[GraphicsArray[Table[{ifromh[n], itos[n]},
          {n, -1, 1}]]];
```

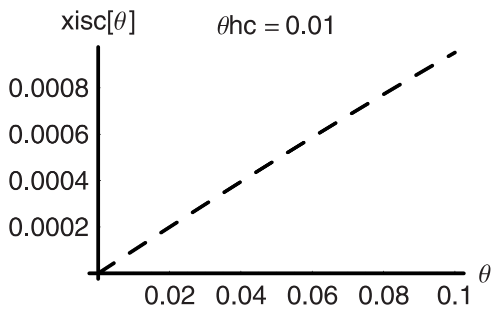


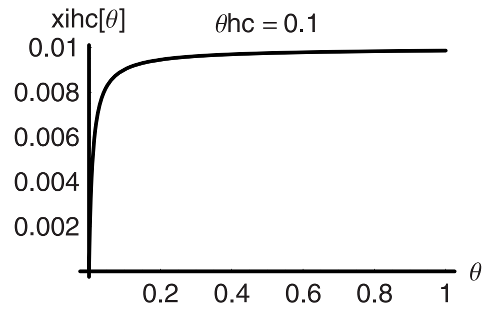


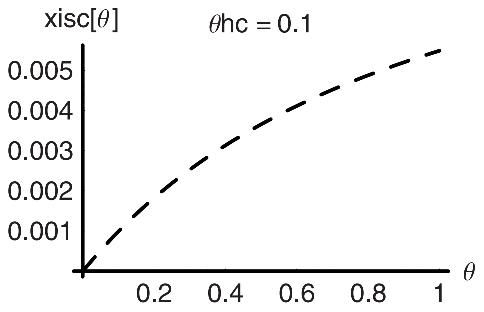
Using these ideas in tandem we can examine the effect of the holding time of the heavy feed in the contactor over a range of 10^6 as shown here:

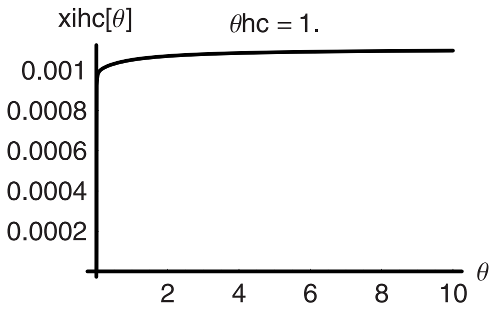
```
In[87]:= Table[{ifromh[n], itos[n]}, {n, -2, 4}];
         Show[GraphicsArray[%]];

         NumberForm::sigz : In addition to the number of digits
          requested, one or more zeros will appear as placeholders.

         NumberForm::sigz : In addition to the number of digits
          requested, one or more zeros will appear as placeholders.

         NumberForm::sigz : In addition to the number of digits
          requested, one or more zeros will appear as placeholders.

         General::stop : Further output of NumberForm::sigz
          will be suppressed during this calculation.
```

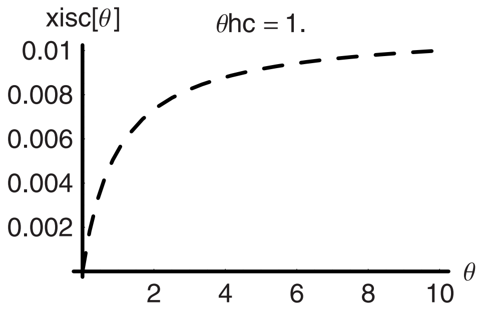

xihc[θ]
θhc = 0.01
0.04
0.03
0.02
0.01
0.02 0.04 0.06 0.08 0.1
θ

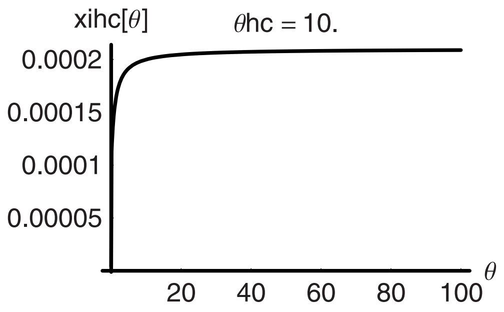

xisc[θ]
θhc = 0.01
0.0008
0.0006
0.0004
0.0002
0.02 0.04 0.06 0.08 0.1
θ

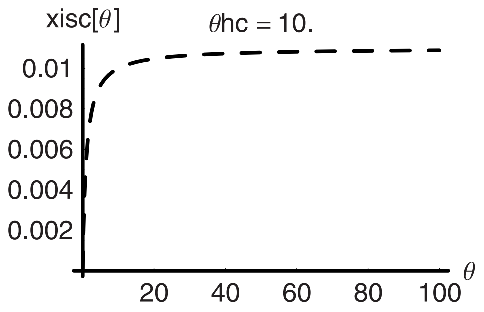

xihc[θ]
θhc = 0.1
0.01
0.008
0.006
0.004
0.002
0.2 0.4 0.6 0.8 1
θ

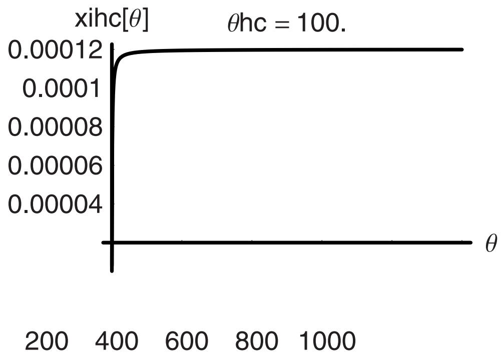

xisc[θ]
θhc = 0.1
0.005
0.004
0.003
0.002
0.001
0.2 0.4 0.6 0.8 1
θ

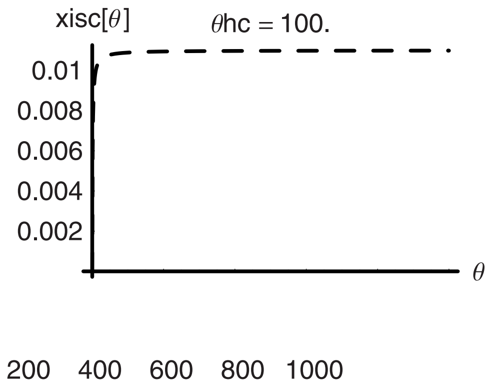

xihc[θ]
θhc = 1.
0.001
0.0008
0.0006
0.0004
0.0002
2 4 6 8 10
θ

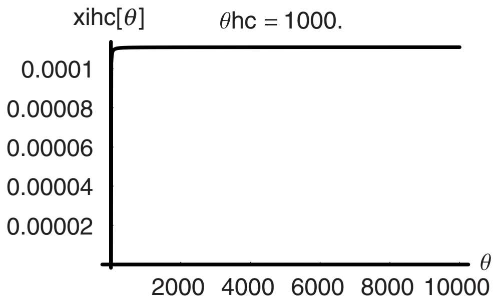

xisc[θ]
θhc = 1.
0.01
0.008
0.006
0.004
0.002
2 4 6 8 10
θ

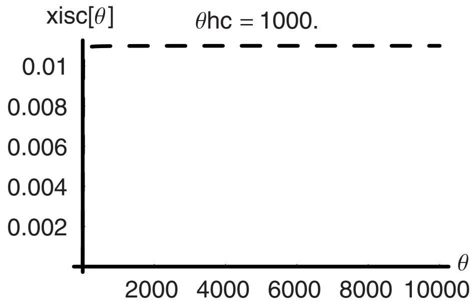

xihc[θ]
θhc = 10.
0.0002
0.00015
0.0001
0.00005
20 40 60 80 100
θ

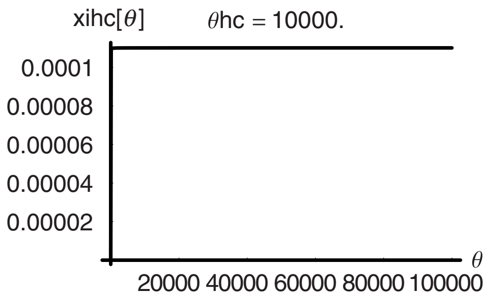

xisc[θ]
θhc = 10.
0.01
0.008
0.006
0.004
0.002
20 40 60 80 100
θ

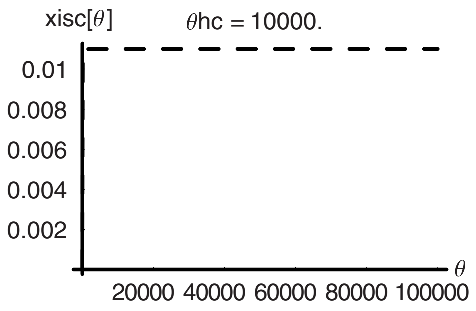












Recall that the $\boldsymbol{\theta}$-axis in these graphs is the reduced time, that is, the real time ratioed to the holding time. (Note: In the last six graphs the time axis is too compressed and the time labels are too close together. This could be overcome by splitting the first set of outputs from the second with two function calls and appropriately different plotting options.) We notice in the first case with a short holding time of 0.01 that the unit has not yet reached a steady state even after 10 holding times have passed. In the cases that follow, 10 holding times are more than enough time to ensure that a steady state has been derived. When we look at the data, we can see that the impurity levels are much reduced in the heavy stream as it exits the reactor. Even in the first case, at 10 holding times, the concentration is ~5% of that of the inlet stream. As we increase the holding times by factors of 10, we see that the picture improves; in fact, we notice that the fractional concentration of the impurity at steady state decreases by a factor of 10 for every factor of 10 increase in the holding time. Remember that holding time is just the ratio of the volume of the impure stream to its volume flow rate through the unit. Increasing the holding time at a fixed flow rate is the same as increasing the volume of the impure feed in the unit, or in other words the same as making the unit bigger. This way a larger contactor unit gives a larger holding time and more effective transfer of the impurity to the extracting solvent phase. Depending upon how far below the inlet feed concentration the exit concentration of impurity needs to be, we would use this calculation to find the volume of the system required. Keeping everything else the same, reducing the concentration by a factor of 10^6 between exit and inlet would require a much larger unit than the unit that would reduce this by only a factor of 10.

We will come back to this overall unit scheme later in our studies when we seek to write models for a group of units. For now we can use the knowledge we have gained here by applying it to some other seemingly different systems that are actually quite similar at the level of analysis.

5.8 Summary

Now that we have seen how mass moves between phases and those factors that control the rate of this process, we can bore in at the molecular level on mass action, especially adsorption.

CHAPTER 6

Adsorption and Permeation

6.1 Adsorption

Net Rate of Adsorption

Adsorption is a fundamental process of separation that is practiced for different purposes; removal of volatile organics contaminants (VOCs) from air is one, removal of water vapor from nitrogen is another. Hydrogen purification, that is, removing trace quantities of hydrocarbons, is important, especially for applications in electronics fabrication processes (such as metal organic chemical vapor deposition (MOCVD)), which require "ten nines" and better purity (that is less than one part impurity in 10^9 parts of the gas!). Although adsorption will not give this level of purification, it is one of the methods that can be used in the process of producing such high purity hydrogen. Diffusion of hydrogen through a palladium membrane gives the highest attainable purity, as hydrogen and only hydrogen can be transported through the metal. We will cover permeation after we examine adsorption.

Although we include adsorption here following the chapter on mass transfer, we should be clear that it is a very specific process in its fullest fundamental meaning. Adsorption is the process by which molecules in the fluid phase in contact with a solid move to the solid surface and interact with it. Once at the solid surface these molecules may be reversible or irreversible adsorbed, that is, they may come back off the surface to the fluid phase with their full molecular integrity intact, or they may be so strongly bound that the rate of removal is for all purposes close enough to zero to be considered zero.

When the discussion turns to removal of some component from a fluid stream by a high surface area porous solid, such as silica gel, which is found in many consumer products (often in a small packet and sometimes in the product itself), then the term "adsorption" becomes more global and hence ambiguous. The reason for this ironically is that mass transfer may be convoluted with adsorption. In other words the component to be adsorbed must move from the bulk gas phase to the near vicinity of the adsorbent particle, and this is termed external mass transfer. From the near external surface region, the component must now be transported through the pore space of the particles. This is called internal mass transfer because it is within the particle. Finally, from the fluid phase within the pores, the component must be adsorbed by the surface in order to be removed from the gas. Any of these processes, external, internal, or adsorption, can, in principle, be the slowest step and therefore the process that controls the observed rate. Most often it is not the adsorption that is slow; in fact, this step usually comes to equilibrium quickly (after all just think of how fast frost forms on a beer mug taken from the freezer on a humid summer afternoon). More typically it is the internal mass transport process that is rate limiting. This, however, is lumped with the true adsorption process and the overall rate is called "adsorption." We will avoid this problem and focus on adsorption alone as if it were the rate-controlling process so that we may understand this fundamentally.

True adsorption is a "mass action" process rather than a mass transfer process. What this means is that it will occur even in the absence of a concentration gradient between the bulk gas and the surface. It comes about due to the rapid and chaotic motion of the fluid phase molecules, and their impingement on the surface. From the elementary kinetic theory of an ideal gas we can compute the number of molecules impinging upon a surface per unit time per unit area at a given temperature and pressure. It is:

$$\frac{\text{Number Molecules}}{\text{area time}} = \frac{1}{4}\text{CL}\bar{v} = \frac{1}{4}\sqrt{\frac{8RT}{\pi\text{MW}}}\frac{\text{PL}}{RT} = P\text{L}\sqrt{\frac{1}{2\pi\, RT\text{MW}}} = \frac{P}{\sqrt{2\pi\, RT\text{MW}}}$$

Hence the number of molecules hitting the surface per unit time per unit area is a flux. Also, it is proportional to the pressure of the gas and the mean speed of the gas molecules and to $T^{-\frac{1}{2}}$. At room temperature and pressure the impingement frequency of nitrogen is:

In[1]:= $\frac{1}{4}\sqrt{\frac{8RT}{\pi\,\text{MW}}\frac{\text{PL}}{RT}}$ == NumberForm[
PowerExpand[
N[$\frac{.8\,\text{atm}}{4}\frac{6.02\,10^{23}\text{mole}^{-1}}{1000\frac{\text{cm}^3}{\text{L}}0.08205\frac{\text{Latm}}{\text{mole K}}300\text{ K}}\sqrt{\frac{88.314\,10^{7}\frac{\text{Joule}}{\text{mole K}}\frac{\frac{\text{g cm}^2}{\text{s}^2}}{\text{Joule}}\;300\text{ K}}{\pi\,28\,\frac{\text{g}}{\text{mole}}}}$]],2]

Out[1]= $\frac{\text{PL}\sqrt{\frac{\text{RT}}{\text{MW}}}}{\sqrt{2\pi}\,\text{RT}} == \frac{2.3\times 10^{23}}{\text{cm}^2\text{s}}$

Thus, nearly one-third of a mole of nitrogen molecules strikes every square centimeter every second. No wonder the time to equilibration of adsorption is so fast!

Irving Langmuir, the Nobel prize-winning industrial physical chemist who worked at General Electric, built an elegant structure upon this foundation in kinetic theory. He reasoned that not every molecule would adsorb, but only some would do so. Furthermore, one reason for this was that to be adsorbed there should be a site for adsorption to occur. It stands to reason then that on the basis of mass action, the rate of adsorption should be proportional to the concentration of molecules in the gas phase and to the number of sites available on the surface. Additionally, the rate should be related at any time to the number of sites not covered at that time rather than to the total number of sites present per unit area. Conversely, and again by the principle of mass action, the rate of desorption should be proportional to the number of sites currently occupied at that time. Using **ka** and **kd** as the proportionality constants (that we will call the rate constant for adsorption and desorption, respectively), we can write the net rate of adsorption for gas phase species i as the difference between the rate of adsorption and the rate of desorption:

$$\text{rate}_{i,\text{ads,net}} = k_{\text{ai}} C_{ig} (C_{i-\text{site,total}} - C_{i-\text{sites,occupied by }i}) - k_{d,s} C_{i-\text{sites,occupied by }i}$$

All the sites are assumed to be identical, and the adsorption at one side does not affect that at another site, that is, they interact with the gas phase independently. In addition to the two rate constants the term $\boldsymbol{C_{i-\text{site,total}}}$ is also a constant and is the number of sites available on the solid per unit area. This raises another point: if this and the concentration of occupied sites are written on a per unit area basis, and the gas phase concentration $\boldsymbol{C_{ig}}$ is written on a per unit volume basis, then what are the dimensions of the rate constant?

The net rate of adsorption is the number or moles of molecules adsorbed per unit area per unit time, where the area is the area made available for adsorption by a given mass or volume of the adsorbent solid. Therefore, the two rates on the right-hand side must also be moles per unit area-time. This means that the rate constants must be dimensioned as follows:

$$\frac{\text{mole}}{\text{length}^2\ \text{time}} = \left[\frac{\textbf{length}^3}{\textbf{mole time}}\right]\left[\frac{\text{mole}}{\text{length}^3}\right]\left(\left[\frac{\text{mole}}{\text{length}^2}\right]\right) - \left[\frac{\textbf{1}}{\textbf{time}}\right]\left(\left[\frac{\text{mole}}{\text{length}^2}\right]\right)$$

These dimensions (bold) are what we expect from mass action kinetics for a second-order and for a first-order rate constant.

Consider now an adsorbent that offers little or no resistance to mass transfer because it is "macroporous." This means that the pores within the solid are large (macro), that is, greater than 20 nm in diameter or width, and that transport of small molecules (0.2 nm) is unhindered and takes place as if they were in the bulk phase surrounding the solid. This means that the bulk gas phase concentration is the same in the pore spaces within the solid as it is outside the solid.

If the gas around the adsorbent solid occupies some fraction ϵ of a volume $\mathbf{V}$, and if this volume contains an adsorbing gas **i**, then the rate of adsorption of the gas onto the adsorbent and the rate of depletion of that species from the gas phase are coupled batch processes. The component mass balances for the gas and solid phases are as follows:

$$\text{Gas phase:} \quad \frac{dC_{i,g}\epsilon V}{dt} = -(1-\epsilon)\text{rate}_{i,\text{ads,net}} A_s \rho_s \text{V}$$

$$\text{Solid phase:} \quad \frac{dC_{i,s}(1-\epsilon)A_{\text{tot}}}{dt} = (1-\epsilon)\text{rate}_{i,\text{ads,net}} A_s \rho_s \text{V}$$

$C_{i,g} = \dfrac{\text{mole } i}{\text{volume}}$ gas phase

$\in =$ void fraction in the bed of solid and gas

$C_{i,s} = \dfrac{\text{mole } i}{\text{area}}$ surface phase

$V =$ total volume occupied by solid and gas

$A_{\text{tot}} = A_s \rho_s V = \text{Length}^2$

$\text{rate}_{i,\text{ads,net}} =$ net rate of adsorption of i on solid

$\text{As} = \dfrac{\text{cm}^2}{g}, \dfrac{\text{Length}^2}{\text{mass}}$

$\rho_s = \dfrac{g}{\text{cm}^3}, \dfrac{\text{mass}}{\text{Length}^3}$

When rearranged and written with the explicit rate of adsorption these become:

$$\text{Gas phase:} \quad \frac{dC_{i,g}}{dt} = -\frac{(1-\epsilon)}{\epsilon}(k_a C_{i,g}(C_{i,s,\text{tot}} - C_{i,s}) - k_{d,s} C_{i,s}) A_s \rho_s \text{V}$$

$$\text{Solid phase:} \quad \frac{dC_{i,s}}{dt} = (k_a, C_{i,g}(C_{i,s,\text{tot}} - C_{i,s}) - k_{d,s} C_{i,s})$$

There are two types of experiments suggested by these equations and that actually are done to obtain the rate constants for adsorption of a species **i** on a given adsorbent. The first experiment is done gravimetrically. The adsorbent is placed in small container suspended from a balance and inside an evacuable enclosure. After heating under dynamic vacuum to remove any water or other adsorbates, the sample is cooled to the experimental temperature, and then the adsorbate is admitted in such a way that its pressure remains constant throughout the course of the experiment. This is done either by constant delivery or by connection to a large ballast volume of the adsorbate gas. The mass uptake is measured as a function of time. The parameters **Cistot** and $\mathbf{\frac{As}{Vs}}$ are typically known from separate measurements. Thus only **ka** and **kd** need to be fitted to the data, either in differential or integral form. To fit to the integral form, we need the expression for mass adsorbed per time. Hence we need to integrate the mass balance for the adsorbate over time:

$$\frac{dC_{i,s}}{dt} = (k_a C_{i,g}(C_{i,s,\text{tot}} - C_{i,s}) - k_{d,s}C_{i,s})$$

$$\frac{d}{dt}C_{i,s}(1-\epsilon)V\rho_s A_s\,\text{Mwi} = (k_a C_{i,g}(C_{i,s,\text{tot}} - C_{i,s}) - k_{d,s}C_{i,s})(1-\epsilon)\rho_s A_s \text{V Mwi}$$

$$\frac{dm_{i,s}}{dt} = (k_a C_{i,g}(C_{i,s,\text{tot}}(1-\epsilon)\rho_s A_s V\,\text{Mwi} - m_{i,s}) - k_{d,s}m_{i,s})$$

$$\frac{dm_{i,s}}{dt} = k_a C_{i,g}(m_{i,s,\text{tot}} - m_{i,s}) - k_{d,s}m_{i,s}$$

$$\text{where}\quad m_{i,s,\text{tot}} = C_{i,s,\text{tot}}(1-\epsilon)\rho_s A_s V\,\text{Mwi}$$

This concentration of **i** in the gas phase is a constant, which makes this equation simple to integrate:

```
In[2]:=  Simplify[DSolve[
           {∂tmis[t] == kaCig (mistot - mis[t]) - kdmis[t], mis[0] == 0},
           mis[t], t]]
```

$$\text{Out[2]= } \{\{\text{mis[t]} \to -\frac{\text{Cig}(-1 + e^{-(\text{Cig ka} + \text{kd})\,t})\,\text{ka mistot}}{\text{Cig ka} + \text{kd}}\}\}$$

We can put some realistic numbers into this equation to see how it would behave. We can take ∈ to be 0.4, which is a reasonable number for a packed bed of particles. The area per unit volume can be taken as 100 m^2 per g (~10^6 cm^2 per g), the density of the solid is on the order of 1 g cm^{-3}, and the number of sites per unit area $\boldsymbol{N_{i,s,tot}}$ is on the order of 10^{14} per cm^2, making $\boldsymbol{C_{i,s,tot}} \sim 10^{-9}$ mole sites cm^2. (On a perfect surface there are ~10^{15} per atoms cm^2, so we have taken 10% of this value as the number of sites, which corresponds to one site in every 1 nm^2. Finally, the mass concentration of the adsorbate (if the latter is ideal) is $\frac{\text{Pi MW}i}{RT}$. We use these numbers and a value of **ka**, which is one order of magnitude larger than **kd**. If the system were to come to equilibrium, then the mass uptake would go to zero. This would be the same when rate of adsorption is balanced exactly by the rate of desorption. We can compute the mass of *i* on the solid when this occurs as follows:

$$\frac{dm_{i,s}}{dt} = 0 = k_a C_{i,g}(m_{i,s,\text{tot}} - m_{i,s}) - k_{d,s}m_{i,s}$$

```
In[3]:=  Clear[mistot, miseq, ka, kd, Cg]
         Solve[kaCg (mistot - miseq) - kdmiseq == 0, miseq]
```

$$\text{Out[4]= } \{\{\text{miseq} \to \frac{\text{Cg ka mistot}}{\text{Cg ka} + \text{kd}}\}\}$$

The full time-dependent solution comes from the solution of the material balance equation. Both solutions are presented here:

```
In[5]:= SetOptions[{Plot, ListPlot}, AxesStyle → {Thickness[0.01]},
          PlotStyle → {PointSize[0.015], Thickness[0.006]},
          DefaultFont → {"Helvetica", 17}];

In[6]:= "The expression on the right needs to be
           divided by grams to make it dimensionless for plotting";
```

$$\texttt{mis[t\_] := N[}-\frac{\texttt{Cg}(-1+e^{-(\texttt{Cg ka}+\texttt{kd})t})\texttt{ka mistot/g}}{\texttt{Cg ka}+\texttt{kd}}\texttt{]}$$

$$\texttt{ka} = 10^{2}\frac{\texttt{cm}^{3}}{\texttt{mole}};$$

```
kd = .001;
pi = 1 atm;
```

$$\texttt{Mwi} = 100\ \frac{\texttt{g}}{\texttt{mole}};$$

```
T = 300 K;
```

$$\texttt{R} = 82.05\ \frac{\texttt{cm}^{3}\texttt{atm}}{\texttt{mole K}};$$

$$\texttt{Cg} = \frac{\texttt{pi}}{\texttt{R T}};$$

$$\epsilon = 0.4;$$

$$\texttt{V} = 1\,\texttt{cm}^{3};$$

$$\rho\texttt{s} = 2.5\ \texttt{g cm}^{-3};$$

$$\texttt{As} = 10^{6}\texttt{cm}^{2}\texttt{g}^{-1};$$

$$\texttt{Vs} = 10\,\texttt{cm}^{3};$$

$$\texttt{Cistot} = 10^{14}\texttt{cm}^{-2}\frac{\texttt{1 mole}}{6.02\ 10^{23}};$$

```
tmax = 1000;
```

$$\texttt{mistot} = (1-\epsilon)\,\texttt{Cistot V As}\,\rho\texttt{s Mwi};$$

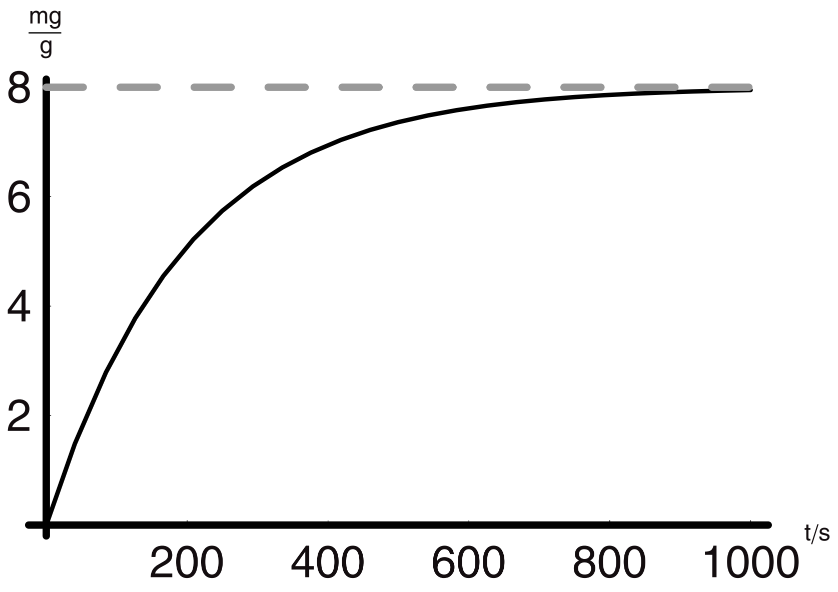
```
"In the following two expressions
     the unit of mass needs to be eliminated for plotting";
```

$$\texttt{miseq} = \frac{\texttt{Cg ka mistot}}{(\texttt{Cg ka}+\texttt{kd})\texttt{g}}\frac{1000}{\rho\texttt{s V/g}};$$

$$\texttt{Plot[}\frac{\texttt{mis[t]1000}}{\rho\texttt{s V/g}}\texttt{, \{t, 0, tmax\},}$$

$$\texttt{AxesLabel} \to \texttt{\{"t/s","}\frac{\texttt{mg}}{\texttt{g}}\texttt{"\},}$$

```
 Epilog → {Thickness[0.01], Dashing[{0.05, 0.05}],
   GrayLevel[0.6], Line[{{0, miseq}, {tmax, miseq}}]}
];
```



In this case, the equilibrium is reached at a modest level of 8 mg of adsorbate per gram of adsorbent, which is a low level of adsorption. Higher values would be on the order of 80 mg per gram.

A different experiment that appears to be simple is to expose the adsorbent to a volume of gas and then measure the pressure change as a function of time. This has the same aim as the procedure we just analyzed but it is much more complex. A brief analysis will show us why. Let $\boldsymbol{V_o}$ be the volume that is occupied by the gas at a known pressure and temperature. Once the two volumes are connected, the total volume of the system is $\mathbf{Vtot} = \boldsymbol{\epsilon}\mathbf{V} + \boldsymbol{V_o}$ on the basis of a solid that was space occupying but not adsorbing. The initial pressure $\boldsymbol{P_1}$, after opening a valve between the two, is given by:

$$P_o V_o = P_1(\epsilon \mathrm{V} + V_o) \Longrightarrow P_1 = \frac{P_o V_o}{(\epsilon V + V_o)}$$

Is this the correct initial pressure to use? Or should we account for the internal void of the adsorbent as well when we compute the initial pressure. To do so would lead to one more term in volume, namely, that of the void fraction within the solid. This is not the void between the solid particles, but that which is within the solid particles. If the mass of the particles is ms and their density is $\boldsymbol{\rho s}$, then the volume of the particles is $\mathbf{Vp} = \frac{\mathbf{ms}}{\boldsymbol{\rho}\mathbf{s}}$, and if the fraction that is unoccupied by solid is ξ, then this extra volume is $\boldsymbol{\xi}\,\mathbf{Vp} = \boldsymbol{\xi}\frac{\mathbf{ms}}{\boldsymbol{\rho}\mathbf{s}}$. The corrected initial pressure would be:

$$P_o V_o = P_1(\epsilon \mathrm{V} + V_o + \xi \mathrm{Vp}) \Longrightarrow P_1 = \frac{P_o V_o}{(\epsilon V + V_o + \xi \mathrm{Vp})}$$

Assuming that we can compute a reasonable initial pressure, then the pressure is measured as a function of time and the data would be fitted either to the differential or integral expressions from these equations:

$$\begin{aligned}
\frac{dC_{i,g}}{dt} &= -\frac{(1-\epsilon)}{\epsilon}(k_a C_{i,g}(C_{i,s,\text{tot}} - C_{i,s}) - k_{d,s}C_{i,s})A_s\rho_s V \\
\frac{dP_i}{dt} &= -\frac{(1-\epsilon)}{\epsilon}(k_a P_i(C_{i,s,\text{tot}} - C_{i,s}) - k_{d,s}\text{RT}\,C_{i,s})A_s\rho_s \text{V} \\
\text{RT}\,\frac{dC_{i,s}}{dt} &= k_a P_i\,(C_{i,s,\text{tot}} - C_{i,s}) - k_{d,s}\,\text{RT}\,C_{i,s}
\end{aligned}$$

These equations appear to be very similar to those we have just seen, and hence they seem to be simple. In fact they are not simple because the pressure of the gas is a function of time as is the concentration on the surface. The previous experiment has the advantage of being designed around an analysis that was simple to carry out and solve for an analytical expression. We can solve these two equations using *Mathematica*, but the closed-form solutions are anything but straightforward. To see this run the **DSolve** code:

```
In[26]:= Clear[ka, kd, p, c, Cstot, As, ρs, V]

         Simplify[DSolve[
          {∂tp[t] == -(kap[t](Cstot - c[t]) - kdc[t])AsρsV,
           ∂tc[t] == (kap[t](Cstot - c[t]) - kdc[t]),
           p[0] == P1,
           c[0] == 0},
          {p[t], c[t]}, t]]

         General::spell1 : Possible spelling error: new symbol name
          "Cstot" is similar to existing symbol "Cistot".

         Solve::ifun : Inverse functions are being used by Solve,
          so some solutions may not be found.
```

Out[27]=

$$\{c[t] \to \frac{1}{(2\,As\,ka\,V\,\rho s} \\ (kd + ka(P1 + As\,Cstot\,V\rho s) \\ + \sqrt{-kd^2 - ka^2(P1 - As\,Cstot\,V\rho s)^2 - 2ka\,kd(P1 + As\,Cstot\,V\rho s)} \\ Tan[\frac{1}{2}t\sqrt{-kd^2 - ka^2(P1 - As\,Cstot\,V\rho s)^2 - 2ka\,kd(P1 + As\,Cstot\,V\rho s)} \\ - ArcTan[\frac{\sqrt{Cstot}\sqrt{P1}\sqrt{\frac{(kd + ka\,P1 + As\,Cstot\,ka\,V\rho s)^2}{As\,Cstot\,ka^2\,P1\,V\rho s}}}{\sqrt{-\frac{kd^2 + ka^2(P1 - As\,Cstot\,V\rho s)^2 + 2ka\,kd(P1 + As\,Cstot\,V\rho s)}{As\,ka^2\,V\rho s}}}]]),$$

$$\mathrm{c[t]} \rightarrow \frac{1}{(2\,\mathrm{As\,ka\,V}\,\rho\mathrm{s}}$$

$$(\mathrm{kd} + \mathrm{ka}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})$$

$$+\sqrt{-\mathrm{kd}^2 - \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})^2 - 2\mathrm{ka\,kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})}$$

$$\mathrm{Tan}[$$

$$\frac{1}{2}\mathrm{t}\sqrt{-\mathrm{kd}^2 - \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})^2 - 2\mathrm{ka\,kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})}$$

$$+\mathrm{ArcTan}[\frac{\sqrt{\mathrm{Cstot}}\sqrt{\mathrm{P1}}\sqrt{\frac{(\mathrm{kd} + \mathrm{ka\ P1} + \mathrm{As\,Cstot\,ka\,V}\,\rho\mathrm{s})^2}{\mathrm{As\,Cstot\,ka}^2\,\mathrm{P1V}\,\rho\mathrm{s}}}}{\sqrt{-\frac{\mathrm{kd}^2 + \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})^2 + 2\mathrm{ka\,kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})}{\mathrm{As\,ka}^2\mathrm{V}\,\rho\mathrm{s}}}}]]),$$

$$\mathrm{p[t]} \rightarrow -\frac{1}{2\mathrm{ka}}$$

$$(\mathrm{kd} - \mathrm{ka\,P1} + \mathrm{As\,Cstot Ka\,V}\,\rho\mathrm{s}$$

$$+\sqrt{-\mathrm{kd}^2 - \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})^2 - 2\mathrm{ka\,kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})}$$

$$\mathrm{Tan}[$$

$$\frac{1}{2}\mathrm{t}\sqrt{-\mathrm{kd}^2 - \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})^2 - 2\mathrm{ka}\ \ \mathrm{kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\ \mathrm{s})}$$

$$-\mathrm{ArcTan}[\frac{\sqrt{\mathrm{Cstot}}\sqrt{\mathrm{P1}}\sqrt{\frac{(\mathrm{kd} + \mathrm{ka\ P1} + \mathrm{As\,Cstot\,ka\,V}\,\rho\mathrm{s})^2}{\mathrm{As\,Cstot\,ka}^2\,\mathrm{P1V}\,\rho\mathrm{s}}}}{\sqrt{-\frac{\mathrm{kd}^2 + \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\,\mathrm{s})^2 + 2\mathrm{ka\,kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})}{\mathrm{As\,ka}^2\,\mathrm{V}\rho\mathrm{s}}}}]]),$$

$$\mathrm{p[t]} \rightarrow -\frac{1}{2\mathrm{ka}}$$

$$(\mathrm{kd} - \mathrm{ka\,P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})$$

$$+\sqrt{-\mathrm{kd}^2 - \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})^2 - 2\mathrm{ka\,kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})}$$

$$\mathrm{Tan}[$$

$$\frac{1}{2}\mathrm{t}\sqrt{-\mathrm{kd}^2 - \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})^2 - 2\mathrm{ka\,kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})}$$

$$+\mathrm{ArcTan}[\frac{\sqrt{\mathrm{Cstot}}\sqrt{\mathrm{P1}}\sqrt{\frac{(\mathrm{kd} + \mathrm{ka\ P1} + \mathrm{As\,Cstot\,ka\,V}\,\rho\mathrm{s})^2}{\mathrm{As\,Cstot\,ka}^2\,\mathrm{P1V}\,\rho\mathrm{s}}}}{\sqrt{-\frac{\mathrm{kd}^2 + \mathrm{ka}^2(\mathrm{P1} - \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})^2 + 2\mathrm{ka\,kd}(\mathrm{P1} + \mathrm{As\,Cstot\,V}\,\rho\mathrm{s})}{\mathrm{As\,ka}^2\,\mathrm{V}\,\rho\mathrm{s}}}}]])\}$$

As we can see from these solutions this is anything but a simple experiment. This is a good illustration of how the analysis can be used to define and indeed to design the experiment.

Semicontinuous Adsorption: Pseudo-Steady State

From the experimental point of view there is one more experiment that can be done to obtain adsorption rate parameters and this is the use of a semicontinuous approach. Here the adsorbate is fed at a mass flow rate that is equal to the rate of adsorption, and a very sensitive pressure transducer is slaved to a mass flow controller for the adsorbate. As this gas is adsorbed, and were there no flow into the system, the pressure would drop. This would lead to the complexities we have just analyzed. If, however, the mass flow controller is slaved in such a way that it opens whenever there is a slight $\delta\mathbf{P}$ of pressure drop below the fixed experimental pressure set point, then the pressure can be maintained as a constant. The mass flow rate into the system is the same as the mass rate of adsorption "out" of the gas phase and "onto" the adsorbent phase II.

The analysis of this experiment begins with a slightly modified version of the equations we have seen:

$$\text{Gas phase:}\quad \frac{dC_{i,g}\,\epsilon\,V}{dt} = C_{i,g}q - (1-\epsilon)\text{rate}_{i,\text{ads,net}}A_s\rho_s\text{V}$$

$$\text{Solid phase:}\quad \frac{dC_{i,s}(1-\epsilon)A_{\text{tot}}}{dt} = (1-\epsilon)\text{rate}_{i,\text{ads,net}}A_s\rho_s\text{V}$$

The gas phase balance includes a convective flow term for the mass flow of species **i** into the system. The pressure would rise were it not for the rate of adsorption, that is, the process that removes **i** from the gas phase and locates it in the second phase, the adsorbent. Now we can make progress in the analysis even before we substitute in the rate expression. The reason is this: in the experiment the rate of adsorption must be equal to the rate of delivery. Therefore we have a *pseudo-steady state* in that the gas phase concentration remains constant all the while the surface concentration is changing:

$$C_{i,g}\text{q} = (1-\epsilon)\ \text{rate}_{i,\text{ads,net}}A_s\rho_s\text{V}$$

$$\therefore \frac{dC_{i,s}(1-\epsilon)A_{\text{tot}}}{dt} = C_{i,g}\text{q}$$

$$\frac{dC_{i,s}}{dt} = \frac{C_{i,g}q}{(1-\epsilon)A_s\rho_s V}$$

Given that the gas phase concentration is constant, this is immediately integrated to a linear form in time:

$$C_{i,s}[\text{t}] = \frac{C_{i,g}q}{(1-\epsilon)A_s\rho_s V}\text{t} = \text{CS t}$$

This can be substituted back into the rate expression to give:

$$
\begin{aligned}
\frac{dC_{i,s}}{dt} &= (k_a C_{i,g}(C_{i,s,\text{tot}} - C_{i,s}) - k_{d,s}C_{i,s}) \\
&= \left(k_a C_{i,g}\left(C_{i,s,\text{tot}} - \frac{C_{i,g}q}{(1-\epsilon)A_s\rho_s V}\,t\right) - k_{d,s}\frac{C_{i,g}q}{(1-\epsilon)A_s\rho_s V}t\right) \\
&= (k_a C_{i,g}(C_{i,s,\text{tot}} - \text{CS}\,t) - k_{d,s}\text{CS}\,t) \\
&= k_a C_{i,g}C_{i,s,\text{tot}} - k_a C_{i,g}\text{CS}\,t - k_{d,s}\text{CS}\,t \\
&= k_a C_{i,g}C_{i,s,\text{tot}} - (k_a C_{i,g} + k_{d,s})\text{CS}\,t
\end{aligned}
$$

```
In[28]:= Clear[Cis, ka, kd, Cig, CS, Cistot, q, As, ρs, ϵ,V]

         Simplify[DSolve[
          {Cis'[t] == kaCigCistot - (kaCig + kd)CSt, Cis[0] == 0},
          Cis[t], t]]
                         1
Out[29]= {{Cis[t] → -─t(-2CigCistotka + CigCSkat + CSkdt)}}
                         2
                           1
In[30]:= Simplify[Solve[-─t(-2CigCistotka + CigCSkat + CSkdt)
                           2
          == CSt, {ka, kd}]]

        Solve::svars : Equations may not give solutions for all
         "solve" variables.

                       2CS + CSkdt
Out[30]= {{ka → ─────────────────────}}
                 2CigCistot - CigCSt
```

To solve for **ka** and **kd** explicitly, we need one more equation. We can get this from the consideration of an equilibrium condition. When the concentration on the surface of the adsorbent is no longer changing, then rates of adsorption and desorption are equal. From this we find:

$$0 = (k_a C_{i,\text{ge}}(C_{i,s,\text{tot}} - C_{i,\text{se}}) - k_{d,s}C_{i,\text{se}})$$

This can be rearranged to give the ratio of the rate constants on the left-hand side:

$$\frac{\text{ka}}{\text{kd}} = \frac{\text{Cise}}{\text{Cige(Cistot} - \text{Cise)}} = \text{Kads}$$

Dividing through by the total number of sites we get the fraction of sites occupied, which is $\boldsymbol{\theta}$, and the well-known Langmuir isotherm:

$$\text{Kads} = \frac{(\text{Cise}/\text{Cistot})}{\text{Cige}\,(\text{Cistot} - \text{Cise})/\text{Cistot}}$$
$$\text{Kads} = \frac{\theta}{\text{Cige}\,(1-\theta)}$$

```
In[31]:= Solve[Kads == θ/(Cige(1 - θ)), θ]

         General::spell1: Possible spelling error: new symbol name
          "Cige" is similar to existing symbol "Cig".

Out[31]= {{θ → Cige Kads/(1 + Cige Kads)}}
```

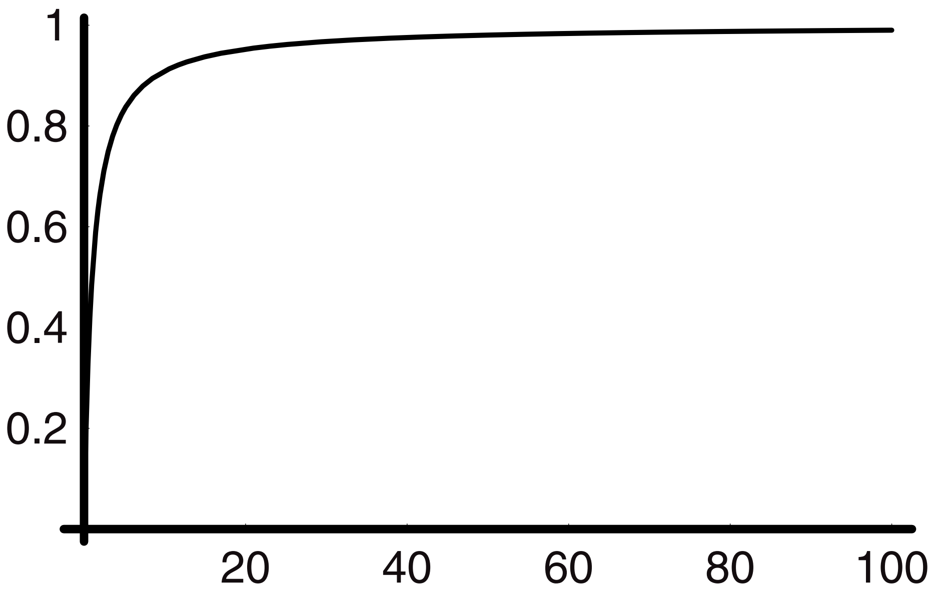
Thus from this one measurement we can find the ratio of the rate constants, **Kads**, and if we have some independent measure of the total number of sites, **Cistot**, then we can compute the rate constants:

$$\text{ka} = \frac{2\text{CS} + \text{CS}\,\text{kd}\,\text{t}}{2\text{Cig}\,\text{Cistot} - \text{Cig}\,\text{CS}\,\text{t}} = \frac{2\text{CS} + \text{CS}\,\frac{\text{ka}}{\text{Kads}}\text{t}}{2\text{Cig}\,\text{Cistot} - \text{Cig}\,\text{CS}\,\text{t}}$$

```
In[32]:= Simplify [Solve [ka == (2CS + CS (ka/Kads) t)/(2Cig Cistot - Cig CS t), ka]]

Out[32]= {{ka → -2 CS Kads/(-2Cig Cistot Kads + CS t + Cig CS Kads t)}}
```

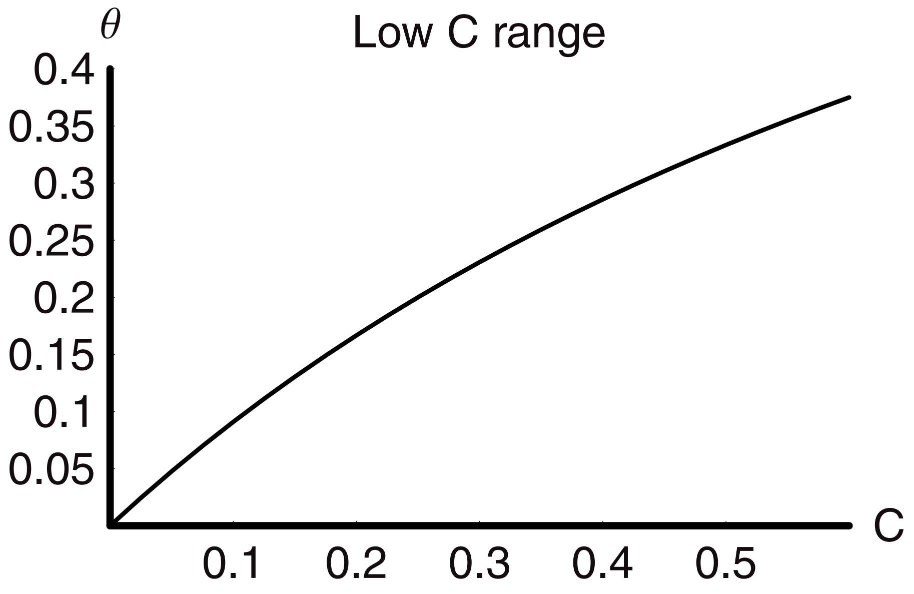
We should note that the fraction of sites occupied at any equilibrium gas phase concentration follows a graph that looks as follows:

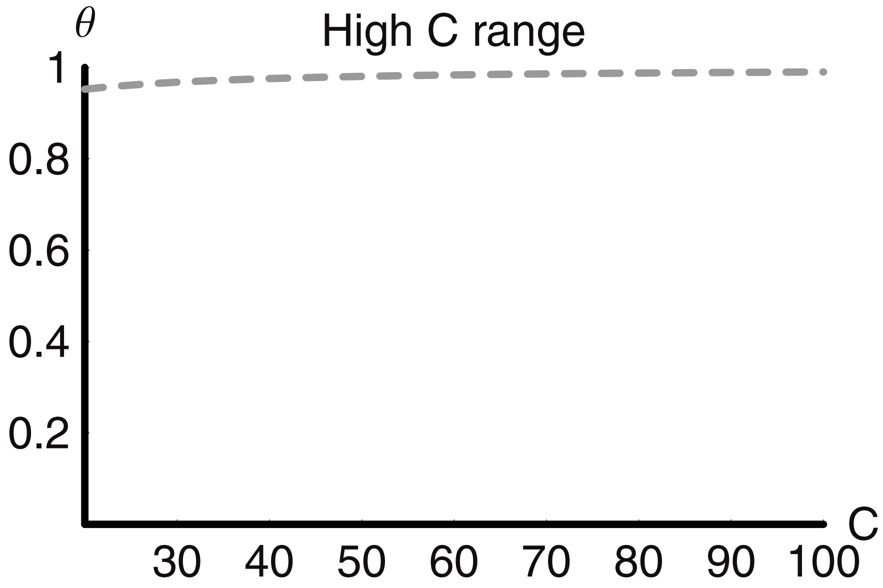
```
In[33]:= K = 1;
         Plot[KC/(1 + KC), {C, 0, 100}];
```



The expression reduces to a constant when the concentration is large relative to **K**, at about 20×, but at low concentration the expression is linear in **C**:

```
In[35]:= Show[
           GraphicsArray[
            Plot[KC/(1 + KC), {C, 0, .6},
             PlotRange → {{0, .6}, {0, .4}},
             AxesLabel → {"C", "θ"},
                 PlotLabel → "Low C range"],
           Plot[KC/(1 + KC), {C, 20, 100},
            PlotRange → {{20, 100}, {0, 1}},
            AxesLabel → {"C", "θ"},
                PlotStyle → {Thickness[0.01],
                 Dashing[{0.025, 0.025}], GrayLevel[0.6]},
                PlotLabel → "High C range"]
              ]
             ];
```





In the low concentration limit the adsorption isotherm is a linear law as was the partition coefficient, and just as the isotherm deviates from linearity outside of the low concentration limit, so too does the partition relation between the two liquid phases.

There is much more that we can say about adsorption and what we can do with it, including the coupling together of mass transfer and adsorption. There is no better example of that kind of process than that which occurs with membranes, and this is called permeation.

6.2 Permeation

Permeation is a process by which mass is transferred through a membrane from a region of higher concentration or pressure to one of lower concentration or pressure. The membrane can be polymeric, metallic or ceramic. Our bodies and all living organisms use membranes as critical structural components of cells. Separating the inside from the outside of the cell provides it with its integrity and specialization. Only certain molecules and ions are allowed to move across the membranes, thus rendering them highly selective. Synthetic membranes seek to emulate this but with much simpler structures and with mechanisms of operation for much less complex separations. Polymeric membranes can be obtained that will separate molecules on the basis of their relative affinities for the interior of the membrane. Those molecules with higher affinities partition themselves to a larger degree in the membrane versus the bulk phases than do their competitors. With higher concentrations within, they also transport across the membranes faster. A classic example of this is the membrane that is used for hemodialysis. Rendered incapable of clearing the blood of toxins, patients with renal dysfunction can be "dialyzed" by passing their blood continuously through the membrane unit. The polymers making up the membranes transport these toxins to a dialysate solution in which they are very soluble, and thereby return the blood in refreshed state to the patient.

Ceramic and metallic membranes hold the promise of conducting small molecule separations continuously and with much less energy than required by other processes. The ceramic membranes offer the opportunity to operate at elevated temperatures (even as part of a chemical reactor, which can offer enhanced conversions and yields of products) by transporting one product away from the reaction zone, selectively and continuously in order to bypass the equilibrium limitations. Metallic membranes of palladium and its alloys are special in that they transport hydrogen and only hydrogen. This makes them particularly interesting for hydrogen purification, recovery, and use. They may also play a role in fuel cells. Before we can begin to work with membranes we must know how to analyze their behavior, which is the goal of this section of the chapter.

6.3 Permeation—Adsorption and Diffusion

Batch. Permeation involves the transport of molecules across a membrane phase. The transport process involves either dissolution or adsorption within the substance of the membrane and then transport from regions of higher to lower "potential" (that is, concentration) within the membrane phase. The global measurement of the rate of transport across the membrane, given in terms of the measurable changes in the concentrations in the bulk above and below the membrane, is permeation. Transport within the membrane, described quantitatively in terms of the concentration within it, is diffusion. The processes that take gas phase species from the bulk either to the surface of the membrane or that lead to their dissolution within the near surface region are adsorption and partitioning (dissolution), respectively.

The rate of transport across the membrane in units of mass (or moles) per unit time per unit area is termed a flux **J** and it is found to be proportional to the difference between the concentrations on either side of the membrane. The proportionality constant is called the permeability $\boldsymbol{P_m}$ with intrinsic dimensions of $\frac{\textbf{Length}}{\textbf{time}}$, the same as the mass transfer coefficient and the same as velocity.

$$\mathrm{J} = P_m(C^I - C^{\mathrm{II}})$$

Higher permeabilities make for higher fluxes as do higher concentrations and pressures. The high concentration side of the membrane from which mass typically flows is termed the retentate, while that side to which mass flows is the permeate.

At relatively low pressures and concentrations, the permeability is the product of two terms—the adsorption constant or partition coefficient and the diffusivity:

$$P_m = \mathrm{KD} \quad (\mathrm{K} = \mathrm{Kd\ \ or\ \ Kads...})$$

We will see how this factors into the analysis as we go through this material.

Consider the following diagram, Figure 1, for a simple system for batch permeation.

The concentrations of **B** and **D** are given as $\boldsymbol{C_B^I}$ and $\boldsymbol{C_D^I}$ on the retentate side and as $\boldsymbol{C_B^{II}}$ and $\boldsymbol{C_D^{II}}$ on the permeate side of the membrane. The permeation process is considered to take place at fixed temperature. The membrane has an area **Am** through which the flux is measured.

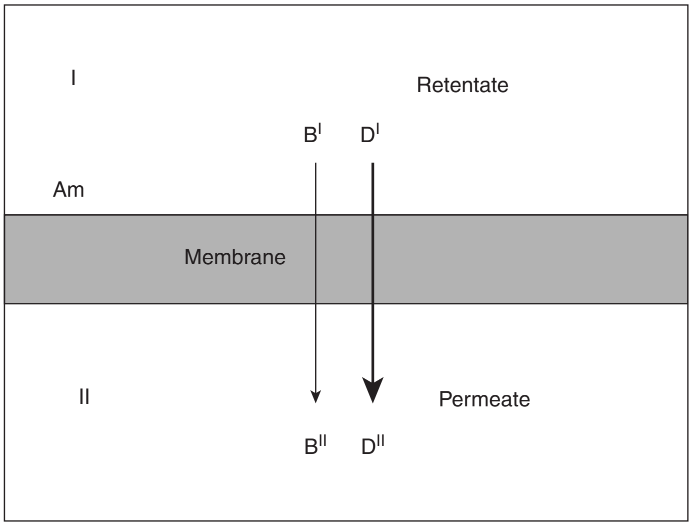


Figure 1

The volumes of the two compartments are V^I and V^{II}. Each molecule will have its own permeability and we will assume that they permeate independently of one another. The material balance equations are:

$$\frac{dC_B^I V^I}{dt} = -P_{m,B}\,\mathrm{Am}\left(C_B^I - C_B^{\mathrm{II}}\right)$$
$$\frac{dC_D^I V^I}{dt} = -P_{m,D}\,\mathrm{Am}\left(C_D^I - C_D^{\mathrm{II}}\right)$$
$$\frac{dC_B^{\mathrm{II}} V^{\mathrm{II}}}{dt} = P_{m,B}\,\mathrm{Am}\left(C_B^I - C_B^{\mathrm{II}}\right)$$
$$\frac{dC_D^{\mathrm{II}} V^{\mathrm{II}}}{dt} = P_{m,D}\,\mathrm{Am}\left(C_D^I - C_D^{\mathrm{II}}\right)$$

There are four equations that describe the system. We see that independent of which side of the membrane is the higher concentration side, these equations still work, as they must if they are to be valid. The concentrations could be in mass per volume or in moles per volume (we will assume the latter). If **B** and **D** were ideal gases, we could express these equations in terms of the pressure. In all four equations the right-hand side is just the flux of the component times the area of the membrane. The volumes of the compartments are constant; thus they can be brought to the right-hand sides:

$$\frac{dC_B^I}{dt} = -\frac{P_{m,B}\,\mathrm{Am}\left(C_B^I - C_B^{\mathrm{II}}\right)}{V^I}$$
$$\frac{dC_D^I}{dt} = -\frac{P_{m,D}\,\mathrm{Am}\left(C_D^I - C_D^{\mathrm{II}}\right)}{V^I}$$
$$\frac{dC_B^{\mathrm{II}}}{dt} = \frac{P_{m,B}\,\mathrm{Am}\left(C_B^I - C_B^{\mathrm{II}}\right)}{V^{\mathrm{II}}}$$
$$\frac{dC_D^{\mathrm{II}}}{dt} = \frac{P_{m,D}\,\mathrm{Am}\left(C_D^I - C_D^{\mathrm{II}}\right)}{V^{\mathrm{II}}}$$

It is interesting that yet again this simple operation provides a useful time constant. This is the ratio of $\frac{\mathbf{Pm\ Am}}{V}$, which has dimensions of reciprocal time. Therefore, the reciprocal of this group is a time, $\frac{V}{\mathbf{Am}}$ is a characteristic length, and $\frac{\mathbf{1}}{\mathbf{Pm}}$ is a time per length; thus their product is a time.

We can solve these equations analytically for the case in which the permeate side is initially evacuated and the retentate side is charged with initial concentrations of **B** and **D**. Also, to simplify the result, we can take the volumes to be equal:

```
In[36]:= Clear["Global`*"]

In[37]:= VI = VII;
```

```
memsol =
 Flatten[
   Simplify[
    DSolve[
     {∂t C1B[t] == -PmBA/VI (C1B[t] - C2B[t]),
      ∂t C1D[t] == -PmDA/VI (C1D[t] - C2D[t]),
      ∂t C2B[t] == +PmBAm/VII (C1B[t] - C2B[t]),
      ∂t C2D[t] == +PmDAm/VII (C1D[t] - C2D[t]),
     C1B[0] == C1Bo, C1D[0] == C1Do,
     C2B[0] == 0, C2D[0] == 0},

    {C1B[t], C2B[t], C1D[t], C2D[t]},
    t]
          ]
         ]
   CIB[t_] := Evaluate[C1B[t] /. memsol[[1]]]
   CIIB[t_] := Evaluate[C2B[t] /. memsol[[2]]]
   CID[t_] := Evaluate[C1D[t] /. memsol[[3]]]
   CIID[t_] := Evaluate[C2D[t] /. memsol[[4]]]
```

```
General::spell1 : Possible spelling error: new symbol
 name "C1Bo" is similar to existing symbol "C1B".

General::spell : Possible spelling error: new symbol
 name "C1Do" is similar to existing symbols {C1Bo, C1D}.
```

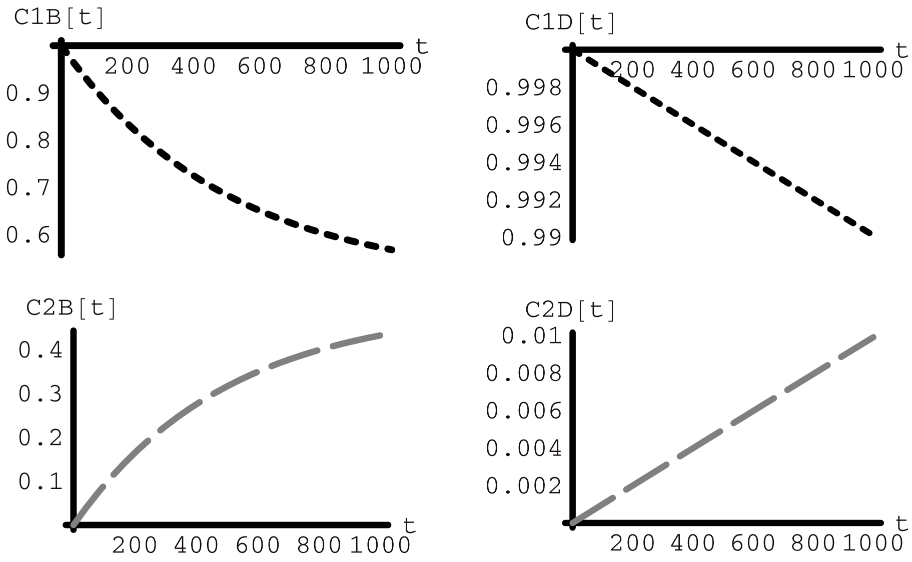
Out[38]= $\{C1B[t] \to \frac{1}{2}C1Bo(1+e^{-\frac{2\,Am\,PmB\,t}{VII}}),\ C2B[t] \to \frac{1}{2}C1Bo(1-e^{-\frac{2\,Am\,PmB\,t}{VII}}),$
$C1D[t] \to \frac{1}{2}C1Do(1+e^{-\frac{2\,Am\,PmD\,t}{VII}}),\ C2D[t] \to \frac{1}{2}C1Do(1-e^{-\frac{2\,Am\,PmD\,t}{VII}})\}$

```
General::spell1 : Possible spelling error: new symbol
 name "CIIB" is similar to existing symbol "CIB".

General::spell : Possible spelling error: new symbol
 name "CIID" is similar to existing symbols {CID, CIIB}.
```

```
In[43]:= C1Bo = 1;
         C1Do = 1;
         PmB = 10^-4;
```

```
PmD = 10^-6;
VII = 1;
Am = 10;

SetOptions[Plot, DefaultFont → {"Hevetica", 10},
 AxesStyle → Thickness[.02]];

                              VI
plI = Plot[CIB[t], {t, 0, ──────},
                            PmBAm
  DisplayFunction → Identity,
  AxesLabel → {"t", "C1B[t]"}, PlotRange → All,
  PlotStyle → {Thickness[0.02],
   Dashing[{0.025, 0.035}]}];

                               VI
plII = Plot[CID[t], {t, 0, ──────},
                             PmBAm
   DisplayFunction → Identity,
   AxesLabel → {"t", "C1D[t]"}, PlotRange → All,
   PlotStyle → {{Thickness[0.02],
    Dashing[{0.025, 0.035}]}}];

                                 VI
plIII = Plot[CIIB[t], {t, 0, ──────},
                               PmBAm
   DisplayFunction → Identity,
   AxesLabel → {"t", "C2B[t]"},
   PlotRange → All,
   PlotStyle → {Thickness[0.02], GrayLevel[0.5],
    Dashing[{{0.15, 0.05}]}];

                                VI
plIV = Plot[CIID[t], {t, 0, ──────},
                              PmBAm
    DisplayFunction → Identity,
    AxesLabel → {"t", "C2D[t]"},
    PlotRange → All,
    PlotStyle → {Thickness[0.02], GrayLevel[0.5],
     Dashing[{0.15, 0.05}]}];

Show[GraphicsArray[{{plI, plII}, {plIII, plIV}}]];
```

```
General::spell1 : Possible spelling error: new symbol name
 "plII" is similar to existing symbol "plI".

General::spell1 : Possible spelling error: new symbol name
 "plIII" is similar to existing symbol "plII".

General::spell : Possible spelling error: new symbol name
 "plIV" is similar to existing symbols {plI, plII}.
```



We see from the preceding graph that over the period of time $\frac{\mathbf{VI}}{\mathbf{PmB\ Am}}$ the concentration of **B** has fallen sharply on the retentate side and has risen as sharply on the permeate side. The two sides are almost at equilibrium with respect to species **B**, with each cell going to 0.5 concentration units. The other species **D** has barely begun to transfer across the membrane. It takes on the order of 10^2 times longer to get to equilibrium, which it nearly reaches in $\frac{\mathbf{VI}}{\mathbf{PmD\ Am}}$ time units. Were this achievable, the selectivity would be nearly perfect as there is so little **D** on the permeate side compared to **B**. It would be very nice to try this continuously to see how well the systems would work.

Continuous Permeation. The continuous process must have feed and exit on the retentate side and at least exit flow on the permeate side. We could have an additional sweep (gas or liquid) feed on the permeate side, which adds very little to the analysis. The new physical situation is as shown in Figure 2:

The material balance equations for components **B** and **D** on the retentate and permeate sides become:

$$\frac{dC_B^I V^I}{dt} = \left(C_{\mathrm{Bf}}^I - C_B^I\right)q^I - P_{m,B}\,\mathrm{Am}\left(C_B^I - C_B^{\mathrm{II}}\right)$$
$$\frac{dC_D^I V^I}{dt} = \left(C_{\mathrm{Df}}^I - C_D^I\right)q^I - P_{m,D}\,\mathrm{Am}\left(C_D^I - C_D^{\mathrm{II}}\right)$$
$$\frac{dC_B^{\mathrm{II}} V^{\mathrm{II}}}{dt} = -C_B^{\mathrm{II}} q^{\mathrm{II}} + P_{m,B}\,\mathrm{Am}\left(C_B^I - C_B^{\mathrm{II}}\right)$$
$$\frac{dC_D^{\mathrm{II}} V^{\mathrm{II}}}{dt} = -C_D^{\mathrm{II}} q^{\mathrm{II}} + P_{m,D}\,\mathrm{Am}\left(C_D^I - C_D^{\mathrm{II}}\right)$$

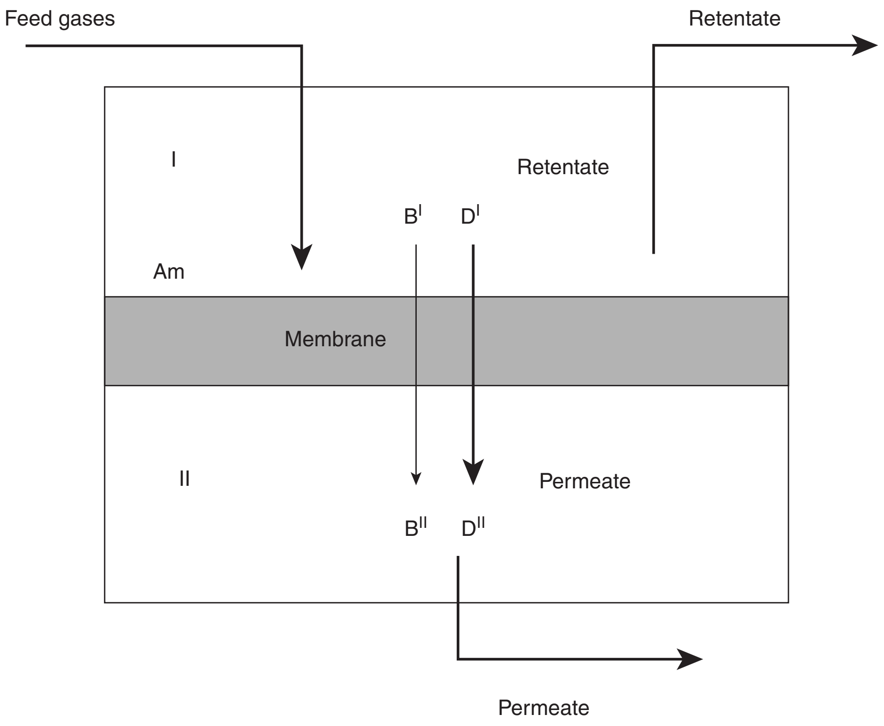


Figure 2

When the system runs at steady state, the derivatives are identically zero. We can divide both sides of both of the equations by the volumes to give:

$$0 = (C_{\mathrm{Bf}}^I - C_B^I)\frac{q^I}{V^I} - \frac{P_{m,B}\,\mathrm{Am}}{V^I}(C_B^I - C_B^{\mathrm{II}})$$
$$0 = (C_{\mathrm{Df}}^I - C_D^I)\frac{q^I}{V^I} - \frac{P_{m,D}\,\mathrm{Am}}{V^I}(C_D^I - C_D^{\mathrm{II}})$$
$$0 = -C_B^{\mathrm{II}}\frac{q^{\mathrm{II}}}{V^{\mathrm{II}}} + \frac{P_{m,B}\,\mathrm{Am}}{V^{\mathrm{II}}}(C_B^I - C_B^{\mathrm{II}})$$
$$0 = -C_D^{\mathrm{II}}\frac{q^{\mathrm{II}}}{V^{\mathrm{II}}} + \frac{P_{m,D}\,\mathrm{Am}}{V^{\mathrm{II}}}(C_D^I - C_D^{\mathrm{II}})$$

There are now two characteristic times in the equations: the first is the holding time $(\frac{q}{V})^{-1}$ and the second is a permeation time $(\frac{P_m\mathbf{Am}}{V})^{-1}$. We have four equations, two inlet concentrations, four outlet concentrations, two flow rates, two volumes, two permeabilities, and one area for a total of 13 variables and parameters. If we know the two inlet concentrations of **B** and **D**, their two permeabilities, the two volumes and the area of the membrane, and the retentate flow rate, then we have eight of these in hand. Hence, there are four to be calculated if we measure one. Let us say we measure the permeate flow rate. Then we should be able to compute the four exit concentrations:

```
In[55]:= Clear["Global`*"]

         VI = VII;
         PmB = 10^-4;
         PmD = 10^-6;
         VII = 1;
         r = 5;
         Am = 10^r;
         CBIf = .1;
         CDIf = 0.1;
         qI = 10;
         qII = qI;

      General::spell1 : Possible spelling error: new symbol name
       "CDIf" is similar to existing symbol "CBIf".

In[66]:= solmem = Flatten[
            Simplify[
             Solve[
              {0 == (CBIf - CBI) qI/VI - PmB Am/VI (CBI - CBII),
               0 == (CDIf - CDI) qI/VI - PmD Am/VI (CDI - CDII),
               0 == -CBII qII/VII + PmB Am/VII (CBI - CBII),
               0 == -CDII qII/VII + PmD Am/VII (CDI - CDII)},
              {CBI, CDI, CBII, CDII}]
                   ]
                  ];

       CB1ss = solmem[[1, 2]];
       CD1ss = solmem[[2, 2]];
       CB2ss = solmem[[3, 2]];
       CD2ss = solmem[[4, 2]];
       (CB2ss/CB1ss)/(CD2ss/CD1ss);
       CB1ss/CBIf;
```

$$\mathbf{\frac{CD1ss}{CDIf};}$$

$$\mathbf{\frac{CB2ss}{CBIf}\frac{qII}{qI}}$$

$$\mathbf{CB2ss\ qII}$$

```
General::spell1 : Possible spelling error: new symbol name
 "CBII" is similar to existing symbol "CBI".

General::spell : Possible spelling error: new symbol name
 "CDII" is similar to existing symbols {CBII, CDI}.

General::spell1 : Possible spelling error: new symbol name
 "CD1ss" is similar to existing symbol "CB1ss".

General::spell1 : Possible spelling error: new symbol name
 "CD2ss" is similar to existing symbol "CB2ss".

Out[74]= 0.333333

Out[75]= 0.333333
```

We might justifiably question how long it would take such a system to reach a steady state. To determine this we can solve the same set of equations that we have just examined at the steady state only now in the full time domain.

$$\textit{In[76]:=}\ \mathbf{Solve[\{0 == -CBII[t]\frac{qIIb}{VI} + \frac{PmBAm}{VI}(CBI[t] - CBII[t]),}$$
$$\mathbf{0 == -CDII[t]\frac{qIId}{VI} + \frac{PmDAm}{VI}(CDI[t] - CDII[t])\},}$$
$$\mathbf{\{qIIb, qIId\}]}$$

```
General::spell1 : Possible spelling error: new symbol name
 "qIIb" is similar to existing symbol "qII".

General::spell : Possible spelling error: new symbol name
 "qIId" is similar to existing symbols {qII, qIIb}.
```

$$\textit{Out[76]=}\ \{\{qIIb \rightarrow \frac{10(CBI[t] - CBII[t])}{CBII[t]},\ qIId \rightarrow -\frac{-CDI[t]\ +\ CDII[t}{10CDII[t]}\}\}$$

Next, we input the parameter values:

```
In[77]:= CIBo = 0.000001;
         CIDo = 0.000001;
         VI = 1;
```

```
PmB = 10^-4;
PmD = 10^-6;
VII = VI;
r = 5;
Am = 10^r;
CBIf = .1;
CDIf = 0.1;
qI = 10;
n = 10;

General::spell1 : Possible spelling error: new symbol name
 "CIBo" is similar to existing symbol "CIB".

General::spell : Possible spelling error: new symbol name
 "CIDo" is similar to existing symbols {CIBo, CID}.
```

Solve the equation and assign the functions:

```
In[89]:= permflow = Flatten[
           Solve[0 == -CBII[t] qIIb/VI + PmB Am/VI (CBI[t] - CBII[t]), qIIb]
                         ]
         permflow[[1, 2]]
Out[89]= {qIIb → 10(CBI[t] - CBII[t])/CBII[t]}
Out[90]= 10 (CBI[t] - CBII[t])/CBII[t]
```

Now we use this solution in the solution of the full set of equations and to make the required plots. The first taks is to set the equations using the new definition for qIIb[t]:

```
In[91]:=  qIIb[t_] := permflow[[1, 2]]

In[92]:=  eqns = {∂tCBI[t] == (CBIf - CBI[t]) qI/VI - PmB Am/VI (CBI[t] - CBII[t]),
            ∂tCDI[t] == (CDIf - CDI[t]) qI/VI - PmD Am/VI (CDI[t] - CDII[t]),
            ∂tCBII[t] == (-CBII[t]) qIIb[t]/VII + PmB Am/VII (CBI[t] - CBII[t]),
            ∂tCDII[t] == (-CDII[t]) qIIb[t]/VII + PmD Am/VII (CBI[t] - CBII[t]),
            CBI[0] == CIBo, CDI[0] == CIDo, CBII[0] == 10^-10, CDII[0] == 0};

In[93]:=  eqns
```

```
Out[93]= {CBI′[t] == 10(0.1 - CBI[t]) - 10(CBI[t] - CBII[t]),
```

$$\mathrm{CDI}'[t] == 10(0.1 - \mathrm{CDI}[t]) + \frac{1}{10}(-\mathrm{CDI}[t] + \mathrm{CDII}[t]),\ \mathrm{CBII}'[t] == 0,$$

$$\mathrm{CDII}'[t] == \frac{1}{10}(\mathrm{CBI}[t] - \mathrm{CBII}[t]) - \frac{10(\mathrm{CBI}[t] - \mathrm{CBII}[t])\mathrm{CDII}[t]}{\mathrm{CBII}[t]},$$

$$\mathrm{CBI}[0] == 1.\times 10^{-6},\ \mathrm{CDI}[0] == 1.\times 10^{-6},\ \mathrm{CBII}[0] == \frac{1}{10000000000},$$

```
         CDII[0] == 0}
```

Next, we solve the equations numerically subject to the initial conditions and over the time range of interest:

```
In[94]:= numsol = Flatten[NDSolve[
             eqns, {CBI[t], CDI[t], CBII[t], CDII[t]},
             {t, 0, n VI/(PmB Am)}]]
         numsol
Out[94]= {CBI[t]  → InterpolatingFunction[{{0., 1.}}, <>][t],
          CDI[t]  → InterpolatingFunction[{{0., 1.}}, <>][t],
          CBII[t]  → InterpolatingFunction[{{0., 1.}}, <>][t],
          CDII[t]  → InterpolatingFunction[{{0., 1.}}, <>][t]}

Out[95]= {CBI[t]  → InterpolatingFunction[{{0., 1.}}, <>][t],
          CDI[t]  → InterpolatingFunction[{{0., 1.}}, <>][t],
          CBII[t]  → InterpolatingFunction[{{0., 1.}}, <>][t],
          CDII[t]  → InterpolatingFunction[{{0., 1.}}, <>][t]}
```

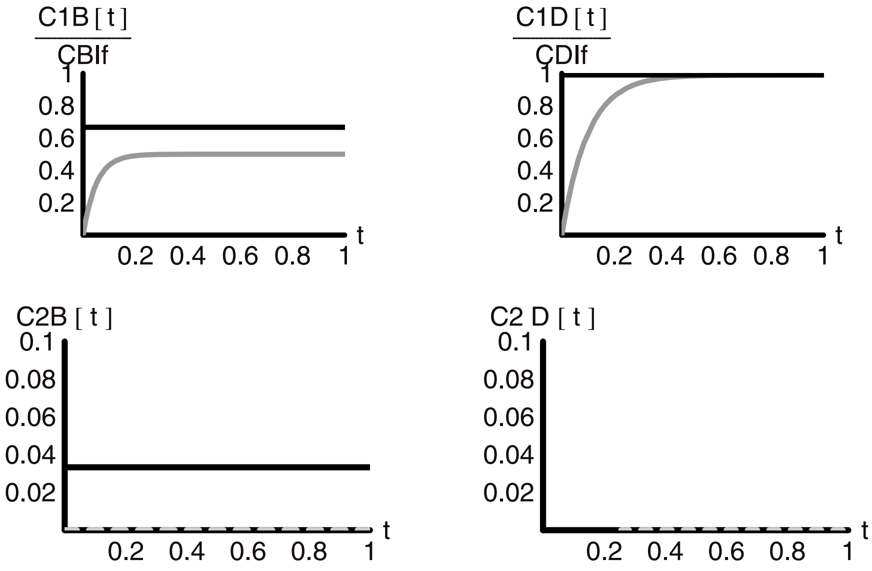
The solutions are assigned to functions:

```
In[96]:= C1B[t_]:= Evaluate[CBI[t] /. numsol[[1]]]
         C1D[t_]:= Evaluate[CDI[t] /. numsol[[2]]]
         C2B[t_]:= Evaluate[CBII[t] /. numsol[[3]]]
         C2D[t_]:= Evaluate[CDII[t] /. numsol[[4]]]
         q2[t_]:= AmPmB (C1B[t] - C2B[t])/C2B[t]
```

Finally, we set the plotting options and then make the plots:

```
In[101]:= SetOptions[Plot, DefaultFont → {"Helvetica", 10}];

          pl1 = Plot[C1B[t]/CBIf, {t, 0, n VI/(PmB Am)},
          PlotStyle → {Thickness[0.02], GrayLevel[0.6]},
          DisplayFunction → Identity,
                            C1B[t]
          AxesLabel → {"t", "------"},
                            CBIf
                               VI
          PlotRange → {{0, n -------}, {0, 1}},
                             PmB Am
          Epilog → {Thickness[0.02],
                     CB1ss       VI      CB1ss
           Line[{{0, -----}, {n -------, -----}}]}
                     CBIf      PmB Am    CBIf
          ];
                  C1D[t]             VI
    pl2 = Plot[------, {t, 0, n -------},
                  CDIf             PmB Am
          PlotStyle → {Thickness[0.02], GrayLevel[0.6]},
          DisplayFunction → Identity,
                            C1D[t]
          AxesLabel → {"t", "------"},
                            CBIf
          PlotStyle → GrayLevel[.4],
                               VI
          PlotRange → {{0, n -------}, {0, 1}},
                             PmB Am
          Epilog → {Thickness[0.02],
                     CD1ss        VI       CD1ss
           Line[{{0, -----}, {n  -------, -----}}]}];
                     CBIf       PmB Am     CBIf
                                        VI
    pl3 = Plot[C2B[t], {t, 0, n -------},
                                      PmB Am
          PlotStyle → {Dashing[{0.04, 0.04}],
           Thickness[0.02], GrayLevel[0.8]},
          DisplayFunction → Identity,
          AxesLabel → {"t", "C2B[t]"},
          PlotStyle → {GrayLevel[0], Dashing[{0.05, 0.05}]},
                               VI
          PlotRange → {{0, n -------}, {0, CBIf}},
                             PmB Am
          Epilog → {{Thickness[0.02],
                                     VI
           Line[{{0, CB2ss}, {n -------, CB2ss}}]}}
                                   PmB Am
          ];
```

```
pl4 = Plot[C2D[t], {t, 0, n VI/(PmB Am)},
    PlotStyle → {Dashing[{0.03, 0.04}],
     Thickness[0.02], GrayLevel[0.8]},
    DisplayFunction → Identity,
    AxesLabel → {"t", "C2D[t]"},
    PlotRange → {{0, n VI/(PmB Am)}, {0, CDIf}},
    AxesOrigin → {0, 0}];

Show[GraphicsArray[{{pl1, pl2}, {pl3, pl4}}]];
```



The numbers all seem reasonable until we examine them a bit more carefully. We notice that the concentration of **B** at steady state on the retentate side divided by its feed concentration is well below the value of 0.66, which we had computed from the purely steady-state analysis. Furthermore, the concentration of **B** on the permeate side is nearly zero for all times! The product of the permeate concentration and flow rate is the molar flow of the impurity **B** out of the system below the membrane. The molar flow is reasonable in magnitude, but when we compute the volume flow we see that it is ludicrously large O (10^9)! Why? The reason is the seemingly reasonable assumption that the permeate side would always be at a steady state, that is, there would be no time lag for the flow to fully develop out of the unit here. But the concentrations at the lower side of the membrane are so low that in order for the mathematics to satisfy the steady state that we have imposed the flow must be compensatingly large. If we set the initial concentration of **B** on the permeate side to zero, the flow must be infinite and so it goes from there with real number values for this concentration. This is a case where the transient and steady state do not mix well!

We have two options: either include an inlet flow on the permeate side of the membrane, or set the exit flow rate. From the mathematical perspective the two amount to the same thing, thus they get us out of the bind. Physically, they are reasonable as well. We can certainly configure a mass flow controller and a pump that would keep the flow out of the system constant, but the inlet flow is easier to do experimentally, and therefore we will include this.

The inlet flow on the permeate side would be that of an inert gas, for instance, which continuously sweeps the lower side of the membrane and clears the permeate. If the flow is large compared to the volume, then we will have near zero concentrations of the permeate gas and maximal permeation rates. (In fact, this is what the steady-state analysis we just did imposed automatically.) We still need to consider only the component balances at this point. We will use all the same numbers and equations except that **qII** will be fixed at the inlet and outlet of the permeate side of the unit:

```
In[107]:= Clear[PmB, PmA, Am, VI, VII, CIBo, CIDo, plI, plII, plIII,
           plIV, t]

In[108]:= CIBo = 0.000001;
          CIDo = 0.000001;
          VI = 1;
          PmB = 10^-4;
          PmD = 10^-6;
          VII = VI;
          r = 5;
          Am = 10^r;
          CBIf = .1;
          CDIf = 0.1;
          qI = 10;
          qII = 10;
          n = 10;

In[121]:=
```

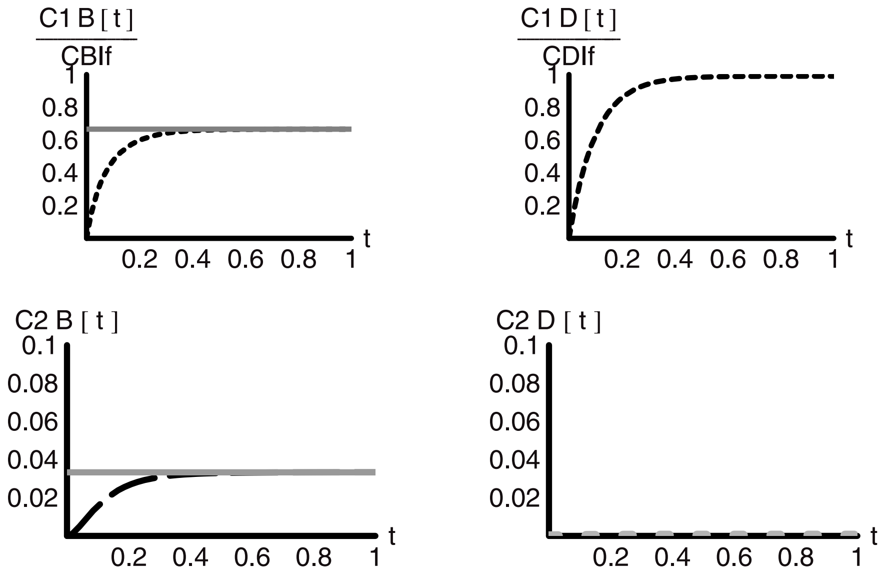
$$\text{eqns}=\{\partial_t \text{CBI}[t] == (\text{CBIf}-\text{CBI}[t])\frac{\text{qI}}{\text{VI}} - \frac{\text{PmB Am}}{\text{VI}}(\text{CBI}[t]-\text{CBII}[t]),$$

$$\partial_t \text{CDI}[t] == (\text{CDIf}-\text{CDI}[t])\frac{\text{qI}}{\text{VI}} - \frac{\text{PmD Am}}{\text{VI}}(\text{CDI}[t]-\text{CDII}[t]),$$

$$\partial_t \text{CBII}[t] == (-\text{CBII}[t])\frac{\text{qII}}{\text{VII}} + \frac{\text{PmB Am}}{\text{VII}}(\text{CBI}[t]-\text{CBII}[t]),$$

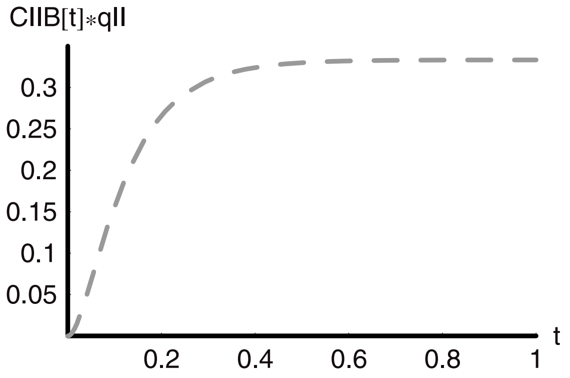
```
                           qII     PmD Am
   ∂tCDII[t] == (-CDII[t])─── + ────── (CBI[t] - CBII[t]),
                           VII      VII
   CBI[0] == CIBo, CDI[0] == CIDo, CBII[0] == 10^-10,
   CDII[0] == 0};

numsol = Flatten[
   NDSolve[
    eqns,
    {CBI[t], CDI[t], CBII[t], CDII[t]},
                 VI
    {t, 0, n ──────}
              PmB Am
              ]
   ];
C1B[t_]:= Evaluate[CBI[t] /. numsol[[1]]]
C1D[t_]:= Evaluate[CDI[t] /. numsol[[2]]]
C2B[t_]:= Evaluate[CBII[t] /. numsol[[3]]]
C2D[t_]:= Evaluate[CDII[t] /. numsol[[4]]]

SetOptions[Plot, DefaultFont → {"Helvetica", 10},
 AxesStyle → Thickness[0.02]];
           C1B[t]              VI
pl1 = Plot[──────, {t, 0, n──────},
            CBIf             PmB Am
   DisplayFunction → Identity,
                          C1B [t]
   AxesLabel → {"t", "───────"},
                           CBIf
   PlotStyle → {Thickness[0.02], Dashing[{0.02, 0.03}]},
                       VI
   PlotRange → {{0, n──────}, {0, 1}},
                     PmB Am
   Epilog →
    {GrayLevel[0.5], Thickness[0.02],
               CB1ss       VI      CB1ss
     Line[{{0, ─────}, {n──────, ─────}}]}];
               CBIf      PmB Am    CBIf
           C1D[t]              VI
pl2 = Plot[──────, {t, 0, n──────},
            CDIf             PmB Am
   DisplayFunction → Identity,
                          C1D[t]
   AxesLabel → {"t", "──────"},
                           CDIf
   PlotStyle → {Thickness[.02], Dashing[{0.03, 0.03}]},
                       VI
   PlotRange → {{0, n──────}, {0, 1}}];
                     PmB Am
```

```
pl3 = Plot[C2B[t], {t, 0, n VI/(PmB Am)},
   DisplayFunction → Identity,
   AxesLabel → {"t", "C2B[t]"},
   PlotStyle → {Thickness[.02], Dashing[{0.15, 0.05}]},
   PlotRange → {{0, n VI/(PmB Am)}, {0, CBIf}},
   Epilog →
     {GrayLevel[.6], Thickness[0.02],
      Line[{{0, CB2ss}, {n VI/(PmB Am), CB2ss}}]}];
pl4 = Plot[C2D[t], {t, 0, n VI/(PmB Am)},
   DisplayFunction → Identity,
   AxesLabel → {"t", "C2D[t]"},
   PlotStyle → {Thickness[.03], Dashing[{.03, .09}],
    GrayLevel[.7]},
   PlotRange → {{0, n VI/(PmB Am)}, {0, CDIf}},
   AxesOrigin → {0, 0}];

Show[GraphicsArray[{{pl1, pl2}, {pl3, pl4}}]];
```



The following is a plot of the mass flow of B from the system that is the product C2B[t]qII:

```
In[133]:= Plot[C2B[t] qII, {t, 0, n VI/(PmB Am)},
            AxesStyle  → Thickness[.01],
            AxesLabel  → {"t", "CIIB[t]*qII"},
            PlotStyle  → {Thickness[.01],
             Dashing[{0.05, 0.05}], GrayLevel[ .6]},
            PlotRange  → {{0, n VI/(PmB Am)}, {0, .35}}];
```



Now the analyses make sense. The steady-state analysis agrees with the transient analysis. However, let us consider all of this one more time. In what manner did the analysis show us we should move if we wish to get the ultimate removal of **B** from the feed stream with the areas, feed flow, and permeances all fixed? The answer is obvious. The pseudo-steady-state analysis showed us that if we could somehow reduce the concentration of **B** on the permeate side to near zero values, then we could remove nearly 50% of it versus only 33% with these conditions. How could we do this? How about raising the sweep flow rate on permeate side? This will have the effect of keeping the concentration of **B** very low and increasing the driving force for **B** across the membrane. How much higher would the sweep flow have to be to accomplish this? The answer is in what follows; on the order of a factor of 10 increase will do it!

```
In[134]:= Clear[PmB, PmA, Am, VI, VII, CIBo, CIDo, plI, plII, plIII,
           plIV, t]

In[135]:= CIBo = 0.000001;
          CIDo = 0.000001;
          VI = 1;
          PmB = 10^-4;
```

```
PmD = 10^-6;
VII = VI;
r = 5;
Am = 10^r;
CBIf = .1;
CDIf = 0.1;
qI = 10;
qII = 10^3;
n = 10;
```

In[148]:= eqns = {∂_tCBI[t] == (CBIf - CBI[t]) $\frac{qI}{VI}$ - $\frac{PmB\ Am}{VI}$(CBI[t] - CBII[t]),

∂_tCDI[t] == (CDIf - CDI[t]) $\frac{qI}{VI}$ - $\frac{PmD\ Am}{VI}$(CDI[t] - CDII[t]),

∂_tCBII[t] == (-CBII[t]) $\frac{qII}{VII}$ + $\frac{PmB\ Am}{VII}$(CBI[t] - CBII[t]),

∂_tCDII[t] == (-CDII[t]) $\frac{qII}{VII}$ + $\frac{PmD\ Am}{VII}$(CBI[t] - CBII[t]),

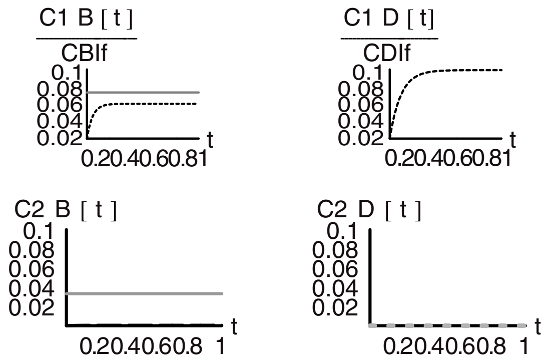
CBI[0] == CIBo, CDI[0] == CIDo, CBII[0] == 10^{-10},

CDII[0] == 0};

In[149]:= numsol = Flatten[
```
   NDSolve[
    eqns,
    {CBI[t], CDI[t], CBII[t], CDII[t]},
```
{t, 0, n $\frac{V}{PmB\ Am}$}
```
         ]
   ];

C1B[t_]:= Evaluate[CBI[t] /. numsol[[1]]]
C1D[t_]:= Evaluate[CDI[t] /. numsol[[2]]]
C2B[t_]:= Evaluate[CBII[t] /. numsol[[3]]]
C2D[t_]:= Evaluate[CDII[t] /. numsol[[4]]]

SetOptions[Plot, DefaultFont → {"Helvetica", 10},
  AxesStyle → Thickness[0.02]];
```

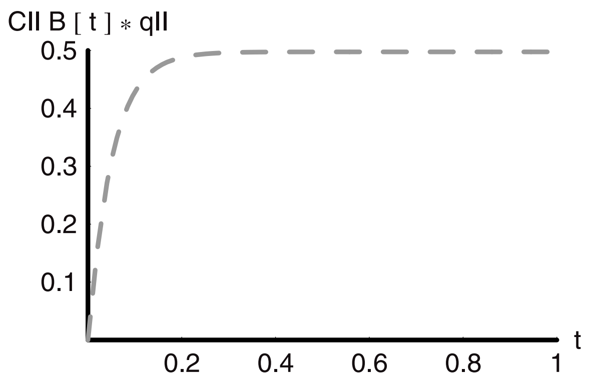
pl1 = Plot[$\frac{C1B[t]}{CBIf}$, {t, 0, n$\frac{VI}{PmB\ Am}$},

```
  DisplayFunction → Identity,
                          C1B[t]
    AxesLabel  →  {"t", " ─────── "},
                          CBIf
    PlotStyle  →  {Thickness[0.02], Dashing[{0.02, 0.03}]},
                          VI
    PlotRange  →  {{0, n ──────── }, {0, 1}},
                        PmB  Am
    Epilog  →  {GrayLevel[0.5], Thickness[0.02],
                   CB1ss          VI      CB1ss
      Line[{{0,  ─────── }, {n ─────── , ─────── }}]}];
                   CBIf         PmB Am    CBIf

              C1D[t]              VI
pl2 = Plot[ ─────── , {t, 0, n ─────── },
              CDIf               PmB Am
                                                        C1D[t]
    DisplayFunction → Identity, AxesLabel → {"t", " ─────── "},
                                                        CDIf
    PlotStyle  →  {Thickness[ .02], Dashing[{0.03, 0.03}]},
                         VI
    PlotRange  →  {{0, n ─────── }, {0, 1}}];
                        PmB Am

                                  VI
pl3 = Plot[C2B[t], {t, 0, n ─────── },
                                 PmB Am
    DisplayFunction  →  Identity,
    AxesLabel  →  {"t", "C2 B[t]"},
    PlotStyle  →  {Thickness[.02], Dashing[{0.15, 0.05}]},
                         VI
    PlotRange  →  {{0, n ─────── }, {0, CBIf}},
                        PmB Am
    Epilog  →  {GrayLevel[.6], Thickness[0.02],
                                VI
     Line[{{0, CB2ss}, {n ─────── , CB2ss}}]}];
                               PmB Am

                                  VI
pl4 = Plot[C2D[t], {t, 0, n ─────── },
                                 PmB Am
    DisplayFunction  →  Identity,
    AxesLabel  →  {"t", "C2D [t]"},
    PlotStyle  →  {Thickness[.03],
     Dashing[{.03, .09}], GrayLevel[.7]},
                         VI
    PlotRange  →  {{0, n ─────── }, {0, CDIf}},
                        PmB Am
    AxesOrigin  →  {0, 0}];

Show[GraphicsArray[{{pl1, pl2}, {pl3, pl4}}]];
```



And once again the total flow of B from the system:

```
In[160]:= Plot[C2B[t]qII, {t, 0, n VI/(PmB Am)},
            AxesStyle → Thickness[.01],
            AxesLabel → {"t", "CII B[t] * qII"},
            PlotStyle → {Thickness[.01], Dashing[{0.05, 0.05}],
             GrayLevel[.6]},
            PlotRange → {{0, n VI/(PmB Am)}, {0, .5}}];
```



6.4 Expanding Cell

Consider the following problem. A spherical cell consists of a thin membrane surrounding a salt solution. Outside of the cell membrane there is a solution that is isotonic with that within the membrane. The cell is removed instantaneously from its surroundings and placed into an environment of pure water. The action of osmosis immediately drives water through the membrane to cause dilution of its contents. The transport across the membrane is a permeation process with a rate of:

$$J_{\mathrm{H_2O}} = \mathrm{Pm}\left(C_{\mathrm{H_2O}}^{\mathrm{Outside}} - C_{\mathrm{H_2O}}^{\mathrm{Inside}}\right)$$

The direction of the flow is from the region of lower salt concentration to higher salt concentration, but from higher water concentration to lower water concentration. The concentration of water on the outside of the cell is taken to be equal to the density of pure water $\boldsymbol{C_{\mathrm{H_2O}}^{\mathrm{Outside}}} = \boldsymbol{\rho_{\mathrm{H_2O}}}$. Inside the membrane the concentration of water is increasing as the density of the solution is decreasing. The density of the cellular content follows the linear relationship:

$$\rho_{\mathrm{cell}}[t] = \rho_{\mathrm{H_2O}} + \gamma C_{\mathrm{salt}}^{\mathrm{Cell}}[t]$$

While water transports osmotically across the cell membrane to dilute the cellular contents, the cell grows. The membrane stretches to accommodate the newly accumulated mass. The geometry of the cell remains constant, that is it grows in every direction through the whole of the solid angle by the same amount in the same time period. The physical situation with this cell is sketched in Figure 3:

We would like to know how the membrane grows as a function of time, how much the cellular contents are diluted in time, and related information about this process. We will find the answers to these questions by modeling the dynamic process. We begin by writing the

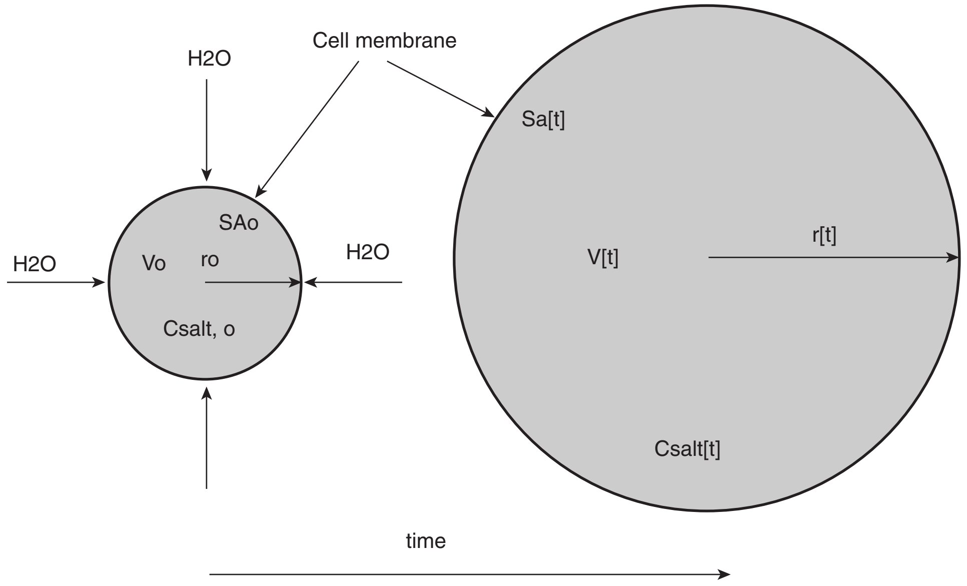


Figure 3

material balance equations within the cell in a general form:

$$\begin{aligned}
\frac{dm_{\text{tot}}[t]}{dt} &= P_m\,\text{SA}\left(C_{\text{H}_2\text{O}}^{\text{Outside}} - C_{\text{H}_2\text{O}}^{\text{Inside}}\right)\\
\frac{dm_{\text{salt}}[t]}{dt} &= 0\\
\frac{dm_{\text{H}_2\text{O}}[t]}{dt} &= P_m\,\text{SA}\left(C_{\text{H}_2\text{O}}^{\text{Outside}} - C_{\text{H}_2\text{O}}^{\text{Inside}}\right)
\end{aligned}$$

The surface area and volume of the cell are functions of the radius:

$$V[t] = \tfrac{4}{3}\pi\,\text{r}[\text{t}]^3 \quad \text{SA}[\text{t}] = 4\pi\,\text{r}[\text{t}]^2$$
$$\frac{dV[t]}{dt} = 4\pi\,\text{r}[\text{t}]^2\frac{dr[t]}{dt} \quad \frac{d\text{SA}[t]}{dt} = 8\pi\,\text{r}[\text{t}]\frac{dr[t]}{dt}$$

We will need to relate the water concentration to the salt concentration and the change in density to the water rather than the salt concentration, as it is the latter that is flowing into the cell:

$$\begin{aligned}
\rho_{\text{cell}}[t] &= \rho_{\text{H}_2\text{O}} + \gamma C_{\text{salt}}^{\text{Cell}}[\text{t}]\\
C_{\text{salt}}^{\text{Cell}}[\text{t}] + C_{\text{H}_2\text{O}}^{\text{Cell}}[\text{t}] &= \rho_{\text{H}_2\text{O}} + \gamma C_{\text{salt}}^{\text{Cell}}[\text{t}]\\
C_{\text{H}_2\text{O}}^{\text{Cell}}[\text{t}] &= \rho_{\text{H}_2\text{O}} + (\gamma-1)C_{\text{salt}}^{\text{Cell}}[\text{t}]\\
C_{\text{salt}}^{\text{Cell}}[\text{t}] &= \frac{C_{\text{H}_2\text{O}}^{\text{Cell}}[t] - \rho_{\text{H}_2\text{O}}}{(\gamma-1)}\\
\rho_{\text{cell}}[t] &= \rho_{\text{H}_2\text{O}} + \gamma\frac{C_{\text{H}_2\text{O}}^{\text{Cell}}[t] - \rho_{\text{H}_2\text{O}}}{(\gamma-1)}\\
\rho_{\text{cell}}[t] &= \frac{\gamma C_{\text{H}_2\text{O}}^{\text{Cell}}[t] - \rho_{\text{H}_2\text{O}}}{(\gamma-1)}
\end{aligned}$$

With these expressions and the component balances we can now find the expression for $\boldsymbol{C_{\text{salt}}^{\text{Cell}}[\text{t}]}$ and $\boldsymbol{C_{\text{H}_2\text{O}}^{\text{Cell}}[\text{t}]}$ as functions of **r[t]**:

$$\begin{aligned}
\frac{dm_{\text{H}_2\text{O}}[t]}{dt} &= P_m\,\text{SA}\left(C_{\text{H}_2\text{O}}^{\text{Outside}} - C_{\text{H}_2\text{O}}^{\text{Inside}}\right)\\
\frac{dm_{\text{salt}}[t]}{dt} &= 0 = \frac{dC_{\text{salt}}^{\text{Cell}}[t]V[t]}{dt} = C_{\text{salt}}^{\text{Cell}}[t]\frac{dV[t]}{dt} + \text{V}[\text{t}]\frac{dC_{\text{salt}}^{\text{Cell}}[t]}{dt}\\
C_{\text{salt}}^{\text{Cell}}[t] &= \frac{C_{\text{salt}}^{\text{Cell}}[0]V[0]}{V[t]} = \frac{C_{\text{salt}}^{\text{Cell}}[0]r_o^3}{r[t]^3}\\
C_{\text{H}_2\text{O}}^{\text{Cell}}[t] &= \rho_{\text{H}_2\text{O}} + (\gamma-1)\frac{C_{\text{salt}}^{\text{Cell}}[0]r_o^3}{r[t]^3}
\end{aligned}$$

Taking the derivative of this, we find the change in salt concentration as a function of time to be:

```
In[161]:= r =.
```

$$\partial_t C^{Cell}_{H_2O}[t] == \partial_t(\rho_{H_2O} + (\gamma - 1)\frac{C^{Cell}_{salt}[0] r_o^3}{r[t]^3})$$

```
Out[162]= (C^Cell_H2O)'[t] == 0
```

$$(C^{Cell}_{H_2O})'[t] == 0$$

Returning to the total material balance, we can use all of the definitions for the variables in terms of **r[t]** to solve for the derivative of **r[t]** in terms of just the cell parameters. Once we have this, we can then solve for **r[t]** explicitly in time:

$$\frac{dm_{tot}[t]}{dt} = P_m\,\mathrm{SA}\left(C^{Outside}_{H_2O} - C^{Cell}_{H_2O}[t]\right)$$
$$\frac{d\rho_{cell}[t]V[t]}{dt} = P_m\,\mathrm{SA}\left(C^{Outside}_{H_2O} - C^{Cell}_{H_2O}[t]\right)$$

The following cell solves and reexpresses, the left- and right-hand sides in terms of **r[t]** and then solves for **r′[t]**. Finally, we solve for **r[t]** and plot the function:

```
In[163]:= r =.
```

$$C_{H_2O}[t\_] := \rho_{H_2O} + (\gamma - 1)\frac{C^{Cell}_{salt,o}\ r_o^3}{r[t]^3}$$

$$\rho_{cell}[t\_] := \rho_{H_2O} + \gamma\frac{C_{H_2O}[t] - \rho_{H_2O}}{(\gamma - 1)}$$

$$V[t\_]\ :\ = \frac{4}{3}\ r[t]^3$$

$$lhs = \partial_t(\rho_{cell}[t]V[t])$$

```
General::spell1 : Possible spelling error: new symbol name
 "cell" is similar to existing symbol "Cell".
```

$$Out[167]= -\frac{4\gamma r_o^3 C^{Cell}_{salt,o} r'[t]}{r[t]} + 4r[t]^2(\rho_{H_2O} + \frac{\gamma r_o^3 C^{Cell}_{salt,o}}{r[t]^3})r'[t]$$

```
In[168]:= SA[t_] := 4πr[t]^2
          rhs = Pm SA[t] (ρH2O - CH2O[t])
          Solve[lhs == rhs,r'[t]]
```

$$Out[169]= -\frac{4\pi\,Pm(-1 + \gamma)r_o^3 C^{Cell}_{salt,o}}{r[t]}$$

$$Out[170]= \{\{r'[t] \to -\frac{\pi\,Pm(-1 + \gamma)r_o^3 C^{Cell}_{salt,o}}{r[t]^3\rho_{H_2O}}\}\}$$

In[171]:= **Simplify[PowerExpand[DSolve[**

$$\mathbf{\{\partial_t r[t] == -\frac{\pi Pm(-1+\gamma) r_o^3 C_{salt,o}^{Cell}}{r[t]^3 \rho_{H_2O}},}$$

$$\mathbf{r[0] == r_o\},}$$

$$\mathbf{r[t],\ t]]]}$$

Out[171]=

$$\{r[t] \to -\frac{(1+i)\pi^{1/4} Pm^{1/4}(-1+\gamma)^{1/4} r_o^{3/4} C_{salt,o}^{Cell/4}\left(t+\frac{r_o \rho_{H_2O} C_{salt,o}^{-Cell}}{4\pi Pm - 4\pi Pm\gamma}\right)^{1/4}}{\rho_{H_2O}^{1/4}},$$

$$r[t] \to -\frac{(1-i)\pi^{1/4} Pm^{1/4}(-1+\gamma)^{1/4} r_o^{3/4} C_{salt,o}^{Cell/4}\left(t+\frac{r_o \rho_{H_2O} C_{salt,o}^{-Cell}}{4\pi Pm - 4\pi Pm\gamma}\right)^{1/4}}{\rho_{H_2O}^{1/4}},$$

$$r[t] \to \frac{(1-i)\pi^{1/4} Pm^{1/4}(-1+\gamma)^{1/4} r_o^{3/4} C_{salt,o}^{Cell/4}\left(t+\frac{r_o \rho_{H_2O} C_{salt,o}^{-Cell}}{4\pi Pm - 4\pi Pm\gamma}\right)^{1/4}}{\rho_{H_2O}^{1/4}},$$

$$r[t] \to \frac{(1+i)\pi^{1/4} Pm^{1/4}(-1+\gamma)^{1/4} r_o^{3/4} C_{salt,o}^{Cell/4}\left(t+\frac{r_o \rho_{H_2O} C_{salt,o}^{-Cell}}{4\pi Pm - 4\pi Pm\gamma}\right)^{1/4}}{\rho_{H_2O}^{1/4}}\}$$

In[172]:=

$$\mathbf{Solve\left[\int_{ro}^{r[t]} r[t]^3\, dr[t] == \int_0^t -\frac{\pi Pm(-1+\gamma) r_o^3 C_{salt,o}^{Cell}}{\rho_{H_2O}}\, dt, r[t]\right]}$$

Out[172]=

$$\{\{r[t] \to -\sqrt{2}\left(\frac{ro^4}{4} - \frac{\pi Pm\, t(-1+\gamma) r_o^3 C_{salt,o}^{Cell}}{\rho_{H_2O}}\right)^{1/4}\},$$

$$\{r[t] \to -i\sqrt{2}\left(\frac{ro^4}{4} - \frac{\pi Pm\, t(-1+\gamma) r_o^3 C_{salt,o}^{Cell}}{\rho_{H_2O}}\right)^{1/4}\},$$

$$\{r[t] \to -i\sqrt{2}\left(\frac{ro^4}{4} - \frac{\pi Pm\, t(-1+\gamma) r_o^3 C_{salt,o}^{Cell}}{\rho_{H_2O}}\right)^{1/4}\},$$

$$\{r[t] \to \sqrt{2}\left(\frac{ro^4}{4} - \frac{\pi Pm\, t(-1+\gamma) r_o^3 C_{salt,o}^{Cell}}{\rho_{H_2O}}\right)^{1/4}\}\}$$

There are four solutions to this equation, and two are real and two are complex. By calling the package **Miscellaneous 'RealOnly'**, only the real solutions are displayed. Of the two real solutions, only the second is physical, as it is growing as a function of time (see what follows here in the In statements and graph):

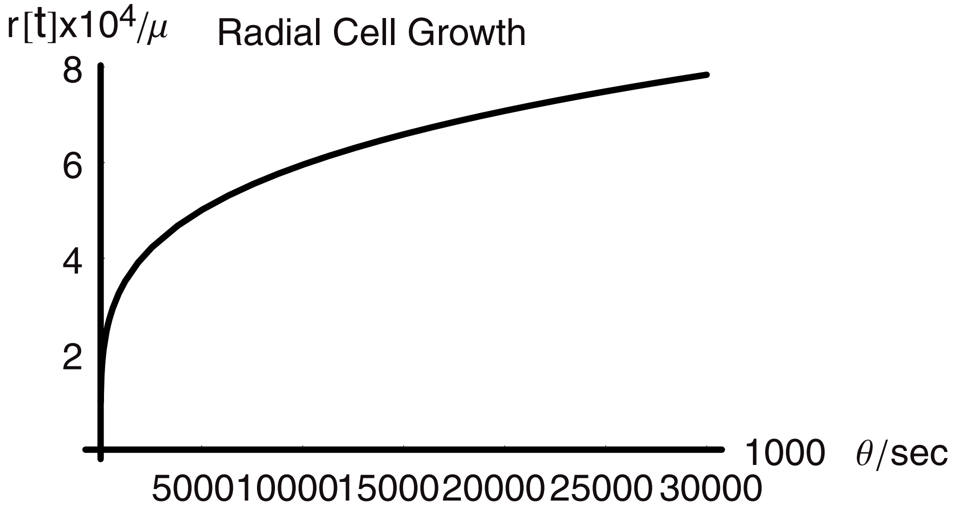
```
In[173]:= SetOptions[{Plot, ListPlot}, AxesStyle → {Thickness[0.01]},
            PlotStyle  → {PointSize[0.015], Thickness[0.01]},
            DefaultFont  → {"Helvetica", 17}];
```

In[174]:= **r =.**

$$\mathbf{r[t\_] := \sqrt{2}\left(\frac{ro^4}{4} - \frac{\pi Pm\, t(-1+\gamma) ro^3 C_{salt,o}}{\rho_{H_2O}}\right)^{1/4}}$$

ro = 10^-4;

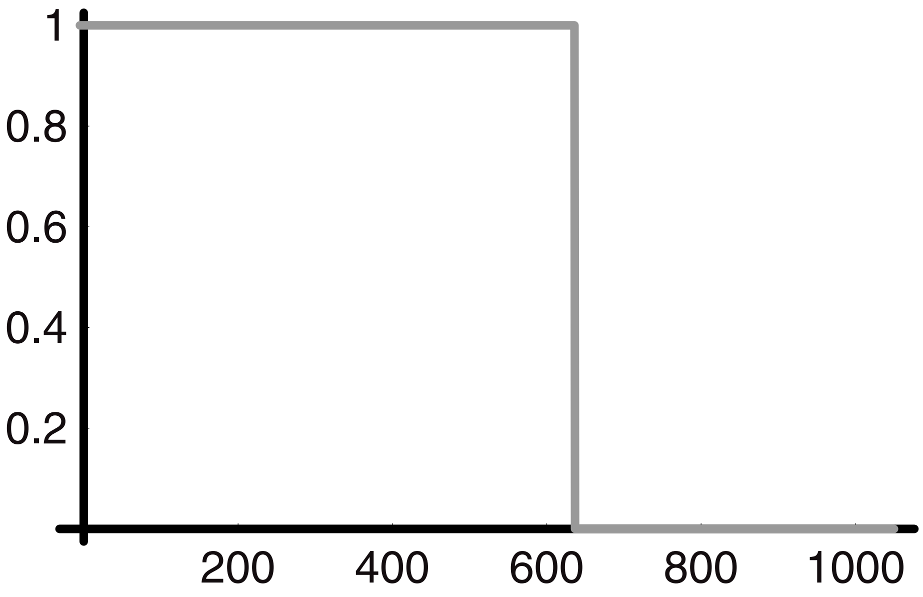
```
Pm = 10^-3;
γ = 0.9;
C_salt,o = 0.01;
ρ_H2O = 1;
tmax = 1000 3Pm/r[0];
Plot[r[t]10^4, {t, 0, tmax},
  AxesLabel → {"1000 θ/sec", "r[t]×10^4/μ"},
  PlotLabel → "Radial Cell Growth"];
```



This solution to the problem shows that it grows very fast at a short time, but then it slows at longer times. There is, however, one noticeable issue that crops up with this solution, which is that the radius continues to grow with increasing time. To be physical the membrane surface would have to be infinitely elastic, which is impossible. Instead we expect the membrane to rupture and explode at some critical radius. Thus the solution should be indicative of this behavior.

We can modify the solution to include this behavior. The critical radius can be expressed as a multiple of the initial radius at time zero. We want the function to literally "blow-up" when we reach this radius. The critical condition can be expressed with the **UnitStep** function. It will be of unit value until the critical condition is reached, whereupon it will go to zero. Recall the behavior of the **UnitStep** given in terms of **r[t]** and the critical condition of **2r[0]**:

```
In[183]:= Plot[1 - UnitStep[r[t] - 3r[0]], {t, -5, 1050},
            PlotStyle → {Thickness[.01], GrayLevel[.6]}];
```

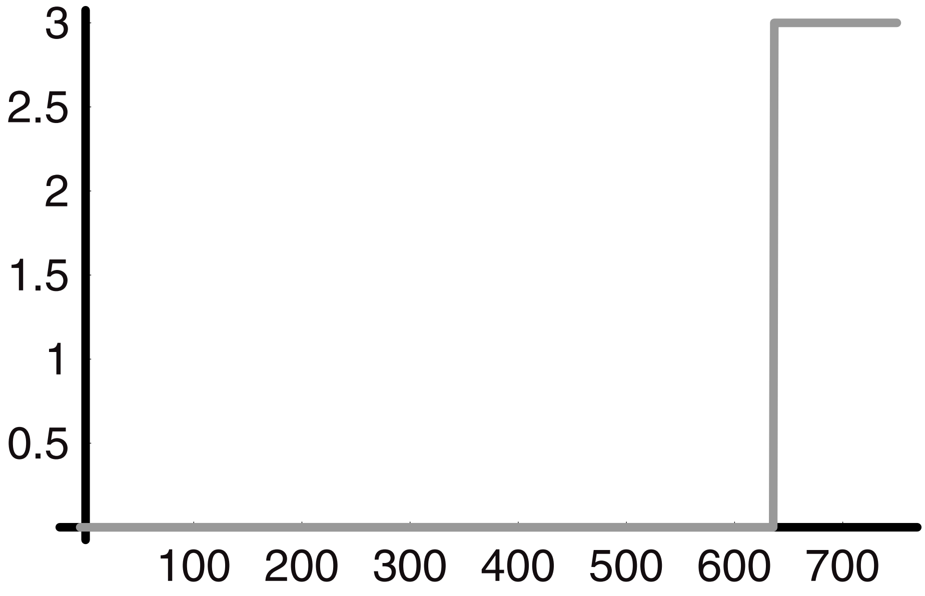


By dividing **r[t]** by the **UnitStep** function we obtain the correct behavior; the solution is meaningful up to the critical radius but not beyond and is shown in what follows in the In statement and two graphs:

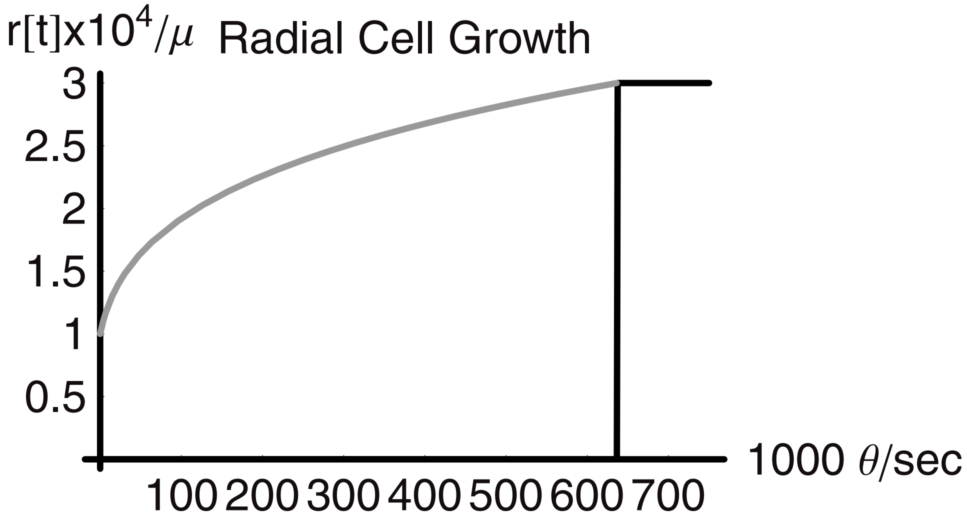
```
In[184]:= Clear[rbl, us]

          rbl[t_] :=
```

$$\frac{\sqrt{2}}{1 - \texttt{UnitStep[r[t] - 3r[0]]}}\left(\frac{ro^4}{4} - \frac{\pi\, Pm\, t(-1 + \gamma) ro^3 C_{salt,o}}{\rho_{H_2O}}\right)^{1/4}$$

```
          ro = 10^-4;
          Pm = 10^-3;
          γ = 0.9;
          C_salt,o = 0.01;
          ρ_H2O = 1;
          tmax = 25 (3 Pm)/(r[0]);

          us = Plot[3UnitStep[r[t] - 3r[0]], {t, -5, tmax},
              PlotStyle → {Thickness[.01], GrayLevel[.6]}];

          Plot[{3UnitStep[r[t] - 3r[0]], rbl[t]10^4}, {t, 0, tmax},
            AxesLabel → {"1000 θ/sec", "r[t]×10^4/μ"},
            PlotLabel → "Radial Cell Growth",
            PlotStyle → {{Thickness[.01]}, {Thickness[.01],
             GrayLevel[.6]}}];
```

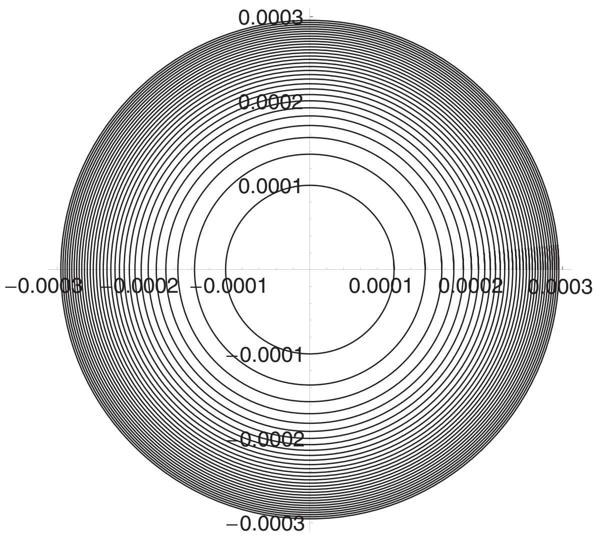


```
Power::infy : Infinite expression 1/0 encountered.

Power::infy : Infinite expression 1/0 encountered.

Power::infy : Infinite expression 1/0 encountered.

General::stop : Further output of Power::infy will be
 suppressed during this calculation.

Plot::plnr : rbl[t] 10^4 is not a machine-size real number
 at t = 658.1365691561368`.

Plot::plnr : rbl[t] 10^4 is not a machine-size real number
 at t = 641.6663540073249`.

Plot::plnr : rbl[t] 10^4 is not a machine-size real number
 at t = 638.1017437770654`.

General::stop : Further output of plot::plnr will be
 suppressed during this calculation.
```



We also might wish to project the growth in the plane. When we do so we can use our expression for the radius as a function of time and then plot the circular surfaces at integer time steps separated by a constant increment:

```
In[194]:= Show[Graphics[Table[{Circle[{0, 0}, rbl[t]]},
            {t, 0, .8tmax, 20}], AspectRatio → Automatic,
            Axes → Automatic]];
```



Another way to visualize the growth of the cell is to watch its contents expand and change structure as the membrane stretches. We can do this by "filling" the cell with smaller circular entities that are fixed in number. Imagine that can form attachments to the cell wall at fixed angular separations all around the membrane. As the membrane grows these attachments will grow in length but not width and thus their structure will emerge as the cell expands.

Let us envision how this can work with simple examples first, one taken at initial time zero and one at some later time. We will set the outer radius equal to that of **rbl[0]**. Inside this membrane we will place 32 smaller bodies with radii that are 3.5% of that of the outer radius of the cell plus one at the center. To arrange these around the interior of the cell we compute their {**x, y**} locations in terms of the radius as follows:

{m rbl[0] Cos [n Pi]//N, m rbl[0] Sin [n Pi]//N}

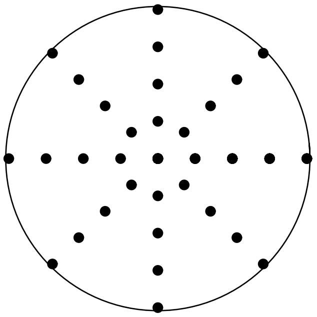
By incrementing **m** we take fractional positions along the radius at some fixed angle, which is given by **[nPi]**. The increment **n** moves the angle around the cellular interior. By nesting two Table functions, one in *n* and the other in *m*, we cover the interior completely. The radius of 0.035 rbl[0] was chosen so that the subcells eventually would overlap and fill the inside of the membrane.

To implement this we use the Disk function for the subcells, because it is filled with an **RGBColor**. This command calls for the position of the center of the disk and then for the radius of the disk. We have implemented as follows:

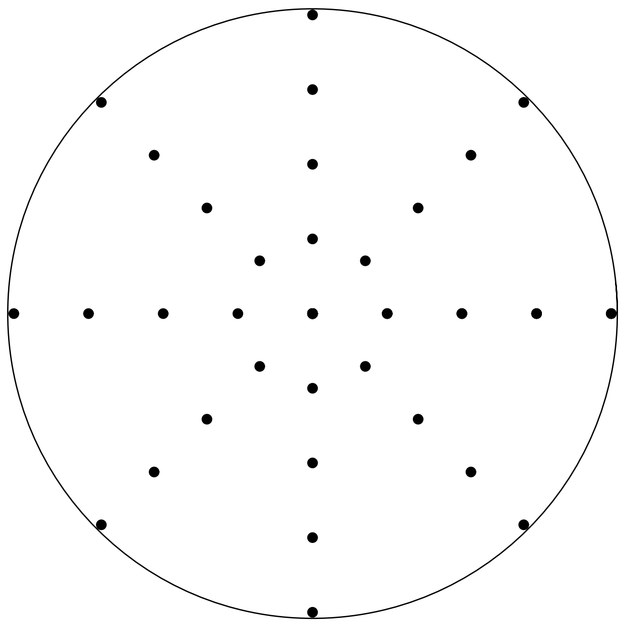
Disk[{m .98rbl[0] Cos [n Pi]//N, m .98rbl[0] Sin [n Pi]//N}, .035rbl[0]]}

The factor of 0.98 is used to keep the cell contents inside the membrane rather than on it. At time $t = 0$ and then at $t = 12$ we can show the cell and its subcellular contents with this code (remember that **r[t]** and **rbl[t]** need to be active first). See In statements [195] and [197] and graphs that follow:

```
In[195]:= circls1 = Flatten[
            Table[
             Table[
              {Graphics
                   [
                   {
                   Disk[
                     {m .98 rbl[0]Cos[n Pi] // N,
                       m .98 rbl[0]Sin[n Pi] // N}, .035 rbl[0]]}
                   ]},
                {n, 0, 2, .25}],
               {m, 0, 1, .25}],
              1];
             Show[{%, Graphics[{Circle[{0, 0}, rbl[0]]}]},
               AspectRatio → Automatic,
               PlotRange → {{-.0002, .0002}, {-.0002, .0002}}];
```



```
In[197]:= t = 120;
          circls2 = Flatten[
              Table[
               Table[
                {Graphics
                   [

                   Disk[
                    {m .98 rbl[t]Cos[n Pi] // N,
                      m .98 rbl[t]Sin[n Pi] // N}, .035rbl[0]]
                   ]},
                {n, 0, 2, .25}],
               {m, 0, 1, .25}],
              1];
          Show[{%, Graphics[{Circle[{0, 0}, rbl[t]]}]},
            AspectRatio → Automatic,
            PlotRange → {{-.0002, .0002}, {-.0002, .0002}}];
```

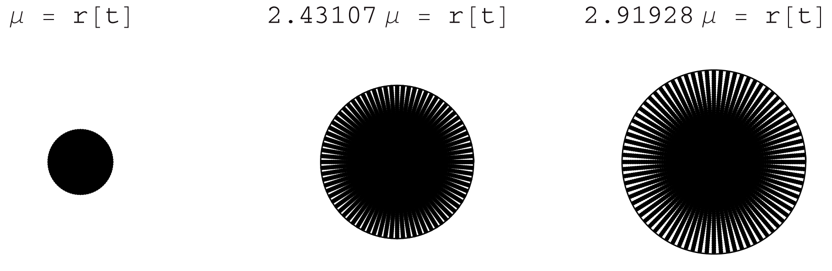


Between these two times we see that the cell has expanded and also that the subcells have moved along fixed radii to larger separation distances. Now, we can decrease the increments *m* and *n* by factors of ten in order to increase the number of subcells in total by a factor of 100 to 3200 + 1. This will fill the cell and make for a much richer visualization. **Note**: This may take a few minutes to render, depending upon the **CPU** speed of the computer you are using:

```
In[200]:=Clear[circls, t, m, n]

         circls = Flatten[
            Table[
             Table[
              {Graphics[
                Disk[{.5 m .98 rbl[t] Cos[n Pi] // N,
                  .5 m .98 rbl[t] Sin[n Pi] // N}, .035 rbl[0]]
                ]},
              {n, 0, 2, .025}],
             {m, 0, 1, .025}],
            1];

         cells = Table[
            Show[
             {circls,
                Graphics[
                  Circle[{0, 0}, .5 rbl[t]]
                          ]
                }, AspectRatio → Automatic,
               PlotRange → {{-.0002, .0002}, {-.0002, .0002}},
               PlotLabel → rbl[t] 10^4"μ = r[t]",
                   DisplayFunction → Identity],
              {t, 0,.8 tmax, 30}
                  ]
          General::spell1: Possible spelling error: new symbol
          name "cells" is similar to existing symbol "cell".

Out[202]= {-Graphics-, -Graphics-, -Graphics-, -Graphics-,
           -Graphics-, -Graphics-, -Graphics-, -Graphics-,
           -Graphics-, -Graphics-, -Graphics-, -Graphics-,
           -Graphics-, -Graphics-, -Graphics-, -Graphics-,
           -Graphics-, -Graphics-, -Graphics-, -Graphics-,
           -Graphics-}
```

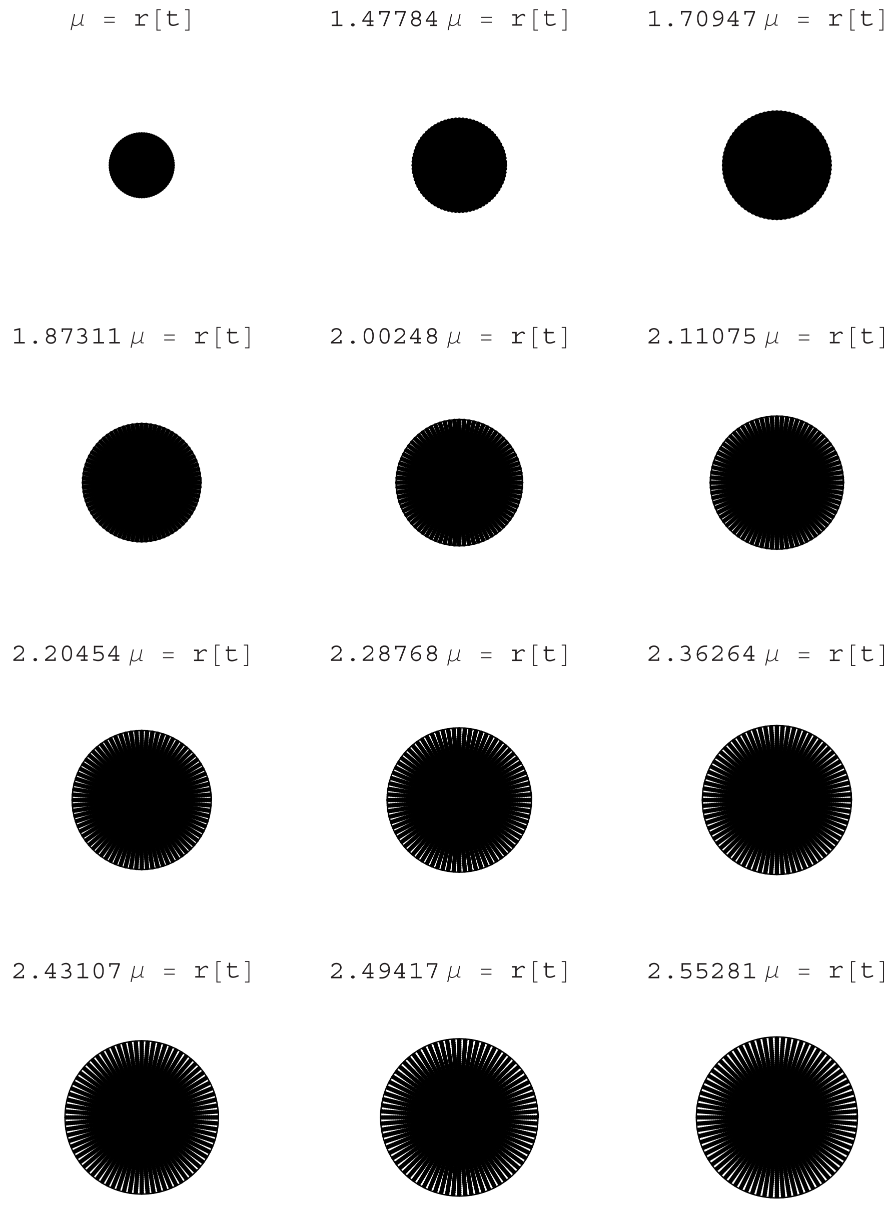
Now we can choose to examine just three of the cell structures that result, for example, 1, 10, and 20:

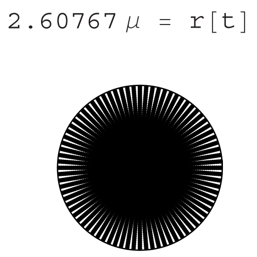
```
In[203]:= Show[GraphicsArray[{cells[[1]], cells[[10]], cells[[20]]}],
            DisplayFunction → $DisplayFunction];
```

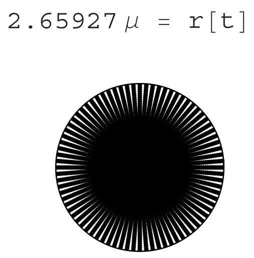


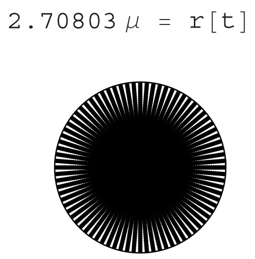
Or we can look at all of the structures in groups of three at a time by utilizing the GraphicsArray command:

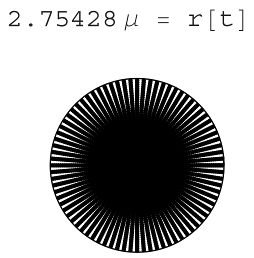
```
In[204]:= Show[GraphicsArray[Table[{cells[[n]], cells[[n + 1]],
             cells[[n + 2]]}, {n, 1, 18, 3}]]];
```

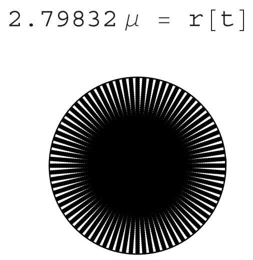

μ = r[t]
1.47784 μ = r[t]
1.70947 μ = r[t]
1.87311 μ = r[t]
2.00248 μ = r[t]
2.11075 μ = r[t]
2.20454 μ = r[t]
2.28768 μ = r[t]
2.36264 μ = r[t]
2.43107 μ = r[t]
2.49417 μ = r[t]
2.55281 μ = r[t]

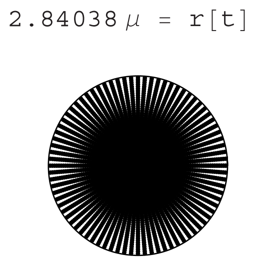












By keeping the number and area of the subcellular bodies constant we can consider these to be conserved during the simulation much as mass of salt would be in the real case. The outcome is that the "mass" is redistributed all along the interior of the cell as it expands. We have chosen to have the "mass" be redistributed along equiangular lines, which leads to an interesting pattern of distribution. In a well-mixed system the distribution would expand evenly everywhere. Yet, at the same time, natural systems do display remarkable patterns especially during growth that resemble the one we have constructed here. In fact the final outcome is reminiscent of the patterns that mollusks create during mineralization of their shells. One wonders what the underlying mechanisms of mass transfer must be in such processes, which can to lead to such "un-mixed" results!

6.5 Summary

We have covered a body of material in this chapter that deals with movement of mass along gradients and between phases. We have examined the commonalities and differences between linear driving forces, net rates of adsorption, and permeation. Each has the common feature that reaction is not involved but does involve transport between apparently well-defined regions. We move now to chemically reactive systems in anticipation of eventually analyzing problems that involve mass transfer and reaction.

CHAPTER 7

Reacting Systems—Kinetics and Batch Reactors

Chemical kinetics are at the heart of industrial chemistry and hence chemical engineering. In addition to being a fascinating subject worthy of scientific inquiry in its own right, *chemical kinetics are the quantitative description of chemically reacting systems*. The concept of *rate* in the context of a chemical reactor is the central issue in chemicals production. The mathematics of kinetics is crucial to such essential tasks as calculating the size and type of reactor that is needed to meet a defined target of production, but also for the prediction of which species, wanted or unwanted, will emerge from that reactor. Although the cost of the reactor is typically small as a percentage of the total cost of an overall production facility, the chemical events that occur within it dictate how much of the theoretical profit associated with a chemical reaction can be captured rather than being surrendered back as costs of manufacturing.

The reason for this is simple. If the reaction chemistry is not "clean" (meaning selective), then the desired species must be separated from the matrix of products that are formed and that is costly. In fact the major cost in most chemical operations is the cost of separating the raw product mixture in a way that provides the desired product at requisite purity. The cost of this step scales with the complexity of the "un-mixing" process and the amount of energy that must be added to make this happen. For example, the heating and cooling costs that go with distillation are high and are to be minimized wherever possible. The complexity of the separation is a function of the number and type of species in the product stream, which is a direct result of what happened within the reactor. Thus the separations are costly and they depend upon the reaction chemistry and how it proceeds in the reactor. All of the complexity is summarized in the kinetics.

Can we predict these costs beforehand? If a company is considering committing capital to a new project, then in order to determine if that capital investment would be a wise one, that is, one that would meet expected rates of return and would be superior to placing the capital in other investments or projects, modeling of the new process must be done to calculate the expected costs of production. This modeling must begin with kinetics.

These are the chemical engineering motivations for studying kinetics. A physical chemist might look at these, then stand back and say there are many other reasons to study kinetics and molecular dynamics that are quite separate and distinct from the industrial production of chemical materials. In contrast to chemical engineers, chemical scientists examining kinetics or chemical dynamics are often investigating simple chemical systems that involve only one type of molecule. However, the processes that they are examining may be detailed and complex, and may, for example, take place entirely within the molecule rather than between molecules. Again the descriptions are made in the context of kinetics. Because of this there is a seamlessness in chemical kinetics from the very applied to the esoteric.

Finally, how do thermodynamics fit in with kinetic descriptions of chemical reactions? Thermodynamics provides information at equilibrium. Yet many chemical reactions take place within chemical reactors and never reach equilibrium. Although some reactions do move to equilibrium quickly, many of industrial interest do not. Instead, these reactions must often be pushed and pushed hard with high temperature and pressure to the product side. Typically, a catalyst must be used to make the reaction go in economic yields, at acceptable costs, and within a reasonable time frame. (The catalyst is a device that is not consumed by the reaction but lowers the temperature required to make a reaction take place. The extent of reaction, however, is dictated by the equilibrium thermodynamics. The catalyst only accelerates the rate to equilibrium.) As powerful as thermodynamics is, it does **not** provide any information about how fast or how slow will be the rate of approach to equilibrium. The relationship of thermodynamics to chemical process engineering is like that of having an itinerary for travel between two cities. This itinerary tells us how far apart each city is but provides no information on the terrain that lies between them. We have no way to estimate what kind of trip it would be, or what kind of vehicle would be best to use to make the trip. This is analogous to the situation we find ourselves in when we have chemical thermodynamics information but are completely lacking kinetics. To drive the analogy a bit further, imagine that you were asked if one could operate a profit-making business by moving clients between the two cities and all you knew was how far they were apart!

7.1 How Chemical Reactions Take Place

We know from elementary chemistry that reactions take place when molecules collide with one another. We also know that reactions often take place faster at higher temperature and that a catalyst often improves the rate further. Enzymes are the prototypical natural catalysts and they work by orienting molecules along specific directions that are preferred for reaction.

We can say then that reactions have strong temperature and orientational dependence, and that collisions alone are not enough for reaction to take place.

We can get a quantitative sense for this by turning once again to the kinetic theory of gases to compute the number of collisions that take place per unit volume and time at fixed temperature and pressure. The collision number Zab between two molecules A and B is given as:

$$\begin{aligned}\mathrm{Zab} &= \pi\frac{(\mathrm{da}+\mathrm{db})^2}{4}\sqrt{\frac{8RT}{\pi\mu}}\,\frac{\mathrm{Na\,Nb\,Nav}^2}{V^2}\\ &= \pi\frac{(\mathrm{da}+\mathrm{db})^2}{4}\sqrt{\frac{8RT}{\pi\mu}}\,\frac{\mathrm{Pa\,Pb}}{(RT)^2}\\ &= A_c\bar{v}C_aC_b\end{aligned}$$

where R = ideal gas constant, [j/mol-K]

T = absolute temperature, [K]

$\mu = \dfrac{\mathrm{Ma\,Mb}}{(\mathrm{Ma}+\mathrm{Mb})}$, mean molecular weight

$\bar{v} = \sqrt{\dfrac{8RT}{\pi\mu}}$, mean molecular speed

M = molecular weight, [g/mol]

$A_c = \dfrac{\pi(\mathrm{da}+\mathrm{db})^2}{4}$, [cm^2]

da, db = molecular diameters of A and B, [cm]

Na, Nb = [number]

V = volume, [cm^3]

Nav = Avogadro's number, [number per mol]

This is the product of the molecular cross section at collision, the mean speed, and the product of the number concentrations of A and B. We can compute this value for standard conditions:

```
In[1]:= da = 2.5 10^-8 cm;
        db = 3 10^-8 cm;
        R1 = 8.314 10^7 g cm^2/(s^2 mol K);
        T = 300 K;
        μ = 30 g/mol;
        Pa = .8 atm;
        Pb = .2 atm;
```

```
R2 = 82.05 cm^3 atm/(mol K);
Nav = 6.02 10^23 1/mol;
Zab == PowerExpand[π (da + db)^2/4 Sqrt[8 R1 T/(π μ)] Pa Pb Nav^2/(R2 T)^2]
meanspeed == Sqrt[8 R1 T/(π μ)]
Out[10]= Zab == 1.04617 × 10^28/(cm^3 s)
Out[11]= meanspeed == 46012.4 Sqrt[cm^2/s^2]
```

This shows that we have O (10^{28}) collisions per cm^3 per second between molecules such as oxygen and nitrogen in air at room temperature. Yet we know that these do not react under these conditions even though oxidation of nitrogen can lead to formation of nitrogen oxides. Also, a rule of thumb for chemical reactions is that every 10-degree rise in temperature leads to a doubling of the rate of reaction. We can see by inspection that the rate of collisions does not rise in this way with temperature. Also, if we were to convert Zab into moles of collisions per unit volume per unit time, it would be on the order of 16,000 moles per cm^3 per second! Clearly, there is much more to chemical reaction kinetics than simply collisions.

From basic chemistry we know that for reaction to take place the energy of the collision must be above a threshold value and the molecules must be oriented properly. Reaction of real molecules is much more complex than what one would expect from the collisions of hard spheres. Molecules have shape and reactive regions and bonds that usually are broken in order for reaction to take place. At the beginning of the twentieth century Arrhenius articulated this in a simple, yet elegant mathematical statement for the rate constant:

$$k[T] = A e^{\frac{-Ea}{RT}}$$

Here, A is a pre-exponential factor—the A factor—that accounts for geometric effects, while the temperature dependence is accounted for in the exponential. The term Ea is the activation energy, the threshold that must be surmounted for a collision to lead to reaction. We can rewrite this statement in terms of the reaction temperature $T_{\mathrm{rxn}} = \frac{\mathrm{Ea}}{R}$.

$$k[T] = A e^{\frac{-\mathrm{Trxn}}{T}}$$

If we take the threshold energy to be 40,000 cal/mol, and given R = 1.98 $\sim 2\frac{\mathrm{cal}}{\mathrm{mol}\,K}$, then the threshold reaction temperature is ~20,000 K. At this threshold temperature for reaction we can see that adding 10 degrees to an initial temperature will indeed double the rate constant and, if all else is the same, the rate of reaction. We also can see that if the threshold temperature is

40,000 K, then a 10-degree temperature rise will nearly quintuple the rate, whereas at 10,000 K the rate is barely raised by 1.5 times its initial value:

```
In[12]:= T2 = T1 + 10;
         T1 = 500;
         Trxn = 20000;
```

$$\texttt{Solve[\{}\frac{k1}{k2} == \frac{e^{\frac{-Trxn}{T1}}}{e^{\frac{-Trxn}{T2}}}\texttt{\}, k2] // N}$$

We have not discussed the issue of the dimensions of the rate constant. The reason is that the dimensions change with the change in the rate dependence upon concentration. Hence we have postponed consideration of dimensions until we reach that point.

Reactions take place in a localized region of space, that is, a system defined by a control volume. The control volume can be real or abstract such as a cell or organelle, or a region of an organelle. They can be macrosized such as a reactor or abstractly macrosized as in the case of the reactions that take place within the nucleus of a star. We choose the control volume according to the dictates of the analysis that we are undertaking. The control volume should be one phase or it may be abstract and treat more than one phase as if it would behave as a single phase.

We will begin with the case of the batch reactor. In this case the vessel defines the control volume. We will move to systems with flow in and both flow in and out. The former is the case of semibatch operation while the latter will be treated as the continuous stirred tank reactor (CSTR) and the plug flow reactor (PFR). All the chemical kinetics that we will need can be introduced within the context of these four different kinds of reactors.

7.2 No-Flow/Batch System

We know from the conservation of mass that when we run a reaction in a batch reactor the mass of products must be equal to the mass of reactants so long as nothing has escaped from the reactor. This holds absolutely and is independent of the chemical reaction type, mechanism, or stoichiometry. All that chemical reactions do is to rearrange the atoms and mass in the molecules. In essence the labels on the mass change but that is all. If the reaction in solution leads to a gas such as the reaction of baking soda with vinegar water (that is, sodium bicarbonate with dilute acetic acid), then a mass change can take place because one of the products is a gas and can escape the vessel:

$$Na_2CO_3 + 2CH_3COOH \rightarrow 2Na(CH_3COO) + CO_2 \uparrow + H_2O$$

On the other hand, if a provision is made to trap the carbon dioxide, say, with a balloon placed over the mouth of the vessel, then the mass of sodium acetate, water, and carbon dioxide will be equal to that of the original sodium carbonate and acetic acid. This is the consequence of the conservation of mass and nothing more.

What is the proper expression for a batch reactor? We know that the total mass balance will be equal to zero based on the conservation of mass and the assumption that nothing escapes the vessel. This means that the net accumulation will be zero:

$$\frac{d\rho V}{dt} = 0$$

If the net accumulation is zero for the overall mass in the vessel, then what can we say about the component balances? We know that the reaction proceeds forward to completion and in so doing the concentrations of the species change with time. Therefore even though the overall mass does not change, the mass of any given species or component does change and this is what we measure. Consider the reaction:

$$A + B \rightarrow D + E$$

This is simple stoichiometrically and we can assume that it is irreversible, which means that reaction proceeds to the right-hand side completely and that product **D** does not return to **A** and **B**. The *component mass balances* become:

$$\frac{d\,Ca\,V}{dt} = \frac{d\,Cb\,V}{dt} = -r_a V = -r_b V$$
$$\frac{d\,Cd\,V}{dt} = \frac{d\,Ce\,V}{dt} = r_d V = r_e V$$
$$r_a = r_b = -r_d = -r_e = r$$

The reactants are considered to be decreasing in concentration, and the products are increasing in concentration. Thus, the rates of the reactants are taken to be negative and the product rate is positive. The important point is that the rates are oppositely signed as is shown by the last expression. If the reaction does not produce a change in volume, then the control volume does not change, and the volume term is a constant that can be cancelled across all the equations:

$$\frac{d\,Ca}{dt} = \frac{d\,Cb}{dt} = -r$$
$$\frac{d\,Cd}{dt} = \frac{d\,Ce}{dt} = r$$

The next step is to find kinetics for the rate of reaction that can be used as a constitutive relationship to replace the rate **r** on the right-hand side. When we say that we look for a constitutive relationship for the kinetics to replace the right-hand side, we are seeking to make the equation *autonomous*. This means that we are seeking a function for the *RHS* that is explicit in the concentration of one or more of the components.

7.3 Simple Irreversible Reactions—Zeroth to *N*th Order

First-Order Kinetics

The simplest case to consider by far is that of first-order or linear kinetics in a constant volume batch reactor. If the rate of reaction is directly proportional to the rate of the reaction, then we call this the first order in the concentration of reactant, and the right-hand side becomes:

$$r = k\,Ca$$
$$\frac{d\,Ca}{dt} = -k\,Ca$$
$$Ca(t) = Cao\, e^{-kt}$$

This is an expression for the rate of decay of the concentration of species **A**. (It should remind us of the expression we derived for the change in level of the draining tank for which we used a linear constitutive relationship between level and rate of flow.) The dimensions of **k** in this case are reciprocal time, that is, $\mathbf{sec^{-1}}$ or $\mathbf{min^{-1}}$ etc. The reason for this is that the rate of reaction is given in dimensions of $\frac{\mathbf{moles}}{\mathbf{volume\ time}}$. Therefore to be dimensionally consistent the first-order rate constant must be in dimensions of inverse time.

As the stoichiometry for the rate of reaction of component **B** is the same we can show that:

$$\frac{d\,Ca}{dt} = \frac{d\,Cb}{dt}$$
$$Ca - Cao = Cb - Cbo$$
$$Cb(t) = Ca(t) - (Cao + Cbo)$$

From which we find:

$$\begin{aligned} Cb(t) &= Cao e^{-kt} - Cao + Cbo \\ &= Cao[e^{-kt} - 1] + Cbo \end{aligned}$$

In a similar fashion we find that the rate of appearance of component **D** is:

$$\begin{aligned}
\frac{d\,Ca}{dt} &= -\frac{d\,Cd}{dt} \\
Ca - Cao &= -(Cd - Cdo) \\
Cd(t) &= (Cao + Cdo) - Ca(t) \\
&= (Cao + Cdo) - Cao\,e^{-kt} \\
&= Cao[1 - e^{-kt}] + Cdo
\end{aligned}$$

The change in concentration of component **E** at any time would follow the same form with the substitution of **Ceo** for **Cdo**.

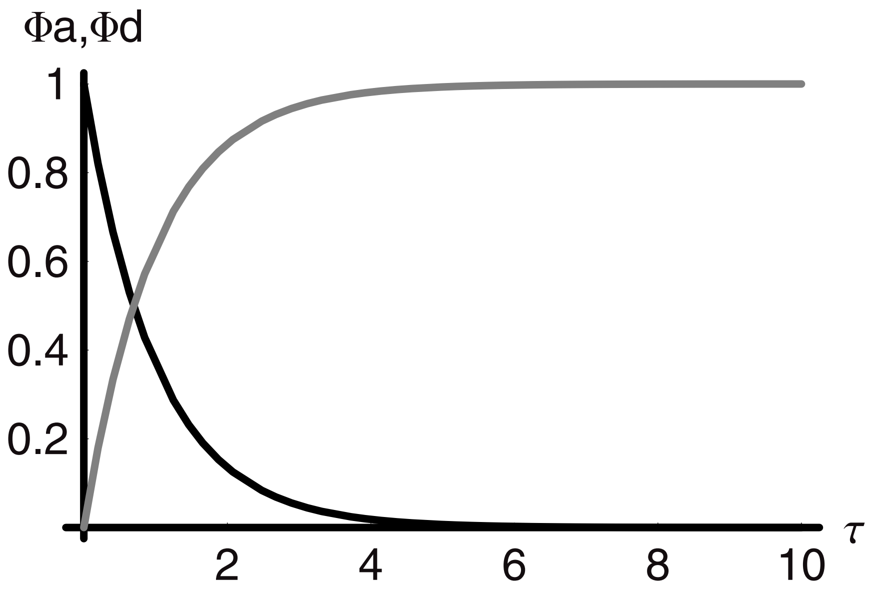
We can now plot these concentration functions so that we can see how a reaction system of this kind would behave in time. To do this we will assume that the concentrations of the products are both zero at time zero and that the initial concentrations of the two reactants are both equal. Also, in order to make the behavior general, we will plot the change in the *ratio of the concentrations of the reactants to an initial concentration* of one of the reactants **Cao**. We can go one step further and normalize the time coordinate with the "inverse reaction time." What is that in this case? Well, for a first-order reaction rate constant, its dimension is the reciprocal of time, that is, inverse time. Thus, in essence for the first-order case, the rate constant is the inverse of the characteristic time for the chemical reaction. Therefore if we multiply the rate constant **k** by real time **t** the result is dimensionless time, which we shall refer to as τ. In fact we already had this result in hand. Look back at the expression for the change in concentration of **A** with time. We notice that the *RHS* has an exponential term, the argument of which is the product **k t**. Because the exponential is a transcendental function, such as sine, cosine, etc., the argument must be a pure number that is dimensionless. Thus the solution of the differential equation that leads to this result naturally generates the dimensionless time τ simply as an outcome of the solution procedure.

Therefore, what we plan to plot will be **Ca(τ)/Cao**, **Cb(τ)/Cao**, **Cd(τ)/Cdo**, and **Ce(τ)/Ceo** against the dimensionless time τ. We can use *Mathematica* to do this, the beauty of which is that we can let the product **kt** $= \tau$ vary as natural numbers without actually assigning a specific value to **k**, and for the same reason the concentrations will vary as natural numbers between zero and unity. To emphasize the nondimensional nature of the concentrations, we can introduce a new variable, namely, the Greek letter **Φ** for dimensionless concentrations. When the initial concentration of **B** is divided by that of **A**, we will call this **Φbo**, and likewise for the other two species. The new expressions in dimensionless form will be:

$$\begin{aligned}
\Phi a &= e^{-\tau} \\
\Phi b &= [e^{-\tau} - 1] + \Phi bo \\
\Phi d &= [1 - e^{-\tau}] - \Phi do \\
\Phi e &= [1 - e^{-\tau}] - \Phi eo
\end{aligned}$$

As the initial concentrations of the products **D** and **E** are taken to be zero, the corresponding dimensionless initial concentrations are also zero. Also, we can see that **Φd** = **Φe**. Thus we will examine only **Φd**, and if **Φbo** = **1**, then **Φa** = **Φb** and we need only consider **Φa**:

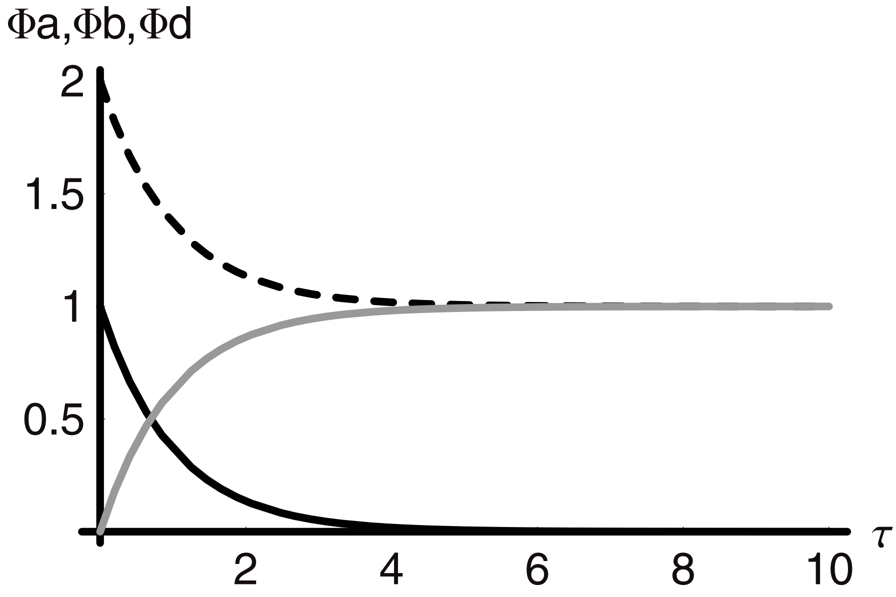
```
In[12]:= SetOptions[{Plot, ListPlot},
           AxesStyle → {Thickness[0.01]},
           PlotStyle → {PointSize[0.015], Thickness[0.006]},
           DefaultFont → {"Helvetica", 17}];

In[13]:= firstordpl1 = Plot[
             {N[Exp[-τ]], N[(1 - Exp[-τ])]}, {τ, 0, 10},
             AxesLabel → {"τ","Φa,Φd"},
             PlotStyle → {{Thickness[.01], Dashing[{0, 0}]},
              {Thickness[.01], GrayLevel[0.5]}}];
```



The plots show what we would expect, that is, the concentration of **A** diminishes exponentially along with **B** while the concentrations of **D** and **E** grow exponentially to their final value, which is the same as that of the initial concentrations of **A** and **B**.

What would be the result if the concentration of **B** were initially twice that of **A**? We can find this result by setting **Φbo** = 2 and plotting the results as we did before:

```
In[14]:= firstordpl2 = Plot[{
             N[Exp[-τ]], N[Exp[-τ] - 1 + 2], N[(1 - Exp[-τ])]},
             {τ, 0, 10},
             AxesLabel → {"τ", "Φa,Φb,Φd"},
PlotStyle → {{Thickness[0.01], Dashing[{0, 0}]},
  {Thickness[0.01], Dashing[{0.025, 0.025}]},
  {Thickness[0.01], GrayLevel[0.6]}}];
```



Now we note that the Y-axis for dimensionless concentration varies from 0 to 2, and the concentration of **B** drops from an initial value of 2 to a final value of 1. This indicates that only half of the original concentration of **B** would be used to produce **C** and **D**, even though all of **A** would have been consumed. In this case we see that component **A** is the "limiting reagent." If we were to make the concentration of **B** initially 100 times that of **A**, we would find that the concentration of **B** would move from 100 to 99, and would be virtually "unchanged" in the process.

Under such conditions, unless our measurements on the concentration of **B** were very accurate, that is, accurate enough to pick up a change this small, ~1% in concentration, we might find that the variation in **B** would be undetectable, which is smaller than our experimental error. If this were to happen, then we would think that the rate of reaction did not depend upon the concentration of component **B**, and in fact under conditions such as these, that would be a good working conclusion. However, it must strike us as odd that if **B** is not present, then the reaction to **C** and **D** from **A** will not take place, and yet we find little rate dependence on **B**.

Perhaps this is what happened when the data were analyzed for this reaction and that is why the kinetics we have used are first order in **A** and "zero order" in **B**, that is, independent of **B**. Maybe this reaction rate only appears to be first order in **A** at the conditions under which the experiment was run, when in fact it is really first order in **A** and in **B**. If this should prove to be the case, then the first-order rate expression for the reaction of **A** and **B** to give **D** and **E** is actually *pseudo-first order*, rather than true first order. A pseudo-first-order reaction may really be second order when we analyze the data and plan the experiments more carefully.

In other words, the second-order rate expression, which is first order in **A** and **B**, may appear to be first order in **A** only if the experiments were done with a large excess of **B** present and its change in concentration went undetected! Let us see how this works out in the next section, by considering second-order kinetics for the same reaction.

Second-Order Kinetics Overall

For the reaction of **A** and **B** to produce **C** and **D**, it is more likely that the kinetics would be second order overall, with first order in the concentration of both **A** and **B** rather than just first order in **A**. If this were the case, then the solution would be different than that which we found in the foregoing and would be derived as follows:

$$\mathrm{r} = \mathrm{k\,Ca\,Cb}$$
$$\frac{\mathrm{d\,Ca}}{\mathrm{dt}} = -\mathrm{k\,Ca\,Cb}$$

This cannot be solved as written; we need an expression for **Cb** in terms of **Ca** in order to substitute and do the integration. This can be obtained by going back to the stoichiometric statement:

$$\begin{aligned}\frac{\mathrm{d\,Ca}}{\mathrm{dt}} &= \frac{\mathrm{d\,Cb}}{\mathrm{dt}}\\ \mathrm{Ca} - \mathrm{Cao} &= \mathrm{Cb} - \mathrm{Cbo}\\ \mathrm{Cb} &= \mathrm{Ca} - \mathrm{Cao} + \mathrm{Cbo}\\ \mathrm{Cb} &= \mathrm{Ca} - (\mathrm{Cao} - \mathrm{Cbo})\end{aligned}$$

Now, we can substitute this expression for **Cb** in terms of **Ca** into the differential equation describing the change in **Ca**, make it autonomous, and derive an expression for the time dependence of **Ca**:

$$\begin{aligned}\frac{\mathrm{d\,Ca}}{\mathrm{dt}} &= -\mathrm{k\ Ca\ Cb}\\ &= -\mathrm{k\ Ca}[\mathrm{Ca} - (\mathrm{Cao} - \mathrm{Cbo})]\\ &= -\mathrm{k\ Ca}^2 + \mathrm{k\ Ca}(\mathrm{Cao} - \mathrm{Cbo})\\ &= -\mathrm{k}\ [\mathrm{Ca}^2 - \mathrm{Ca}(\mathrm{Cao} - \mathrm{Cbo})]\end{aligned}$$

Using *Mathematica* we have two primary choices on how to proceed with the solution to this equation—we can rearrange it into its separable components and then integrate both sides of the equation or we can solve it directly; we will do the latter.

```
In[15]:= DSolve[{Ca'[t] == -k Ca[t]^2 + k Ca[t](Cao - Cbo),
           Ca[0] == Cao}, Ca[t], t]
```

```
Out[15]= {{Ca[t] → Cao(Cao - Cbo)e^(Cao kt) / (Cao e^(Caokt) - Cbo e^(Cbo kt))}}
```

This solution is one that we would like to explore as we did with the previous solution for the first-order rate equation. We could at this point convert the solution for **Ca** into one that is "nondimensionalized," but as it stands we might make errors in doing so. In fact it would be easier to have "nondimensionalized" the equation to be solved in the first place. Therefore, instead of working on the solution, we will rework the differential equation and resolve it:

$$\frac{d\,Ca}{dt} = -k\,Ca^2 + k\,Ca(Cao - Cbo)$$

First, we will let **Φa = Ca/Cao** and **Φb = Cb/Cao** once again. In this case we can multiply through on both sides of the equation by **Cao/Cao**. In the case of the first term on the *RHS*, we will multiply by **Cao²/Cao²**. This yields:

$$Cao\frac{d\Phi a}{dt} = -k\,Cao^2\Phi a^2 + k\,Cao\,\Phi a(Cao - Cbo)$$

If we divide each side by the residual **Cao** on the *LHS*, then we find:

$$\frac{d\Phi a}{dt} = -k\,Cao\,\Phi a^2 + k\,\Phi a(Cao - Cbo)$$

What about the rate constant that appears in both terms of the equation? Can we clear this as well? We can because we have already said that **kt = τ**, for the first-order case and so this meant that **k dt = d(kt) = dτ**. But what about in this case, with second-order kinetics? Can we still do this? To understand what is happening, we need to be very mindful of what the dimensions are for each term. If we were to simply divide by **k**, and make the substitution with **dτ**, we would have:

$$\frac{d\Phi a}{d\tau} = -Cao\,\Phi a^2 + \Phi a(Cao - Cbo)$$

Look carefully at this equation because it is misleading. Our goal was to nondimensionalize it. Therefore we expect the accumulation term, that is, the *LHS*, to be dimensionless. But how can it be? The *RHS* clearly is not dimensionless, and it has units of concentration! This is the key to seeing where we went wrong; our error was in assuming that the rate constant **k** for this second-order rate expression had the same units as those used for the first-order system. It does not. *The dimensions for this second-order rate are vol/mol/tim; in other words, inverse time and inverse concentration.* Why? Because the accumulation term on the *LHS* must have the same dimensions of mol/volume/time regardless of the order or complexity of the rate expression on the *RHS*. Therefore the units of the rate constant will always be dictated by the form of the rate expression and the need for proper dimensions on the *LHS*.

Recalling the steps we took to nondimensionalize, we see that the error we made came about when we expressed the dimensionless time variable. Instead of **kt** = $\boldsymbol{\tau}$, for the second-order case we have found that **kCao t** = $\boldsymbol{\tau}$. We can group the rate constant and the initial concentration of A parenthetically to give:

$$\begin{aligned} \mathrm{k\,Cao\,t} &= \tau \\ (\mathrm{k\,Cao})\,\mathrm{t} &= \tau \\ \mathrm{k't} &= \tau \end{aligned}$$

The product of **k** and **Cao** has units of inverse time, the same as the first-order rate constant. Thus, we can identify the product (**k Cao**) as the new rate constant **k**′ and this is now a *pseudo-first-order rate constant*. Of course the choice of **Cao** was arbitrary; if we had chosen to nondimensionalize in terms of **Cbo**, then **k**′ would still be pseudo-first order, but it would be the product of **Cbo** and **k**. We can immediately see that if we had run a kinetics experiment with **B** in such great excess over **A**, its concentration change would have been undetectable, and we would observe first-order kinetics rather than second-order overall kinetics.

Let us return to the nondimensionalization of the equation. Before we replaced time **t** inconsistently with the true dimension of the second-order **k**, we had the following form of the equation:

$$\mathrm{Cao}\frac{\mathrm{d}\Phi\mathrm{a}}{\mathrm{dt}} = -\mathrm{k\,Cao}^2\Phi\mathrm{a}^2 + \mathrm{k\,Cao}\,\Phi\mathrm{a}(\mathrm{Cao}-\mathrm{Cbo})$$

By dividing both sides of this equation by (**k Cao²**), we get the result that we were seeking, which is properly dimensionless on both sides:

$$\frac{\mathrm{d}\Phi\mathrm{a}}{\mathrm{k\,Cao\,dt}} = -\Phi\mathrm{a}^2 + \Phi\mathrm{a}\frac{(\mathrm{Cao}-\mathrm{Cbo})}{\mathrm{Cao}}$$

The last step we take is to recognize that (**k Cao dt**) is the same as **d**$\boldsymbol{\tau}$, rendering the equation as:

$$\frac{\mathrm{d}\Phi\mathrm{a}}{\mathrm{d}\tau} = -\Phi\mathrm{a}^2 + \Phi\mathrm{a}\frac{(\mathrm{Cao}-\mathrm{Cbo})}{\mathrm{Cao}}$$

Following the procedure taken in the latter half of the last section, we can now experiment with the behavior of this equation by solving it and plotting the dimensionless results. To do so let **M** replace the ratio of initial concentrations:

```
In[16]:= DSolve[{Φa'[τ] == -Φa[τ]^2 + MΦa[τ], Φa[0] == Φa0},
          Φa[τ], τ]
```

Out[16]= $\{\{\Phi_a[\tau] \rightarrow \frac{e^{M\tau} M\Phi_{a0}}{M - \Phi_{a0} + e^{M\tau}\Phi_{a0}}\}\}$

Thus, we find that:

$$\Phi_a[\tau] \rightarrow \frac{e^{M\tau} M\Phi_{a0}}{M - \Phi_{a0} + e^{M\tau}\Phi_{a0}}$$

This can be plotted to determine how the dimensionless concentration of **A** changes with time. We know that the concentration change for **B** will follow that of **A** based on the stoichiometry. The change in concentration of the products **D** and **E** will also track each other; thus we need only solve for one. To solve for the change in the dimensionless concentration of **D**, we recall that:

$$\frac{d\,Ca}{dt} = -\frac{d\,Cd}{dt}$$
$$Ca - Cao = Cdo - Cd$$
$$\therefore Cd\ (t) = Cdo - Ca\ (t) + Cao$$

If we divide every term by **Cao** to render this expression dimensionless, we find:

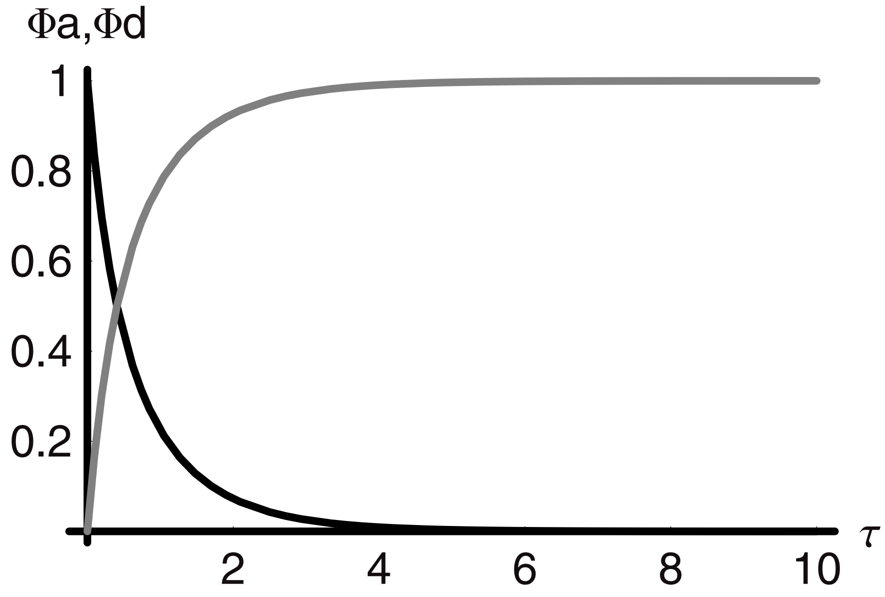
$$\Phi_d(\tau) = \Phi_{d0} - \Phi_a(\tau) + 1$$

Taking the initial concentration of **D** as zero and replacing for **Φa(τ)** we have:

$$\Phi_d(\tau) = 1 - \frac{e^{M\tau} M\Phi_{a0}}{M - \Phi_{a0} + e^{M\tau}\Phi_{a0}}$$

These two equations can now be plotted as shown in the following graph to determine their behavior after we assign initial values to **Φao**, **Cao**, **Cbo** and to **M**. The simplest case is that of **Φao** = 1, and **2Cao** = **Cbo** making **M** = − 1:

```
In[17]:= Φa0 = 1;
         M = -1;
         secordpl1 = Plot[{N[(e^(Mτ) MΦa0)/(M - Φa0 + e^(Mτ) Φa0)],
                      N[1 - (e^(Mτ) MΦa0)/(M - Φa0 + e^(Mτ) Φa0)]}, {τ, 0, 10},
           AxesLabel → {"τ", "Φa, Φd"},
        PlotStyle → {{Thickness[.01], Dashing[{0, 0}]},
                      {Thickness[.01], GrayLevel[0.5]}}];
```



What happens if **Cbo** is equal to **Cao**? Then we find that **M** = 0, and any of the second-order solutions that we have just derived become zero, or, in other words, meaningless! Why? The reason is that when we solved these equations either in regular or dimensionless form, we implicitly assumed that **Cao** ≠ **Cbo**. If they are equal, then the equation changes and we obtain:

$$\begin{aligned}\frac{d\,Ca}{dt} &= -k\,Ca\,Cb \\ &= -k\,Ca^2 + k\,Ca(Cao - Cbo) \\ &= -k\,Ca^2\end{aligned}$$

or in dimensionless form:

$$\begin{aligned}\frac{d\Phi a}{d\tau} &= -\Phi a^2 + \Phi a\frac{(Cao - Cbo)}{Cao} \\ \frac{d\Phi a}{d\tau} &= -\Phi a^2\end{aligned}$$

Working with the dimensionless form we have:

```
In[20]:= Φa0 =.
         DSolve[{Φa'[τ] == -Φa[τ]^2, Φa[0] == Φa0}, Φa[τ], τ]
```

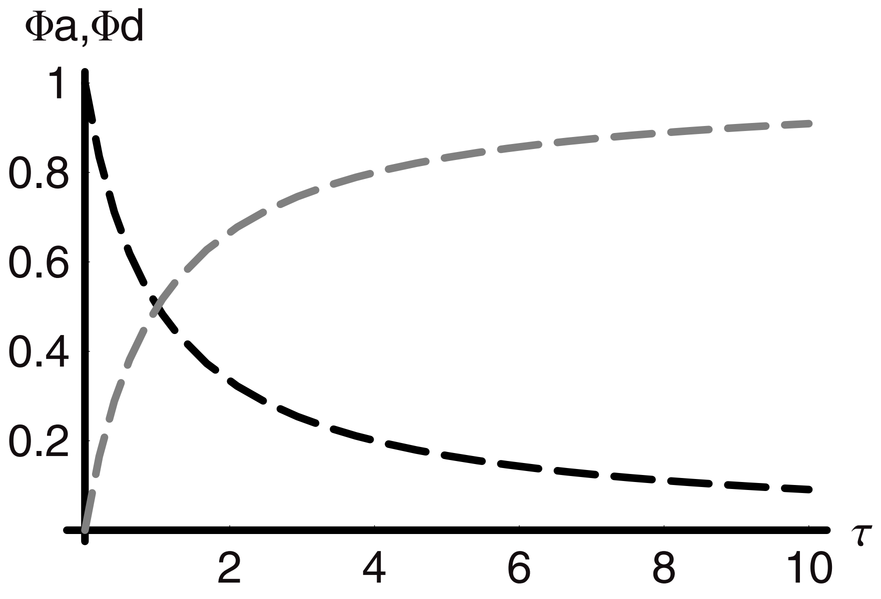
```
Out[21]= {{Φa[τ] → Φa0/(1 + τΦa0)}}
```

By the same stoichiometric relationship between **D** and **A**, but using this new solution for dimensionless **A** concentration we find:

$$\begin{aligned}\Phi_d(\tau) &= \Phi_{d0} - \Phi_a(\tau) + 1\\ &= \Phi_{d0} - \frac{\Phi_{a0}}{1+\tau\Phi_{a0}} + 1\end{aligned}$$

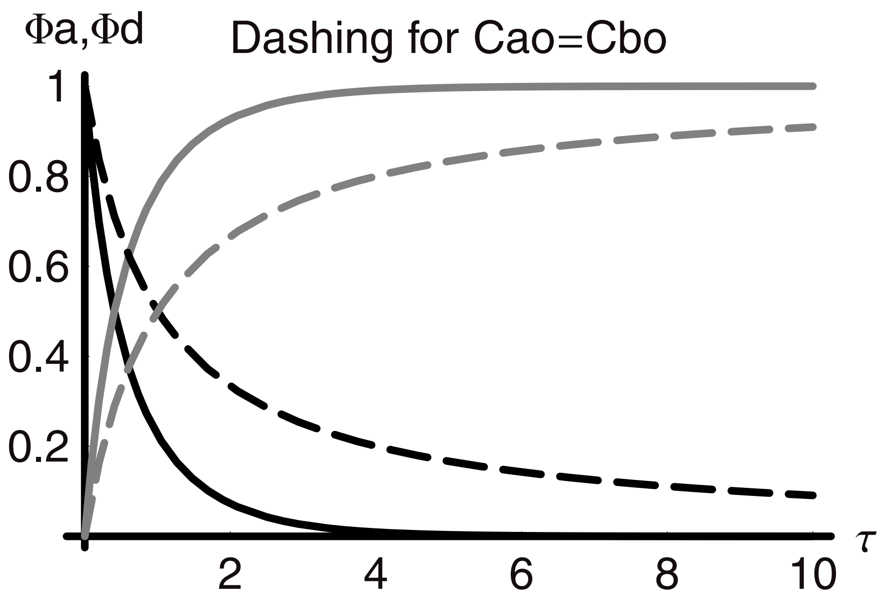
Using the same procedure as we used before we can solve for and plot the new solution subject to the condition that **Cao** = **Cbo** and we obtain:

```
In[22]:= Φa0 = 1;
         Φd0 = 0;

         secordpl2 = Plot[
           {N[Φa0/(1 + τΦa0)], N[Φd0 - Φa0/(1 + τΦa0) + 1]}, {τ, 0, 10},
           AxesLabel → {"τ", "Φa, Φd"},
           PlotStyle → {{Thickness[.01], Dashing[{0.05, 0.025}]},
             {Thickness[.01], Dashing[{0.05, 0.025}],
              GrayLevel[0.5]}}];
```



This set of two solutions can be compared to the set of two solutions that we obtained earlier with **Cao** ≠ **Cbo**.

```
In[25]:= Show[secordpl1, secordpl2,
           PlotLabel → "Dashing for Cao=Cbo"];
```



The results show that the solutions obtained when **Cao** ≠ **Cbo** are sharper and more steeply rising and falling than the corresponding solutions when **Cao** = **Cbo**.

Given the form of the rate expression with **k Ca²**, it is natural to wonder if it uniquely applies to the situation we just analyzed. The answer is that it does not. The solution we derived for the case of **A** reacting with **B** and with equal initial concentrations to produce **D** and **E** is also a description of the similar case in which **A** reacts with itself to give **D**:

$$\mathbf{A} \rightarrow \mathbf{D}$$

If this reaction happens to follow second-order kinetics and for every mole of **A** reacted we get one mole of **D**, then the resultant analysis will lead to the same result we have just seen:

$$\frac{d\,\mathrm{Ca}}{dt} = -\frac{d\,\mathrm{Cd}}{dt}$$

$$\mathrm{Ca} - \mathrm{Cao} = \mathrm{Cdo} - \mathrm{Cd}$$

$$\mathrm{Cd} = \mathrm{Cdo} + \mathrm{Cao} - \mathrm{Ca}$$

$$\frac{d\,\mathrm{Ca}}{dt} = -\mathrm{k}\,\mathrm{Ca}^2$$

$$\mathrm{Ca}[t] = \frac{\mathrm{Cao}}{1 + \mathrm{Cao}\,\mathrm{k}\,t}$$

and

$$\mathrm{Cd[t]} = \mathrm{Cdo} + \mathrm{Cao}\left(1 - \frac{1}{1 + \mathrm{Cao\,k\,t}}\right)$$

These are just the same expressions we had already obtained for the second-order case in which two different species were involved, and they had the same initial concentrations.

*N*th *Order*

The order of a reaction may not be as simple as first or second order. We often find nonintegral order in what is called "power-law" kinetics. This typically indicates that the "reaction" rate we have measured is not for a single reaction, which is one elementary step, but for several elementary steps taking place simultaneously, the sum of which is the overall reaction that we observe. Normally, we refer to rate expressions such as these as *global rates* or *kinetics* (global in the sense of overall or measurable as opposed to *intrinsic* or *fundamental rates* and *kinetics*). Consider the reaction of **A** to **B**:

$$\mathrm{A} \rightarrow \mathrm{B}$$
$$r_{A-} = k\,C_A^n$$
$$\frac{dC_A}{dt} = -k\,C_A^n$$
$$\frac{dC_B}{dt} = +k\,C_A^n$$

When we nondimensionalize, these become:

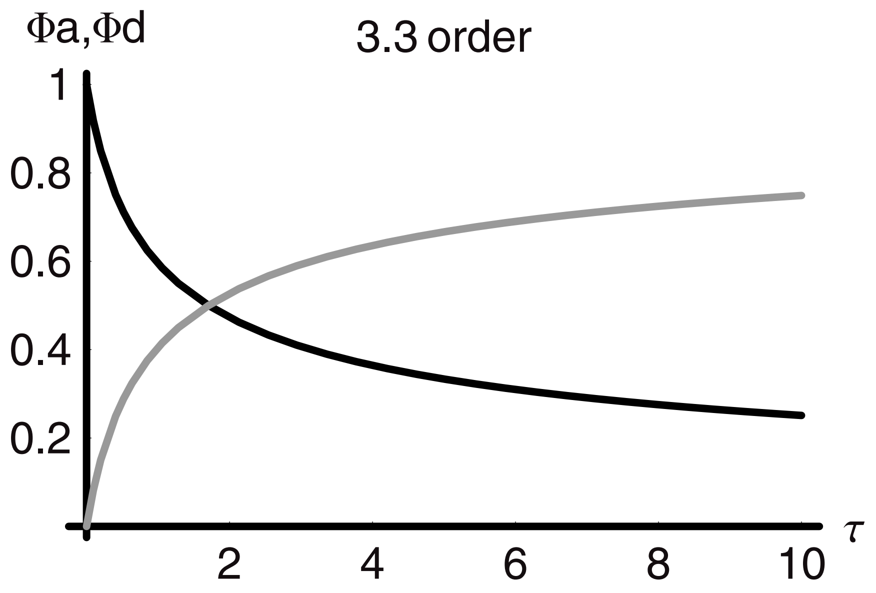
$$\frac{d\Phi_A}{d\tau} = -\Phi_A^n$$
$$\frac{d\Phi_B}{d\tau} = +\Phi_A^n$$

Solving for the concentrations of **A** and **B** we find:

```
In[26]:= Remove[Ca, ca, solnord]

In[27]:= solnordA = Simplify[
          DSolve[
           {Ca'[t] == -k Ca[t]^n, Ca[0] == Cao},
           {Ca[t]}, t]
              ]
         ca[t_] := solnordA[[1, 2]]
```

```
          solnordB = Simplify[
           DSolve[
            {Cb'[t] == +kca[t]^n, Cb[0] == Cbo},
            {Cb[t]}, t]
               ]
          cb[t_] := solnordB[[1, 2]]

          Solve::ifun : Inverse functions are being used by Solve,
            so some solutions may not be found.

          Solve::ifun : Inverse functions are being used by Solve,
            so some solutions may not be found.

                         1                  1
Out[27]= {Ca[t] → (( ───)^(-1+n) + k(-1 + n)t  ───}
                        Cao                1-n

          General::spell1 : Possible spelling error: new symbol
            name "solnordB" is similar to existing symbol
            "solnordA".

                                              1                            1
Out[29]= {{Cb[t] → Cbo - k(-1 + n)t((( ───)^(-1+n) + k(-1 + n)t ───)^n
                                             Cao                          1-n

              1               1          1              1
           + ( ───)^(-1+n) ((((  ───)^(-1+n)) ───)^n - ((( ───)^(-1+n)
              Cao            Cao        1-n            Cao

                          1
           + k(-1 + n)t ───)^n)}}
                         1-n

In[31]:= Clear["Global`*"]

In[32]:= nda = DSolve[{Φa'[τ] == -Φa[τ]^n, Φa[0] == Φao},
           Φa[τ], τ]

           ΦA[τ_] := nda[[1]][[2]]

           ΦA[τ]

           ndb = Simplify[
            DSolve[
              {Φb'[τ] == ΦA[τ]^n, Φb[0] == 0}, Φb[τ], τ]
          ]

          ΦB[τ_] := ndb[[1, 1, 2]]

          ΦB[τ]

          General::spell1 : Possible spelling error: new symbol
           name "Φa" is similar to existing symbol "Φ".

          General::spell1 : Possible spelling error: new symbol
           name "Φao" is similar to existing symbol "Φa".

          Solve::ifun : Inverse functions are being used by Solve,
           so some solutions may not be found.
          Solve::ifun : Inverse functions are being used by Solve,
           so some solutions may not be found.

                                       1    -1+n        1    -1+n
                                     (---)           n(---)
                                      Φao               Φao               1
Out[32]= {Φa[τ] → (-τ + nτ -  --------------- + ---------------)  ---  }
                                   -1 + n            -1 + n              1-n

          General::spell : Possible spelling error: new symbol
           name "ΦA" is similar to existing symbols {Φ, Φa}.

                         1    -1+n        1    -1+n
                       (---)           n(---)
                        Φao               Φao               1
Out[34]= (-τ + nτ -  --------------- + ---------------)  ---
                         -1 + n            -1 + n          1-n

          General::spell : Possible spelling error: new symbol
           name "Φb" is similar to existing symbols {Φ, Φa}.

                                                      1    -1+n    1
Out[35]= {{Φb[τ] → -(-1 + n)τ (((-1 + n)τ + (---)      )  ---  )^n
                                                     Φao          1-n

                                    1    -1+n    1
           + (-(((-1 + n)τ + (---)      )  ---  )^n
                                   Φao          1-n

                     1    -1+n    1             1    -1+n
           + ((( ---)      )  ---  )^n) ( ---)      }}
                    Φao          1-n           Φao

          General::spell : Possible spelling error: new symbol
           name "ΦB" is similar to existing symbols {Φ, ΦA, Φb}.

                                         1    -1+n    1
Out[37]= -(-1 + n)τ (((-1 + n)τ + (---)      )  ---  )^n
                                        Φao          1-n

                                  1    -1+n    1
          + (-(((-1 + n)τ + (---)      )  ---  )^n
                                 Φao          1-n

                    1    -1+n    1             1    -1+n
          + ((( ---)      )  ---  )^n) ( ---)
                   Φao          1-n           Φao
```

```
In[38]:= SetOptions[{Plot, ListPlot},
           AxesStyle → {Thickness[0.01]},
           PlotStyle → {PointSize[0.015], Thickness[0.006]},
           DefaultFont → {"Helvetica", 17}];

In[39]:= n = 3.3;
         Φao = 1;
         Plot[{ΦA[τ], ΦB[τ]}, {τ, 0, 10},
           AxesLabel → {"τ", "Φa,Φd"},
           PlotStyle → {
            {Thickness[0.01], GrayLevel[0]},
            {Thickness[0.01], GrayLevel[0.6]}},
           PlotLabel → n "order"];
```



7.4 Reversible Reactions—Chemical Equilibrium

Chemical reactions do not move in the forward direction only but in either direction and come to a "resting" point of concentrations known as the position of chemical equilibrium. Our goal in this section is to understand how we analyze such a common situation and at the same time to discover the interrelationships between kinetics and thermodynamics as they apply to chemical systems.

Take as a starting point the simplest most, reversible reaction:

$$A = B$$

This is "simple" because the stoichiometry is one mole of reactant goes to one mole of product, and because the conversion of **A** to **B** follows first-order kinetics, as does the conversion of **B** back to **A**. Thus, when we assemble the two-component mass balance equations in a constant volume batch reactor, we find:

$$\frac{d\,Ca}{dt} = -r_a + r_b$$

$$\frac{d\,Cb}{dt} = r_a - r_b$$

These expressions reflect the fact that the overall rate of "accumulation" of either species will be the difference between their rates of formation and depletion, that is, the net rate. If we take both r_a and r_b as first order in **Ca** and **Cb**, respectively, then we have:

$$\frac{d\,Ca}{dt} = -k_a Ca + k_b Cb$$
$$\frac{d\,Cb}{dt} = k_a Ca - k_b Cb$$

These two equations can be solved simultaneously to give **Ca[t]** and **Cb[t]** for any arbitrary initial concentrations of **A** and **B**.

The following set of commands shows us the variable names used to this point in the notebook and that they are indeed removed by the **Remove** command.

```
In[42]:= Names["Global‘*"]
         Remove["Global‘*"]
         Names["Global‘*"]

Out[42]= {a, a0, atm, ca, Ca, Cao, cb, Cb, Cbo, cm, d0, da, db,
          firstordpl1, firstordpl2, g, k, M, meanspeed, mol, n,
          Nav, nda, ndb, Pa, Pb, R1, R2, s, secordpl1, secordpl2,
          solnordA, solnordB, t, T, Zab, μ, τ, Φ, Φa, ΦA, Φao,
          Φb, ΦB, $1, $2, $3, $4, $5}

Out[44]= {}
```

Now we set up the solution of the rate equations that express the reversible chemical process:

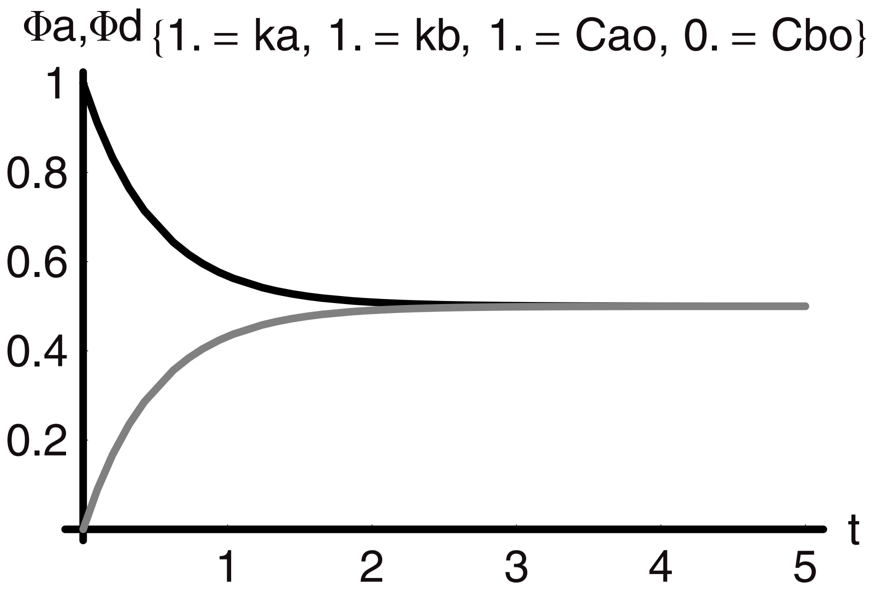
```
In[45]:= reversol1 = Simplify[DSolve[
             {Ca’[t] == -ka Ca[t] + kb Cb[t],
              Cb’[t] == +ka Ca[t] - kb Cb[t],
              Ca[0] == Cao, Cb[0] == Cbo},
             {Ca[t], Cb[t]}, t]];

         Φa[t_] := reversol1[[1, 1, 2]]/Cao
         Φb[t_] := reversol1[[1, 2, 2]]/Cao

         Φa[t]
         Φb[t]

         General::spell1 : Possible spelling error: new symbol
          name "Φb" is similar to existing symbol "Φa".
```

$$\text{Out[48]=}\ \frac{\text{Cbo}(1 - e^{-(\text{ka}+\text{kb})t})\text{kb} + \text{Cao}(e^{-(\text{ka}+\text{kb})t}\text{ka} + \text{kb})}{\text{Cao}(\text{ka} + \text{kb})}$$

```
Out[49]=  Cao(ka - e^(-(ka+kb)t) ka) + Cbo(ka + e^(-(ka+kb)t) kb)
          -------------------------------------------------------
                              Cao(ka + kb)
```

We could have nondimensionalized these equations completely, as we have done for other cases, but then we would lose the individual contributions of **ka** and **kb**. Instead we have referenced to the initial concentration of **A**, but we have retained the real time **t**.

```
In[50]:= SetOptions[{Plot, ListPlot},
           AxesStyle → {Thickness[0.01]},
           PlotStyle → {PointSize[0.015], Thickness[0.006]},
           DefaultFont → {"Helvetica", 17}];

In[51]:= ka = 1.;

         kb = 1.;

         Cao = 1.;

         Cbo = 0.;

         Plot[{N[Φa[t]], N[Φb[t]]}, {t, 0, 5},
          PlotRange → All,
          AxesLabel → {"t", "Φa, Φd"},
          PlotStyle → {{Thickness[0.01], Dashing[{0, 0}]},
           {Thickness[0.01], GrayLevel[0.5]}},
          PlotLabel → {ka "= ka", kb "= kb", Cao "= Cao",
           Cbo "= Cbo"}];
```



What we observe is that the concentrations of **A** and **B** move to a value of 0.5 by **t** = 3 and then they remain unchanged. This is the *equilibrium point* for the parameters that we set. Could we have calculated this before solving the differential equations explicitly? The answer is yes. The reason is that at equilibrium the rates of the forward and reverse reaction are equal, which is why the system appears unchanging. Given that this is the case, we can reason that the accumulation terms (that is, the differentials on the *LHS*) are zero-valued because their arguments are no longer time-dependent. Thus, after fully nondimensionalizing, we can see that:

$$\frac{d\,\Phi a}{d\tau} = -\Phi a + \frac{k_b}{(k_a + k_b)}(1 + \Phi bo)$$

$$\left.\frac{d\Phi a}{d\tau}\right|_{eq} = 0$$

$$\therefore\ \Phi a|_{eq} = \frac{k_b}{(k_a + k_b)}(1 + \Phi bo)$$

Because we chose **ka** = **kb** and **Φbo** = 0, we find that:

$$\Phi a|_{eq} = 0.5$$

We can also note that the kinetics relate directly to the thermodynamics (equilibrium) in this manner:

$$0 = -\Phi a + \frac{k_b}{(k_a + k_b)}(1 + \Phi bo)$$

$$0 = -\Phi a\,(k_a + k_b) + k_b(1 + \Phi bo)$$

$$0 = -\Phi a\,k_a - \Phi a\,k_b + k_b(1 + \Phi bo)$$

$$0 = -\Phi a\,k_a + k_b(1 + \Phi bo - \Phi a)$$

But we have already shown that:

$$\Phi b = 1 + \Phi bo - \Phi a$$

$$\therefore 0 = -\Phi a\,\mathbf{k_a} + \Phi b\,\mathbf{k_b}$$

$$\frac{\Phi b}{\Phi a} = \frac{\mathbf{k_a}}{\mathbf{k_b}} = K_{eq}$$

This well-known result provides the kinetics definition of chemical equilibrium and relates the rate constant from thermodynamics to the ratio of the forward and reverse rate constants.

The case that we have analyzed is the simplest and one more example of mixed order is worth studying in the same way. Let us take as an example the case of one molecule dividing into two different molecules. Examples abound—**PCl_5** reacts to give **PCl_3** and **Cl_2**, ethylbenzene (**$C_6H_5\ CH_2\ CH_3$**) reacts to give styrene (**$C_6H_5\ CH{=}CH_2$**) and dihydrogen (**H_2**). We can generalize this type of reaction to:

$$A = B + D$$

For this case we will assume that in the direction from **A** to **B** and **D** the rate is first order in **A** and in the opposite direction we will take it as second order overall, first order in **B** and **D** each.

$$r_A = k_A C_A$$
$$r_B = r_D = k_B C_B C_D$$

These are constitutive kinetics that we need to complete the model for this type of reaction taking place in a constant volume batch reactor. The component equations are:

$$\frac{d\,\mathbf{C_A}}{dt} = -k_A C_A + k_B C_B C_D$$
$$\frac{d\,\mathbf{C_B}}{dt} = k_A C_A - k_B C_B C_D$$
$$\frac{d\,\mathbf{C_D}}{dt} = k_A C_A - k_B C_B C_D$$

If we try to solve the three simultaneous equations in their initial form, an error message is the result we get back:

```
In[56]:= Remove[Ca, Cb, Cd, ka, kb]

In[57]:= DSolve[
          {Ca'[t] == -ka Ca[t] + kb Cb[t] Cd[t],
           Cb'[t] == +ka Ca[t] - kb Cb[t] Cd[t],
           Cd'[t] == +ka Ca[t] - kb Cb[t] Cd[t],
           Ca[0] == Cao, Cb[0] == Cbo, Cd[0] == Cdo},
          {Ca[t], Cb[t], Cd[t]}, t]

         Solve::tdep : The equations appear to involve the
          variables to be solved for in an essentially
          non-algebraic way.

         DSolve::dsing : Unable to fit initial/boundary
          conditions {Ca[0] == 1, Cb[0] == 0, Cd[0] == Cdo}.

Out[57]= {}
```

If, however, we can say that the initial concentrations of **B** and **D** are equal, then we can reexpress **Cd** in terms of **Cb** as they are equal. Now, we can solve analytically as follows:

```
In[58]:= Names["Global'*"]
         Remove["Global'*"]
         Names["Global'*"]

Out[58]= {Ca, Cao, Cb, Cbo, Cd, Cdo, ka, kb, reversol1, t, Φa, Φb}
```

```
Out[60]= {}

In[61]:= reversol2 = Simplify[DSolve[{Ca'[t] == -ka Ca[t]
              + kb Cb[t]^2, Cb'[t] == +ka Ca[t] - kb Cb[t]^2,
              Ca[0] == Cao, Cb[0] == Cbo}, {Ca[t], Cb[t]}, t]];

         ca[t_] := reversol2[[2, 2]]
         ca[t]
         cb[t_] := reversol2[[4, 2]]
         cb[t]

         Simplify[∂t(ca[t]) == -ka ca[t] + kb cb[t]^2]
         Simplify[∂t(cb[t]) == +ka ca[t] - kb cb[t]^2]

         Solve::ifun : Inverse functions are being used by
          Solve, so some solutions may not be found.
```

Out[63]=

$$\frac{1}{2kb}(ka + 2(Cao + Cbo)kb + \sqrt{-ka(ka + 4(Cao + Cbo)kb)}\ \mathrm{Tan}[\frac{1}{2}\sqrt{-ka(ka + 4(Cao + Cbo)kb)}t + \mathrm{ArcTan}[\frac{\sqrt{4Cao\,ka - 4Cbo^2\,kb}\sqrt{\frac{(ka+2Cbo\,kb)^2}{kb(-Cao\,ka+Cbo^2\,kb)}}}{2\sqrt{ka}\sqrt{4Cao + 4Cbo + \frac{ka}{kb}}}]])$$

Out[65]=

$$-\frac{1}{2kb}(ka + \sqrt{-ka(ka + 4(Cao + Cbo)kb)}\ \mathrm{Tan}[\frac{1}{2}\sqrt{-ka(ka + 4(Cao + Cbo)kb)}t + \mathrm{ArcTan}[\frac{\sqrt{4Cao\,ka - 4Cbo^2\,kb}\sqrt{\frac{(ka+2Cbo\,kb)^2}{kb(-Cao\,ka+Cbo^2\,kb)}}}{2\sqrt{ka}\sqrt{4Cao + 4Cbo + \frac{ka}{kb}}}]])$$

```
Out[66]= True

Out[67]= True

In[68]:= Simplify[ca[t]]
         Simplify[cb[t]]
```

Out[68]=

$$\frac{1}{2kb}(ka + 2(Cao + Cbo)kb + \sqrt{-ka(ka + 4(Cao + Cbo)kb)}\ \mathrm{Tan}[\frac{1}{2}\sqrt{-ka(ka + 4(Cao + Cbo)kb)}t + \mathrm{ArcTan}[\frac{\sqrt{4Cao\,ka - 4Cbo^2\,kb}\sqrt{\frac{(ka + 2Cbo\,kb)^2}{kb(-Cao\,ka + Cbo^2\,kb)}}}{2\sqrt{ka}\sqrt{4Cao + 4Cbo + \frac{ka}{kb}}}]])$$

Out[69]= $-\frac{1}{2kb}(ka + \sqrt{-ka(ka + 4(Cao + Cbo)kb)}$
$\mathrm{Tan}[\frac{1}{2}\sqrt{-ka(ka + 4(Cao + Cbo)kb)}t$
$+ \mathrm{ArcTan}[\frac{\sqrt{4Cao\,ka - 4Cbo^2\,kb}\sqrt{\frac{(ka + 2Cbo\,kb)^2}{kb(-Cao\,ka + Cbo^2\,kb)}}}{2\sqrt{ka}\sqrt{4Cao + 4Cbo + \frac{ka}{kb}}}]])$

These solutions are still somewhat cumbersome and we have already constrained them to equal initial concentrations of **A** and **B**. Let us relax this constraint and solve in nondimensional form. We can express the concentrations of **B** and **D** in terms of the concentration of **A** through the stoichiometric relationships:

$$\frac{d\mathbf{C_A}}{dt} = -\frac{d\mathbf{C_B}}{dt} = -\frac{d\mathbf{C_D}}{dt}$$
$$C_A - C_{Ao} = C_{Bo} - C_B = C_{Do} - C_D$$
$$C_B = C_{Ao} + C_{Bo} - C_A$$
$$C_D = C_{Ao} + C_{Do} - C_A$$

Rewriting we have:

$$\frac{d\mathbf{C_A}}{dt} = -k_A C_A + k_B(C_{Ao} + C_{Bo} - C_A)(C_{Ao} + C_{Do} - C_A)$$
$$\frac{d\mathbf{C_B}}{dt} = k_A C_A - k_B(C_{Ao} + C_{Bo} - C_A)(C_{Ao} + C_{Do} - C_A)$$

We can collect the terms in the concentration of A on the right-hand side and then simplify to put the equations in a simpler looking form prior to solving them:

```
In[70]:= Clear["Global‘*‘"]
```

In[71]:= **$\frac{dC_A}{dt}$ == Simplify[**
Collect[$-k_A C_A + k_B(C_{Ao} + C_{Bo} - C_A)(C_{Ao} + C_{Do} - C_A)$, C_A]]
$\frac{dC_B}{dt}$ == Simplify[
Collect[$k_A C_A - k_B(C_{Ao} + C_{Bo} - C_A)(C_{Ao} + C_{Do} - C_A)$, C_A]]

Out[71]= $\frac{dC_A}{dt}$ == $C_A^2 k_B + (C_{Ao} + C_{Bo})(C_{Ao} + C_{Do})k_B$
$- C_A(k_A + (2C_{Ao} + C_{Bo} + C_{Do})k_B)$

Out[72]= $\frac{dC_B}{dt}$ == $-C_A^2 k_B - (C_{Ao} + C_{Bo})(C_{Ao} + C_{Do})k_B$
$+ C_A(k_A + (2C_{Ao} + C_{Bo} + C_{Do})k_B)$

These can be nondimensionalized with $\mathbf{C_{Ao}}$ as follows:

$$\mathbf{C_{Ao}\frac{d\Phi_A}{dt} = C_{Ao}^2\Phi_A^2 k_B + C_{Ao}^2(\Phi_{Ao}+\Phi_{Bo})(\Phi_{Ao}+\Phi_{Do})k_B - C_{Ao}\Phi_A(k_A + (2C_{Ao} + C_{Bo} + C_{Do})k_B)}$$

$$\mathbf{\frac{d\Phi_A}{dt} = C_{Ao}\Phi_A^2 k_B + C_{Ao}(\Phi_{Ao}+\Phi_{Bo})(\Phi_{Ao}+\Phi_{Do})k_B - \Phi_A(k_A + (2C_{Ao} + C_{Bo} + C_{Do})k_B)}$$

Recognizing that $\mathbf{C_{Ao}\,k_B}$ is in every term, we can divide through by the product of this concentration parameter and the rate constant for the reverse reaction. This product has units of inverse time as the rate constant is second order. Therefore on the left-hand side we have $\mathbf{\frac{1}{C_{Ao}k_B}\frac{d\Phi_A}{dt}}$, which is just the same as $\mathbf{\frac{d\Phi_A}{d\tau}}$, where $\mathbf{d\tau = C_{Ao}k_B dt}$. This puts the equations in complete dimensionless form:

$$\frac{d\Phi_A}{d\tau} = \Phi_A^2 + (\Phi_{Ao}+\Phi_{Bo})(\Phi_{Ao}+\Phi_{Do}) - \Phi_A\frac{(k_A + (2C_{Ao}+C_{Bo}+C_{Do})k_B)}{C_{Ao}k_B}$$

$$\frac{d\Phi_B}{d\tau} = -\Phi_A^2 - (\Phi_{Ao}+\Phi_{Bo})(\Phi_{Ao}+\Phi_{Do}) + \Phi_A\frac{(k_A + (2C_{Ao}+C_{Bo}+C_{Do})k_B)}{C_{Ao}k_B}$$

The group of constants $\mathbf{\frac{(k_A+(2C_{Ao}+C_{Bo}+C_{Do})k_B)}{C_{Ao}k_B}}$, which we shall call **M**, that make up the coefficient of the linear term in $\mathbf{\Phi_A}$ are worth looking at in more detail. Recall that k_B is a second-order rate constant with dimensions of vol/mol/time. When this is multiplied by the sum of the initial concentrations ($\mathbf{2C_{Ao} + C_{Bo} + C_{Do}}$), the resultant dimensions are 1/time, the same as that of $\mathbf{k_A}$ (the other term in the numerator), and as the product $\mathbf{C_{Ao}k_B}$ seen in the denominator. This makes sense and the overall group is dimensionless. We also see that if we provide the relative magnitudes of the rate constants and the initial concentrations, then this term can be evaluated. To solve these equations we stop just short of this and give initial dimensionless concentrations and the ratio of the rate constants:

```
In[73]:= Names["Global'*"]
         Remove["Global'*"]
         Names["Global'*"]

Out[73]= {A, Ao, B, Bo, ca, Ca, Cao, cb, Cb, Cbo, d, dt, k, ka,
           kb, reversol2, t}

Out[75]= {}

In[76]:= ndreversol3 =.
         Φao = Cao = 1;
         Φbo = Φdo = Cbo = Cdo = 0;
         kb = 10 ka;

         ndreversol3 = Simplify[DSolve[{Φa'[τ] == Φa[τ]^2
               + (Φao + Φbo) (Φao + Φdo) - Φa [τ] M,
        Φb'[τ] == -Φa[τ]^2 - (Φao + Φbo)(Φao + Φdo)
         + Φa [τ] M,
        Φa[0] == Φao, Φb[0] == Φbo},
          {Φa[τ], Φb[τ]}, τ]]

    ΦA[τ_] := ndreversol3[[1, 2]]
    ΦB[τ_] := ndreversol3[[2, 2]]

    Simplify[∂τ(ΦA[τ]) == ΦA[τ]^2 + (Φao + Φbo) (Φao + Φdo)
          - ΦA[τ] M]
    Simplify[∂τ(ΦB[τ]) == -ΦA[τ]^2 - (Φao + Φbo) (Φao + Φdo)
          + ΦA[τ] M]

    General::spell1 : Possible spelling error: new symbol
     name "Φbo" is similar to existing symbol "Φao".

    General::spell : Possible spelling error: new symbol
     name "Φdo" is similar to existing symbols {Φao, Φbo}.

    General::spell1 : Possible spelling error: new symbol
     name "Φa" is similar to existing symbol "Φao".

    General::spell : Possible spelling error: new symbol
     name "Φb" is similar to existing symbols {Φa, Φbo}.

    Solve::ifun : Inverse functions are being used by
     Solve, so some solutions may not be found.
```

Out[80]=
$$\left\{\Phi a[\tau]\to\frac{1}{2}\left(M+\sqrt{4-M^2}\,\mathrm{Tan}\left[\frac{1}{2}\sqrt{4-M^2}\,\tau+\mathrm{ArcTan}\left[\frac{2-M}{\sqrt{4-M^2}}\right]\right]\right),\right.$$
$$\left.\Phi b[\tau]\to\frac{(-2+M)\left(-\sqrt{4-M^2}+(2+M)\,\mathrm{Tan}\left[\frac{1}{2}\sqrt{4-M^2}\,\tau+\mathrm{ArcTan}\left[\frac{2-M}{\sqrt{4-M^2}}\right]\right]\right)}{2\sqrt{4-M^2}}\right\}$$

```
General::spell1 : Possible spelling error: new symbol
 name "ΦA" is similar to existing symbol "Φa".

General::spell : Possible spelling error: new symbol
 name "ΦB" is similar to existing symbols {ΦA, Φb}.

Out[83]= True

Out[84]= True
```

The solutions are more complex than we have seen before, but the check we have put them through indicates their validity. The complexity arises from the fact that this problem is one that is fully transient until the equilibrium point is reached. It is important to realize that there is a marked difference between equilibrium and steady state, as we will see when we examine flow reactors. We can have a steady state in a flow reactor, which is far from equilibrium.

We can test these solutions further after we have applied a specific value for the forward rate constant; we take $\mathbf{k_a} = 10^{-3}\ \text{min}^{-1}$. Taking the limit as τ goes to zero should give us unity and zero for dimensionless **A** and **B**. At long times they should go to the equilibrium values.

```
In[85]:= Φao = Cao = 1;
         Φbo = Φdo = Cbo = Cdo = 0;
         ka = .001;
         kb = 10 ka;
              (ka + (2Cao + Cbo + Cdo)kb)
         M = ----------------------------- ;
                        Cao kb
         ndreversol3 // N

         Limit[ndreversol3[[1, 2]], τ → 0]
         Limit[ndreversol3[[2, 2]], τ → 0]

         ndreversol3[[1, 2]] /. τ → 10^6
         ndreversol3[[2, 2]] /. τ → 10^6

Out[90]= {Φa[τ] → 0.5 (2.1 - 0.640312 Tanh[(0.157462 + 0.i)
           + 0.320156 τ]),
          Φb[τ] → (0.-0.0780869i) ((0.-0.640312i)
           + (0. + 4.1i)Tanh[(0.157462 + 0.i) + 0.320156τ])}

Out[91]= 1. + 0.i

Out[92]= 0. + 0.i

Out[93]= 0.729844 + 0.i

Out[94]= 0.270156 + 0.i
```

We see that at zero time the values of dimensionless **A** and **B** concentration are as they should be, and at long time they tend to 0.73 and 0.27, respectively. We can check this by computing the equilibrium extent of reaction α from the expression for the equilibrium constant. Recall that the magnitude of the equilibrium constant at any temperature is given by the ratio of the forward to the reverse rate constants; and the concentration of the products at equilibrium in this case is just $\alpha\ \mathbf{C_{Ao}}$ and the reactant is $(\mathbf{1} - \alpha)\ \mathbf{C_{Ao}}$. This gives the following expression to be solved:

In[95]:= $\text{Solve}\left[\frac{k_A}{k_B} == C_{Ao}\frac{\alpha^2}{1-\alpha}, \alpha\right]$

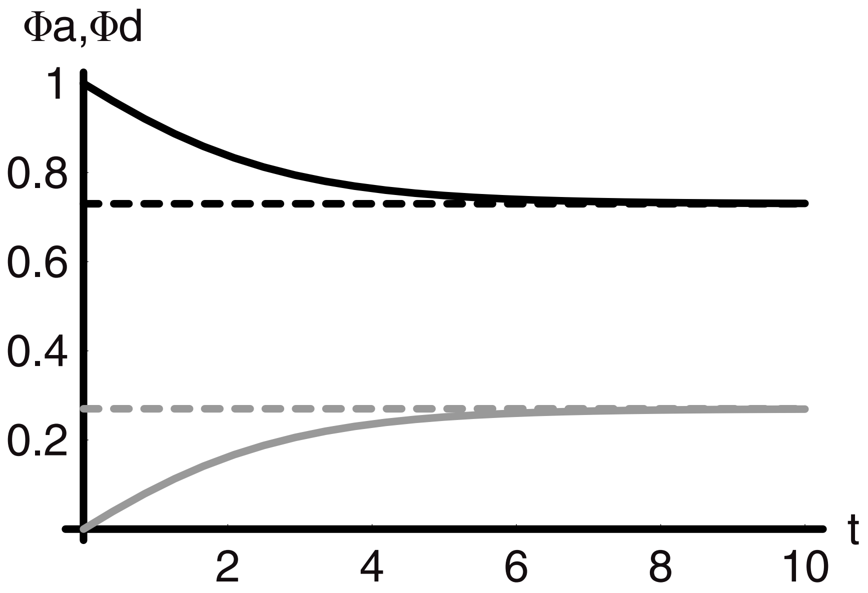
Out[95]= $\left\{\left\{\alpha \to \frac{-k_A - \sqrt{k_A}\sqrt{k_A + 4C_{Ao}k_B}}{2C_{Ao}k_B}\right\}, \left\{\alpha \to \frac{-k_A + \sqrt{k_A}\sqrt{k_A + 4C_{Ao}k_B}}{2C_{Ao}k_B}\right\}\right\}$

Clearly, the extent of reaction must be positive and the value of 0.27 agrees exactly with the value derived from the kinetics. Finally, we can graph the concentrations of **A** and **B** in dimensionless form as a function of dimensionless time. But before we can do so we need to examine the solutions carefully. We know they are correct, but we also notice that the term 0.i appears in both. In order to plot these solutions we must have fully real forms; that is, even if the coefficient of ⅈ is zero, we cannot graph such an expression in the real plane because *Mathematica* takes this as a complex number. Let us look at these solutions before and after we use **Complex Expand** on them:

```
In[96]:= ndreversol3[[1, 2]]
         ndreversol3[[2, 2]]
         Simplify[ComplexExpand[ndreversol3[[1, 2]]]]
         Simplify[ComplexExpand[ndreversol3[[2, 2]]]]

          1
Out[96]= --- (2.1 - 0.640312 Tanh[(0.157462 + 0.ⅈ) + 0.320156τ])
          2

Out[97]= (0.- 0.0780869ⅈ) (-0.640312ⅈ
          + 4.1ⅈ Tanh[(0.157462 + 0.ⅈ) + 0.320156τ])

                                 0.ⅈ
Out[98]= 1.05 + ------------------------------------
                 1.+ Cosh[0.314925 + 0.640312τ]

           0.320156 Sinh[0.314925 + 0.640312τ]
         - ------------------------------------
            1. + Cosh[0.314925 + 0.640312 τ]

                                         0. + 0.ⅈ
Out[99]= (-0.05 + 0.ⅈ) + ------------------------------------
                          1. + Cosh[0.314925 + 0.640312 τ]

            (0.320156 + 0.ⅈ) Sinh[0.314925 + 0.640312τ]
          + ---------------------------------------------
                 1. + Cosh[0.314925 + 0.640312τ]
```

It is easy to see that the first expression is fully real, but the second expression for dimensionless **B** is less clear until we expand it. After expansion we can see that although the solutions appear to involve complex numbers the coefficients of ⅈ are all identically zero (see what follows in the next graph):

```
                             (0.320 Sinh[0.315 + 0.640τ])
In[100]:= ΦA[τ_] := 1.05 - -------------------------------
                             (1 + Cosh[0.315 + 0.640τ])

                    (0.320) Sinh[0.315 + 0.640τ]
          ΦB[τ_] := ---------------------------- - .05
                    (1 + Cosh[0.315 + 0.640τ])

          Plot[
               {ΦA[τ], ΦB[τ]}, {τ, 0, 10},
            PlotRange → All,
            AxesLabel → {"t", "Φa, Φd"},
            AxesStyle → {Thickness[0.01]},
            PlotStyle → {{Thickness[0.01], GrayLevel[0]},
             {Thickness[0.01], GrayLevel[0.6]}},
            Epilog → {
              {GrayLevel[0.6], Dashing[{0.02, 0.02}],
               Thickness[.01], Line[{{0, 0.27}, {10, 0.27}}]},
              {GrayLevel[0], Dashing[{0.02, 0.02}], Thickness[.01],
                Line[{{0, 0.73}, {10, 0.73}}]}
            }, DefaultFont → {"Helvetica", 17}
               ];
```



7.5 Complex Reactions

Series

Reactions rarely take place in isolation of other reactions. Reversibility is one example of the simultaneity of reaction chemistries. Another classical problem is the one that arises when a reaction is immediately preceded by another reaction. When reactions occur in series, they are referred to as being consecutive. An example would be:

$$A \rightarrow B \rightarrow D$$

If each of these proceeds via a first-order rate process, then this can be analyzed readily. Higher-order reaction rates follow the same analysis, but they require a bit more mathematical effort. We can begin by writing the key material balance equations:

$$\frac{d\,C_A}{dt} = -k_A C_A$$

$$\frac{d\,C_B}{dt} = k_A C_A - k_B C_B$$

$$\frac{d\,C_D}{dt} = k_B C_B$$

We can see that the first of these equations can be integrated immediately to give:

$$Ca = Cao\,\exp\,(-k_A t)$$

This can be substituted into the second equation and the integration can be done for the concentration of **B**. Subsequently, we substitute this into the equation for the rate of change of

D and integrate once more for the full solution. Alternatively, we can let the **DSolve** algorithm do for us all at once:

```
In[103]:= Clear["Global‘*"]

In[104]:= Simplify[
            DSolve[
             {cA’[t] == -ka cA[t],
              cB’[t] == ka cA[t] - kb cB[t],
              cD’[t] == kb cB[t],
              cA[0] == cAo,
              cB[0] == 0,
              cD[0] == 0},
             {cA[t], cB[t], cD[t]},
             t]
                ]

          General::spell1 : Possible spelling error: new symbol
           name "cAo" is similar to existing symbol "Cao".
```

Out[104]= $\{\{cA[t] \to cAo\, e^{-ka\, t},\ cB[t] \to \dfrac{cAo\,(e^{-ka\, t} - e^{-kb\, t})\, ka}{-ka + kb},$

$cD[t] \to \dfrac{cAo\,(ka - e^{-kb\, t} ka + (-1 + e^{-ka\, t})\, kb)}{ka - kb}\}\}$

We could also have chosen to nondimensionalize the differential equations before solving them in order to find a general solution in fewer absolute parameters. We can divide all by $\mathbf{k_A C_{Ao}}$, which will give us:

$$\left[\frac{1}{k_A C_{Ao}}\right]\frac{d\,C_A}{dt} = -\Phi_A$$

$$\left[\frac{1}{k_A C_{Ao}}\right]\frac{d\,C_B}{dt} = \Phi_A - \frac{k_B}{k_A}\Phi_B$$

$$\left[\frac{1}{k_A C_{Ao}}\right]\frac{d\,C_D}{dt} = \frac{k_B}{k_A}\Phi_B$$

However, $\mathbf{k_A dt = d\tau}$ because $\mathbf{k_A}$ is an inverse time constant associated with the rate of the first chemical reaction and τ is “reduced” time. This gives us the following three equations:

$$\frac{d\Phi_A}{d\tau} = -\Phi_A$$

$$\frac{d\Phi_B}{d\tau} = \Phi_A - \frac{k_B}{k_A}\Phi_B$$

$$\frac{d\Phi_D}{d\tau} = \frac{k_B}{k_A}\Phi_B$$

Now the ratio of $\frac{k_B}{k_A}$ is not an equilibrium constant because both reactions are considered to be irreversible. It is simply the ratio of the rate constants and we will leave it as such.

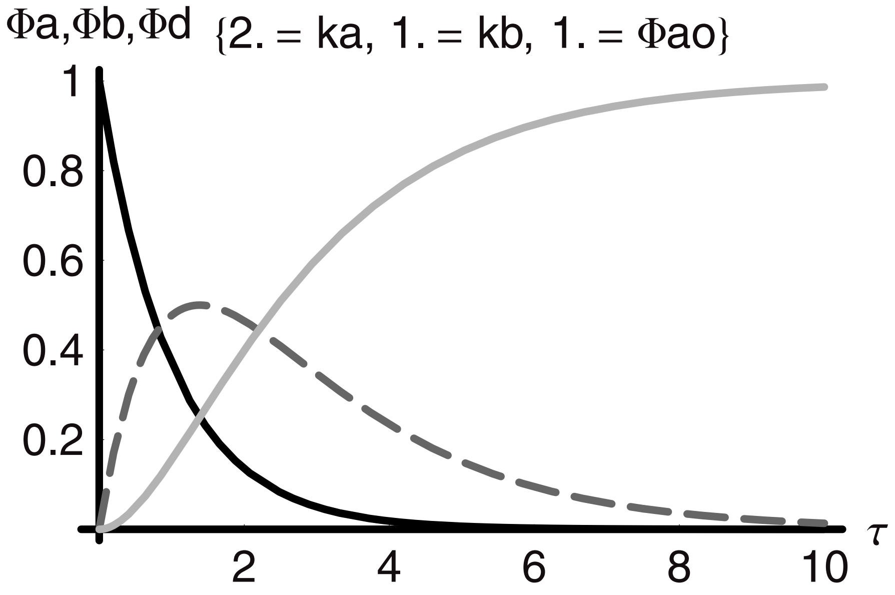
```
In[105]:= Clear["Global‘*"]

In[106]:= ka =.
          kb =.
          Φao =.
          sersol1 = Simplify[DSolve[
              {Φa'[τ] == -Φa[τ],
               Φb'[τ] == +Φa[τ] - kb/ka Φb[τ],
               Φd'[τ] == +kb/ka Φb[τ],
               Φa[0] == Φao,
               Φb[0] == 0,
               Φd[0] == 0},
              {Φa[τ], Φb[τ], Φd[τ]},
              τ]
                   ];

          ΦA[τ_] := sersol1[[1, 1, 2]]
          ΦB[τ_] := sersol1[[1, 2, 2]]
          ΦD[τ_] := sersol1[[1, 3, 2]]
          Φa[τ] == ΦA[τ]
          Φb[τ] == ΦB[τ]
          Φd[τ] == ΦD[τ]
          ka = 2.;
          kb = 1.;
          Φao = 1.;

          Plot[
            {ΦA[τ], ΦB[τ], ΦD[τ]},
            {τ, 0, 10},
            PlotRange → All,
            AxesLabel → {"τ", "Φa,Φb,Φd"},
            AxesStyle → {Thickness[0.01]},
            PlotStyle →
             {{Thickness[0.01], Dashing[{0, 0}]},
              {Thickness[0.01], Dashing[{0.05, 0.025}],
               GrayLevel[0.4]},
              {Thickness[0.01], GrayLevel[0.7]}},
            DefaultFont → {"Helvetica", 20}},
            PlotLabel → {ka "= ka", kb "= kb", Φao "= Φao"}];
```

```
General::spell : Possible spelling error: new symbol
 name "Φd" is similar to existing symbols {Φ, Φa,
 Φb, Φdo}.

General::spell : Possible spelling error: new symbol
 name "ΦD" is similar to existing symbols {Φ, ΦA,
 ΦB, Φd}.
```

Out[113]= $\Phi a[\tau] == e^{-\tau}\Phi ao$

Out[114]= $\Phi b[\tau] == \dfrac{(e^{-\tau} - e^{-\frac{kb\tau}{ka}})\, ka\, \Phi ao}{-ka + kb}$

Out[115]= $\Phi d[\tau] == \dfrac{(ka - e^{-\frac{kb\tau}{ka}} ka + (-1 + e^{-\tau})\, kb)\, \Phi ao}{ka - kb}$



The preceding graph shows the behavior expected for a set of reactions taking place in series. We see the reactant being depleted, and the intermediate concentration grows and then falls while the final product grows monotonically throughout the process. It would be handy to be able to look at this “A to B to D” process with different rate constants in order to gain a better understanding of how the concentration profiles for each species vary in character with changes in the magnitudes of the rate constants. However, this would be cumbersome if we were to use the code we have just written. Instead it makes much more sense to write a Module function based on this code, which can be invoked and utilized like any other command. Here is the means to do that:

```
In[120]:= Clear["Global‘*"]
```

```
In[121]:= sersol2[ka_, kb_, Φao_, τmax_] :=
           Module[{Φa, Φb, Φd},
           Φa[τ_] := e^-τ Φao;
```

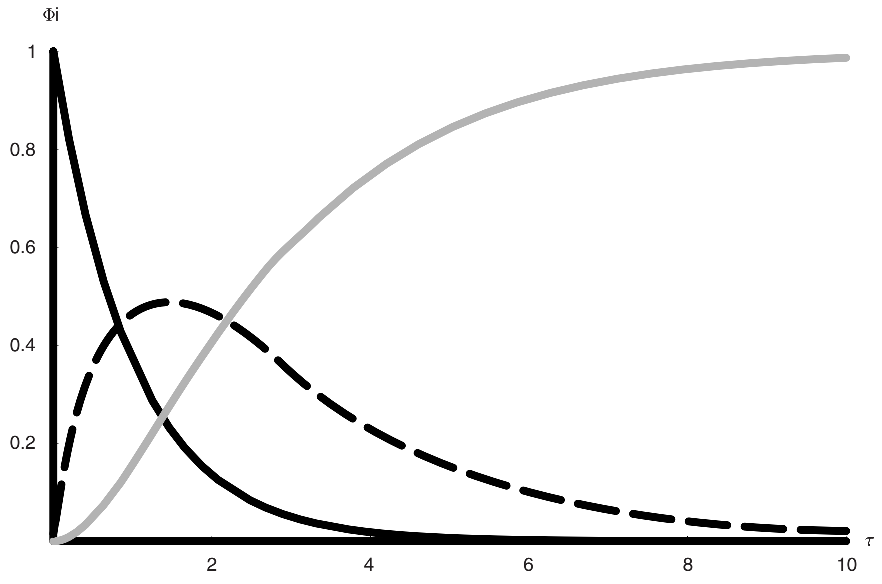
$$\Phi b[\tau\_] := \frac{(e^{-\tau} - e^{-\frac{kb\,\tau}{ka}})\,ka\,\Phi ao}{-ka + kb};$$

$$\Phi d[\tau\_] := \frac{(ka - e^{-\frac{kb\,\tau}{ka}}ka + (-1 + e^{-\tau})kb)\Phi ao}{ka - kb};$$

```
           SetOptions[plot, DefaultFont → {"Helvetica", 8}];
           Plot[
            {Φa[τ], Φb[τ], Φd[τ]},
            {τ, 0, τmax},
            PlotRange → {{0, τmax}, {0, 1}},
            AxesLabel → {"τ", "Φi"},
            AxesStyle → {Thickness[0.01]},
            PlotStyle →
             {{Thickness[0.01], Dashing[{0, 0}]},
              {Thickness[0.01], Dashing[{0.05, 0.025}]},
              {Thickness[0.01], GrayLevel[0.7]}},
            DisplayFunction → Identity]
          ]
```

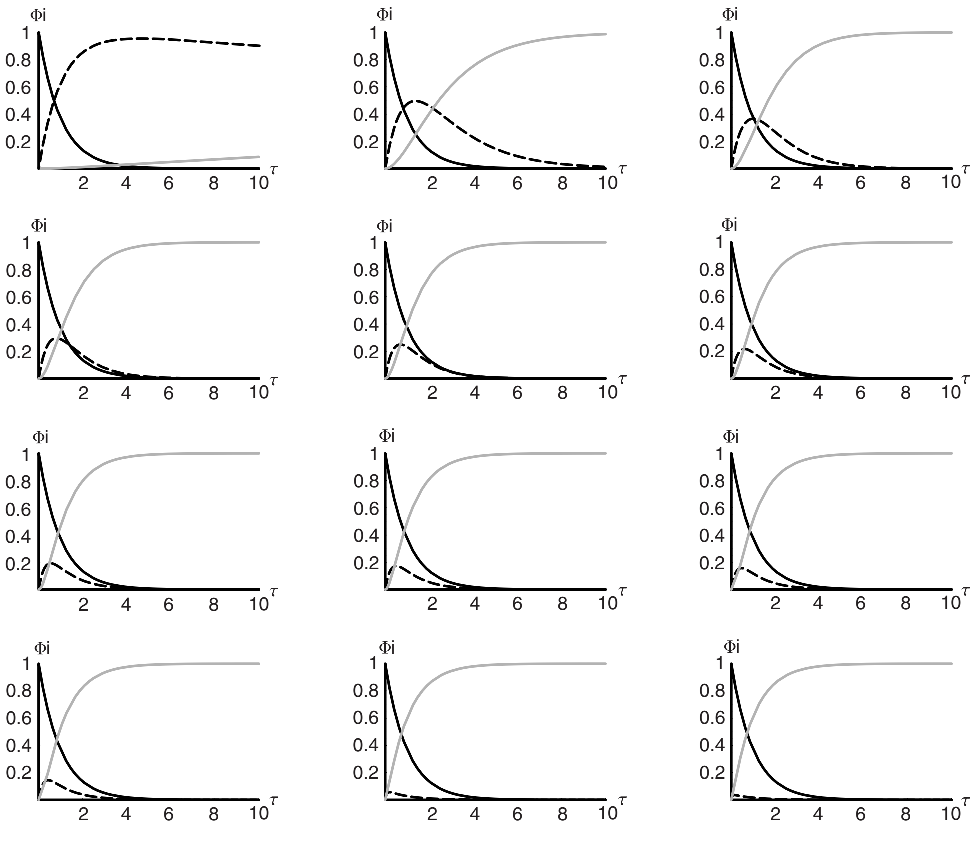
Now if we input the Module and then run it with parameter values as shown, we obtain the same result as that which we had in the preceding:

```
In[122]:= Show[sersol2[1., .5, 1., 10],
           DisplayFunction → $DisplayFunction];
```



Furthermore one can now run as many cases as one should like in order to compare the effects of different parameters. We have left the semicolon out after the **Plot** routine in the **Module** function, so that **Graphics** are a bonafide output. This allows us to use sersol as part of **GraphicsArray**.

```
In[123]:= Show[
             GraphicsArray[{Table[sersol2[1., n, 1., 10],
               {n, 0.01, 1.01, .5}],
               Table[sersol2[1., n, 1., 10], {n, 1.51, 2.52, .5}],
               Table[sersol2[1., n, 1., 10], {n, 3.01, 4.02, .5}],
               Table[sersol2[1., n, 1., 10], {n, 4.51, 24.51, 10}]
             }, DisplayFunction → $DisplayFunction]];
```



This array shows the full gamut of the effects that the magnitude of **kb** at fixed **ka** has upon the chemistry. We see that when **kb** is 10^2 smaller than **ka**, the reaction appears to be that of **A** $\longrightarrow$ **B**. When we begin to increase the magnitude of **kb** we see that the intermediacy of **B** grows as does the final amount of **D** at 10τ. With longer times, the amount of **B** would grow to be equal to the original amount of **A**, but we are concerned here with a fixed batch holding time of 10τ. As **kb** increases and overtakes **ka**, the maximum amount of **B** continuously diminishes and shifts to earlier τ. Finally with **kb** at ~25**ka**, the maximum in **B** is shifted to very short τ and to a value of less than 0.1.

If we were to change the kinetics so that the first reaction was second order in **A** and the second reaction was first order in **B**, then we would see largely the same picture emerging in the graphs of dimensionless concentration versus time. There would of course be differences, but not large departures in the trends from what we have observed for this all first-order case. But what if the reactions have rate expressions that are not so readily integrable? What if we have widely differing, mixed-order concentration dependencies? In some cases one can develop fully analytical (closed-form) solutions like the ones we have derived for the first-order case, but in other cases this is not possible. We must instead turn to numerical methods for efficient solution.

Suppose that the following reaction is a series network with square kinetics for the first reaction and half-order kinetics for the second:

$$2A \rightarrow B \rightarrow D$$

Then accounting for the stoichiometry of two going to one we have the following set of equations to solve:

$$\begin{aligned}
\frac{d\,Ca}{dt} &= -ka\,Ca^2 \\
\frac{d\,Cb}{dt} &= +\frac{ka\,Ca^2}{2} - kb\sqrt{Cb} \\
\frac{d\,Cd}{dt} &= +kb\sqrt{Cb} \\
\frac{d\,Cb}{dt} &= -\frac{1}{2}\frac{d\,Ca}{dt} - \frac{d\,Cd}{dt}
\end{aligned}$$

Nondimensionalizing must be done carefully. We begin with the first equation, which gives the expected result:

$$\begin{aligned}
\frac{Cao}{Cao}\frac{d\,Ca}{dt} &= -\frac{Cao^2}{Cao^2}ka\,Ca^2 \\
Cao\frac{d\Phi a}{dt} &= -Cao^2\,ka\,\Phi a^2 \\
\frac{d\Phi a}{dt} &= -Cao\,ka\,\Phi a^2 \\
\frac{d\Phi a}{d\tau} &= -\Phi a^2
\end{aligned}$$

As we are going to need the nondimensionalized form for the rate of change of **D** to obtain **B** we proceed with this equation next:

$$\begin{aligned}\frac{d\,Cd}{dt} &= +kb\sqrt{Cb}\\ \frac{1}{ka\,Cao^2}\frac{d\,Cd}{dt} &= +\frac{1}{ka\,Cao^2}kb\sqrt{Cb}\\ \frac{d\Phi d}{d\tau} &= +\frac{kb}{ka\sqrt{Cao^3}}\sqrt{\Phi b}\end{aligned}$$

Finally, we take these two results and combine them to derive the nondimensionalized form for the rate of change of **B**. We first show that the overall nondimensionalized equation is parallel in form to the fully dimensional equations and then make the appropriate substitutions and so on:

$$\begin{aligned}\frac{d\,Cb}{dt} &= -\frac{1}{2}\frac{d\,Ca}{dt} - \frac{d\,Cd}{dt}\\ \frac{1}{ka\,Cao^2}\frac{d\,Cb}{dt} &= \frac{1}{ka\,Cao^2}\left(-\frac{1}{2}\frac{d\,Ca}{dt} - \frac{d\,Cd}{dt}\right)\\ \frac{d\,\Phi b}{d\tau} &= -\frac{1}{2}\frac{d\,\Phi a}{d\tau} - \frac{d\,\Phi d}{d\tau}\\ \frac{d\,\Phi b}{d\tau} &= \frac{\Phi a^2}{2} - \frac{kb}{ka\sqrt{Cao^3}}\sqrt{\Phi b}\end{aligned}$$

If we try to solve this analytically, we find that we cannot do it, at least not directly with **DSolve**:

```
In[124]:= Names["Global‘*"]
          Remove["Global‘*"]
          Names["Global‘*"]

Out[124]= {A, Ao, B, cA, cAo, Cao, cB, Cbo, cD, Cdo, k, ka, kb,
           M, n, ndreversol3, sersol1, sersol2, t, α, τ, τmax, τ$,
           Φ, Φa, ΦA, Φao, Φa$, Φb, ΦB, Φbo, Φb$, Φd, ΦD, Φdo,
           Φd$, $1}

Out[126]= {}

In[127]:= DSolve[
           {Φa'[τ] == -Φa[τ]^2,
```

$$\Phi b'[\tau] == +\frac{\Phi a[\tau]^2}{2} - \frac{kb}{2ka\sqrt{Cao^3}}\sqrt{\Phi b[\tau]},$$

```
  Φd'[τ] == +kb/(2 ka√Cao³) √Φb[τ],
  Φa[0] == Φao,
  Φb[0] == 0,
  Φd[0] == 0},
 {Φa[τ], Φb[τ], Φd[τ]},
  τ]
```

```
General::spell1 : Possible spelling error: new symbol
 name "Φb" is similar to existing symbol "Φa".

General::spell : Possible spelling error: new symbol
 name "Φd" is similar to existing symbols {Φa, Φb}.

General::spell1 : Possible spelling error: new symbol
 name "Φao" is similar to existing symbol "Φa".
```

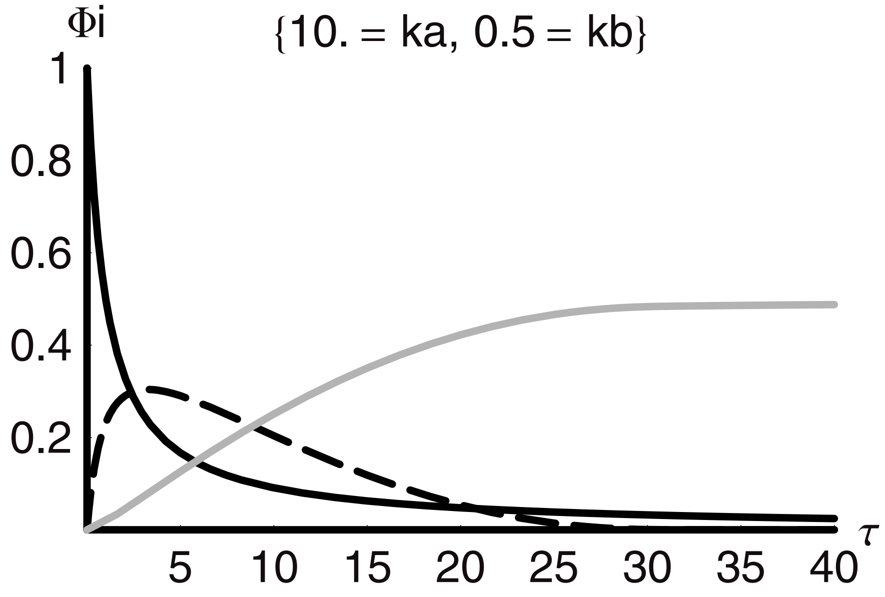
```
Out[127]= DSolve[{Φa′ [τ] == -Φa[τ]²,
           Φb′ [τ] == Φa[τ]²/2 - kb√Φb[τ]/(2√Cao³ ka),
           Φd′ [τ] == kb√Φb[τ]/(2√Cao³ ka), Φa[0] == Φao, Φb[0] == 0,
           Φd[0] == 0}, {Φa[τ], Φb[τ], Φd[τ]}, τ]
```

We turn then to numerical methods in **NDSolve** and find the solution readily, as long as we specify parameters.

```
In[128]:= Remove["Global'*"]

In[129]:= ka = 10.;

          kb = .5;

          Cao = 1;

          τmax = 40;

          sersol3 = NDSolve[{Φa'[τ] == -Φa[τ]²,
              Φb'[τ] == +Φa[τ]²/2 - kb/(ka√Cao³) √Φb[τ],
              Φd'[τ] == +kb/(ka√Cao³) √Φb[τ],
              Φa[0] == 1,
              Φb[0] == 0,
              Φd[0] == 0},
              {Φa[τ], Φb[τ], Φd[τ]},
              {τ, 0, τmax}];
```

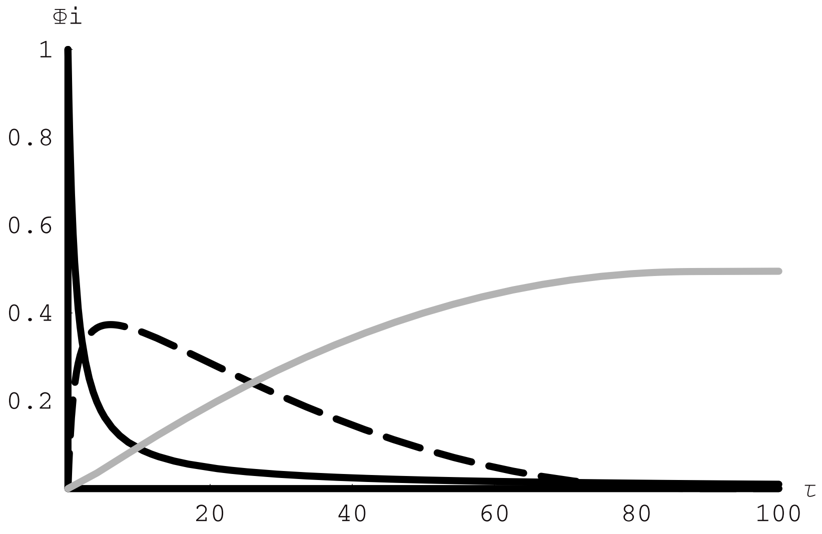
```
ΦA[τ_] := sersol3[[1, 1, 2]]
ΦB[τ_] := sersol3[[1, 2, 2]]
ΦD[τ_] := sersol3[[1, 3, 2]]

Plot[{ΦA[τ], ΦB[τ], ΦD[τ]},
  {τ, 0, τmax},
  PlotRange → {{0, τmax}, {0, 1}},
  AxesLabel → {"τ", "Φi"},
  PlotStyle →
   {{Thickness[.01], Dashing[{0, 0}]},
    {Thickness[0.01], Dashing[{0.05, 0.025}]},
    {Thickness[.01], GrayLevel[0.7]}},
  PlotLabel → {ka "= ka", kb "= kb"}];
```

```
General::spell1 : Possible spelling error: new symbol
 name "Φb" is similar to existing symbol "Φa".

General::spell : Possible spelling error: new symbol
 name "Φd" is similar to existing symbols {Φa, Φb}.

General::spell1 : Possible spelling error: new symbol
 name "ΦA" is similar to existing symbol "Φa".

General::spell : Possible spelling error: new symbol
 name "ΦB" is similar to existing symbols {ΦA, Φb}.

General::spell : Possible spelling error: new symbol
 name "ΦD" is similar to existing symbols{ΦA, ΦB, Φd} .
```

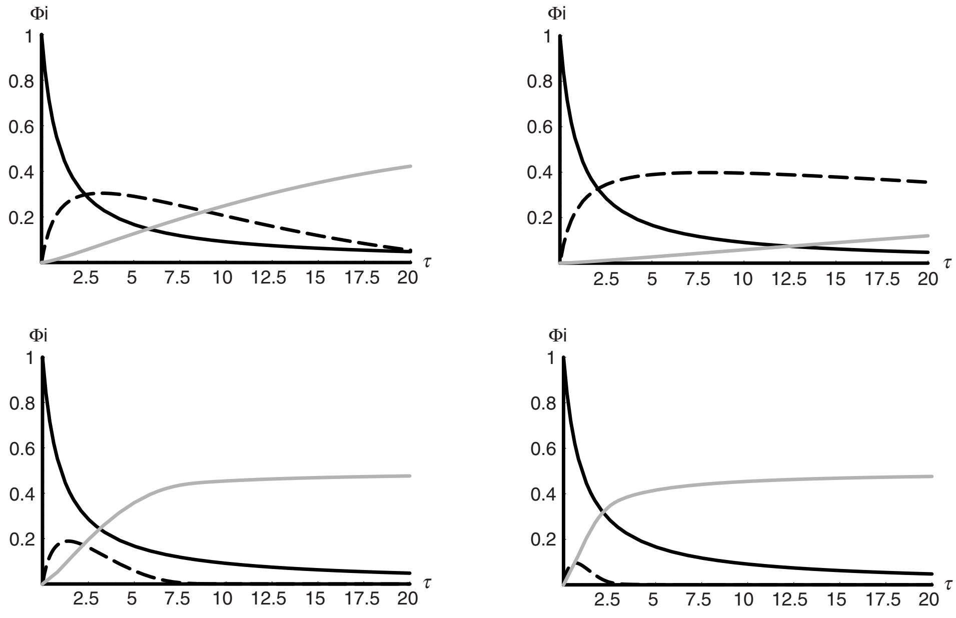


The beauty of an analytical solution (see preceding graph) is that it allows us to see the function and all of its parametric dependencies "all at once." The disadvantage of the numerical solution is that it does not allow for this, at least not directly. On the other hand, we do obtain solutions where there may not have been any if we lacked the numerical tools. *Mathematica* allows us to approach this problem by creating a **Module** function of the numerical routine. With this **Module** we can use a **Table** loop to find how the solutions vary with different parameters. We can do this as follows:

```
In[138]:= Clear["Global'*"]

In[139]:= mixord[ka_, kb_, Φao_, Cao_, τmax_] := Module
           [{sersol, Φa, Φb, Φd, A, B, D, τ},
            sersol = NDSolve[
             {Φa'[τ] == -Φa[τ]^2,
```
$$\Phi b'[\tau] == +\frac{\Phi a[\tau]^2}{2} - \frac{kb}{ka\sqrt{Cao^3}}\sqrt{\Phi b[\tau]},$$
$$\Phi d'[\tau] == +\frac{kb}{ka\sqrt{Cao^3}}\sqrt{\Phi b[\tau]},$$
```
               Φa[0] == Φao,
               Φb[0] == 0,
               Φd[0] == 0},
             {Φa[τ], Φb[τ], Φd[τ]},
             {τ, 0, τmax}];

            A[τ] = Evaluate[Φa[τ] /. sersol];
            B[τ] = Evaluate[Φb[τ] /. sersol];
            D[τ] = Evaluate[Φd[τ] /. sersol];

            SetOptions[Plot, DefaultFont → {"Hevetica", 8}];

            Plot[{A[τ], B[τ], D[τ]},
             {τ, 0, τmax},
             PlotRange → {{0, τmax}, {0, 1}},
             AxesLabel → {"τ", "Φi"},
             PlotStyle →
              {{Thickness[.01], Dashing[{0, 0}]},
               {Thickness[0.01], Dashing[{0.05, 0.025}]},
               {Thickness[.01], GrayLevel[0.7]}},
            DisplayFunction → Identity]
          ]

In[140]:= Show[mixord[30, .5, 1, 1, 100],
           DisplayFunction → $DisplayFunction];
```

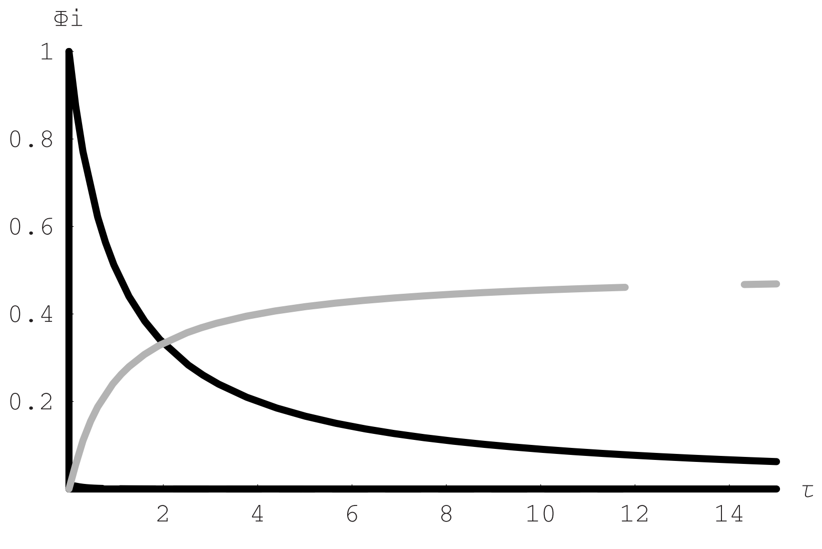


```
In[141]:= Show[
            GraphicsArray[
             {Table[mixord[n, .5, 1., 1., 20], {n, 10, 50, 40}],
              Table[mixord[5., m, 1., 1., 20], {m, 1, 3, 2}]}
            ]];
```



We notice that at constant **kb** as the value of **ka** increases so does the concentration of **A** at shorter times. However, notice that the concentrations of **A** and **B** have much different parametric sensitivities than they did in the other cases. Even with a small value of $\mathbf{ka} = 0.1$, we find that with $\mathbf{kb} = 0.5$ nearly all the **A** is converted to **D** in about 15τ. In simpler terms every time **A** is converted to **B**, then **B** is immediately converted to **D**. Thus we see very little **A**. As **kb** increases at constant **ka**, the trend is reversed. We also note that these equations are numerically "stiff" for some values of their parameters. For example, if we choose $\mathbf{ka} = 0.1$ and $\mathbf{kb} = 0.5$, the integration becomes unstable after about 6τ.

```
In[142]:= Show[mixord[.1, .5, 1, 1, 15],
           DisplayFunction → $DisplayFunction];

          Plot::plnr : B$536[τ$536] is not a machine-size real
           number at τ$536 = 11.846017793583648`.

          Plot::plnr : B$536[τ$536] is not a machine-size real
           number at τ$536 = 11.76600439063428`.

          Plot::plnr : B$536[τ$536] is not a machine-size real
           number at τ$536 = 11.75609901499778`.

          General::stop : Further output of Plot::plnr will be
           suppressed during this calculation.
```



Series-Parallel Reactions

Next, we shall consider the series-parallel reaction system. Here, we shall examine the case where the reactions are all first order. This keeps the math simple and allows us observe the general behavior of such a group of reactions. If the order of the rates of reaction becomes higher, or nonintegral, then numerical methods such as those used in the last section may be employed.

If instead of the one species reacting to one other species, we will look at the situation in which there can be two products formed by competing reactions. We can also let one of the two primary products react to produce one other product. Thus, this set of reactions, or reaction *network* will involve four components and three rate constants as follows:

$$\begin{tikzcd} A \arrow[r, "k_1"] \arrow[dr, "k_3"'] & B \arrow[r, "k_2"] & D \\ & E & \end{tikzcd}$$

This is a rather simple network of reactions that can be solved readily by employing the same analysis methods that we have used to this point:

$$\frac{d\,C_A}{dt} = -k_1\,C_A - k_3\,C_A$$
$$\frac{d\,C_B}{dt} = k_1\,C_A - k_2\,C_B$$
$$\frac{d\,C_D}{dt} = k_2\,C_B$$
$$\frac{d\,C_E}{dt} = k_3\,C_A$$

We note that in the first equation the rate constants are the proportionality factors that determine how much of **A** proceeds to **B** and **E**. Also, the rate of depletion of **A** follows an observed rate constant that is the sum of the two rate constants for the parallel forward reactions. The other equations are much as we would expect. We can nondimensionalize using the sum $\mathbf{k1}+\mathbf{k3}$ and of course $\boldsymbol{C_{Ao}}$:

$$\frac{d\Phi_A}{d\tau} = -\Phi_A$$
$$\frac{d\Phi_B}{d\tau} = \kappa_1\Phi_A - \kappa_2\,\Phi_B$$
$$\frac{d\Phi_D}{d\tau} = \kappa_2\,\Phi_B$$
$$\frac{d\Phi_E}{d\tau} = \kappa_3\,\Phi_A$$

where

$$\kappa_1 = \frac{k_1}{(k_1+k_3)}, \quad \kappa_2 = \frac{k_2}{(k_1+k_3)}, \quad \kappa_3 = \frac{k_3}{(k_1+k_3)}$$

This is the **DSolve** routine for this network of reactions. We have used **Φi** to denote the dimensionless concentration of component **i**. One more routine is added here. We have nested the **DSolve** routine in the **Simplify** function "**Simplify[DSolve[<>]]**." This ensures that the output statement corresponds to an algebraically reduced form.

```
In[143]:= Clear["Global'*"]

          Simplify[DSolve[{Φa'[τ] == -Φa[τ], Φb'[τ] == κ1Φa[τ]
             - κ2Φb[τ], Φd'[τ] == κ2Φb[τ], Φe'[τ] == κ3Φa[τ],
             Φa[0] == Φao, Φb[0] == Φbo, Φd[0] == Φdo,
             Φe[0] == Φeo}, {Φa[τ], Φb[τ], Φd[τ], Φe[τ]}, τ]]

          General::spell : Possible spelling error: new symbol
           name "Φe" is similar to existing symbols { Φa, Φb, Φd}.

          General::spell : Possible spelling error: new symbol
           name "Φbo" is similar to existing symbols {Φao, Φb}.

          General::spell : Possible spelling error: new symbol
           name "Φdo" is similar to existing symbols {Φao, Φbo, Φd}.

          General::stop : Further output of General::spell
           will be suppressed during this calculation.
```

Out[144]=

$$\{\{\Phi a[\tau] \to e^{-\tau}\Phi ao,$$

$$\Phi b[\tau] \to \frac{e^{-(1+\kappa 2)\tau}(e^{\kappa 2\tau}\kappa 1\Phi ao - e^{\tau}(\kappa 1\Phi ao + \Phi bo - \kappa 2\Phi bo))}{-1+\kappa 2},$$

$$\Phi d[\tau] \to \frac{(e^{-(1+\kappa 2)\tau}(-e^{\kappa 2\tau}\kappa 1\kappa 2\Phi ao + e^{\tau}(\kappa 1\Phi ao + \Phi bo - \kappa 2\Phi bo) + e^{\tau+\kappa 2\tau}(-1+\kappa 2)(\kappa 1\Phi ao + \Phi bo + \Phi do)))}{-1+\kappa 2},$$

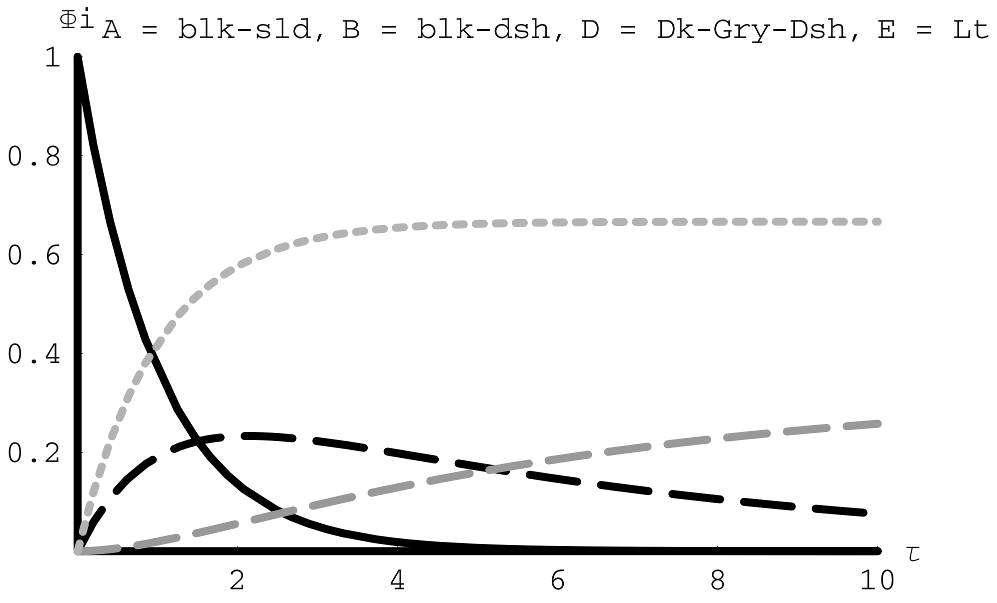
$$\Phi e[\tau] \to \kappa 3(\Phi ao - e^{-\tau}\Phi ao) + \Phi eo\}\}$$

These are the output statements. We note that the loss of **A** from the systems goes as a typical exponential decay, but recall that τ is made nondimensional as the product of real time and the sum of the rate constants for the two reactions that consume **A**. If there were three **A**-consuming reactions, then we would use the sum of all three. If there were reactions consuming **A** and reactions producing **A** simultaneously, then we would still take the sums of the rate constants, but the signs would be positive and negative. Thus we would have a sum and difference in the argument leading to τ.

It is also noteworthy that the stoichiometry will be controlled by the rate constants $\mathbf{k_1}$ and $\mathbf{k_3}$. This is clear and evident in the expression for **Φe** [τ]. If **Φeo** is zero, then at large τ, **Φe[**τ**]** $\to \kappa_3$**Φao**, where $\kappa_3 = \frac{\kappa_1}{(\kappa_1+\kappa_3)}$, the ratio of $\mathbf{k_3}$ to the sum of $\mathbf{k_1}$ and $\mathbf{k_3}$. This ratio κ_3 is also a measure of the *selectivity* of the reaction network.

In what follows in In statement [145] and the graph, we have assigned values to the parameters of the system. The rate constant $\mathbf{k_1}$ has been set to unity for simplicity and all the others are set in relation to it. In this case, the rate constant of the third step, which leads to **E**, is set at twice the value of that of the step leading to **B**. The rate constant between **B** and **D** is taken as half the magnitude of $\mathbf{k_1}$. The initial concentration of **A** is unity and zero for the other species. The solutions derived from **DSolve** are implemented as local functions **Φi[**τ**_]**.

```
In[145]:= Clear["Global‘*"]

          k1 = 1.;

          k2 = 0.5 k1;

          k3 = 2 k1;

          τmax = 10;

                  k1
          κ1 = ─────────;
               (k1 + k3)

                  k2
          κ2 = ─────────;
               (k1 + k3)

                  k3
          κ3 = ─────────;
               (k1 + k3)

          Φao = 1;

          Φbo = Φdo = Φeo = 0;

          Φa[τ_] := E^-τ Φao

                        1
          Φb[τ_] := ────────(E^-(1+κ2)τ (-(E^τ - E^κ2τ)κ1Φao
                    -1 + κ2
           + E^τ(-1 + κ2)Φbo))

                        1
          Φd[τ_] := ──────── (κ1(-1 + E^-κ2τ + κ2 - E^-τ κ2)Φao
                    -1 + κ2
           + E^-κ2τ(-1 + κ2) ((-1 + E^κ2τ)Φbo + E^κ2τ Φdo))
          Φe[τ_] := κ3(Φao - E^-τ  Φao) + Φeo

          SetOptions[Plot, DefaultFont → {"Helvetica", 12}];

          Plot[{Φa[τ], Φb[τ], Φd[τ], Φe[τ]}, {τ, 0, τmax},
            PlotRange → {{0, τmax}, {0, 1}},
            AxesLabel → {"τ", "Φi"},
            PlotStyle →
             {
              {Thickness[.01], Dashing[{0, 0}]},
              {Thickness[.01], Dashing[{0.06, 0.03}]},
              {Thickness[.01], Dashing[{0.05, 0.025}],
                GrayLevel[0.6]},
              {Thickness[.01], Dashing[{0.01, 0.015}],
                GrayLevel[0.7]}
             },
            PlotLabel → {"   A = blk-sld”, ”B = blk-dsh”,
          "D = Dk-Gry-Dsh", "E = Lt-Gry-Dsh"}];
```



Because **A** is consumed rapidly by the two pathways to **B** and **E**, its concentration profile drops sharply with time. The concentration of **E** rises very rapidly in response to the drop in **A**, but **B** lags behind. The reason is that **B** is not only formed more slowly, but as it is formed it is depleted by reaction to produce **D**, albeit at a comparably slower rate than the other reactions. The ultimate products **E** and **D** finally reach constant values over a long time and their magnitudes are 0.66 and 0.33, each corresponding to the rate constant ratios as we predicted.

This is an illustration of the intricacies that can develop even in a very simple reaction network. Imagine the kind of complexity that arises in some petroleum processing steps that involve numerous reactant molecules and many potential pathways for reaction (thermal, acid-catalyzed, metal catalyzed . . .). This is what reaction selectivity is all about and why chemists and engineers spend so much time dwelling on the topic. Nature has spent eons "dwelling on the topic" as well, and the result is reactions that ultimately are as highly specific as is possible. The critical factor in making this possible in natural systems is the enzyme catalyst with its "lock and key" mechanism for rejecting unwanted substrates (reactants) and driving to specific products and all at ambient temperature, where the rates of most chemical reactions as we know from Arrhenius ($k = A \exp(-Ea/RT)$) are relatively low. A high degree of molecular specificity or *molecular recognition* combined with slow but steady rates gives natural systems the advantage over the best man-made catalysts. This quest for selectivity is what drives so much fundamental and applied chemical research in catalysis and bio-technology.

Langmuir-Hinshelwood-Hougen-Watson Kinetics

In heterogeneous catalysis, the kinetics we use must account for the fact that the reaction takes place not in the gas phase but on the surface of the solid. Hence heterogeneous catalysis is also referred to as contact catalysis in the older literature. In fact reaction takes place in

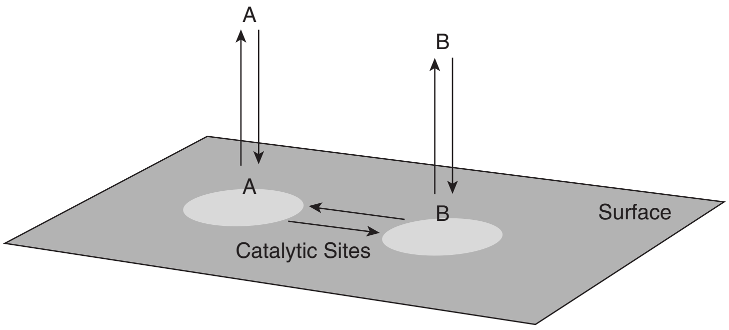


Figure 1

combination with adsorption. We have already seen how adsorption can be treated from the point of view of mass action. Now we need to couple the adsorption with the mass action kinetics for the surface reaction. To do this we will assume that the rates of adsorption and desorption are fast compared to the rates of surface chemical reaction. This is a good assumption for many cases, but not all. To go into the cases where adsorption or desorption rates limit the rate of chemical reaction would be to go beyond the bounds of the present discussion.

We will consider first the case of a simple surface reaction that takes **A** into **B**, for example, an isomerization. The reactant **A** adsorbs onto a site where it reacts at that site to form **B**; then **B** desorbs to the gas phase, relinquishing the site for another round of reaction. This is pictured on two equivalent sites in the schematic shown in Figure 1.

Given that adsorption and desorption of **A** and **B** are at the same site, they are in essence competing for the sites. We account for this is in the adsorption rate term, as shown for **A** in what follows:

$$r_{A,ads} = k_{A,ads}C_A[C_{tot} - C_{A,surf} - C_{B,surf}] - k_{A,des}C_{A,surf}$$

At adsorption-desorption equilibrium this rate goes to zero. Then we have:

$$k_{A,ads}C_A[C_{tot} - C_{A,surf} - C_{B,surf}] = k_{A,des}C_{A,surf}$$

$$C_A\frac{k_{A,ads}}{k_{A,des}} = \frac{C_{A,surf}}{(C_{tot} - C_{A,surf} - C_{B,surf})}$$

$$C_AK_A = \frac{C_{A,surf}}{(C_{tot} - C_{A,surf} - C_{B,surf})}\frac{C_{tot}}{C_{tot}}$$

$$C_AK_A = \frac{C_{A,surf}}{C_{tot}}\frac{C_{tot}}{(C_{tot} - C_{A,surf} - C_{B,surf})}$$

$$C_AK_A = \theta_A\frac{1}{(1 - \theta_A - \theta_B)}$$

where θ_A stands for the fraction of the total sites occupied by **A** on the surface and the same meaning is attributed to θ_B for species **B**. Repeating this analysis for species **B** we find:

$$C_B K_B = \theta_B \frac{1}{(1-\theta_A-\theta_B)}$$

We can solve the two equations simultaneously for θ_A and for θ_B to get them both in terms of the gas-phase concentrations and the adsorption constants for each:

```
In[161]:= Clear["Global`*"]
          θ =.
          K =.
          Solve[{C_A K_A == θ_A 1/(1 - θ_A - θ_B),
            C_B K_B == θ_B 1/(1 - θ_A - θ_B)}{θ_A, θ_B}]

Out[164]= {{θ_A → C_A K_A/(1 + C_A K_A + C_B K_B), θ_B → C_B K_B/(1 + C_A K_A + C_B K_B)}}
```

The surface reaction is reversible and is first order in the surface concentrations of **A** and of **B**:

$$r_{A\to B,surf} = k_{A\to B,surf} C_{A,surf} - k_{B\to A,surf} C_{B,surf}$$

Multiplying through by $\frac{C_{tot}}{C_{tot}}$ provides these expressions in terms of the fractional surface concentrations:

$$r_{A\to B,surf} = k_{A\to B,surf} C_{tot} \theta_A - k_{B\to A,surf} C_{tot} \theta_B$$

Replacing with the expressions for the fractional surface concentrations:

$$r_{A\to B,surf} = \frac{k_{A\to B,surf} C_{tot} C_A K_A - k_{B\to A,surf} C_{tot} C_B K_B}{1 + C_A K_A + C_B K_B}$$

The reversible surface reaction has associated with it an equilibrium constant, which is just the ratio of the forward to the reverse surface rate constants:

$$K_{A\Leftrightarrow B,surf} = \frac{k_{A\to B,surf}}{k_{B\to A,surf}}$$
$$\therefore k_{B\to A,surf} = \frac{k_{A\to B,surf}}{K_{A\Leftrightarrow B,surf}}$$

Making this substitution and factoring out the product of the forward surface rate constant and the total surface concentration of sites:

$$r_{A\to B,surf} = \frac{k_{A\to B,surf}C_{tot}\left(C_AK_A - \frac{C_BK_B}{K_{A\Leftrightarrow B,surf}}\right)}{1 + C_AK_A + C_BK_B}$$

On the far right of the numerator we have two parameters that may be difficult to obtain independently. They are the adsorption equilibrium constant for **B** and the surface equilibrium constant for the reaction $\boldsymbol{A_{surf}} \Leftrightarrow \boldsymbol{B_{surf}}$. Our goal is to clear these by reexpressing them in terms of something that is unchanging. After all both of these may be strong functions of the catalyst structure and composition. The overall reaction **A** $\Leftrightarrow$ **B** is, however, one which is fixed at any temperature and pressure by the overall equilibrium constant. *This is independent of the catalyst.* Therefore we want to use this in the reaction rate expression. Here is how we do it:

$$\begin{aligned} K_{eq} = \frac{C_B}{C_A} &= \frac{C_B}{C_{B,surf}}\frac{C_{B,surf}}{C_{A,surf}}\frac{C_{A,surf}}{C_A} \\ &= \frac{1}{K_B}K_{A\Leftrightarrow B,surf}K_A \\ \therefore \frac{K_B}{K_{A\Leftrightarrow B,surf}} &= \frac{K_A}{K_{eq}} \end{aligned}$$

and

$$r_{A\to B,surf} = \frac{k_{A\to B,surf}C_{tot}\left(C_AK_A - \frac{C_BK_A}{K_{eq}}\right)}{1 + C_AK_A + C_BK_B}$$

$$r_{A\to B,surf} = \frac{k_{A\to B,surf}K_AC_{tot}\left(C_A - \frac{C_B}{K_{eq}}\right)}{1 + C_AK_A + C_BK_B}$$

$$r_{A\to B,surf} = \frac{k_{f,AB}\left(C_A - \frac{C_B}{K_{eq}}\right)}{1 + C_AK_A + C_BK_B}$$

The product $\mathbf{k_{A\to B,surf}K_AC_{tot}}$ is usually taken as the "global" forward rate constant on the surface $\mathbf{k_{f,AB}}$. Now we can proceed to see how this equation behaves.

Consider a batch reactor of volume **V** into which the catalyst that does the conversion of **A** to **B** has been placed. The catalyst occupies a fraction $(1 - \epsilon)$ of the reactor volume.

The component balances for **A** and **B** are:

$$\frac{d[C_A \epsilon V]}{dt} = -(1-\epsilon)\frac{k_{f,AB}\left(C_A - \frac{C_B}{K_{eq}}\right)}{1 + C_A K_A + C_B K_B} V$$

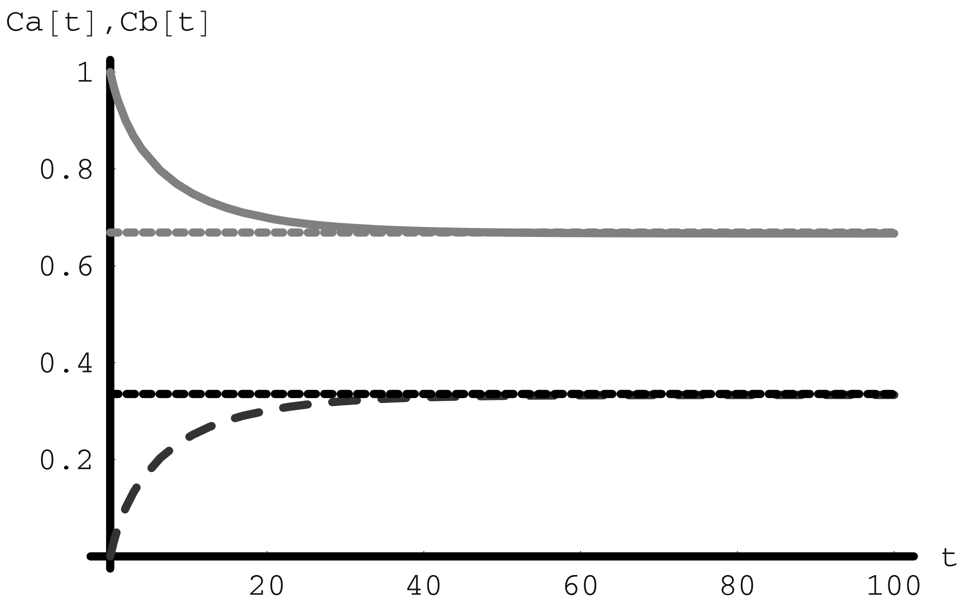
$$\frac{d[C_B \epsilon V]}{dt} = +(1-\epsilon)\frac{k_{f,AB}\left(C_A - \frac{C_B}{K_{eq}}\right)}{1 + C_A K_A + C_B K_B} V$$

In the following cell we compute the two concentrations as functions of time, and in the **Epilog** we compute the equilibrium levels of **A** and **B** and graph horizontal lines corresponding to each. The equilibrium level of reaction in this case is given by:

$$\text{Keq} = \frac{\alpha}{1-\alpha}$$

```
In[165]:= Clear["Global'*"]

In[166]:= ϵ = 0.4;
          kf = 10^-1;
          Ka = 1;
          Kb = 10;
          Keq = .5;
          Cao = 1;
          Cbo = 0;
          tmax = 100;
          LHHW1 = NDSolve[
              {Ca'[t] == -(1 - ϵ)/ϵ  kf (Ca[t] - Cb[t]/Keq)/(1 + Ka Ca[t] + Kb Cb[t]),
               Cb'[t] == +(1 - ϵ)/ϵ  kf (Ca[t] - Cb[t]/Keq)/(1 + Ka Ca[t] + Kb Cb[t]),
               Ca[0] == Cao,
               Cb[0] == Cbo},
             {Ca[t], Cb[t]},
             {t, 0, tmax}];
          CA[t_] := Evaluate[Ca[t] /. LHHW1]
          CB[t_] := Evaluate[Cb[t] /. LHHW1]
          SetOptions[Plot, DefaultFont → {"Hevetica", 12},
           AxesStyle → {Thickness[0.01]}];
```

```
Plot[{CA[t], CB[t]},
  {t, 0, tmax},
  AxesLabel → {”t”, ”Ca[t],Cb[t]”},
  PlotStyle → {{Thickness[0.01], GrayLevel[0.5]},
    {Dashing[{0.03, 0.03}], Thickness[0.01],
     GrayLevel[0.2]}},
  Epilog →
          {
    {Thickness[0.01], Dashing[{0.01, 0.01}],
     Line[{{0, Flatten[NSolve[Keq == α/(1 - α), α]]
        [[1, 2]] + .002},
       {tmax, Flatten[NSolve[Keq == α/(1 - α), α]]
        [[1, 2]] + .002}}
         ]},
    {Thickness[0.01], GrayLevel[0.5],
     Dashing[{0.01, 0.01}],
    Line[{{0, (1 - Flatten[NSolve[Keq == α/(1 - α), α]]
      [[1, 2]] + .002)},
       {tmax, (1 - Flatten[NSolve[Keq == α/(1 - α), α]]
       [[1, 2]] + .002)}}
         ]}
           }
         ];
```



The hydrogenation and dehydrogenation of alkenes and alkanes are reversible processes that favor the alkane and involve multiple sites. For this reason it is worth considering a prototypical general case to see how we progress with the LHHW analysis.

The reaction proceeding from alkene to alkane can be taken as the forward direction as it is the thermodynamically favored direction. The overall reaction is:

$$A + H_2 \Longleftrightarrow B$$

The individual steps including the sites ($\otimes$) are as follows:

$$\begin{array}{rl} A + \otimes \Longleftrightarrow A\otimes & \text{Adsorption/Desorption} \\ H_2 + 2\otimes \Longleftrightarrow 2H\otimes & \text{Dissociative Adsorption/Desorption} \\ A\otimes + 2H\otimes \Longleftrightarrow B\otimes + 2\otimes & \text{Surface Reaction} \\ B\otimes \Longleftrightarrow B + \otimes & \text{Adsorption/Desorption} \end{array}$$

In this mechanism the dihydrogen molecule must dissociatively adsorb prior to reacting with the alkene **A**. This requires two sites (see Figure 2). When the surface reaction takes place to convert the alkene and two hydrogen atoms into one alkane, the two sites are regenerated. Therefore, we need to examine how the dissociative adsorption step is handled and what ramification this has upon the rate expression, assuming all the adsorption-desorption steps are at equilibrium.

The dissociative adsorption-desorption of hydrogen follows this rate expression:

$$r_{H2,ads} = k_{H2,ads}C_{H2}(C_{tot} - C_{H,surf} - C_{A,surf} - C_{B,surf})^2 - k_{H,des}C_{H,surf}^2$$

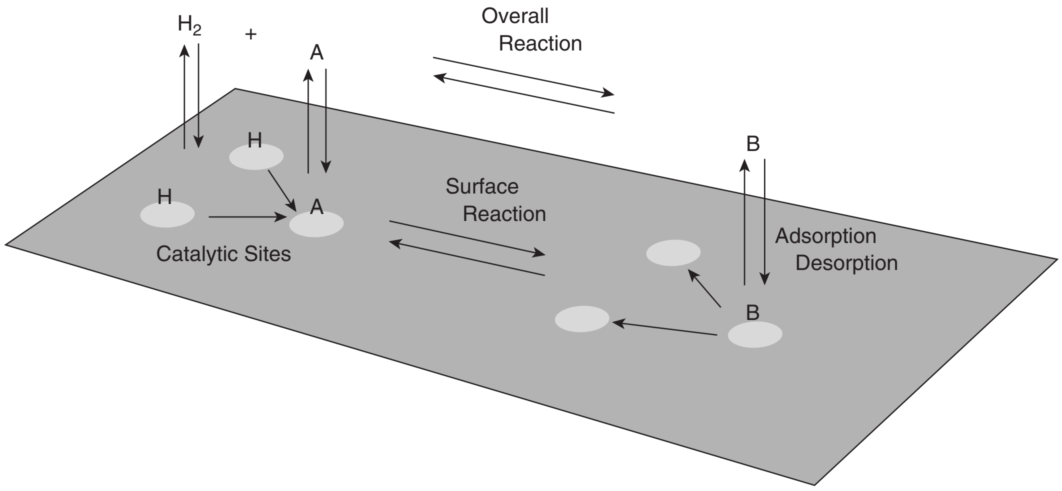


Figure 2

The adsorption includes the difference between the total concentration of sites and the sites occupied by hydrogen atoms **A** and **B**. Notice also that this term is squared because there are two sites involved in the adsorption process. On the desorption side of the expression we see that the rate depends upon the square of the concentration of surface hydrogen atoms. Once again we assume this step and the other adsorption-desorption steps to be at equilibrium:

$$k_{H2,ads}C_{H2}(C_{tot} - C_{H,surf} - C_{A,surf} - C_{B,surf})^2 = k_{H,des}C_{H,surf}^2$$

$$\frac{k_{H2,ads}}{k_{H,des}}C_{H2} = \frac{C_{H,surf}^2}{(C_{tot} - C_{H,surf} - C_{A,surf} - C_{B,surf})^2}$$

$$K_{H2,ads}C_{H2} = \frac{C_{H,surf}^2}{(C_{tot} - C_{H,surf} - C_{A,surf} - C_{B,surf})^2}\frac{C_{tot}^2}{C_{tot}^2}$$

$$K_{H2,ads}C_{H2} = \left(\frac{C_{H,surf}^2}{C_{tot}^2}\right)\left(\frac{C_{tot}^2}{(C_{tot} - C_{H,surf} - C_{A,surf} - C_{B,surf})^2}\right)$$

$$K_{H2,ads}C_{H2} = \frac{\theta_{H,surf}^2}{(1 - \theta_{H,surf} - \theta_{A,surf} - \theta_{B,surf})^2}$$

$$\sqrt{K_{H2,ads}C_{H2}} = \frac{\theta_{H,surf}}{(1 - \theta_{H,surf} - \theta_{A,surf} - \theta_{B,surf})}$$

From the same analyses of the adsorption-desorption processes for **A** and **B** we find:

$$K_{A,ads}C_A = \frac{\theta_{A,surf}}{(1 - \theta_{H,surf} - \theta_{A,surf} - \theta_{B,surf})}$$

$$K_{B,ads}C_B = \frac{\theta_{B,surf}}{(1 - \theta_{H,surf} - \theta_{A,surf} - \theta_{B,surf})}$$

The term for **B** is written in the form that it would have if **B** were adsorbing in order to keep the meaning $\mathbf{K}_B$ uniform with the other adsorption constants. We can solve for the fractional surface concentrations to get:

```
In[179]:= Clear["Global‘*"]

In[180]:= θ =.
          K =.

          Simplify[
           Solve[
                                θ_A
            {C_A K_A ==  ─────────────────────,
                          1 - θ_H - θ_A - θ_B
```

```
        C_B K_B == θ_B / (1 - θ_H - θ_A - θ_B),
        √(K_H2 C_H2) == θ_H / (1 - θ_H - θ_A - θ_B)},
        {θ_A, θ_B, θ_H}]
            ]

Out[182]= {{θ_A → C_A K_A / (1 + C_A K_A + C_B K_B + √(C_H2 K_H2)),
            θ_B → C_B K_B / (1 + C_A K_A + C_B K_B + √(C_H2 K_H2)),
            θ_H → C_H2 K_H2 / (C_H2 K_H2 + (1 + C_A K_A + C_B K_B) √(C_H2 K_H2))}}
```

The surface reaction rate begins to take shape:

$$r_{A+2H\Leftrightarrow B,surf} = k_{A+2H\rightarrow B,surf}C_{A,surf}C^2_{H,surf} - k_{B\rightarrow A+2H,surf}C_{B,surf}C^2_{empty,sites}$$

$$r_{A+2H\Leftrightarrow B,surf} = k_{A+2H\rightarrow B,surf}C_{A,surf}C^2_{H,surf}\left(\frac{C^3_{tot}}{C^3_{tot}}\right) - k_{B\rightarrow A+2H,surf}C_{\mathbf{B},surf}C^2_{empty,sites}\left(\frac{C^3_{tot}}{C^3_{tot}}\right)$$

$$r_{A+2H\Leftrightarrow B,surf} = k_{A+2H\rightarrow B,surf}C^3_{tot}\theta_A\theta_{H^2} - k_{B\rightarrow A+2H,surf}C^3_{tot}\theta_{\mathbf{B}}\theta^2$$

$$r_{A+2H\Leftrightarrow B,surf} = k_{A+2H\rightarrow B,surf}C^3_{tot}\theta_A\theta_{H^2} - k_{B\rightarrow A+2H,surf}C^3_{tot}\theta_{\mathbf{B}}(1-\theta_H-\theta_A-\theta_B)^2$$

$$r_{A+2H\Leftrightarrow B,surf} = k_{A+2H\rightarrow B,surf}C^3_{tot}\left(\theta_A\theta^2_H - \frac{Q_{\mathbf{B}}(1-\theta_H-\theta_A-\theta_B)^2}{K_{A+2H\Leftrightarrow B,surf}}\right)$$

This begins to look a bit formidable because of the algebra that would be involved in manipulating this expression. We will let *Mathematica* do most of the algebraic manipulations by following these steps that use **PowerExpand[ ]** to expand the higher-order terms **Together[ ]**, which brings separate terms over the same denominator and **FullSimplify[ ]**, which does just that. Here is the result of this approach to the parenthetical expression of the right-hand side of the rate expression:

```
In[183]:= θ_A = C_A K_A / (1 + C_A K_A + C_B K_B + √(C_H2 K_H2));
          θ_B = C_B K_B / (1 + C_A K_A + C_B K_B + √(C_H2 K_H2));
          θ_H = C_H2 K_H2 / (C_H2 K_H2 + (1 + C_A K_A + C_B K_B) √(C_H2 K_H2));
```

```
FullSimplify[
        Together[
        PowerExpand[θA θH^2 - θB(1 - θH - θA - θB)^2 / K_{A + 2H ⇔ B, surf}]
               ]
           ]
```

Out[186]= $\frac{-C_B K_B + C_A C_{H2} K_A K_{H2} K_{A+2H\Leftrightarrow B,surf}}{(1 + C_A K_A + C_B K_B + \sqrt{C_{H2}}\sqrt{K_{H2}})^3 K_{A+2H\Leftrightarrow B,surf}}$

We can manipulate this into the form that we are interested in as follows:

$$r_{A+2H\Leftrightarrow B,surf} = k_{A+2H\to B,surf} C_{tot}^3 \frac{(C_A C_{H2} K_A K_{H2} - \frac{C_B K_B}{K_{A+2H\Leftrightarrow B,surf}})}{(1 + C_A K_A + C_B K_B + \sqrt{C_{H2} K_{H2}})^3}$$

$$\begin{aligned} K_{eq} &= \frac{C_B}{C_A C_{H2}} = \frac{C_B}{C_{B,surf}} \frac{C_{B,surf}}{C_{A,surf}} \frac{C_{A,surf}}{C_A} \frac{C_{H,surf}^2}{C_{H,surf}^2} \frac{1}{C_{H2}} \\ &= \frac{C_B}{C_{B,surf}} \frac{C_{B,surf}}{C_{A,surf}} \frac{C_{A,surf}}{C_A} \frac{1}{C_{H,surf}^2} \frac{C_{H,surf}^2}{C_{H2}} \\ &= \frac{C_B}{C_{B,surf}} \frac{C_{B,surf}}{C_{A,surf} C_{H,surf}^2} \frac{C_{A,surf}}{C_A} \frac{C_{H,surf}^2}{C_{H2}} \\ &= \frac{K_{A+2H\Leftrightarrow B,surf} K_A K_{H2}}{K_B} \\ \therefore \frac{K_B}{K_{A+2H\Leftrightarrow B,surf}} &= \frac{K_A K_{H2}}{K_{eq}} \end{aligned}$$

$$r_{A+2H\Leftrightarrow B,surf} = k_{A+2H\to B,surf} K_A K_{H2} C_{tot}^3 \frac{(C_A C_{H2} - \frac{C_B}{K_{eq}})}{(1 + C_A K_A + C_B K_B + \sqrt{C_{H2} K_{H2}})^3}$$

$$r_{A+2H\Leftrightarrow B,surf} = k_{global} \frac{(C_A C_{H2} - \frac{C_B}{K_{eq}})}{(1 + C_A K_A + C_B K_B + \sqrt{C_{H2} K_{H2}})^3}$$

Now we can write some code that will evaluate the kinetics and the equilibrium and then graph the relevant gas and surface phase concentrations for us all at once. The equilibrium extent of reaction can be computed as follows for any given value of the equilibrium constant:

```
In[187]:= Keq = .5;
          α =.
          Cao = 1;

          NSolve[Keq == α / Cao(1 - α)^2, α]

Out[190]= {{α → 3.73205}, {α → 0.267949}}
```

We see that the second of the two evaluations is the correct one. We imbed this into the **Epilog** as the *y*-coordinate of a line for the equilibrium concentration of the product at time zero and tmax, and we take one minus this value to get the line corresponding to the equilibrium concentration of reactant. By doing it this way we can set the magnitude of the equilibrium constant and the code will automatically compute these concentrations for the graph. This allows us to visualize immediately where the concentrations are at any time relative to the equilibrium concentrations:

```
In[191]:= Clear["Global'*"]

In[192]:= ϵ = .4;

          kglo = .01;

          Ka = .1;

          Kb = .01;

          KH2 = .5;

          Keq = .5;

          Cao = 1;

          CH2o = 1;

          Cbo = 0;

          tmax = 500;

          LHHW2 = NDSolve[
             {∂tCa[t] ==
```

$$-\frac{(1-\epsilon)}{\epsilon}\frac{\mathrm{kglo}\left(\mathrm{Ca[t]}\,\mathrm{CH2[t]}-\frac{\mathrm{Cb[t]}}{\mathrm{Keq}}\right)}{\left(1+\mathrm{Ka}\,\mathrm{Ca[t]}+\mathrm{Kb}\,\mathrm{Cb[t]}+\sqrt{\mathrm{KH2}\,\mathrm{CH2[t]}}\right)^3},$$

```
              ∂tCH2[t] ==
```

$$-\frac{(1-\epsilon)}{\epsilon}\frac{\mathrm{kglo}\left(\mathrm{Ca[t]}\,\mathrm{CH2[t]}-\frac{\mathrm{Cb[t]}}{\mathrm{Keq}}\right)}{\left(1+\mathrm{Ka}\,\mathrm{Ca[t]}+\mathrm{Kb}\,\mathrm{Cb[t]}+\sqrt{\mathrm{KH2}\,\mathrm{CH2[t]}}\right)^3},$$

```
              ∂tCb[t] ==
```

$$+\frac{(1-\epsilon)}{\epsilon}\frac{\mathrm{kglo}\left(\mathrm{Ca[t]}\,\mathrm{CH2[t]}-\frac{\mathrm{Cb[t]}}{\mathrm{Keq}}\right)}{\left(1+\mathrm{Ka}\,\mathrm{Ca[t]}+\mathrm{Kb}\,\mathrm{Cb[t]}+\sqrt{\mathrm{KH2}\,\mathrm{CH2[t]}}\right)^3},$$

```
             Ca[0] == Cao,
             CH2[0] == CH2o,
             Cb[0] == Cbo},
             {Ca[t], CH2[t], Cb[t]},
             {t, 0, tmax}];
```

```
           cA[t_] := Evaluate[Ca[t] /. LHHW2[[1]]]
           cH2[t_] := Evaluate[CH2[t] /. LHHW2[[1]]]
           cB[t_] := Evaluate[Cb[t] /. LHHW2[[1]]]
           cA[t]
           cH2[t]
           cB[t]

           General::spell1 : Possible spelling error: new symbol
            name "cH2" is similar to existing symbol "CH2".

Out[206]= InterpolatingFunction[{{0., 500.}}, <>][t]

Out[207]= InterpolatingFunction[{{0., 500.}}, <>][t]

Out[208]= InterpolatingFunction[{{0., 500.}}, <>][t]
```

In[209]:= $\theta A[t\_] := \dfrac{Ka\,cA[t]}{1 + Ka\,cA[t] + Kb\,cB[t] + \sqrt{KH2\,cH2[t]}}$

$\theta H[t\_] := \dfrac{KH2\,cH2[t]}{1 + Ka\,cA[t] + Kb\,cB[t] + \sqrt{KH2\,cH2[t]}}$

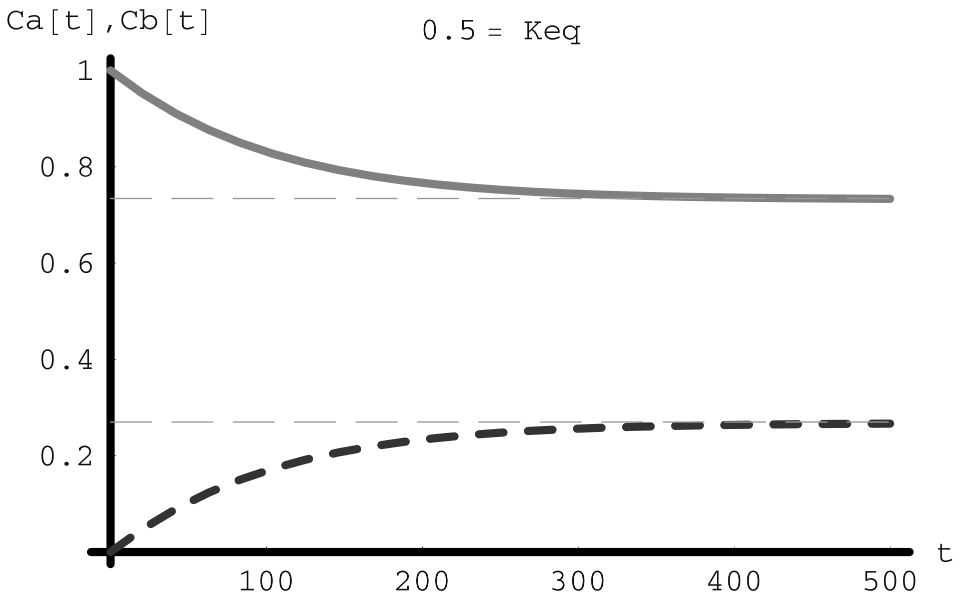
$\theta_A[t]$
$\theta_H[t]$

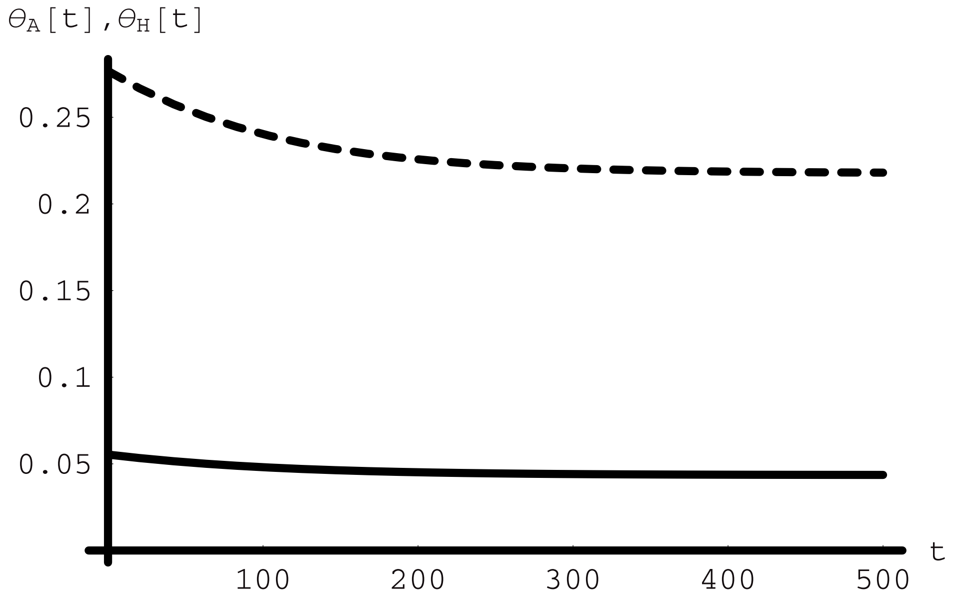
```
           General::spell : Possible spelling error: new symbol
            name "θA" is similar to existing symbols {θ, ΦA}.

           General::spell : Possible spelling error: new symbol
            name "θH" is similar to existing symbols {θ, θA}.
```

Out[211]= $\dfrac{C_A K_A}{1 + C_A K_A + C_B K_B + \sqrt{C_{H2} K_{H2}}}[t]$

Out[212]= $\dfrac{C_{H2} K_{H2}}{C_{H2} K_{H2} + (1 + C_A K_A + C_B K_B)\sqrt{C_{H2} K_{H2}}}[t]$

```
In[213]:= SetOptions[Plot, DefaultFont → {"Helvetica", 12},
            AxesStyle → {Thickness[0.01]}];
          Plot[{cA[t], cB[t]}, {t, 0, tmax},
            AxesLabel → {"t", "Ca[t],Cb[t]"},
            PlotStyle → {{Thickness[0.01], GrayLevel[0.5]},
              {Dashing[{0.03, 0.03}], Thickness[0.01],
                GrayLevel[0.2]}},
            PlotLabel → Keq "= Keq",
            Epilog → {
              {GrayLevel[0.6], Dashing[{0.05, 0.025}],
          Line[{{0, Flatten[NSolve[Keq == α/(1 - α)^2, α]]
           [[2, 2]] + .002},
           {tmax, Flatten[NSolve[Keq == α/(1 - α)^2, α]]
             [[2, 2]] + .002}}
                 ]},
        {GrayLevel[0.6], Dashing[{0.05, 0.025}],
          Line[{{0, (1 - Flatten[NSolve[Keq == α/(1 - α)^2, α]]
           [[2, 2]] + .002)},
           {tmax, (1 - Flatten[NSolve[Keq == α/(1 - α)^2, α]]
             [[2, 2]] + .002)}}
                 ]}
                    }
                 ];
Plot[{θA[t], θH[t]}, {t, 0, tmax},
   AxesLabel → {"t", "θA[t],θH[t]"},
   PlotStyle →
     {Thickness[0.01], Dashing[{0, 0}]},
     {Thickness[0.01], Dashing[{0.02, 0.02}]}}];
```





The next step could be to make the preceding into a **Module** so that we can test parametric sensitivity readily.

Microbial Population Dynamics

At present there is an unprecedented research explosion in the biological sciences. The breakthroughs in the basic sciences of genomics and related disciplines have brought us to the threshold of a new era in biological technology. Paramount to this new technology is the use of microbes (that is, cellular organisms) as reactors. Organisms have evolved mechanisms for dealing with environmental stress (such as the presence of a new substrate chemical in their surroundings) by rerouting their metabolic pathways. Metabolic engineers can take advantage of this through a procedure of accelerated adaptation in order to generate new microbes that consume a given substrate and produce a specific target chemical.

Microbes use enzymes as catalysts to obtain the desired or beneficial reaction and typically under mild conditions. The brewing of beer and fermentation of fruit and vegetable mass high in starches to produce consumable ethanol are the oldest and most familiar examples of using microbial action to achieve a desired end. But now much more has been demonstrated, from the production of essential human hormones to the synthesis of specialty chemicals.

In a reactor containing substrate a colony of microbes is innoculated and brought to maturity. As the colony grows the substrate is consumed to supply the microbes with their building blocks. Some fraction of the substrate is necessarily diverted into the formation of biomass (that is, cells—their membranes and organelles) but some other fraction is used to produce the target molecule. In a batch process when the substrate has been consumed, the microbial colony either dies rapidly or if the process is to be stopped prior to complete substrate consumption, it is killed by a rapid change in conditions (for example, by raising the temperature as is done in the pasteurization of raw milk). From this point the problem of recovering the target molecule is one of separating it from the biomass and aqueous medium.

The basis of life is molecular. Therefore we can describe the rates of substrate consumption, product formation, and even microbe population growth in much the same way that we would describe the rates of molecular-level chemical processes.

We will take the microbe, substrate, and product concentrations to be **a[t], b[t], c[t]**, respectively. The ways in which these kinetics are written are somewhat different. The equations that describe the rates of change of each of these are shown in the following:

$$a'[t] == \left(\mu\mathrm{max}\frac{b[t]}{Ks + b[t]} - k\right)a[t],$$
$$b'[t] == -\left(\frac{\mu\mathrm{max}}{ys}\frac{b[t]}{Ks + b[t]}\right)a[t],$$
$$c'[t] == \left(\alpha + \beta\ \mu\mathrm{max}\frac{b[t]}{Ks + b[t]}\right)a[t],$$

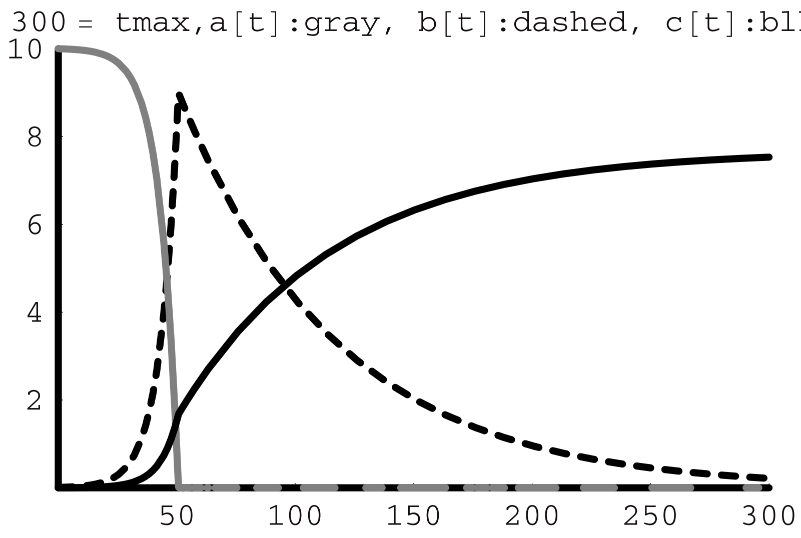
The kinetic expressions are highly nonlinear because they include the following rate term:

$$\mu\mathrm{max} = \frac{b[t]}{Ks + b[t]}a[t]$$

where **μmax** is a maximum rate constant, **Ks** is a saturation concentration, and **ys** is a dimensionless parameter that is similar to a stoichiometric coefficient. Likewise, α and β are dimensionless numbers that are also similar to stoichiometric coefficients; they relate the rate of production of the desired molecule to the rate of growth of microbial cell mass. In the cell that follows we build a model for these kinetics to examine how they behave:

```
In[216]:= Clear["Global‘*"]

In[217]:= μ = .15; "μmax";

          K = .04; "Ks";

          y = 1; "ys";

          α = 10^-n;

          n = 2;
```

```
β = .1;

tmax = 300;

k = 0.1;

"a[t] is the change in microbial concentration;
 decreasing";
"b[t] is the change in the substrate concentration;
 increasing";
"c[t] is the change in product concentration;
 increasing";

bugs1 = NDSolve[{
                              b[t]
             a'[t] == μ(─────────── - k)a[t],
                          K + b[t]
                        μ   b[t]
             b'[t] == - ─ ──────────a[t],
                        y K + b[t]
                                   b[t]
             c'[t] == (α + βμ ──────────)a[t],
                                K + b[t]
             a[0] == .01,
             b[0] == 10,
             c[0] == 0
             },
          {a[t], b[t], c[t]},
          {t, 0, tmax}];

a1[t_] := Evaluate[a[t] /. bugs1];
b1[t_] := Evaluate[b[t] /. bugs1];
c1[t_] := Evaluate[c[t] /. bugs1];

pa1 = Plot[a1[t], {t, 0, tmax},
   DisplayFunction → Identity,
   PlotStyle → {Thickness[0.01], Dashing[{0.02, 0.02}]},
   PlotRange → {{0, tmax}, {0, b1[0][[1]]}}];

pb1 = Plot[b1[t], {t, 0, tmax},
   DisplayFunction → Identity,
   PlotStyle → {Thickness[0.01], GrayLevel[0.5]}];

pc1 = Plot[c1[t], {t, 0, tmax},
   PlotStyle → {Thickness[0.01]},
   DisplayFunction → Identity];

Show[pa1, pb1, pc1, DisplayFunction → $DisplayFunction,
  PlotLabel → tmax ″= tmax,a[t]:gray, b[t]:dashed,
   c[t]:blk″];
```



```
Out[217]= Null
```
[16]

How do we interpret these results? The substrate concentration (gray) falls slowly at very short time after innoculation of the microbial colony. This is an induction period over which the colony grows slowly. After this induction time, the colony of microbes (dashed) suddenly grows "explosively" and reaches a maximum. At the same time that the explosive growth occurs the substrate is diminished at a precipitous rate. After the substrate is used up, the colony begins to diminish in number. This occurs in the present case at a much slower rate than their growth. During the period of explosive growth and shortly after the maximum is attained in microbial population, the product concentration (solid black) increases and then levels to a constant with time as the colony finally expires.

7.6 Summary

In this chapter we have covered a wide spectrum of chemical kinetics from the simplest rate laws with relatively straightforward forms and interpretations to those involving catalysts and enzymes, which are more complex and necessarily more abstruse. As complex as the kinetics may have been, we have throughtout this chapter assumed the most simplistic of chemical reactors—that of the batch reactor. Ironically, although we speak of the batch reactor as being simple, in fact its description as we have seen can be not at all simple due to the fully transient nature of the processes occurring within it. Interestingly, if we introduce flows of reactant and product to and from the control volume, we will find that the system will have a condition that we refer to as the steady-state condition that is totally independent of time. At steady state the flow reactor is simple to describe even though the reactor seems to be more complex. Before we analyze such systems we will first cover the semi- or fed-batch reactor, which involves flow of reactant into the system continuosly or intermittently. This type of reactor may attain a steady state in which case its mathematics are “simple,” but it may also operate transiently making the mathematics complex due to their time dependency. Whatever type of reactor we examine, in every case there will be some form of chemical reaction to consider and the rate of that reaction may be described using the rate laws and methods that we have developed in this chapter.

CHAPTER 8

Semi-Continuous Flow Reactors

8.1 Introduction to Flow Reactors

A batch reactor is useful for laboratory studies and the production of small quantities of materials. Its disadvantage is that each time it is used it must be charged, operated, and then discharged in three separate stages. The time spent in charging and discharging is time lost to production. Nonetheless, if the value of the product is very high and the production quantities required are low, then the batch system is often an optimal reactor choice. As it is not a dedicated unit, many different kinds of products can be scheduled and processed in the same unit. This can be done effectively for many pharmaceuticals and for some specialty chemicals. However, as the production requirements rise and the value-added in the product falls, efficient production is a must and the reactor must be used as continuously as possible. There should be relatively few shut-downs and the process should be operated continuously for as long as possible. With respect to reactor size and the need for process continuity, the petroleum refinery lies at one extreme of the spectrum with pharmaceutical production at the other.

There are three idealized flow reactors: *fed-batch* or *semibatch, continuously stirred tank, and the plug flow tubular.* Each of these is pictured in Figure 1. The fed-batch and continuously stirred reactors are both taken as being well mixed. This means that there is no spatial dependence in the concentration variables for each of the components. At any point within the reactor, each component has the same concentration as it does anywhere else. The consequence

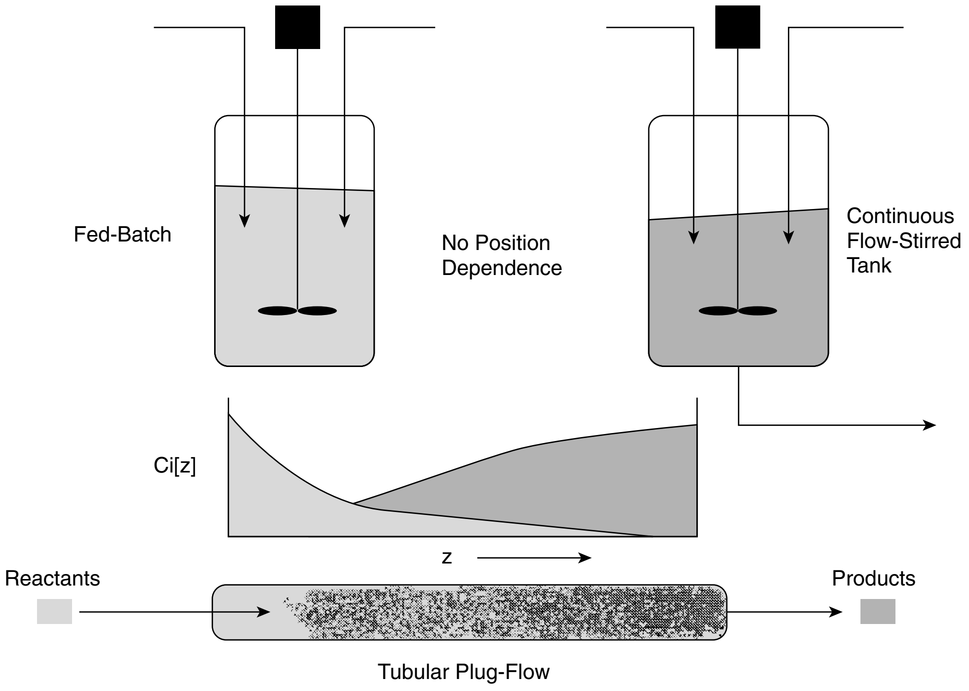


Figure 1

of this assumption is that as soon as the reactants cross the boundary from outside to inside of the reactor, their concentrations go from their feed stream values to the exit stream concentration values. This is true even if the reactor is considered to be operating transiently rather than in a steady state. The conversion of reactants is the same everywhere and, as we will see, it is set by the holding time in the reactor. Of course the exit stream has product and reactant concentrations that are exactly the same as those within the reactor. All this is a consequence of the mathematical assumption of perfect mixing.

In contrast to the first two reactors, concentrations within the tubular flow reactor are characterized by position dependence. When we assume *plug-flow*, we take the concentrations to be independent of their radial positions. (The axial direction z is along the horizontal axis in the diagram; the radial direction is taken from the central axis out to the wall and is perpendicular to the axial direction.) The term plug means that there is no concentration profile in the radial direction; the gas moves through the cylindrical tube as if it were a "plug" of material translating through the volume. So in this case we must account not only for time dependence, but also for position of the front of the plug of gas. Near the entrance of the tube the gas is nearly 100% reactant and at the exit it is a mix of reactant and product, if the conversion is $<100\%$. At axial positions between the two ends the gas is a mix of products and reactants. Our goal will be to predict how the mix changes as a function of position and flow parameters, that is, the holding time.

Some very interesting consequences of complete mixing versus partial mixing can be defined in terms of reactor efficiencies. For positive-order kinetics the fully mixed reactor will require a larger volume than the partially mixed tubular system to achieve the same conversion and at the same holding time. This is a very important result that requires using analysis to understand it. At the same time, although conversion may be higher, so too may selectivity be lower, if multiple reactions are involved. There is much to learn about the systems in which chemical reactions are conducted, even if we assume these systems to be at the extremes of ideal behavior.

8.2 Semicontinuous Systems

Fed-Batch Reactors

The fed-batch reactor is a special system that can be used whenever the need arises to carefully control the reaction rate in a batch system. For example, if a reaction is highly exothermic, then mixing the reactants at their full stoichiometric ratios can lead to uncontrollable temperature rises, which are referred to as thermal excursions. In simple terms the reaction produces heat at a rate that is faster than the rate at which heat can be transferred away from the vessel. As a result the temperature in the vessel rises. The higher temperature leads to faster reaction rates and even higher rates of heat production. And so it goes with the heat of reaction feeding back into the kinetics and the kinetics rising with the increased temperature. The outcome of a thermal excursion can be, at a minimum, reduced selectivity and, at its worst, total loss of control of the reacting system with dire consequences. This phenomenon is called reactor runaway and it can lead to detonation of the system. However, by adding one of the reactants slowly or intermittently, we can control the system and maintain good heat transfer away from the vessel. As the amount of reactant is limited, the rate of reaction proceeds at a much

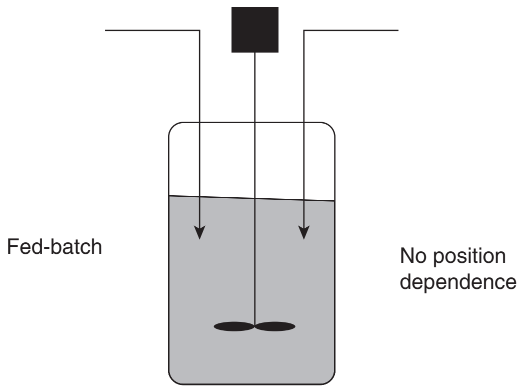


Figure 2

reduced average rate, which allows the rate of heat transfer to keep pace with the rate of reactions. This approach is well known and much utilized by synthetic chemists at the bench and it is also used for all the same reasons that a production chemist or engineer would use it on a larger scale. The concentration changes within the reactor are periodically changing if the mass and volume of reactant are added intermittently, but they become continuously variant in time, that is, transient, if the flow to the system is continuous. For example, bioreactors are in a very real sense fed-batch systems in that oxygen may be fed continuously to the microbial colony for its sustenance, even if the substrate is fully charged at the beginning of the batch, since the volume of solution changes very little throughout the course of the process.

If the rate of the irreversible chemical reaction of A and B to form D is given by r_{AB}, and the flow rates of reactants are given by q_A and q_B with corresponding feed concentrations of C_{Af} and C_{Bf}, then the component balance equations for the fed-batch reactor that produces D are:

$$\frac{\mathrm{d}\,C_A[t]V[t]}{\mathrm{dt}} = C_{\mathrm{Af}}\,q_A - r_{\mathrm{AB}}\,\mathrm{V[t]}$$

$$\frac{\mathrm{d}\,C_B[t]V[t]}{\mathrm{dt}} = C_{\mathrm{Bf}}\,q_B - r_{\mathrm{AB}}\,\mathrm{V[t]}$$

$$\frac{\mathrm{d}\,C_D[t]V[t]}{\mathrm{dt}} = r_{\mathrm{AB}}\,\mathrm{V[t]}$$

These are the three relevant equations we need to solve for this problem. The immediate question that arises is that of the form of the kinetics. We will assume that the reaction between A and B is first order in A and first order in B, that is, second order overall. The equations become:

$$\frac{\mathrm{d}\,C_A[t]V[t]}{\mathrm{dt}} = C_{\mathrm{Af}}\,q_A - k_{\mathrm{AB}}C_A[t]C_B[t]\,\mathrm{V[t]}$$

$$\frac{\mathrm{d}\,C_B[t]V[t]}{\mathrm{dt}} = C_{\mathrm{Bf}}\,q_B - k_{\mathrm{AB}}C_A[t]C_B[t]\,\mathrm{V[t]}$$

$$\frac{\mathrm{d}\,C_D[t]V[t]}{\mathrm{dt}} = k_{\mathrm{AB}}\,C_A[t]C_B[t]\,\mathrm{V[t]}$$

8.3 Negligible Volume Change

In some cases the volume change may be negligible from the start to the finish of the batch. For example, one reagent may be added in a very concentrated form to a dilute solution of the second reactant in the reactor. This can lead to a situation in which the volume of added reactant is quite small compared to that of the initial volume of solution. The equations for this become:

$$\frac{\mathrm{d}\,C_A[t]}{\mathrm{dt}} = \frac{C_{\mathrm{Af}}q_A}{V} - k_{\mathrm{AB}}C_A[t]C_B[t]$$
$$\frac{\mathrm{d}\,C_B[t]}{\mathrm{dt}} = -k_{\mathrm{AB}}C_A[t]C_B[t]$$
$$\frac{\mathrm{d}\,C_D[t]}{\mathrm{dt}} = k_{\mathrm{AB}}C_A[t]C_B[t]$$

We can attempt to find a complete solution for this system of equations:

```
In[1]:= DSolve[
         {CA'[t] == CAf qA - kab CA[t] CB[t],
          CB'[t] == -kab CA[t] CB[t],
          CD'[t] == kab CA[t] CB[t],
              CA[0] == 0,
              CB[0] == CBo,
              CD[0] == 0},
         {CA[t], CB[t], CD[t]},
                 t]

Out[1]= DSolve[{CA′[t] == CAf qA - kab CA[t] CB[t],
          CB′[t] == -kab CA[t] CB[t], CD′[t] == kab CA[t] CB[t],
          CA[0] == 0, CB[0] == CBo, CD[0] == 0},
          {CA[t], CB[t], CD[t]}, t]
```

What we see is that this set of seemingly naive equations is not readily soluble analytically. The combination of the second-order kinetics plus the convective flow term is enough to require the use of numerical methods. To prove this to ourselves, we can redo the problem after removing the convective flow term. That is done in the cell that follows.

```
In[2]:= Clear["Global'*"]

In[3]:= Simplify[
          DSolve[
            {C1'[t] == -k C1[t] C2[t],
             C2'[t] == -k C1[t] C2[t],
             C3'[t] == +k C1[t] C2[t],
             C1[0] == C1o,
             C2[0] == C2o,
             C3[0] == 0},
            {C1[t], C2[t], C3[t]},
            t]
         ]
```

```
Solve::verif : Potential solution {C[2] → 0, C[3] → 0}
 (possibly discarded by verifier) should be checked
 by hand. May require use of limits.

Solve::ifun : Inverse functions are being used by
 Solve, so some solutions may not be found.

Power::infy : Infinite expression 1/0 encountered.

Power::infy : Infinite expression 1/0^2 encountered.

∞::indet : Indeterminate expression 0k ComplexInfinity
 encountered.

Power::infy : Infinite expression 1/0 encountered.

General::stop : Further output of Power::infy will be
 suppressed during this calculation.

∞::indet : Indeterminate expression 0k ComplexInfinity
 encountered.

∞::indet : Indeterminate expression 0 ComplexInfinity
 encountered.

General::stop : Further output of ∞::indet will be
 suppressed during this calculation.
```

Out[3]= $\{C1[t] \to \frac{C1o(C1o - C2o)}{C1o - C2o\, e^{(-C1o+C2o)kt}},\ C1[t] \to \text{Indeterminate},$

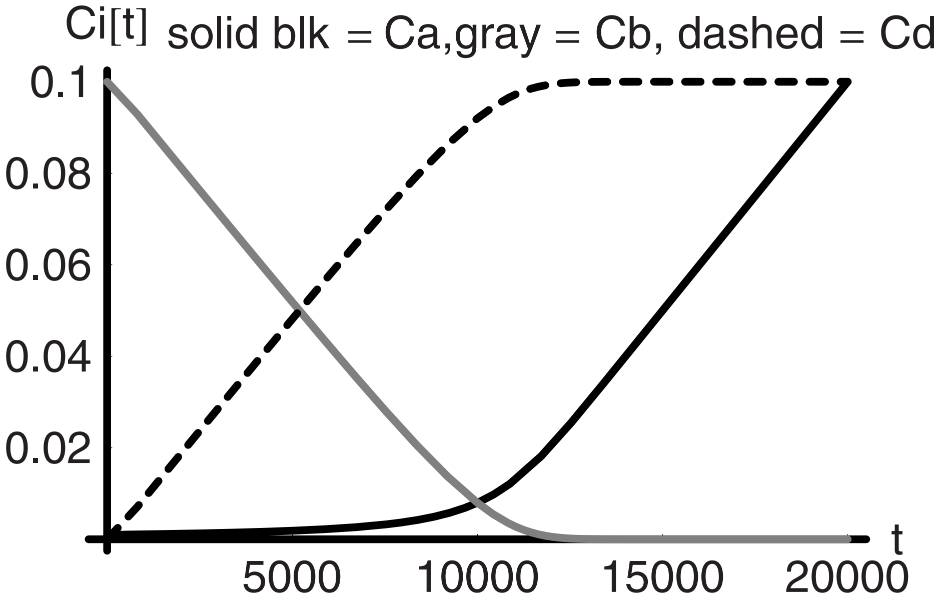
$C2[t] \to -\frac{(C1o - C2o)C2o\, e^{C2o\,k\,t}}{-C1o\, e^{C1o\,k\,t} + C2o\, e^{C2o\,k\,t}},\ C2[t] \to \text{Indeterminate},$

$C3[t] \to \text{Indeterminate},\ C3[t] \to \frac{C1o\, C2o(-1 + e^{(-C1o+C2o)kt}}{-C1o + C2o\, e^{(-C1o+C2o)kt}}\}$

Having removed the flow term, the analytical solution is found; however, we also see that along the way the solver found indeterminance in addition to the closed-form solutions. If we look back at Chapter 5, we find that we already solved this problem, but there we made a substitution for C2[t] in terms of C1[t], which thereby made the solution process easier and avoided an encounter with the infinite expression. Nonetheless, we see that including the constant flow term makes the analytical solution difficult to obtain. On the other hand, the numerical solution is trivial to implement, just as long as we have proper parameter values to apply.

```
In[4]:= SetOptions[{Plot, ListPlot}, AxesStyle → {Thickness[0.01]},
          PlotStyle → {PointSize[0.015], Thickness[0.006]},
          DefaultFont → {"Helvetica", 17}];
```

```
In[5]:= Clear["Global'*"]
        CAf = 1.;
        CAo = 0;
        CBo = .1;
        CDo = 0;
        kab = .1;
        qAf = .001;
        tmax = 20000;
        $D = 5;
        $B = 5;
        $A = 4;
        Mwd = 60;
        Mwb = 40;
        Mwa = 20;
        Vr = 100;

        fbsol1 = NDSolve[
                             CAf qAf
          {CA'[t] ==  ------- - kab CA[t] CB[t],
                               Vr
           CB'[t] == -kab CA[t] CB[t],
           CD'[t] == kab CA[t] CB[t],
               CA[0] == CAo,
               CB[0] == CBo,
               CD[0] == CDo},
         {CA[t], CB[t], CD[t]},
                {t, 0, tmax}];

        ca[t_] := Evaluate[CA[t] /. fbsol1]
        cb[t_] := Evaluate[CB[t] /. fbsol1]
        cd[t_] := Evaluate[CD[t] /. fbsol1]
        ep[t_] := $D cd[t] Vr Mwd - $B cb[t] Vr Mwb
         - $A CAf qAf Mwa t

        Plot[{ca[t], cb[t], cd[t]},
          {t, 0, tmax},
          PlotRange → All,
          PlotStyle → {
            {Dashing[{0.0, 0.0}], Thickness[0.01]},
            {GrayLevel[0.5], Thickness[0.01]},
            {Dashing[{0.02, 0.02}], Thickness[0.01]}},
          AxesLabel → {"t", "Ci[t]"},
          PlotLabel → "solid blk = Ca, gray = Cb,
            dashed = Cd"];
```

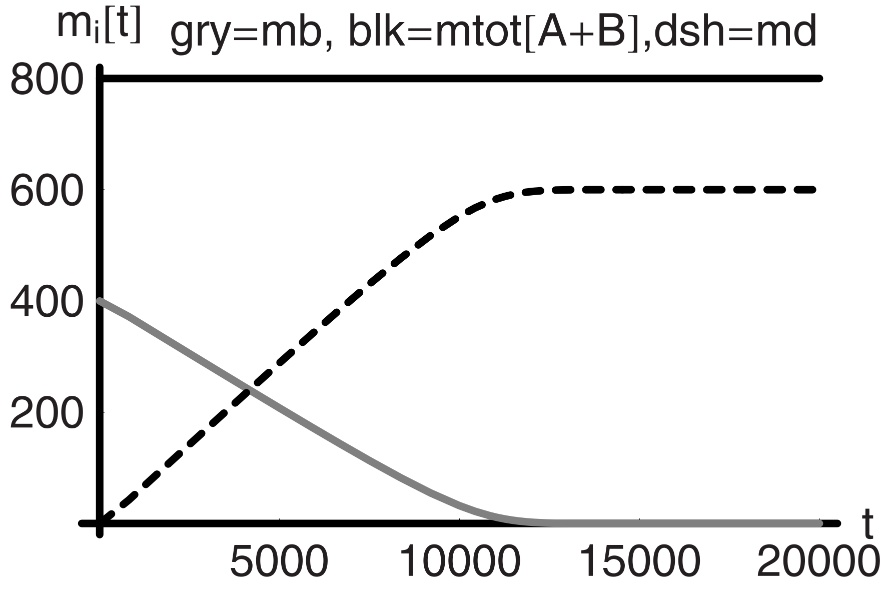


The preceding graph shows the time-dependent concentrations of each component. The profile for B drops nearly linearly with time and that of product rises the same way. The concentration of A is very small until most of B is used up and then it rises sharply with time.

The following graph is most important. Here we have computed the total mass of A added at tmax plus the total mass of B present initially and compared this with the masses of B and D at any time:

$$\text{Total Mass of A} + \text{B} = (C_{Af}\, q_{Af}\, \text{Mwa tmax}) + C_B[0]\text{Vr Mwb}$$

```
In[26]:= Plot[{(CAf qAf Mwa tmax) + cb[0] Vr Mwb, cb[t] Vr Mwb,
               cd[t] Vr Mwd}, {t, 0, tmax},
           PlotStyle → {{Dashing[{0.0, 0.0}], Thickness[0.01]},
             {GrayLevel[0.5], Thickness[0.01]},
              {Dashing[{0.02, 0.02}], Thickness[0.01]}},
           AxesLabel → {"t", "mi[t]"},
           PlotLabel → "gry=mb, blk=mtot[A+B], dsh=md"];
```



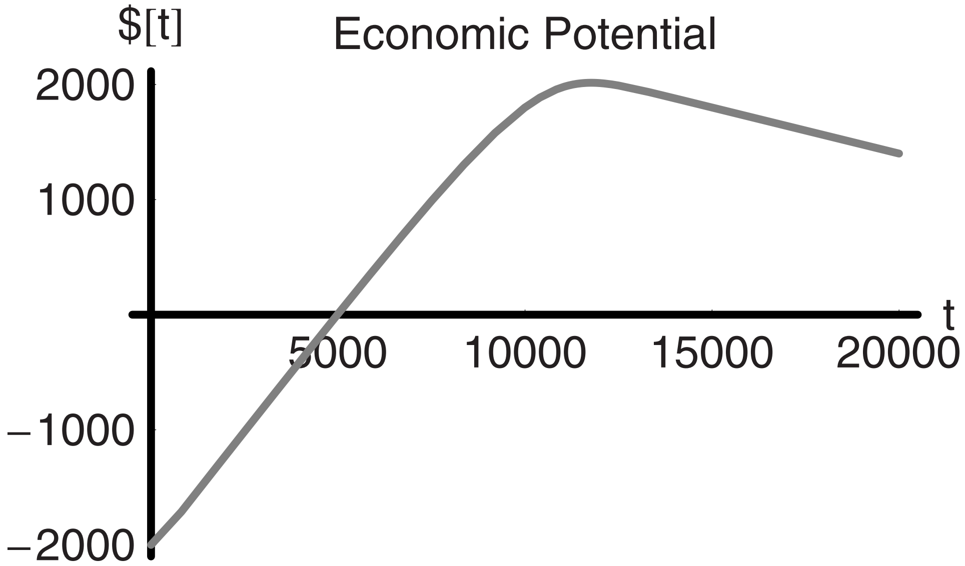
The mass of **D** cannot exceed the mass of **A** + **B**, and it does not. This provides the necessary check on the model. We have included the mass of **B** as a function of time and it goes to zero as we would expect.

Next we can introduce and compute the economic potential of the mixture as a function of time:

$$\mathbf{ep[t\_] := \$D\ C_D[t]Vr\ Mwd - \$B\ C_B[t]Vr\ Mwb - \$A\ C_{Af}\ q_{Af}\ Mwa\ t}$$

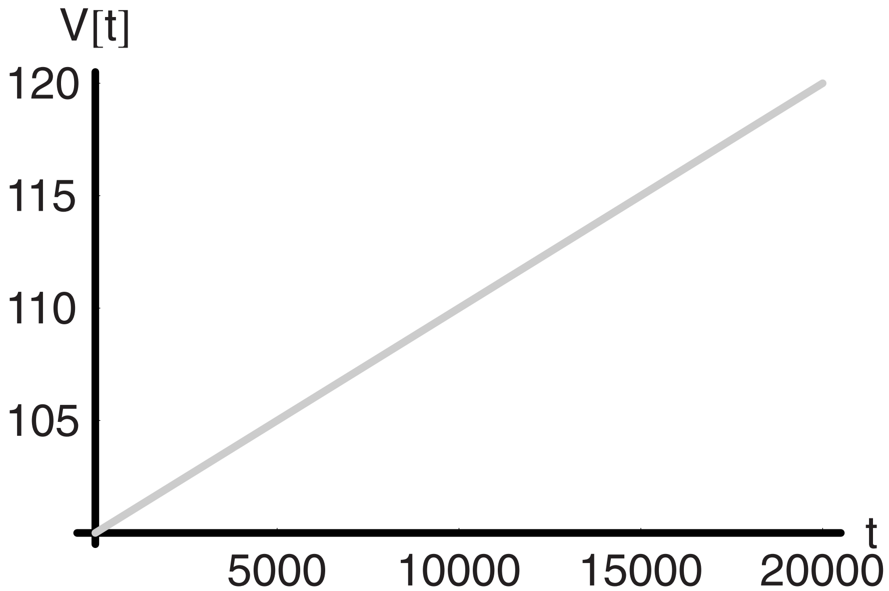
The maximum economic potential is the difference between the values of the products and the reactants. The terms **$D**, **$B**, and **$A** are the values per unit mass of each component. Because this is a semibatch process, the economic potential goes through a maximum. We can plot this below and show this behavior for the case we are considering:

```
In[27]:= $D = 5;
         $B = 5;
         $A = 4;
         Plot[ep[t], {t, 0, tmax},
            PlotStyle → {{Thickness[0.01], GrayLevel[0.5]}},
            PlotRange → All,
            AxesLabel → {"t", "$[t]"},
            PlotLabel → "Economic Potential"];
```



In this case the maximum value of the mixture is reached after 10,000 time units. Before this time we have not converted all of the reactant to product, but after this time we begin to merely dilute the product in reactant A. The next calculation and plot we shall make is the change in total volume calculated on the basis of the flow rate of reactant:

```
In[31]:= Plot[{(qAf t + Vr)}, {t, 0, tmax},
           PlotRange  → All,
           AxesLabel  → {"t", "V[t]"},
           PlotStyle  → {{Thickness[0.01], GrayLevel[0.8]}}
          ];
```



The volume change in this case is on the order of 20%, which is really too large to be acceptable within the context of an analysis in which it was assumed that negligible volume change would occur. Hence we are motivated to do the analysis again without this simplifying assumption.

8.4 Large Volume Change

The equations for variable volume are:

$$\frac{d\,\mathrm{Ca}[t]V[t]}{\mathrm{dt}} = \mathrm{Caf\,qaf} - \mathrm{kab\,Ca[t]Cb[t]\,V[t]}$$
$$\frac{d\,\mathrm{Cb}[t]V[t]}{\mathrm{dt}} = -\mathrm{kab\,Ca[t]Cb[t]\,V[t]}$$
$$\frac{d\,\mathrm{Cd}[t]V[t]}{\mathrm{dt}} = +\mathrm{kab\,Ca[t]Cb[t]\,V[t]}$$

As there are four time-dependent variables and only three equations, we need another equation, which is obtained in the total material balance. The mass in the control volume increases only by the additional mass admitted through the feed stream:

$$\frac{d\ \rho\,\mathrm{V[t]}}{\mathrm{dt}} = \rho\,\mathrm{qaf}$$

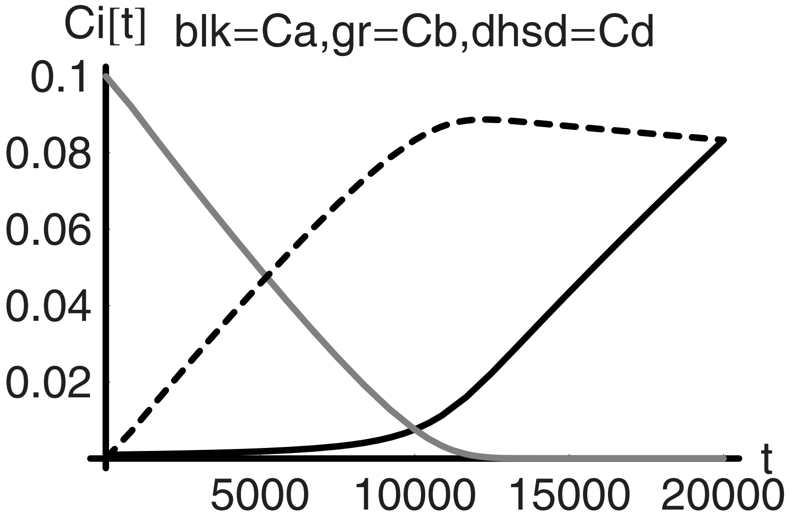
If the density is essentially unchanging and if the flow rate in is a constant, then the volume change is linear in time:

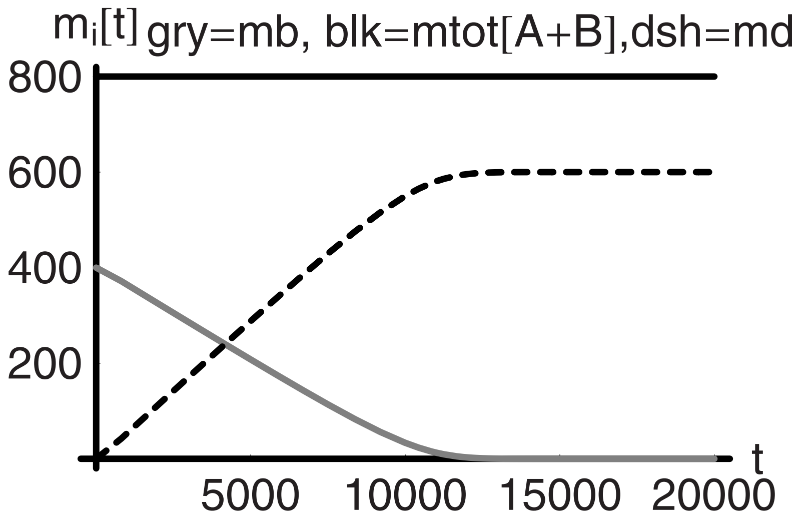
$$\mathrm{V[t]} = \mathrm{Vo} + \mathrm{qaf\,t}$$

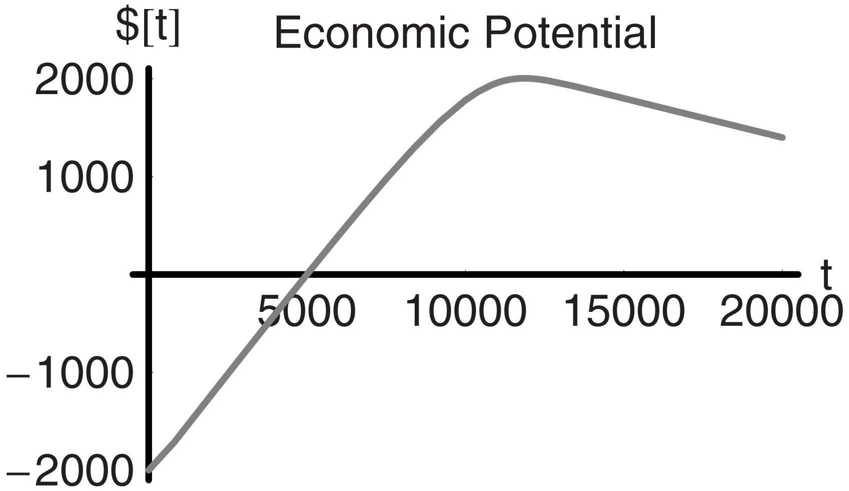
```
In[32]:= Clear["Global‘*"]

In[33]:= CAf = 1.;
         CAo = 0;
         CBo = .1;
         CDo = 0;
         kab = .1;
         qAf = .001;
         tmax = 20000;
         $D = 5;
         $B = 5;
         $A = 4;
         Mwd = 60;
         Mwb = 40;
         Mwa = 20;
         Vr = 100;
```

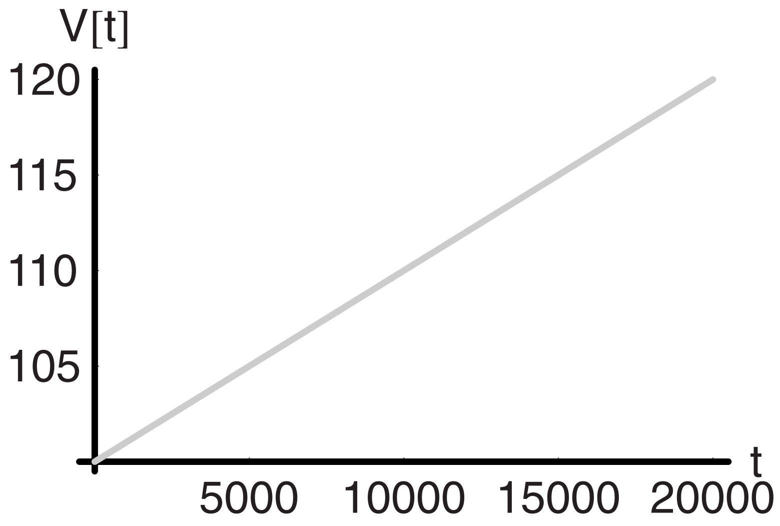
```
fbsol2 = NDSolve[
  {∂t V[t] == qAf,
   ∂t (CA[t] V[t]) == CAf qAf - kab CA[t] CB[t] V[t],
   ∂t (CB[t] V[t]) == -kab CA[t] CB[t] V[t],
   ∂t (CD[t] V[t]) == kab CA[t] CB[t] V[t],
        V[0] == Vr,
        CA[0] == CAo,
        CB[0] == CBo,
        CD[0] == CDo},
  {V[t], CA[t], CB[t], CD[t]},
        {t, 0, tmax}];

v[t_] := Evaluate[V[t] /. fbsol2]
ca[t_] := Evaluate[CA[t] /. fbsol2]
cb[t_] := Evaluate[CB[t] /. fbsol2]
cd[t_] := Evaluate[CD[t] /. fbsol2]

ep[t_] := $D cd[t] v[t] Mwd - $B cb[t] v[t] Mwb
 - $A CAf qAf Mwa t

Plot[{ca[t], cb[t], cd[t]},
  {t, 0, tmax},
  PlotRange → All,
  PlotStyle → {{Dashing[{0.0, 0.0}], Thickness[0.01]},
    {GrayLevel[0.5], Thickness[0.01]},
    {Dashing[{0.02, 0.02}], Thickness[0.01]}},
  AxesLabel → {"t", "Ci[t]"},
  PlotLabel → "blk=Ca,gr=Cb,dhsd=Cd"];

Plot[{(CAf qAf Mwa tmax) + cb[0] Vr Mwb, cb[t] v[t] Mwb,
  cd[t] v[t] Mwd}, {t, 0, tmax},
  PlotStyle → {{Dashing[{0.0, 0.0}], Thickness[0.01]},
    {GrayLevel[0.5], Thickness[0.01]},
    {Dashing[{0.02, 0.02}], Thickness[0.01]}},
  AxesLabel → {"t", "mi[t]"},
  PlotLabel → "gry=mb, blk=mtot[A+B], dsh=md"];

Plot[ep[t], {t, 0, tmax},
  PlotStyle → {{Thickness[0.01], GrayLevel[0.5]}},
  PlotRange → All,
  AxesLabel → {"t", "$[t]"},
  PlotLabel → "Economic Potential"];
```

```
Plot[{(v[t])}, {t, 0, tmax},
  PlotRange  → All,
  AxesLabel  → {"t", "V[t]"},
  PlotStyle  → {{Thickness[0.01], GrayLevel[0.8]}}
 ];
```







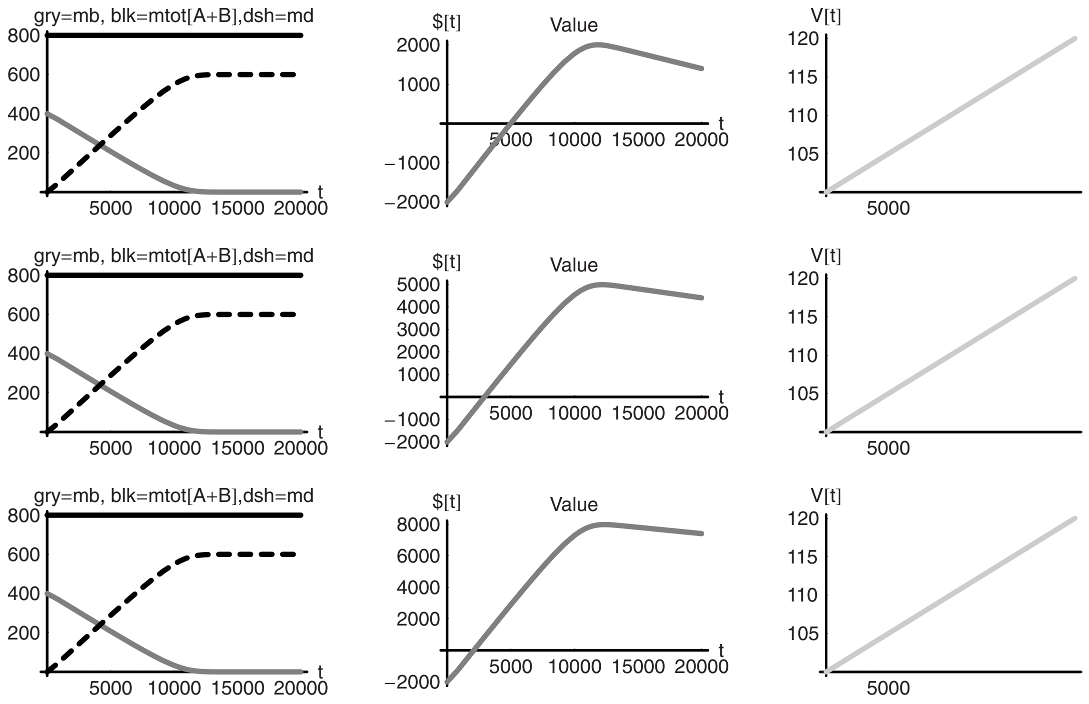


The next step we take is to create a **Module** function **semi2** from this code so that we may run many different cases of this semibatch reactor. The groups of similar variables and parameters are grouped within curly brackets:

```
In[57]:= semi2[{CAf_, CAo_, CBo_, CDo_}, kab_, qAf_,
           {$D_, $B_, $A_}, {Mwd_, Mwb_, Mwa_}, Vr_, tmax_]:=
          Module[
           {V, CA, CB, CD, v, ca, cb, cd, fbsol2, ep, t},

             fbsol2 = NDSolve[
                {∂t V[t] == qAf,
                 ∂t(CA[t] V[t]) == CAf qAf - kab CA[t] CB[t] V[t],
    ∂t(CB[t] V[t]) == - kab CA[t] CB[t] V[t],
    ∂t(CD[t] V[t]) == kab CA[t] CB[t] V[t],
        V[0] == Vr,
        CA[0] == CAo,
        CB[0] == CBo,
        CD[0] == CDo},
    {V[t], CA[t], CB[t], CD[t]},
            {t, 0, tmax}];

v[t] = Evaluate[V[t] /. fbsol2];
ca[t] = Evaluate[CA[t] /. fbsol2];
cb[t] = Evaluate[CB[t] /. fbsol2];
cd[t] = Evaluate[CD[t] /. fbsol2];

ep[t] = ($D cd[t] v[t] Mwd - $B cb[t] v[t] Mwb -
 $A CAf qAf Mwa t);

SetOptions[Plot, DefaultFont → {"Helvetica", 10}];

Plot[{ca[t], cb[t], cd[t]},
 {t, 0, tmax},
 PlotRange → All,
 PlotStyle → {{Dashing[{0.0, 0.0}], Thickness[0.02]},
   {GrayLevel[0.5], Thickness[0.02]},
   {Dashing[{0.04, 0.04}], Thickness[0.02]}},
 AxesLabel → {"t", "Ci[t]"},
 PlotLabel → "blk=Ca, gr=Cb, dhsd=Cd",
 DisplayFunction → Identity];

 {{Graphics[Plot[
    {(CAf qAf Mwa tmax) + CBo Vr Mwb, cb[t] v[t] Mwb,
     cd[t] v[t] Mwd}, {t, 0, tmax},
    AxesLabel → {"t", ""},
    PlotStyle → {{Dashing[{0.0, 0.0}], Thickness[0.02]},
      {GrayLevel[0.5], Thickness[0.02]},
      {Dashing[{0.04, 0.04}], Thickness[0.02]}},
    AxesLabel → {"t", "mi[t]"},
    PlotLabel → "gry=mb, blk=mtot[A+B], dsh=md",
    DisplayFunction → Identity]]},

  {Graphics[Plot[ep[t], {t, 0, tmax},
     PlotStyle → {{Thickness[0.02], GrayLevel[0.5]}},
     PlotRange → All,
     AxesLabel → {"t", "$[t]"},
     PlotLabel → "Value",
     DisplayFunction → Identity]]},
            {Graphics[Plot[{(v[t])}, {t, 0, tmax},
               PlotRange → All,
               AxesLabel → {"t", "V[t]"},
               PlotStyle → {{Thickness[0.02], GrayLevel[0.8]}},
               DisplayFunction → Identity]]}}
          ]
```

This function can now be used to examine how the behavior of the fed-batch reactor system behaves with variation in its parameters. In the example that follows, the parameters are held constant, except for the values of the product which are varied from 5 to 15 in three increments of 5 each. We have done this by making a table of the **Module** "**semi2.**" The output from **semi2** is three graphics, one for each relevant graph. Our goal is to show these as an array of plots. We do this by flattening the output "**solgrp**," to remove all the internal curly brackets. Then these are partitioned into groups of three and finally we "**Show**" the results as a set of **GraphicsArray** as follows:

```
In[58]:= solgrp = Table[
            semi2[{1, 0, .1, 0}, .1, .001, {n, 5, 4}, {60, 40, 20},
            100, 20000], {n, 5, 15, 5}
               ];
          Partition[Flatten[solgrp], 3];
          Show[GraphicsArray[%]];
```



The overall value in $ of the economic potential increases significantly with the increased value of the product, but each case shows the same maximum and at the same point in time. The changes in mass of **B** and **D** are apparently linear in time over most of the run up to ~10,000 time units. Why should this be? Linear time dependence indicates a constancy of slope. However, this is a fully transient, that is, time-dependent, system. How can we have a constant slope, that is, a constant rate of change, in the concentrations for a fully transient system? To understand this we must reintroduce the concept of the *pseudo-steady state.*

8.5 Pseudo-Steady State

A situation that can arise in the well-stirred fed-batch reactor is one in which the rate of consumption of the added component is balanced exactly by its rate of addition. In this case the rate of change of the mass of **A** in the reactor is effectively zero as long as there is sufficient **B** present for reaction to take place. This leads to a period of operation that can be considered to be a steady state, but we refer to it is a *pseudo-steady state* because at the same time the rate of change of **B** is real and constant. We will go back to the equations of change to understand what this means:

$$\frac{\mathrm{d\,Ca}[t]V[t]}{\mathrm{dt}} = \mathrm{Caf\,qaf} - \mathrm{kab\,Ca[t]Cb[t]\,V[t]}$$
$$\frac{\mathrm{d\,Cb}[t]V[t]}{\mathrm{dt}} = -\mathrm{kab\,Ca[t]Cb[t]V[t]}$$
$$\frac{\mathrm{d\,Cd}[t]V[t]}{\mathrm{dt}} = +\mathrm{kab\,Ca[t]Cb[t]V[t]}$$

If the rate of change of the concentration of **A** is zero, then the following simplifications apply:

$$0 = \mathrm{Caf\,qaf} - \mathrm{kab\,Ca[t]Cb[t]V[t]}$$
$$\frac{\mathrm{d\,Cb}[t]V[t]}{\mathrm{dt}} = -\mathrm{Caf\,qaf}$$
$$\frac{\mathrm{d\,Cd[t]V[t]}}{\mathrm{dt}} = +\mathrm{Caf\,qaf}$$

If the feed flow rate and concentration of **A** are constant, then we would find that the concentrations of **B** and **D** are linear in time and with oppositely signed slopes. We use "**stst**" to designate the steady-state time-dependent concentrations of **B** and **D**.

```
In[61]:= Clear["Global‘*"]

         Simplify[DSolve[
           {∂t(V[t]) == qaf,
     ∂t (Cbstst[t] V[t]) == -Caf qaf,
     ∂t (Cdstst[t] V[t]) == +Caf qaf,
     V[0] == Vo,
     Cbstst[0] == Cbo,
     Cdstst[0] == Cdo},
    {V[t], Cbstst[t], Cdstst[t]},
    t]]

General::spell1 : Possible spelling error: new symbol
 name "qaf" is similar to existing symbol "qAf".

General::spell1 : Possible spelling error: new symbol
 name "Caf" is similar to existing symbol "CAf".

General::spell1 : Possible spelling error: new symbol
 name "Cdstst" is similar to existing symbol "Cbstst".

General::stop : Further output of General::spell1 will
 be suppressed during this calculation.
```

Out[62]= $\{\text{Cbstst}[t] \to \frac{-\text{Caf qaf } t + \text{Cbo Vo}}{\text{qaf } t + \text{Vo}},$

$\text{Cdstst}[t] \to \frac{\text{Caf qaf } t + \text{Cdo Vo}}{\text{qaf } t + \text{Vo}}, \ V[t] \to \text{qaf } t + \text{Vo}\}$

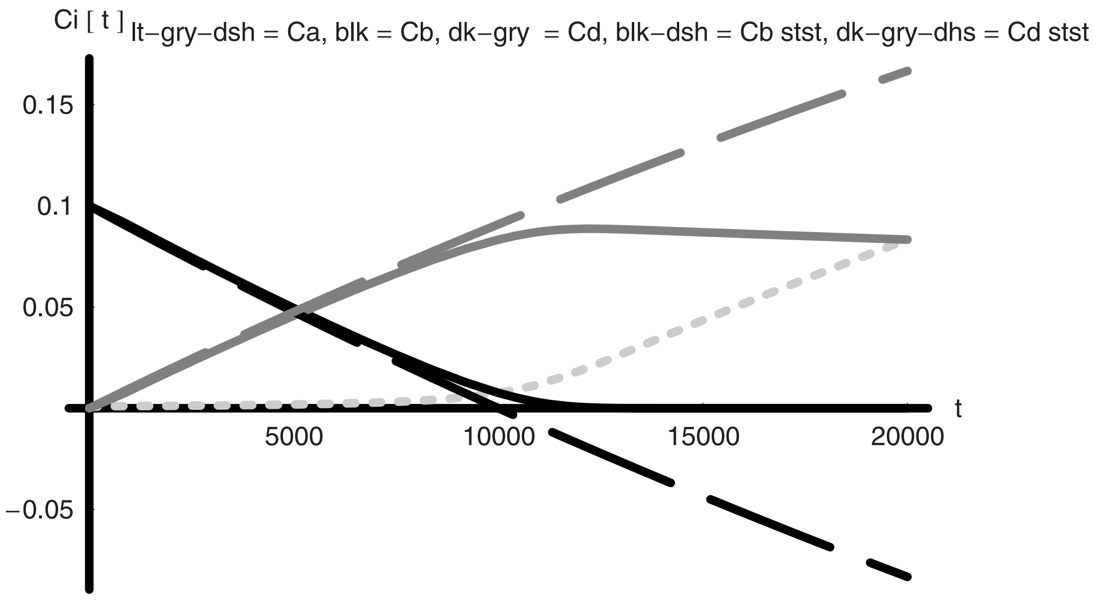
We can go back to the full time-dependent solution and define a flow condition for **A** that would lead to these steady-state results. The key to the pseudo-steady state is that the mass flow of **A** into the system be balanced by the rate of chemical reaction. We can write a new **Module** function that takes the solutions that we just derived for the steady state and compares them to those that we had already obtained for the fully time-dependent case. This is constructed in what follows by copying those pieces of "**semi2**" that we need and adding in the steady-state solutions.

```
In[63]:= semi3[{CAf_, CAo_, CBo_, CDo_}, kab_, qAf_, Vr_,
          tmax_]:=
          Module[
           {V, CA, CB, CD, v, ca, cb, cd, castst, cbstst, cdstst,
            fbsol3, t},

           fbsol3 = NDSolve[
             {∂t V[t] == qAf,
              ∂t (CA[t] V[t]) == CAf qAf - kab CA[t] CB[t] V[t],
              ∂t (CB[t] V[t]) == -kab CA[t] CB[t] V[t],
              ∂t (CD[t] V[t]) == kab CA[t] CB[t] V[t],
            V[0] == Vr,
            CA[0] == CAo,
            CB[0] == CBo,
            CD[0] == CDo},
        {V[t], CA[t], CB[t], CD[t]},
               {t, 0, tmax}];

      v[t] = Evaluate[V[t] /. fbsol3];
      ca[t] = Evaluate[CA[t] /. fbsol3];
      cb[t] = Evaluate[CB[t] /. fbsol3];
      cd[t] = Evaluate[CD[t] /. fbsol3];
```

$$\text{cbstst[t]} = \frac{\text{CBoVr - CAfqAft}}{\text{qAft + Vr}};$$

$$\text{cdstst[t]} = \frac{\text{CAfqAft + CDoVr}}{\text{qAft + Vr}};$$

```
      Plot[{ca[t], cb[t], cd[t], cbstst[t], cdstst[t]},
       {t, 0, tmax},
       PlotRange → All,
       PlotStyle → {{Thickness[0.01], GrayLevel[0.8],
        Dashing[{0.01, 0.02}]},
        {Thickness[0.01], GrayLevel[0]},
        {Thickness[0.01], GrayLevel[0.5]},
        {Thickness[0.01], Dashing[{0.15, 0.05}],
        GrayLevel[0]},
        {Thickness[0.01], Dashing[{0.15, 0.05}],
        GrayLevel[0.5]}},
       AxesLabel → {"t", "Ci[t]"},
       PlotLabel → "lt-gry-dsh = Ca, blk = Cb,
          dk-gry = Cd, blk-dsh = Cb stst,
          dk-gry-dhs = Cd stst",
       DisplayFunction → Identity]
      ]

    General::spell : Possible spelling error: new symbol name
     "cbstst" is similar to existing symbols {castst, Cbstst}.

    General::spell : Possible spelling error: new symbol name
     "cdstst" is similar to existing symbols {castst, cbstst,
     Cdstst}.

In[64]:= semi3[{1, 0, .1, 0}, .1, .001, 100, 20000];
          Show[%, DisplayFunction → $DisplayFunction];
```



The results show in the preceding graph that the steady-state solutions (dashed) map well onto the transient solutions for the concentrations of **B** and **D** at early time. Beyond ~8000 time units, the steady-state concentrations begin to deviate noticeably from the full solutions. This is also the time at which the concentration of **A** begins to rise above near-zero values. The steady-state solutions are useful because they allow us to compute the flow rate of reagent **A** and the time dependence of the systems with very simple equations, but we cannot push such an analysis too far beyond its region of applicability. From the perspective of analysis, the pseudo-steady state is important to us because it explains the behavior of the more complex and complete model in a very straightforward way.

8.6 Summary

In this chapter we have found that a reactor type that is familiar to us and that has intuitively obvious usefulness, namely, the well-mixed semibatch reactor, is also very complex to treat—at least analytically—due to its transient behavior. It is also evident that we would never use this kind of reactor to evaluate even the most basic chemical kinetics. Thus we need a simpler type of reactor that is mathematically more tractable and experimentally more feasible to operate. We will see instances of these in the next chapter. Along the way we have now added the final element that we needed in our *Mathematica* toolbox, the writing of Modules. We will build on this to produce even more useful Packages in what follows.

CHAPTER 9

Continuous Stirred Tank and the Plug Flow Reactors

The two most useful idealizations of chemical reactors are the continuously stirred tank reactor (**CSTR**) and the plug flow reactor (**PFR**). Both are idealizations in that they are two different and quite distinct extremes of mixing. Real reactors are more complex, but often they can be analyzed approximately in terms of these idealizations. Furthermore, when starting from scratch to consider the design of a new reactor system, these simplified models are used to estimate the size of the system that will be required and, in some cases, which mixing regime will lead to better results. Finally, the ideal reactors allow us to do analyses that will give us insight into how real reactors operate, which factors are most important, and how to control them for better performance. Therefore, although the **CSTR** and **PFR** are idealizations, they are quite powerful models for chemically reacting systems and we have much to gain from a study of them. We begin first with the perfectly mixed system.

9.1 Continuous Flow-Stirred Tank Reactor

The name continuous flow-stirred tank reactor is nicely descriptive of a type of reactor that frequently for both production and fundamental kinetic studies. Unfortunately, this name, abbreviated as **CSTR**, misses the essence of the idealization completely. The ideality arises from the assumption in the analysis that the reactor is perfectly mixed, and that it is *homogeneous*. A better name for this model might be *continuous perfectly mixed reactor* (**CPMR**).

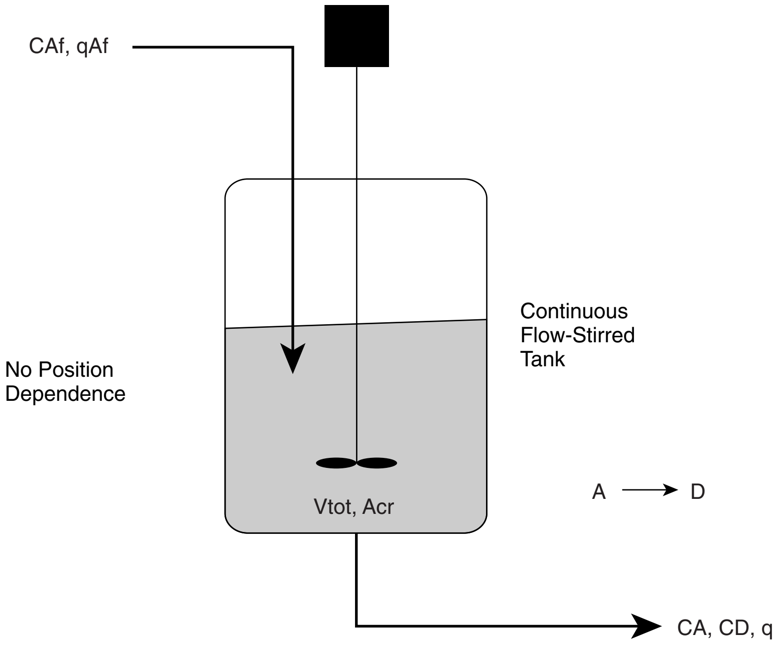


Figure 1

Nonetheless, as long as we realize this and its mathematical consequences, and that the terminology refers as much to the mathematics as it does to any specific configuration, then the more prevalent name **CSTR** is serviceable.

Because of the well-mixed assumption, it is natural to think of the **CSTR** as a liquid phase reactor with a mixer as shown in Figure 1:

The chemistry in this case is the irreversible conversion of **A** to **B**, which follows simple, linear kinetics. When we write the time-dependent mass balances for this system we have:

$$\frac{d\,\mathrm{Ca}[t]V[t]}{dt} = \mathrm{Caf\,qaf} - \mathrm{Ca[t]q} - \mathrm{kad\,Ca[t]\,V[t]}$$

$$\frac{d\,\mathrm{Cd}[t]V[t]}{dt} = -\mathrm{Cd[t]q} + \mathrm{kad\,Ca[t]\,V[t]}$$

$$\frac{d\,\rho[t]V[t]}{dt} = \rho\mathrm{af\,qaf} - \rho\mathrm{[t]q}$$

If the system is at steady state, then the total mass in must be balanced by the total mass out. Furthermore, if the densities of the feed and product are nearly the same, then we can take the flow rate in as equal to the flow rate out. The equations at steady state become:

$$0 = (\mathrm{Caf} - \mathrm{Ca})\mathrm{q} - \mathrm{kad}\,\mathrm{Ca}\,\mathrm{V}$$
$$0 = -\mathrm{Cd}\,\mathrm{q} + \mathrm{kad}\,\mathrm{Ca}\,\mathrm{V}$$
$$0 = \mathrm{qaf} - \mathrm{q}$$

We can divide both of the component balances by the product **Caf V** to find:

$$0 = \frac{(1 - \Phi\mathrm{a})}{\theta} - \mathrm{kad}\,\Phi\mathrm{a}$$
$$0 = -\frac{\Phi\mathrm{d}}{\theta} + \mathrm{kad}\,\Phi\mathrm{a}$$

where $\frac{V}{q} = \theta$ is the holding time for the **CSTR**. If we now solve for the dimensionless exit concentration, we find:

```
In[19]:= Solve[0 == (1 - Φa)/θ - kadΦa, Φa]

Out[19]= {{Φa → 1/(1 + kadθ)}}
```

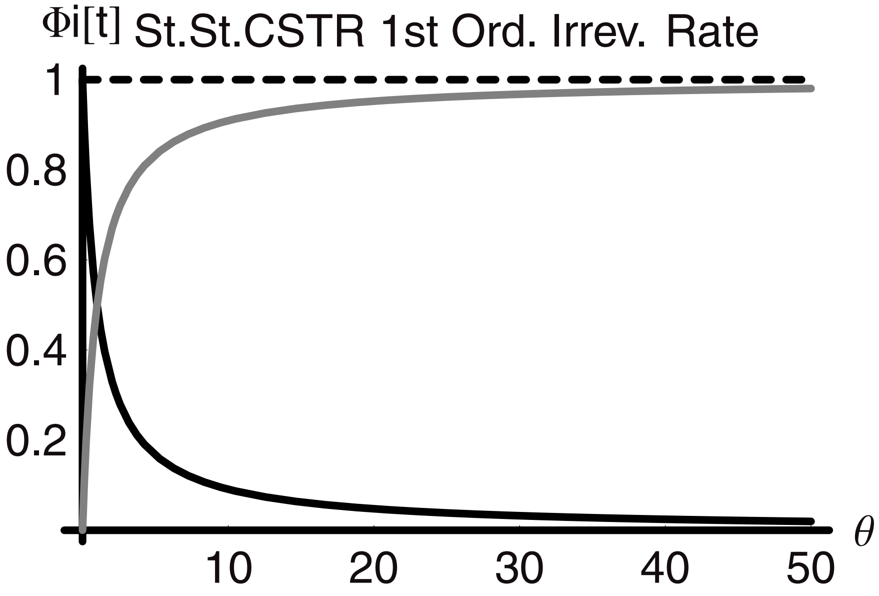
Solving both equations simultaneously to find **Φd:**

```
In[20]:= Clear["Global`*"]
          Solve[{0 == (1-Φa)/θ - kadΦa, 0 == -Φd/θ, kadΦa}, {Φa,Φd}]

         General::spell1: Possible spelling error: new symbol name
          "Φd" is similar to existing symbol "Φa".

Out[21]= {{Φd → kadθ/(1 + kadθ), Φa → 1/(1 + kadθ)}}
```

The concentration of **A** leaving the reactor is the reciprocal of the sum of one plus the product of the first-order rate constant and the holding time. The first-order rate constant, we recall, has dimensions of reciprocal time, and the holding time is just time, so their product is dimensionless. In fact this product is actually the ratio of the holding time to the characteristic time required for the chemistry to occur. If the rate constant is taken to be of order unity, then we will see how the concentrations of **A** and **D** change with holding time.

```
In[22]:= SetOptions[{Plot, ListPlot},
          AxesStyle → {Thickness[0.01]},
          PlotStyle → {PointSize[0.015], Thickness[0.006]},
           DefaultFont → {"Helvetica", 17}];
```

```
In[23]:= Φa[θ_] := 1/(1 + kad θ)
         Φd[θ_] := kad θ/(1 + kad θ)
         kad = 1;

         Plot[{Φa[θ], Φd[θ]}, {θ, 0, 50},
           PlotStyle → {{Thickness[0.01],GrayLevel[0]},
             {Thickness[0.01], GrayLevel[0.5]}},
              AxesLabel → {"θ", "Φi[t]"},
              Epilog → {Thickness[0.01],Dashing[{0.02, 0.02}],
              Line[{{0, 1},{50, 1}}]},
              PlotLabel → "St.St.CSTR 1st Ord. Irrev. Rate"];
```



When the holding time has become a factor of ten larger than the characteristic time for the reaction chemistry, then we find that the concentration of **A** has dropped by ~90% of its feed value. Hence, the concentration of product **D** is said to *tend toward unity asymptotically*.

There is another important way to view these equations. If we go back to the dimensional form it will be more evident. Typically, the **CSTR** is considered to be operated at steady state, which greatly simplifies the problem as we will see.

$$0 = (\mathrm{Caf} - \mathrm{Ca})\mathrm{q} - \mathrm{kad\,Ca\,V}$$

$$0 = -\mathrm{Cd\,q} + \mathrm{kad\,Ca\,V}$$

We can rearrange these as follows:

$$(\mathrm{Caf} - \mathrm{Ca})\frac{q}{V} = \mathrm{kad\,Ca}$$
$$\frac{(\mathrm{Caf} - \mathrm{Ca})}{\theta} = \mathrm{kad\,Ca}$$
$$\frac{(\mathrm{Caf} - \mathrm{Ca})}{\theta} = \mathrm{r_{ad}}$$

This last equation shows why the **CSTR** at steady state is such a valuable tool to the experimentalist seeking kinetic parameters. The rate of reaction, *independent of the form of the kinetics*, is simply the change in concentration of **A** between the feed and exit streams divided by the holding time. We will see this repeatedly.

There is something to learn from rearranging the second equation also:

$$0 = -\mathrm{Cd\,q} + \mathrm{kad\,Ca\,V}$$
$$\frac{\mathrm{Cd}\,q}{V} = r_{\mathrm{ad}}$$

Given the rate of a chemical reaction, and the target production rate **Cd q** we can compute the volume necessary for a well-mixed reactor to achieve this output. Thus in a very real sense this becomes a useful design equation to be employed in the earliest stages of a study of economic feasibility.

9.2 Steady-State CSTR with Higher-Order, Reversible Kinetics

The first-order, irreversible chemical rate case is useful in terms of providing us with insight into what are the consequences of perfect mixing and with a sense of how the characteristic times for reaction and flow are related. On the other hand, it is limited in usefulness because it represents highly simplified chemistries and correspondingly simple kinetics. Often the actual kinetics are far more complex. Let us consider the same chemistry as that we examined in the fed-batch reactor, namely, that of A and B reacting to give D (see Figure 2). The rate law will be second order overall and first order in each component. However, this time we will assume that it is reversible and that the rate law for the reverse reaction will be second order in D:

$$r_{A+B\Leftrightarrow D} = \mathrm{kab\,Ca\,Cb} - \mathrm{kd\,Cd^2}$$
$$\mathrm{Keq} = \frac{\mathrm{kd}}{\mathrm{kab}}$$

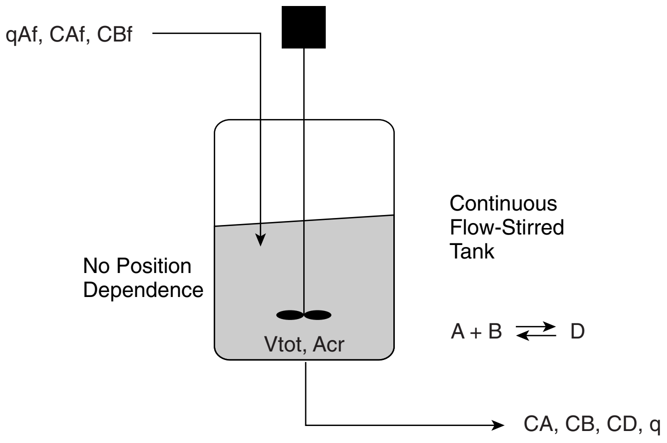


Figure 2

The transient balance equations for this system will be:

$$\frac{d\,\mathrm{Ca}[t]V[t]}{dt} = \mathrm{Caf\,qf} - \mathrm{Ca[t]q} - \mathrm{kab\,Ca[t]\,Cb[t]\,V[t]} + \mathrm{kd\,Cd[t]^2\,V[t]}$$

$$\frac{d\,\mathrm{Cb}[t]V[t]}{dt} = \mathrm{Cbf\,qf} - \mathrm{Cb[t]q} - \mathrm{kab\,Ca[t]\,Cb[t]\,V[t]} + \mathrm{kd\,Cd[t]^2\,V[t]}$$

$$\frac{d\,\mathrm{Cd}[t]V[t]}{dt} = -\mathrm{Cd[t]q} + \mathrm{kab\,Ca[t]\,Cb[t]\,V[t]} - \mathrm{kd\,Cd[t]^2}$$

$$\frac{d\,\rho[t]V[t]}{dt} = \rho\mathrm{f\,qbf} - \rho\mathrm{[t][t]q}$$

At the steady-state condition the mass input must be the same as the mass output. Furthermore, the net rate of change in concentration of each of the components is zero. This makes the differentials each zero in all four equations. The inlet densities of the liquid reactant streams are typically not too different from each other or from the density of the outlet stream including products. The inlet flow rates and concentrations are also equal. Thus the equations reduce to:

$$0 = \mathrm{Caf\,qf} - \mathrm{Ca\,q} - \mathrm{kab\,Ca\,Cb\,V} + \mathrm{kd\,Cd^2\,V}$$

$$0 = \mathrm{Cbf\,qf} - \mathrm{Cb\,q} - \mathrm{kab\,Ca\,Cb\,V} + \mathrm{kd\,Cd^2\,V}$$

$$0 = -\mathrm{Cd\,q} + \mathrm{kab\,Ca\,Cb\,V} - \mathrm{kd\,Cd^2\,V}$$

$$0 = \mathrm{qf} - \mathrm{q}$$

These are four equations that include a total of 10 variables and parameters, and only three of the four equations are independent, as three can be solved to find the fourth. The solutions to these equations are "easy" to find, in the sense that they are a set of simultaneous algebraic equations rather than differential equations. We can solve the three component balances for the concentrations at the exit of the reactor using Solve.

```
In[27]:= Clear["Global`*"]
         cstr1 = Simplify[
           Solve[
            {0 == (Caf - Ca)q - kabCaCbV + kdCd²V,
             0 == (Cbf - Cb)q - kabCaCbV + kdCd²V,
             0 == -Cdq + kabCaCbV - kdCd²V}, {Ca, Cb, Cd}]
                ]
```

Out[28]=

$$\begin{aligned}
\{\{\text{Ca} &\to -\frac{q+\text{Cbf kab V}-\text{Caf}\,(\text{kab}-2\text{kd})\,V+\sqrt{4\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2}}{2\,(\text{kab}-\text{kd})\,V},\\
\text{Cb} &\to -\frac{1}{2\,(\text{kab}-\text{kd})\,V}\Big(q+\text{Caf kab V}-\text{Cbf kab V}+2\text{Cbf kd V}+\\
&\qquad\sqrt{4\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2}\Big),\\
\text{Cd} &\to \frac{q+\text{Caf kab V}+\text{Cbf kab V}+\sqrt{4\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2}}{2\,(\text{kab}-\text{kd})\,V},\\
\{\text{Ca} &\to \frac{-q-\text{Cbf kab V}+\text{Caf}\,(\text{kab}-2\text{kd})\,V+\sqrt{4\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2},}{2\,(\text{kab}-\text{kd})\,V}\\
\text{Cb} &\to \frac{1}{2\,(\text{kab}-\text{kd})\,V}\Big(q-\text{Caf kab V}+\text{Cbf kab V}-2\text{Cbf kd Vkern3pt}+\\
&\qquad\sqrt{4\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2}\Big),\\
\text{Cd} &\to \frac{q+\text{Caf kab V}+\text{Cbf kab V}-\sqrt{4\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2}}{2\,(\text{kab}-\text{kd})\,V}\}\}
\end{aligned}$$

The result is that we get two sets of symbolic solutions for the concentrations. The first set appears to be the appropriate one as the leading coefficient is positive, whereas for the second set the same term is negative, suggesting that for real positive values of the parameters it would return negative concentrations, which are unphysical. Thus we can extract the first set of solutions with the bracketed number 1:

```
In[29]:= cstr1[[1]]
```

Out[29]=

$$\begin{aligned}
\{\text{Ca} &\to -\frac{q+\text{Cbf kab V}-\text{Caf}\,(\text{kab}-2\,\text{kd})\,V+\sqrt{4\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2}}{2\,(\text{kab}-\text{kd})\,V},\\
\text{Cb} &\to -\frac{q+\text{Caf kab V}-\text{Cbf kab V}+2\text{Cbf kd},\,V+\sqrt{4\,\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2}}{2\,(\text{kab}-\text{kd})\,V},\\
\text{Cd} &\to \frac{q+\text{Caf kab V}+\text{Cbf kab V}+\sqrt{4\text{Caf Cbf kab}\,(-\text{kab}+\text{kd})\,V^2+(q+(\text{Caf}+\text{Cbf})\,\text{kab V})^2}}{2\,(\text{kab}-\text{kd})\,V}\}
\end{aligned}$$

We note that in order to solve for these concentrations, we would have to know the values of the two rate constants, the two inlet concentrations, the three flow rates, two in and one out, and the one reactor volume for a total of eight known quantities out of 11. This makes perfect sense as we have only three independent equations and 11 possible unknowns. To extract the right-hand sides of the three solutions and to apply them as functions we use the sequence of bracketed numbers as follows for the concentration of A:

```
In[30]:= cstr1[[1, 1, 2]]
```

$$\text{Out[30]}= -\frac{q+\text{Cbf kab V}-\text{Caf}(\text{kab}-2\text{kd})\text{V}+\sqrt{4\,\text{Caf Cbf kab}(-\text{kab}+\text{kd})\text{V}^2+(q+(\text{Caf}+\text{Cbf})\text{kab V})^2}}{2(\text{kab}-\text{kd})\text{V}}$$

Now we can really see why the CSTR operated at steady state is so different from the transient batch reactor. If the inlet feed flow rates and concentrations are fixed and set to be equal in sum to the outlet flow rate, then, because the volume of the reactor is constant, the concentrations at the exit are completely defined for fixed kinetic parameters. Or, in other words, if we need to evaluate **kab** and **kd**, we simply need to vary the flow rates and to collect the corresponding concentrations in order to fit the data to these equations to obtain their magnitudes. We do not need to do any integration in order to obtain the result. Significantly, we do not need to have fast analysis of the exit concentrations, even if the kinetics are very fast. We set up the reactor flows, let the system come to steady state, and then take as many measurements as we need of the steady-state concentration. Then we set up a new set of flows and repeat the process. We do this for as many points as necessary in order to obtain a statistically valid set of rate parameters. This is why the steady-state flow reactor is considered to be the best experimental reactor type to be used for gathering chemical kinetics.

Why is it that the flow rate should change the concentrations at the exit of the reactor? To see this we should nondimensionalize our equations. We will divide each component balance by **V** and by **Caf**:

$$\begin{aligned}0 &= \frac{(\text{Caf} - \text{Ca})q - \text{kab}\,\text{Ca}\,\text{Cb}\,V + \text{kd}\,\text{Cd}^2\,V}{\text{Caf}\,V}\\ &= (1-\Phi\text{a})\frac{q}{V} - \text{kab}\,\Phi\text{a}\,\text{Cb} + \text{kd}\,\Phi\text{d}\,\text{Cd}^2\end{aligned}$$

We can multiply the last two terms by $\frac{\textbf{Caf}}{\textbf{Caf}}$ in order to express each concentration in non-dimensional terms:

$$0 = (1-\Phi\text{a})\frac{1}{\theta} - \text{kab}\,\Phi\text{a}\frac{\text{Caf}}{\text{Caf}}\text{Cb} + \text{kd}\,\Phi\text{d}\frac{\text{Caf}}{\text{Caf}}\text{Cd}$$

For each of the components we obtain by this procedure:

$$0 = (1 - \Phi a)\frac{1}{\theta} - \text{kab}\,\text{Caf}\,\Phi a\,\Phi b + \text{kd}\,\text{Caf}\,\Phi d^2$$
$$0 = \left(\frac{\text{Cbf}}{\text{Caf}} - \Phi b\right)\frac{1}{\theta} - \text{kab}\,\Phi a\,\Phi b + \text{kd}\,\Phi d^2$$
$$0 = -\Phi d\frac{1}{\theta} + \text{kab}\,\Phi a\,\Phi b - \text{kd}\,\Phi d^2$$

The flow rate through the reactor is **q**, and thus the holding time is $\frac{V}{q}$, which is $\boldsymbol{\theta}$. Because the stoichiometry is 1:1 we can take **Cbf** = **Caf**:

$$0 = (1 - \Phi a) - \text{kab}\,\text{Caf}\,\theta\,\Phi a\,\Phi b + \text{kd}\,\text{Caf}\,\theta\,\Phi d^2$$
$$0 = (1 - \Phi b) - \text{kab}\,\text{Caf}\,\theta\,\Phi a\,\Phi b + \text{kd}\,\text{Caf}\,\theta\,\Phi d^2$$
$$0 = -\Phi d + \text{kab}\,\text{Caf}\,\theta\,\Phi a\,\Phi b - \text{kd}\,\text{Caf}\,\theta\,\Phi d^2$$

```
In[31]:= Clear["Global`*"]

In[32]:= ndcstr1 = Simplify[
           Solve[
            {0 == (1 - Φa) - kab Caf θ Φa Φb + kd Caf θ Φd^2,
             0 == (1 - Φb) - kab Caf θ Φa Φb + kd Caf θ Φd^2,
             0 == -Φd + kab Caf θ Φa Φb - kd Caf θ Φd^2},
            {Φa, Φb, Φd}]]

         General::spell: Possible spelling error: new symbol name
         "Φb" is similar to existing symbols {Φa,Φd}.
```

$$\text{Out[32]= } \{\{\Phi a \to \frac{1 + 2\text{Caf kd}\,\theta - \sqrt{1 + 4\text{Caf kab}\,\theta + 4\text{Caf}^2\,\text{kab kd}\,\theta^2}}{-2\text{Caf kab}\,\theta + 2\text{Caf kd}\,\theta},$$
$$\Phi b \to \frac{1 + 2\text{Caf kd}\,\theta - \sqrt{1 + 4\text{Caf kab}\,\theta + 4\text{Caf}^2\,\text{kab kd}\,\theta^2}}{-2\text{Caf kab}\,\theta + 2\text{Caf kd}\,\theta},$$
$$\Phi d \to \frac{1 + 2\text{Caf kad}\,\theta - \sqrt{1 + 4\text{Caf kab}\,\theta + 4\text{Caf}^2\,\text{kab kd}\,\theta^2}}{2\text{Caf kab}\,\theta - 2\text{Caf kd}\,\theta}\},$$
$$\{\Phi a \to \frac{1 + 2\text{Caf kd}\,\theta + \sqrt{1 + 4\text{Caf kab}\,\theta + 4\text{Caf}^2\,\text{kab kd}\,\theta^2}}{-2\text{Caf kab}\,\theta + 2\text{Caf kd}\,\theta},$$
$$\Phi b \to \frac{1 + 2\text{Caf kd}\,\theta + \sqrt{1 + 4\text{Caf kab}\,\theta + 4\text{Caf}^2\,\text{kab kd}\,\theta^2}}{-2\text{Caf kab}\,\theta + 2\text{Caf kd}\,\theta},$$
$$\Phi d \to \frac{1 + 2\text{Caf kad}\,\theta + \sqrt{1 + 4\text{Caf kab}\,\theta + 4\text{Caf}^2\,\text{kab kd}\,\theta^2}}{2\text{Caf kab}\,\theta - 2\text{Caf kd}\,\theta}\}\}$$

These are the same solutions as before, but in nondimensionalized form they are more compact and more easily understood.

```
In[33]:= φa[θ_] := ndcstr1[[1, 1, 2]]
         φb[θ_] := ndcstr1[[1, 2, 2]]
         φd[θ_] := ndcstr1[[1, 3, 2]]

         Caf = .25;
         kd = 1.5;
         kab = 1.1;
         tmin = 0.001;
         tmax = 100;

         αeq = NSolve[kab/kd == α^2/(1 - α)^2, α]

         Plot[{φa[θ],φd[θ]},
           {θ, tmin, tmax},
           PlotStyle → {{Dashing[{0.15, 0.05}], Thickness[0.01],
            GrayLevel[0]}, {Dashing[{0.15, 0.05}],
            Thickness[.01], GrayLevel[.5]}},
           PlotLabel → "rd = φa; bl = φd; lines = eq",
           AxesLabel → {"θ", "φi[θ]"},
           Epilog → {
            {Dashing[{0.01, 0.01}], GrayLevel[.5],
             Line[{{tmin, αeq[[2, 1, 2]]},{tmax, αeq[[2, 1, 2]]}}]},
            {Dashing[{0.01, 0.01}], Line[{{tmin, 1-αeq[[2, 1, 2]]},
             {tmax, 1 - αeq[[2, 1, 2]]}}]}}
          ];
```

9.3 Time Dependence—The Transient Approach to Steady-State and Saturation Kinetics

Although the steady-state CSTR is simple to operate and analyze and even though it offers real advantages to the kineticist, it is also true that these systems must go through a start-up. They do not start up and necessarily achieve steady state instantaneously. The time period in which the system moves toward a steady-state condition is called the *transient*, meaning that the system is in transition from one that is time-dependent to one that is time-independent.

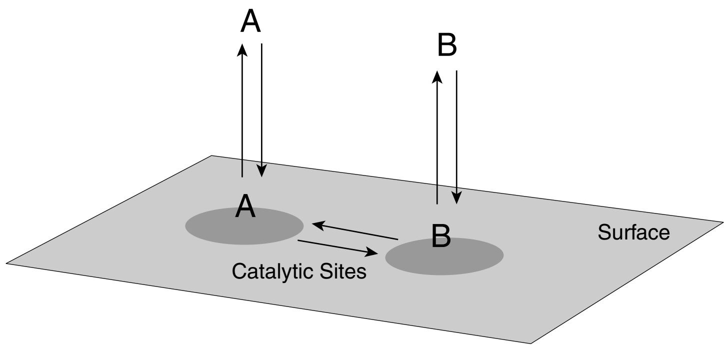


Figure 3

We have no way of knowing how long it will take a given reaction or set of reactions to achieve a steady state in the CSTR before we either do an experiment or solve the time-dependent model equations. If we choose to do experiments as a means to assessing this, then we need to be prepared to do many of them. But if we already know the kinetics, then we do the analysis and the math instead. If we do it correctly, then it is fast and it provides us with insights that complement the experiments and in many cases provides interpretations that a purely experimental approach cannot yield. Therefore, in this problem we will consider just such a case with a more complex set of kinetics.

Consider the reaction of a molecule that takes place on a solid catalyst surface. This reaction simply involves converting one form of the molecule into another: in other words, it is an isomerization reaction. But the reaction in question only takes place on the catalyst surface and not without the catalyst. (See Figure 3.)

As we saw in Chapter 6, when we analyze a reaction of this kind we find that at least two steps are involved—adsorption and surface reaction. The adsorption equilibrium steps take place by the interaction of the molecule in the bulk phase with a so-called adsorption site on the solid surface. The adsorption site is the locus of points on the surface that interact directly with the molecule:

$$A_{\text{bulk}} + \text{site} \rightleftarrows A_{\text{surface}}$$
$$A_{\text{surface}} \rightarrow B_{\text{surface}}$$
$$B_{\text{surface}} \rightleftarrows B_{\text{bulk}} + \text{site}$$

Once **B** is formed, it too undergoes adsorption and desorption. The desorption carries **B** from the surface and into the bulk fluid phase. In this case we will assume that the reaction is irreversible and that the rate of this reaction is first order in the surface concentration of **A**. It also is first order in the concentration of surface sites. Thus the kinetics follow a simple surface rate law:

$$r_{A-} = k'_{\text{surface}} C_{A,\text{surface}} C_{\text{sites}}$$

The surface concentration is difficult to measure; thus we need to reexpress it in terms of the bulk phase concentration of species **A**. To do this we take advantage of the fact that the molecules often adsorb and desorb so quickly that they come to equilibrium rapidly with the surface sites. Therefore, subject to this assumption, we can express the surface concentration in terms of the equilibrium. The equilibrium gives rise to the following relationship for the surface concentration of **A** in terms of the bulk concentration of **A**:

$$C_{A,\text{surface}} = \frac{K_A C_A}{1 + K_A C_A}$$

We can substitute this expression into the rate expression for the reaction. This leads to this rate in terms of the bulk phase concentrations:

$$r_{A-} = k'_{\text{surface}} C_{\text{sites}} \frac{K_A C_A}{1 + K_A C_A}$$

The concentration of sites can be incorporated into the rate constant by rewriting the product of the surface site concentration and the surface rate constant simply as a rate constant:

$$\mathrm{k} = k'_{\text{surface}} C_{\text{sites}}$$

We can do this because the surface site concentration is also a constant. Thus the overall rate for this catalytic reaction is:

$$r_{A-} = \frac{k K_A C_A}{1 + K_A C_A}$$

The time-dependent component balance equations for **A** and **B** in the CSTR are as follows:

$$\frac{\mathrm{d}C_A \epsilon \mathrm{V}}{\mathrm{dt}} = (C_{\mathrm{Af}} - C_A)\mathrm{q} - (1 - \epsilon) r_{A-} \mathrm{V}$$
$$\frac{\mathrm{d}C_B \epsilon V}{\mathrm{dt}} = -C_B \mathrm{q} + (1 - \epsilon) r_{A-} \mathrm{V}$$

Recall that the solid catalyst occupies a fraction $\mathbf{1} - \epsilon$ of the reactor volume leaving a fraction ϵ for the fluid phase volume. We write the balances in terms of the fluid phase. The kinetics have been written also in terms of the fluid phase concentration, but they are written for a process that occurs within the second phase, which is the catalyst. This is called the *pseudo-homogeneous* approximation. In this case we take that phase as homogeneous and continuous, and occupying the $(\mathbf{1}-\epsilon)$ of the reactor volume. We can substitute into these equations the kinetics we just derived:

$$\frac{\mathrm{d}C_A\epsilon\mathrm{V}}{\mathrm{dt}} = (C_{\mathrm{Af}} - C_A)\mathrm{q} - (1-\epsilon)\frac{kK_AC_A}{1+K_AC_A}\mathrm{V}$$
$$\frac{\mathrm{d}C_B\epsilon\mathrm{V}}{\mathrm{dt}} = -C_B\mathrm{q} + (1-\epsilon)\frac{kK_AC_A}{1+K_AC_A}\mathrm{V}$$

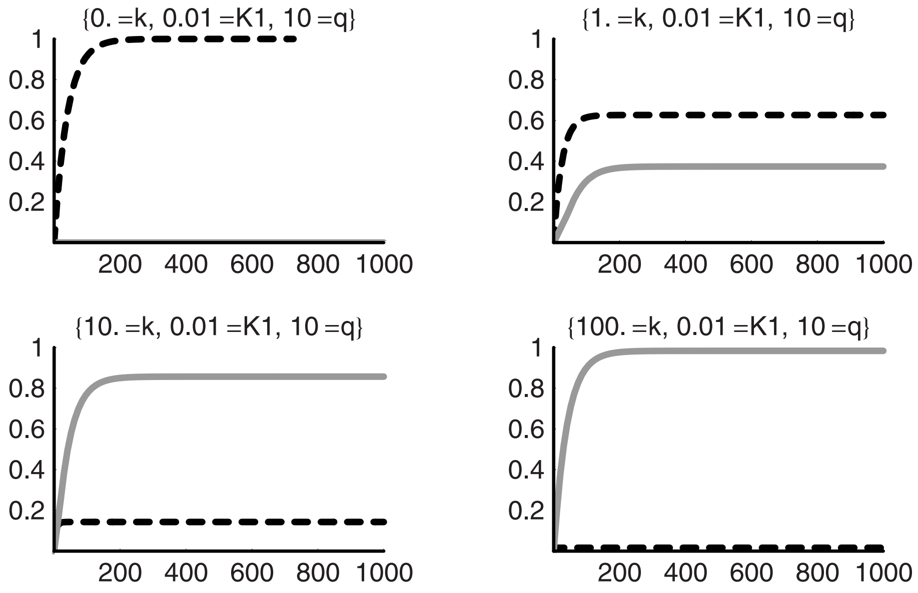
The integration of these two equations in time will show us how long it will take the reactor to achieve a steady-state conversion of **A** and production of **B**.

The first step is to set up a solution to these equations and a graphical display of the results. Using **NDSolve**, we can solve these time-dependent equations to find the concentrations as functions of time. We make a new **Module** function "**cstr4**" to handle this.

```
In[33]:= cstr4[k_, K1_, q_, tmax_], :=
          Module[
           {Caf = 1, V = 1000, Cao = 0, Cbo = 0, ϵ = .4, solns,
            Ca, Cb, CA, CB, t},
           solns = NDSolve[{
         ϵCa'[t] == (Caf - Ca[t]) q/V - (1 - ϵ) kK1Ca[t]/(1+K1Ca[t]),
         ϵCb'[t] == -Cb[t] q/V + (1 - ϵ) k K1 Ca[t]/(1+K1 Ca[t]),
             Ca[0] == Cao, Cb[0] == Cbo},
         {Ca[t], Cb[t]}, {t, 0, tmax}];

         CA[t] = Evaluate[Ca[t] /. solns];
         CB[t] = Evaluate[Cb[t] /. solns];

           SetOptions[{Plot}, AxesStyle  → {Thickness[0.01]},
            PlotStyle  → {Thickness[0.006]},
            DefaultFont  → {"Helvetica", 10}];

         Plot[{CA[t], CB[t]}, {t, 0, tmax},
            PlotStyle  → {{Thickness[0.02], Dashing[{0.04, 0.04}],
             GrayLevel[0]}, {Thickness[0.02], GrayLevel[ .6]}},
            PlotRange  → {{0, tmax}, {0, Caf}},
            PlotLabel  → {k "=k", K1 "=K1", q "=q"},
            DisplayFunction  → Identity]
          ]
```

We can examine the solution at a few extremes to try and understand how the parameter values affect its behavior. We can take $\mathbf{k} = 0$ first to see how the system responds to the flow of **A**. This gives us a sense of how long it takes the flow and mixing to come to steady state in the absence of reaction. Experimentally, we could do this with a noncatalytic solid present. Keeping all else the same, we vary the rate constant **k** from 0 to 100 in multiples of 10.

```
In[34]:= Show[GraphicsArray[{{cstr4[0., .01, 10, 1000],
            cstr4[1., .01, 10, 1000]}, {cstr4[10., .01, 10, 1000],
            cstr4[100., .01, 10, 1000]}}]];
```



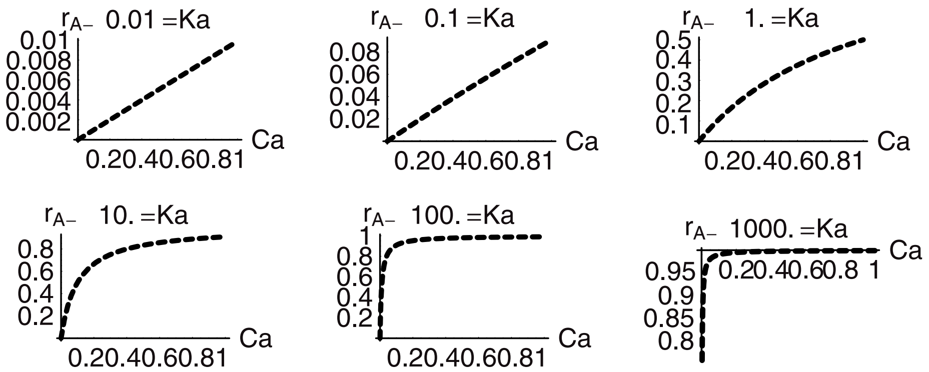
In the first plot, with $\mathbf{k} = 0$, we note that it takes the system about 200 time units at this flow rate to reach a steady state. As we raise the rate constant from unity to 10 and then to 100, the steady-state concentrations of **A** (dashed) drop from 0.6 to less than 0.2 to nearly zero. We also see that the time to reach steady state for the product **B** (solid) is about 200 time units in each case, whereas for **A** it is always less than that time, and the time to steady state shortens as the rate of chemical reaction increases. This is because the concentration of **A** at steady state decreases as **k** increases; thus the time required to reach the plateau is less.

Looking back at the rate expression we see:

$$r_{A-} = \frac{kK_AC_A}{1 + K_AC_A}$$

If $\boldsymbol{K_AC_A}$ is large compared to unity, then the rate reduces to just **k**, that is, a constant or "zeroth-order" rate. Alternatively, when $\boldsymbol{K_AC_A}$ is small compared to unity, the rate becomes $\boldsymbol{kK_AC_A}$ or first order with respect to $\boldsymbol{C_A}$. Letting **k** be unity, for example, we can vary $\boldsymbol{K_A}$ from 10^{-2} to 10^3 over a range of $\boldsymbol{C_A}$ from zero to unity and then plot the results as an array to see the effect of the adsorption constant:

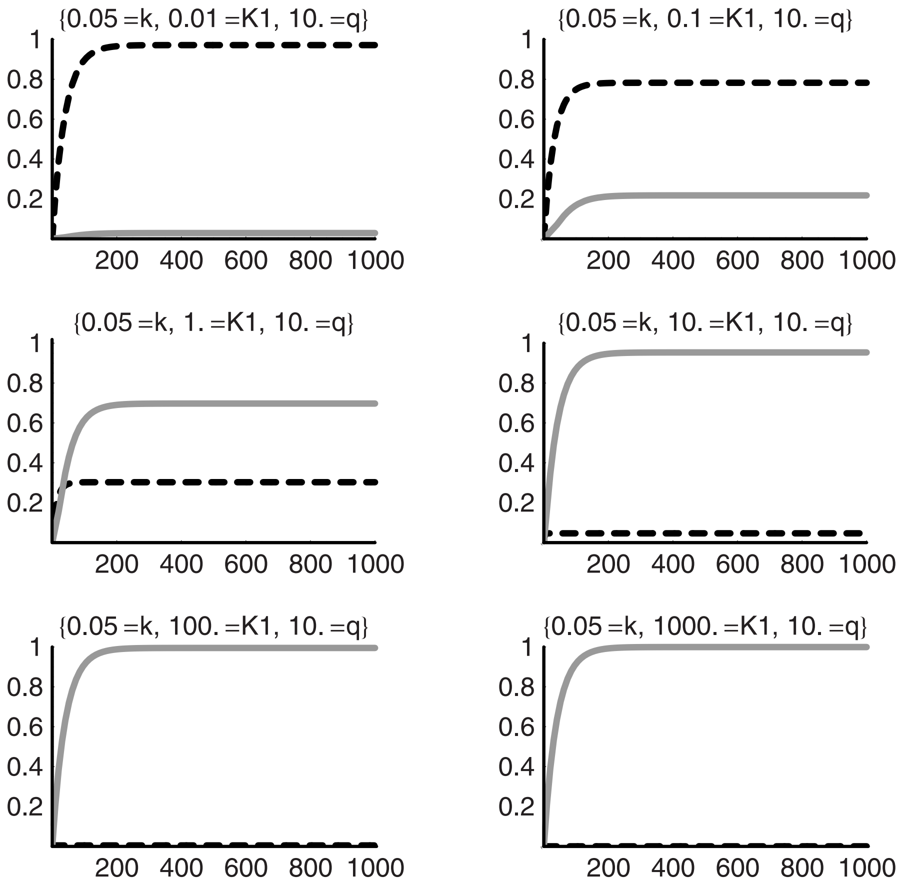
```
In[35]:= k = 1;
         SetOptions[{Plot},
           AxesStyle  → {Thickness[0.01]},
           DefaultFont  → {"Helvetica", 10}];
         Show[
           GraphicsArray[
            {Table[Plot[k 10^n Ca/(1 + 10^n Ca), {Ca, 0, 1},
               DisplayFunction  → Identity,
                PlotStyle  → {{Thickness[0.03],
                 Dashing[{0.04, 0.04}], GrayLevel[0]}},
                AxesLabel  → {"Ca", "rA-"},
                PlotLabel  → 10^n "=Ka"],
                {n, -2., 0}],
             Table[Plot[k 10^n Ca/(1 + 10^n Ca),{Ca,0,1},
               DisplayFunction  → Identity,
               PlotStyle → {{Thickness[0.03],
                Dashing[{0.04, 0.04}], GrayLevel[0]}},
               AxesLabel  → {"Ca","rA–"},
               PlotLabel  → 10^n "=Ka"],
               {n, 1., 3}]}
                         ]
               ];
```



In the first two plots, $K_A C_A$ is small compared to unity and we see that the rate is first order in concentration C_A over the whole range. The last two plots are cases in which $K_A C_A$ is large compared to unity at most values of C_A, except for the very smallest ones. Hence the rate becomes constant at larger values of C_A and this is called saturation. It means that the rate cannot increase in magnitude even though the concentration of reactant has been increased and the rate is an apparently strong function of C_A. In fact, when $K_A C_A$ is large, we see that the rate is a very weak function of C_A. Hence Langmuir-Hinshelwood (and Michaelis-Menton) kinetics are often referred to as "saturation kinetics." The intermediate values of **Ka** lead to intermediate results and rate behavior.

How will the effect of saturation kinetics show up in the evolution to the steady state in a CSTR? We can find this out by letting **K1** vary over this range of magnitudes from 10^{-2} to 10^3 within the **Module** function "**cstr4**." We also have taken the rate constant down from 1.0 to 0.05 to make the differences more evident for the same values of **q** and **V**, that is, the holding time. This does not change the effect of **K1** because we are comparing its product with **Ca** to unity:

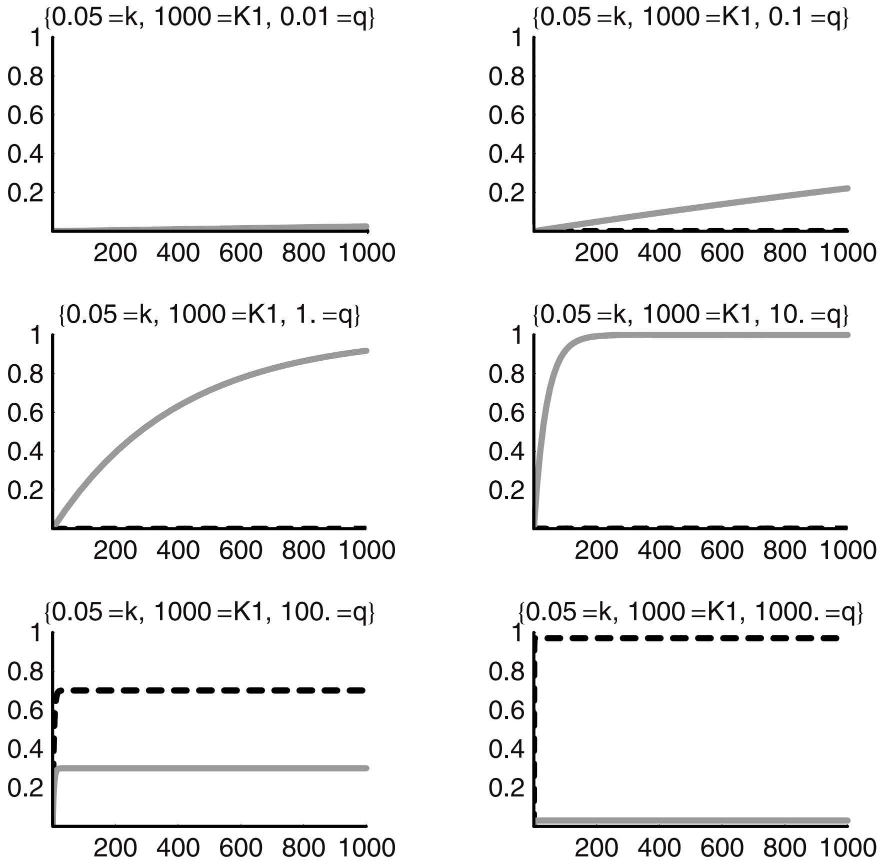
```
In[38]:= Show[
           GraphicsArray[
                Partition[
                     Table[cstr4[.05, 10^n, 10., 1000], {n, -2., 3}],
                         2]
                        ]
                     ];
```



The results are not dramatically different than what we had seen before. The saturation kinetics at these flow rates, that is, holding times, give rise to complete conversion as we see in the last two plots in which **K1** has values of 10^2 and 10^3.

Finally, the last calculation prompts the question of holding time effect. If we vary the flow rate **q** at fixed **V**, keeping **k** and **K1** constant we should see the conversion rise with longer holding times, that is, lower flow rates:

```
In[39]:= Show[
              GraphicsArray[
                   Partition[
                        Table[cstr4[.05, 10^3, 10^n, 1000], {n, -2., 3}],
                          2]
                        ]
                 ];
```

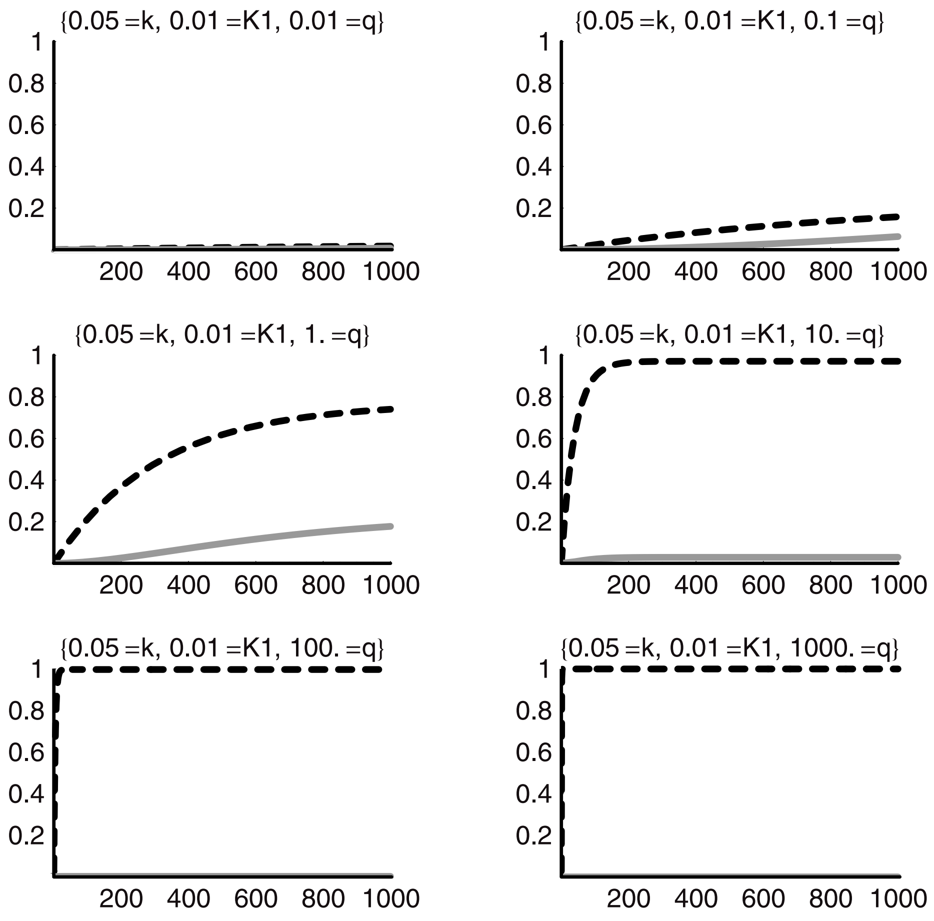


The effect is dramatic. We have taken **K1** = 1000, which puts the kinetics in the zeroth-order regime. We see in the upper-left plot that the conversion appears to be complete, but even after

1000 time units the system is still far from the steady state. At the lower right, the conversion is essentially zero, and the system comes to steady state nearly instantaneously. The other plots show self-consistent behaviors. Notice that with **q** = 10, the approach to steady state is fast and the conversion is essentially complete.

But what would happen if we took the adsorption constant **K1** to be quite small, say, on the order of 10^{-2}? We will find out in the following:

```
In[40]:= Show[
           GraphicsArray[
                Partition[
                     Table[cstr4[.05, 1.10^-2, 10^n, 1000], {n, -2., 3}],
                          2]
                        ]
              ];
```



Aha! This is very interesting and instructive. Here we see that **K1** is so small that at any of the holding times (even the highest ones) the rate is so small that the conversion is effectively zero throughout the range. This means that the catalyst just lacks the adsorption forces necessary to make the reaction run fast. In other words, for the reaction to take place efficiently, the reactant **A** must be adsorbed. If it is not adsorbed to an appreciable extent, then the rate is always going to be small unless the rate constant for the surface reaction is very high.

Before we go on to the next section we should do some "housekeeping." The command **Names** is shown below with its *Mathematica* explanation:

```
In[41]:= ?Names

            Names["string"] gives a list of the names of symbols
             which match the string. Names["string",
             SpellingCorrection  → True] includes names which match
             after spelling correction.
```

As we have worked through this session, or any other session, we have generated many new **Names** for functions. These show up in the **Global** context. If we ask for them we will get a list of those that we have created and used so far. (We show this in the following, but we have suppressed the output.)

```
In[42]:= Names["Global`*"];
```

To clean up the **Global** context, we can **Remove** everything we have created in the **Global** context as follows. Asking for **Names** in this context once again returns an empty set:

```
In[43]:= Remove["Global`*"]
         Names["Global`*"]

Out[44]= {}
```

9.4 The Design of an Optimal CSTR

Several questions arise with respect to the design of an optimal CSTR for a given chemistry. Chief among these are several equations that relate conversion, cost and profitability to each other. As we can plainly see, the volume of the CSTR controls the extent of conversion. Thus the magnitude of the volume goes up with the requirement of conversion. It would seem, then, that we might simply want the largest volume reactor that we can build as this will provide the highest conversion of reactant to product. However, it is self-evident that such logic is highly flawed because the cost of the reactor must scale with its size. Therefore, how do we decide what level of conversion and reactor size is appropriate? The value of the product relative to that of the reactant must be a critical factor. We can do an analysis of the following reaction to see how this works:

$$\mathrm{A} \rightarrow \mathrm{B}$$
$$r_{A-} = r_{B+} = \mathrm{k}C_A$$

The steady-state CSTR equations for components A and B are as follows:

$$(C_{\mathrm{Af}} - C_A)\mathrm{q} - \mathrm{k}C_A\mathrm{V} = 0$$
$$-C_B\mathrm{q} + \mathrm{k}C_A\mathrm{V} = 0$$

The inlet concentration $\boldsymbol{C_{\mathrm{Af}}}$ is a good reference point for reaction. We can normalize the equations by dividing through by $\boldsymbol{C_{\mathrm{Af}}}$:

$$\left(1 - \frac{C_A}{C_{\mathrm{Af}}}\right)\mathrm{q} - \mathrm{k}C_A\mathrm{V} = 0$$
$$-\frac{C_B}{C_{\mathrm{Af}}}\mathrm{q} + \mathrm{k}\frac{C_A}{C_{\mathrm{Af}}}\mathrm{V} = 0$$

The conversion of A can be written as $(\mathbf{1} - \frac{C_A}{C_{\mathrm{Af}}})$ and $\boldsymbol{C_A} = \boldsymbol{C_{\mathrm{Af}}}\ (\mathbf{1} - \boldsymbol{X_A})$. Hence the equation for **A** can be rewritten in terms of $\boldsymbol{X_A}$:

$$\mathrm{X}_A\mathrm{q} - \mathrm{kV}(1 - X_A) = 0$$
$$X_A - \mathrm{k}\frac{V}{q}(1 - X_A) = 0$$
$$X_A - \mathrm{k}\tau(1 - X_A) = 0$$

where τ is the holding time in the CSTR. We can see that we can solve for it in terms of the conversion and the first-order rate constant. The inverse of this rate constant has dimensions of time.

$$\tau = \frac{X_A}{k(1 - X_A)}$$
$$V = \frac{X_A}{k(1 - X_A)}\mathrm{q}$$

We can immediately see the direct relationship between the volume of the reactor and the conversion of **A**. To make this absolutely clear we can rearrange one more time to give:

$$\frac{1}{1+\frac{q}{kV}} = X_A$$

When we consider the extremes we acquire a feeling for the behavior of this relationship:

$$\begin{array}{llll} \mathrm{V} \to 0; & \dfrac{q}{kV} \to \infty; & \dfrac{1}{1+\infty} \to 0 \to X_A \\ \mathrm{V} \to \infty; & \dfrac{q}{kV} \to 0; & \dfrac{1}{1+0} \to 1 \to X_A \end{array}$$

On this basis we can see that all we really have learned is what we already knew intuitively—making the reactor as large as possible will provide the highest possible conversion. Thus we really need a better measure of our objective than simply the conversion.

When a chemical engineer balances an equation it must be done first from the perspective of stoichiometry, and second taking value and profit into consideration. Thus for a given reaction:

$$\mathrm{aA + bB \to dD + eE}$$

As in Chapter 7 we showed that the maximum profit potential is:

$$\text{max. profit potential} = \left[\left(\mathrm{d}\frac{\$}{\text{mole } D} + \mathrm{e}\frac{\$}{\text{mole } E}\right) - \left(\mathrm{a}\frac{\$}{\text{mole } A} + \mathrm{b}\frac{\$}{\text{mole } B}\right)\right]$$

The conversion dictates how much product will be made and how much of the starting material will be left and this thus dictates the value of the mixture. For the simpler case we are considering we can write:

$$\begin{aligned} \text{max. profit potential} &= \left[\left(\mathrm{b}\frac{\$}{\text{mole } D}\right) - \left(\mathrm{a}\frac{\$}{\text{mole } A}\right)\right] \\ &= \left[\left(C_b \mathrm{q}\frac{\$}{\text{mole } B} t\right) - \left(C_a \mathrm{q}\frac{\$}{\text{mole } A} t\right)\right] \\ &= \left[\left(C_b \mathrm{q}\frac{\$}{\text{mole } B} t\right) - \left(C_a \mathrm{q}\frac{\$}{\text{mole } A} t\right)\right] \end{aligned}$$

$$\begin{aligned}
&= \left[\left(C_{\mathrm{af}} X_A \mathrm{q} \frac{\$}{\text{mole } B} t\right) - \left(C_{\mathrm{af}}\, \mathrm{q}(1 - X_A) \frac{\$}{\text{mole } A} t\right)\right] \\
&= C_{\mathrm{af}}\, \mathrm{q}\, t \left[\left(X_A \frac{\$}{\text{mole } B}\right) - (1 - X_A) \frac{\$}{\text{mole } A}\right] \\
&= C_{\mathrm{af}}\, \mathrm{q}\, t \left[X_A \frac{\$}{\text{mole } B} - \frac{\$}{\text{mole } A} + X_A \frac{\$}{\text{mole } A}\right] \\
&= C_{\mathrm{af}}\, \mathrm{q}\, t \left[X_A \frac{\$}{\text{mole } B} + X_A \frac{\$}{\text{mole } A} - \frac{\$}{\text{mole } A}\right] \\
&= C_{\mathrm{af}}\, \mathrm{q}\, t \left[X_A \left(\frac{\$}{\text{mole } B} + \frac{\$}{\text{mole } A}\right) - \frac{\$}{\text{mole } A}\right]
\end{aligned}$$

The maximum profit potential for a chemical reaction is only a crude, zeroth-order measure of value. To gain a better measure of the economics, we must have a fuller analysis of the process. The maximum profit potential can never be achieved because it costs money both to invest in the process hardware and to operate the process. If we simply were to try to maximize the potential profit by maximizing the conversion X_A, then we would need a reactor of infinite volume:

$$\begin{aligned}
&X_A \to 1 \\
\Rightarrow &\frac{q}{kV} \to 0 \\
&\Rightarrow \mathrm{V} \to \infty
\end{aligned}$$

However, as $\boldsymbol{V} \longrightarrow \infty$ the cost of the reactor also goes to infinity and the net profit must go to zero. Hence the net profit must be a better measure of value generation and this must consist of at least the cost of the reactor and its operation in addition to the cost of the reactant and the value of the product.

\$ Net Profit = {\$ Value of Product − \$ Cost of Reactant − \$ Investment in Reactor − \$ Cost of Operation}

This is of course just another accounting statement of the kind that we have used all along to this point. Generally, it says that the net rate of accumulation of a measurable is simply the difference between the rate input minus the rate output:

Rate of Accumulation = {Rate of Input − Rate of Output}

The individual components of this equation for profit can be written in terms of a preset time over which we will evaluate the project $\boldsymbol{t_p}$ and the relevant parameters and variables of the problems. We will assume we are still considering the same simple reaction as before and the reactor is a steady-state CSTR and we have the following terms:

$$\textbf{\$ Value of Product} = C_B \mathbf{q} \frac{\$}{\textbf{mol } B} t_p$$

$$\textbf{\$ Cost of Reactants} = C_{\text{Af}}\, \mathbf{q} \frac{\$}{\textbf{mol } A} t_p$$

$$\textbf{\$ Investment in Reactor} = V_{\textbf{reactor}} \alpha$$

$$\textbf{\$ Cost of Reactor Operation} = t_p \beta$$

$$\textbf{\$ Net Profit} = C_B \mathbf{q} \frac{\$}{\textbf{mol } B} t_p - C_{\text{Af}} \mathbf{q} \frac{\$}{\textbf{mol } A} t_p - V_{\textbf{reactor}} \alpha - t_p \beta$$

$$C_B = C_{\text{Af}} X_A \qquad V_{\textbf{reactor}} = \frac{q\, X_A}{k(1 - X_A)}$$

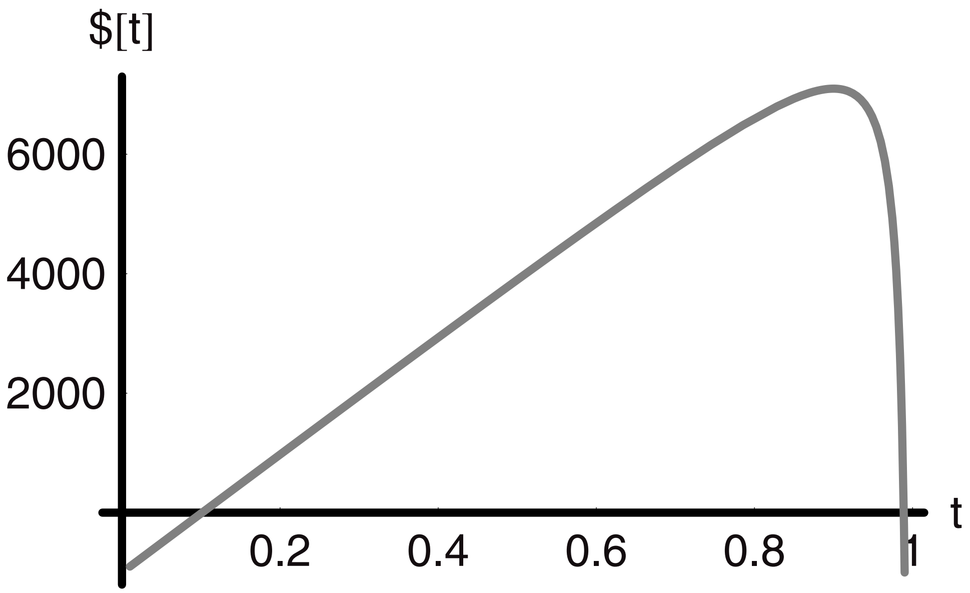
$$\textbf{\$ Net Profit} = C_{\text{Af}} X_A \mathbf{q} \frac{\$}{\textbf{mol } B} t_p - C_{\text{Af}} \mathbf{q} \frac{\$}{\textbf{mol } A} t_p - \frac{\mathbf{q} X_A}{k(1 - X_A)} \alpha - t_p \beta$$

$$\textbf{\$ Net Profit} = C_{\text{Af}} \mathbf{q} t_p \left(X_A \frac{\$}{\textbf{mol } B} - \frac{\$}{\textbf{mol } A} - \frac{\beta}{C_{\text{Af}}\, q} \right) - \frac{\alpha q X_A}{k(1 - X_A)}$$

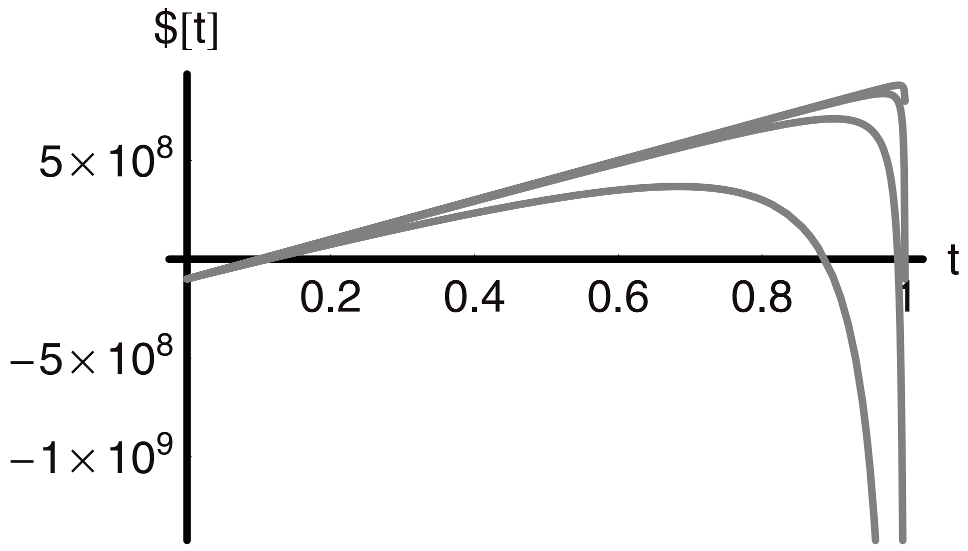
This equation now allows us to compute an optimal profit as a function of the conversion or, in other words, as a function of the reactor size or holding time.

```
In[45]:= netprofit[Caf_, q_, t_, $a_, $b_, k_, $v_, $t_, xa_]:=
           Caf qt((xa/$b) - 1/$a) - q(xa)/(k(1-xa)$v) - t/$t

In[46]:= Caf = 1;
         q = 10;
         t = 100;
         $a = 1;
         $b = .1;
         k = 0.01;
         $v = 10;
         $t = 50;

In[54]:= netprofit[Caf, q, t, $a, $b, k, $v, $t, x]
         SetOptions[{Plot}, AxesStyle → {Thickness[0.01]},
           PlotStyle → {Thickness[0.006]},
           DefaultFont → {"Helvetica", 17}];
         Plot[netprofit[Caf, q, t, $a, $b, k, $v, $t, xa],
          {xa, 0.01, .99}, AxesOrigin → {0, 0},
           PlotStyle → {GrayLevel[0.5], Thickness[0.01]},
            AxesLabel → {"t", "$[t]"}];
```

Out[54]= $-2 - \frac{100.x}{1-x} + 1000(-1 + 10.x)$



```
In[57]:= Caf = 1;
         q =.
         t = 100;
         $a = 1;
         $b = .1;
         k =.
         $v = 10;
         $t = 50;
         xa =.
         b = Table[netprofit[Caf, 10^6, t, $a, $b, 10^-n, $v, $t, xa],
                   {n, 0, 6, 1}];
         %;
         Plot[{b[[1]], b[[2]], b[[3]], b[[4]]}, {xa, 0, .999},
           PlotStyle → {{GrayLevel[0.5], Thickness[0.01]}},
            AxesLabel → {"t", "$[t]"}];
```



The curves from the preceding graph show a clear optimum with respect to the holding time and parametric in the rate constant.

9.5 Plug Flow Reactor

We turn now to the plug flow reactor. Here, as we have said, there is absolutely no mixing in the direction of flow but perfect mixing perpendicular to it, that is, between the centerline and the walls. This special case of a tubular reactor can be operated transiently or in the steady state, but it is the latter mode that is most often considered for kinetics and design. Consider the reactor shown in Figure 4 in which **A** is converted to **B** irreversibly and with linear kinetics.

For the first time we must as a consequence of the plug flow take into account spatial variation as well as time dependence. This means that the concentrations of **A** and **B** will have **z**- and **t**-dependence and the equations describing them will be made up of partial rather than ordinary differentials. We can derive the equation that describes the plug flow system by first visualizing a zone of reaction (Figure 5) that corresponds to a differential control volume **Acr dz**.

The total differential of the concentration is equal to the rate of chemical reaction in this zone over some differential time **dt**. In *Mathematica* the total differential of **f[x, y]** is given as Dt[f[*x*, *y*]]:

```
In[69]:= Dt[f[x, y]]

Out[69]= Dt[y]f^(0,1)[x,y] + Dt[x]f^(1,0)[x,y]
```

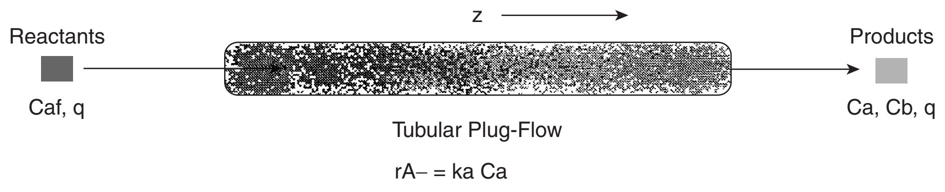


Figure 4

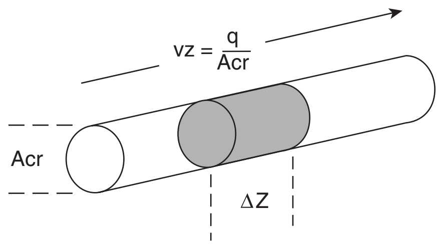


Figure 5

This output statement means:

$$\partial_y f[x, y]\,\mathrm{dy} + \partial_x f[x, y]$$

or

$$\frac{\partial f[x, y]}{\partial y}\,\mathrm{dy} + \frac{\partial f[x, y]}{\partial x}\,\mathrm{dx}$$

Taking the total differential of the concentration and the rate, we obtain:

```
In[70]:=  z =.
          t =.
          Dt[Ca[z, t]] == -ra Dt[t]
```

$$\mathit{Out[72]=}\ \mathrm{Dt[t]Ca}^{(0,1)}\mathrm{[z,t]} + \mathrm{Dt[z]Ca}^{(1,0)}\mathrm{[z,t]}\ == \ \mathrm{-ra\,Dt[t]}$$

We can use the shortcut of dividing through by **Dt[t]**, that is, the derivative of the time **dt** to obtain the following:

$$\mathit{In[73]:=}\ \mathbf{Simplify}\left[\frac{\mathbf{Dt[t]Ca}^{(0,1)}\mathbf{[z,t]} + \mathbf{Dt[z]Ca}^{(1,0)}\mathbf{[z,t]}}{\mathbf{Dt[t]}}\right]\ == \ \frac{\mathbf{-ra\,Dt[t]}}{\mathbf{Dt[t]}}$$

$$\mathit{Out[73]=}\ \mathrm{Ca}^{(0,1)}\mathrm{[z,t]} + \frac{\mathrm{Dt[z]Ca}^{(1,0)}\mathrm{[z,t]}}{\mathrm{Dt[t]}}\ == \ \mathrm{-ra}$$

In the second term on the left-hand side, we have the ratio of $\frac{\mathrm{Dt[z]}}{\mathrm{Dt[t]}}$, ($\frac{dz}{dt}$). This is the velocity of the plug of gas through the differential volume in the axial direction. We will use the symbol **vz** for the axial velocity. Making this substitution we have:

```
In[74]:=  Ca(0,1)[z,t] + vz Ca(1,0)[z,t] == -ra

Out[74]=  Ca(0,1)[z,t] + vz Ca(1,0)[z,t] == -ra
```

Written in traditional form this states:

$$\frac{\partial \mathrm{Ca}[z,t]}{\partial t} = -v_z \frac{\partial \mathrm{Ca}[z,t]}{\partial z} - r_{A-}$$

In words, this mathematical "sentence" states that the partial derivative of the concentration of **A** with respect to time is equal to the negative of the sum of the product of the axial velocity and the partial derivative of the concentration of **A** with respect to position and the rate of reaction of **A**.

For the product **B** we would have the following equation:

$$\frac{\partial \mathrm{Cb}[z,t]}{\partial t} = -v_z \frac{\partial \mathrm{Cb}[z,t]}{\partial z} + r_{A-}$$

A more intuitive way to arrive at these equations begins at the well-mixed approximation. Imagine that within the region **Δz** the fluid phase is "well mixed." The volume of this region is **ΔV = AcrΔz**. The flow rate across the volume is taken to be **q**, and the concentrations in the volume element are $\boldsymbol{C_{i,1}}$ while those exiting are $\boldsymbol{C_{i,2}}$. Writing the mass balance for **A** we have:

$$\frac{d\mathrm{Ca}\Delta V}{dt} = (C_{a,1} - C_{a,2})q - r_{A-}\Delta V$$

$$\frac{d\mathrm{Ca\,Acr}\Delta z}{dt} = (C_{a,1} - C_{a,2})q - r_{A-}\mathrm{Acr}\Delta z$$

$$\frac{d\mathrm{Ca}}{dt} = \frac{(C_{a,1} - C_{a,2})q}{\mathrm{Acr}\Delta z} - r_{A-}$$

$$\frac{d\mathrm{Ca}}{dt} = v_z \frac{-\Delta C_a}{\Delta z} - r_{A-}$$

We can write the corresponding differential difference equation for component **B**. When we take the limit as **Δz ⟶ 0**, these two differential equations become partial differential equations:

$$\underset{\Delta z \to 0}{\mathrm{Limit}}\left[\frac{d\mathrm{Ca}}{dt} = -v_z \frac{\Delta C_a}{\Delta z} - r_{A-}\right] \Rightarrow \frac{\partial \mathrm{Ca}}{\partial t} = -v_z \frac{\partial C_a}{\partial z} - r_{A-}$$

$$\underset{\Delta z \to 0}{\mathrm{Limit}}\left[\frac{d\mathrm{Cb}}{dt} = -v_z \frac{\Delta C_b}{\Delta z} + r_{A-}\right] \Rightarrow \frac{\partial \mathrm{Cb}}{\partial t} = -v_z \frac{\partial C_b}{\partial z} + r_{A-}$$

Thinking physically, it is as if we have taken a slice out of a poorly mixed reactor that is of infinitesimal thickness, but across which the mixing is perfect. This is a "**CSTR**" of vanishing or differential thickness **Δz** and cross-sectional area **Acr**.

We are usually interested in solving these equations for the concentrations at steady state. In this condition the time differentials are as usual identically zero. Recall that v_z divided into a distance **z** would be the time $\boldsymbol{\tau}$ required to translate across that distance. If we divide v_z into the differential distance **dz** then we have the differential time $\mathbf{d}\boldsymbol{\tau}$ required to translate across the infinitesimal distance. Thus $\frac{v_z}{dz}$ is just $\frac{\mathbf{1}}{\mathbf{d}\boldsymbol{\tau}}$, and the equations transform at the steady state as follows:

$$\frac{q}{\text{Acr}}\frac{\text{d}C_a}{\text{dz}} = -r_{A-} \Rightarrow v_z\frac{\text{d}C_a}{\text{dz}} = -r_{A-} \Rightarrow \frac{\text{d}C_a}{\text{d}\tau} = -r_{A-}$$
$$\frac{q}{\text{Acr}}\frac{\text{d}C_b}{\text{dz}} = +r_{A-} \Rightarrow v_z\frac{\text{d}C_b}{\text{dz}} = +r_{A-} \Rightarrow \frac{\text{d}C_b}{\text{d}\tau} = +r_{A-}$$

Remarkably, the form of the steady-state **PFR** equations is identical to the form of the fully transient well-mixed batch reactor with no volume change. The only difference is that instead of the derivative with respect to real time, the **PFR** equations involve the derivative with respect to reduced time. This is a very significant result. It shows us why the steady-state **PFR** is also such a useful reactor for kinetic studies—its model equations are quite simple! Whichever form of the equations used is simply a matter of preference; they all mean the very same thing.

9.6 Solution of the Steady-State PFR

First, we begin by solving for the concentrations with linear kinetics, and we do this in complete form. We will do this with **DSolve** as an exercise, even though the equations are trivial to solve:

```
In[75]:= Clear["Global`*"]
         Simplify[
         DSolve[
          {vz∂zCa[z] == -kabCa[z],
           vz∂zCb[z] == +kabCa[z],
           Ca[0] == Caf,
           Cb[0] == 0},
          {Ca[z], Cb[z]},
          z]
                 ]
```

Out[76]= $\{\{\text{Ca[z]} \to \text{Caf}\, e^{-\frac{\text{kab z}}{\text{vz}}}, \text{Cb[z]} \to \text{Caf} - \text{Caf}\, e^{-\frac{\text{kab z}}{\text{vz}}}\}\}$

Recall that **kab** is just $\frac{1}{\tau_{rxn}}$, that is, the reciprocal of the characteristic time for reaction and $\frac{z}{v_z}$ is the same as $\frac{V}{q}$ or the holding time $\boldsymbol{\tau}$ for the **PFR**. Therefore, these solutions become:

$$\mathrm{Ca}[z] = \mathrm{Caf}\, e^{-\frac{\mathrm{kab}\, z}{\mathrm{vz}}} \Rightarrow \mathrm{Ca}[\tau] = \mathrm{Caf}\, e^{-\mathrm{kab}\,\tau} \Rightarrow \mathrm{Ca}[\tau] = \mathrm{Caf}\, e^{-\frac{\tau}{\tau_{\mathrm{rxn}}}}$$
$$\mathrm{Cb}[z] = \mathrm{Caf}(1 - e^{-\frac{\mathrm{kab}\, z}{\mathrm{vz}}}) \Rightarrow \mathrm{Cb}[\tau] = \mathrm{Caf}(1 - e^{-\mathrm{kab}\,\tau}) \Rightarrow \mathrm{Cb}[\tau] = \mathrm{Caf}(1 - e^{-\frac{\tau}{\tau_{\mathrm{rxn}}}})$$

When we solved the transient, well-mixed batch reactor with linear kinetics, we obtained the same solution functionally, but instead of **kab** $\boldsymbol{\tau}$, we had **kab t** as the argument of the differential, that is, in terms of real time instead of holding time.

We return now to the Langmuir-Hinshelwood kinetics from the **CSTR** section to see how the **PFR** will behave and to compare the **CSTR** and the **PFR**. As in the case of the steady-state **CSTR**, we will write a steady-state **PFR Module** function. Recall that the rate law was:

$$r_{A-} = \frac{kK_AC_A}{1 + K_AC_A}$$

Therefore, once again invoking the pseudo-homogeneous approximation, the equations we must solve are:

$$v_z\frac{\mathrm{d}C_a}{\mathrm{d}z} = -\frac{(1-\epsilon)}{\epsilon}\frac{kK_AC_A}{1 + K_AC_A}$$
$$v_z\frac{\mathrm{d}C_b}{\mathrm{d}z} = +\frac{(1-\epsilon)}{\epsilon}\frac{kK_AC_A}{1 + K_AC_A}$$

The **Module** function for this **PFR** will be called "**pfr1.**" The basic backbone of the code was borrowed from **cstr4** with appropriate changes to the latter having been made. The arguments in "**pfr1**" are the rate constant **k**, the adsorption equilibrium constant **K1**, the volume flowrate **q**, and the radius of the reactor cross section **r**. So that we can make comparisons to the **CSTR**, we have kept the total volume the same at 1000. Once we specify the radius, that fixes the circular cross section. This divided into the volume gives the length of the reactor **zmax**. Therefore, instead of specifying the reactor length, we simply specify the reactor radius and at constant volume this fixes **zmax**. Everything else will be kept the same for comparison sake, especially ϵ and the holding time $\frac{V}{q}$.

```
In[77]:= pfr1[k_, K1_, q_, r_] :=
          Module[
           {Caf = 1, V = 1000, vz, Acr, Cao = 1, Cbo = 0, ϵ = .4,
            pfrsolns, Ca, Cb, CA, CB, zmax}
            Acr = N[πr²];
                   q
            vz = ———;
                  Acr
                     V
            zmax = ———;
                    Acr
```

```
   pfrsolns = NDSolve[{
```

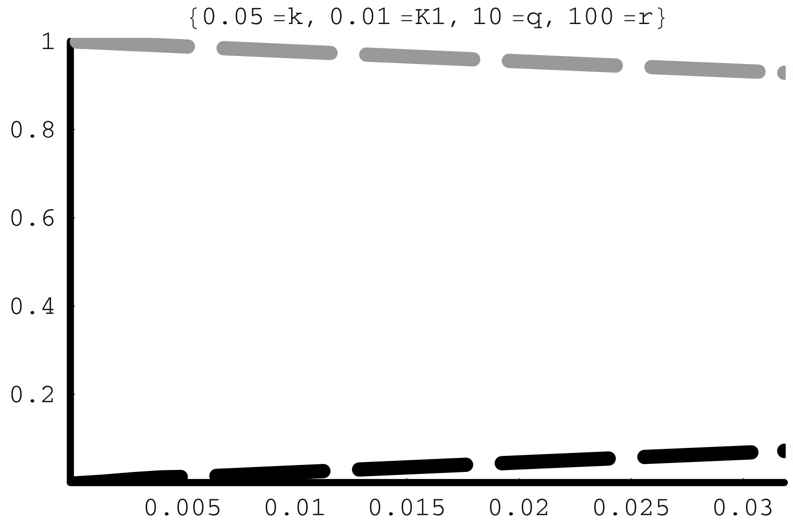
$$\mathrm{vz\,Ca}'[z] == -\frac{(1-\epsilon)}{\epsilon}\frac{\mathrm{k\,K1\,Ca}[z]}{1+\mathrm{K1\,Ca}[z]},$$

$$\mathrm{vz\,Cb}'[z] == \frac{(1-\epsilon)}{\epsilon}\frac{\mathrm{k\,K1\,Ca}[z]}{1+\mathrm{K1\,Ca}[z]},$$

```
      Ca[0] == Cao, Cb[0] == Cbo},
{Ca[z], Cb[z]}, {z, 0, zmax}];

  SetOptions[{Plot}, DefaultFont → {"Hevetica", 10}];

CA[z] = Evaluate[Ca[z] /. pfrsolns];
CB[z] = Evaluate[Cb[z] /. pfrsolns];

Plot[{CA[z], CB[z]}, {z, 0, zmax},
  PlotStyle → {{Dashing[{0.15, 0.05}], GrayLevel[0.6],
    Thickness[.02]},
  {Dashing[{0.15, 0.05}], GrayLevel[0], Thickness[.02]}},
  PlotRange → {{0, zmax}, {0, Caf}},
  PlotLabel → {k "=k", K1 "=K1", q "=q", r "=r"},
  DisplayFunction → Identity]
 ]
```

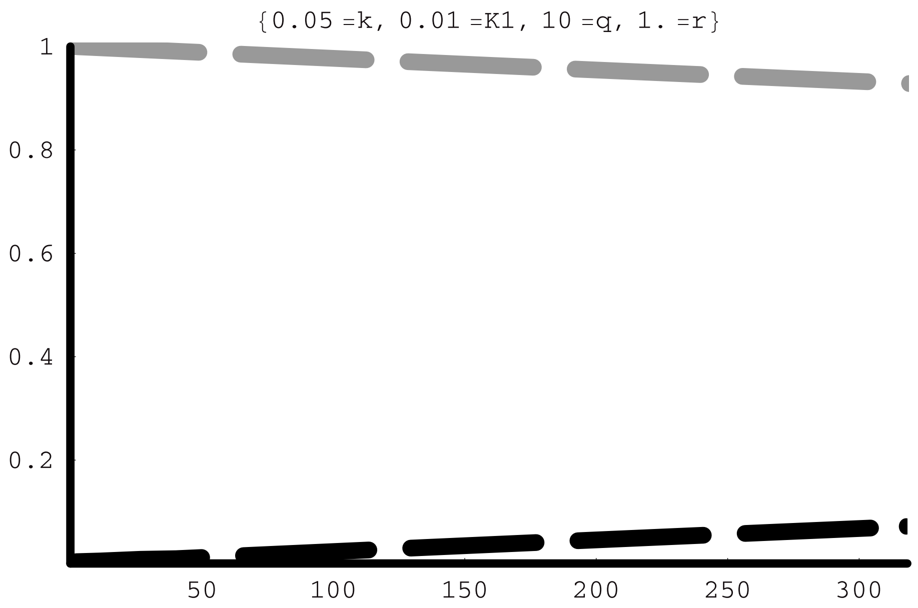
To see how the program runs we will try it out for a radius of 100 units. This makes **zmax** only 10 units. Such a reactor would be odd to find because it would have an aspect ratio of **L:D:**1:200. Run at very high volume flow rates, that is, high v_z, units of this kind are called *short contact time* reactors because the gases are within the reactor volume for so little time. In the present case that time is not short—it is on the order of $100 = \frac{1000}{10}$.

```
In[78]:= Show[pfr1[0.05, .01, 10, 100],
          DisplayFunction → $DisplayFunction];
```



We can see that with this particular flow rate the concentrations exiting at the end of the **PFR** correspond to approximately 10% conversion. Would this have changed if the radius were smaller, say, only 1? The answer is no. The reason is that the holding time will be the same because the volume and flow rate are the same.

```
In[79]:= Show[pfr1[0.05, .01, 10, 1.],
          DisplayFunction → $DisplayFunction];
```

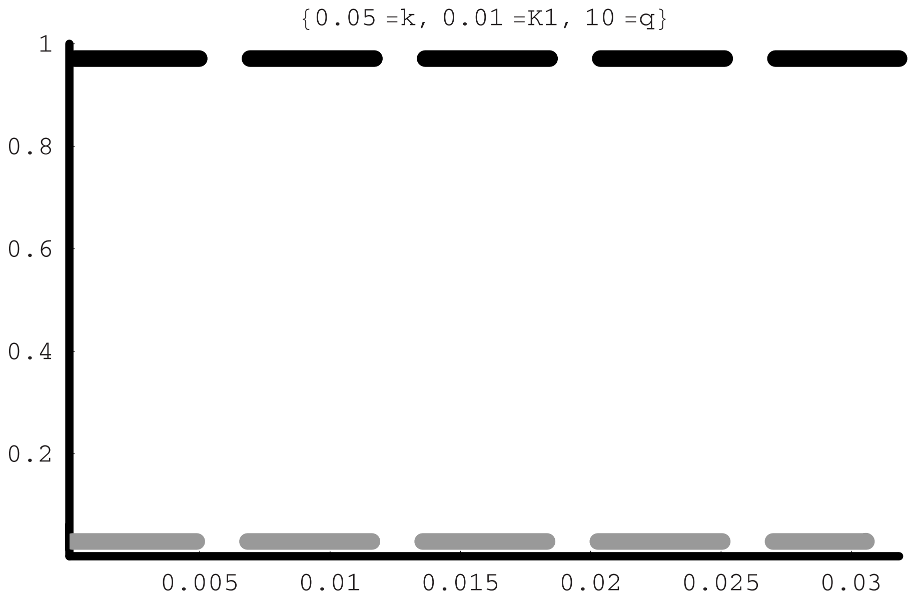


Now for comparison to the **CSTR**, we can build a new **Module** function that allows us to create a plot of the exit concentration from a **CSTR** under the same conditions and also as a "function" of **z**. Of course, there is no functional *z*-dependence for the **CSTR** as we shall see; the plots will be simply horizontal lines, but graphed over the same range as the axial distance through the **PFR**. In this way we can put the two on one graph for comparison. Here is the steady-state **CSTR Module**:

```
In[80]:= cstrstst[k_, K1_, q_, r_] :=
          Module[
           {Caf = 1, Acr, V = 1000, ϵ = .4, ststsolns,
            Ca, Cb, CA, CB, t},
           Acr = N[πr^2];
           zmax = V/Acr;
```

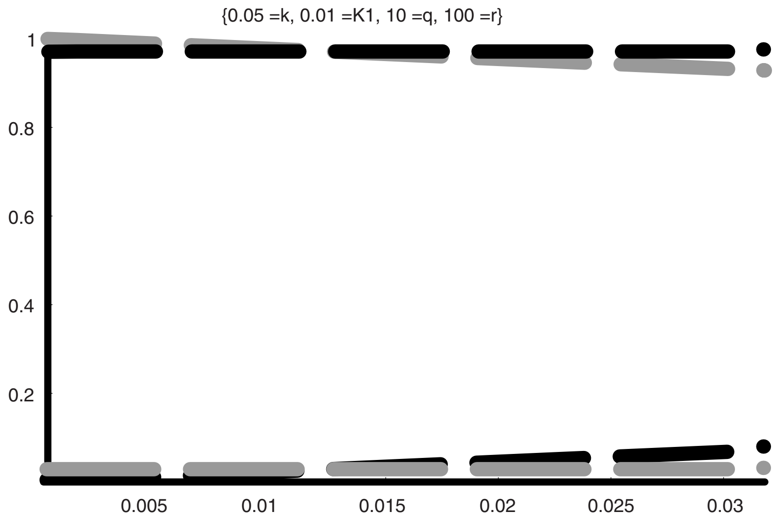
```
 ststsolns = Solve[
   {0 == (Caf - Ca) q/V - (1 - ε) (k K1 Ca)/(1 + K1 Ca),
    0 == -Cb q/V + (1 - ε) (k K1 Ca)/(1 + K1 Ca)},
 {Ca, Cb}];
   SetOptions[{Plot}, DefaultFont → {"Hevetica", 8}];

Ca = ststsolns[[1, 1, 2]];
  Cb = ststsolns[[1, 2, 2]];
  Plot[{Ca, Cb}, {t, 0, zmax},
     PlotStyle → {{Dashing[{0.15, 0.05}], GrayLevel[0.6],
       Thickness[.02]}, {Dashing[{0.15, 0.05}],
       GrayLevel[0], Thickness[.02]}},
    PlotRange → {{0, zmax}, {0, Caf}},
    PlotLabel → {k "=k", K1 "=K1", q "=q"},
    DisplayFunction → Identity]
]
```

```
In[81]:= Show[cstrstst[0.05, .01, 10, 100],
          DisplayFunction → $DisplayFunction];
```



When we put the **PFR** and **CSTR** results on the same graph we find:

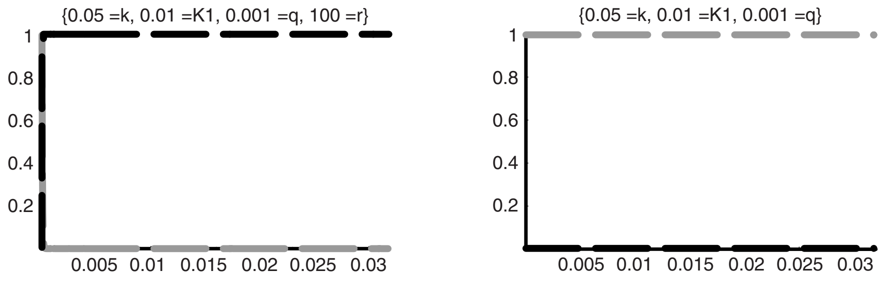
```
In[82]:= Show[{pfr1[0.05, .01, 10, 100],
          cstrstst[0.05, .01, 10, 100]},
           DisplayFunction → $DisplayFunction];
```

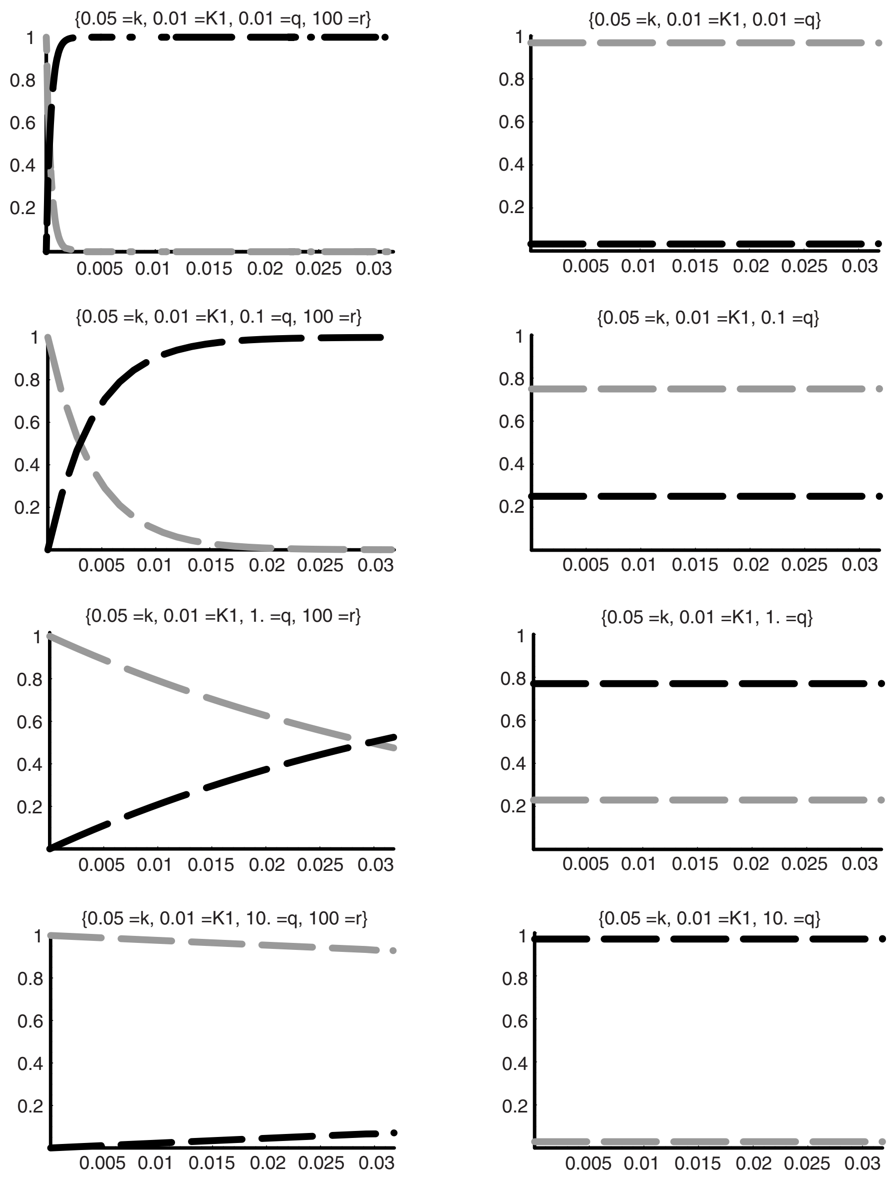


Although the conversions are relatively small, we see that conversion at the exit of the **PFR** is larger than that of the **CSTR** at the same condition by nearly a factor of two!

We can vary the flow rate over five orders of magnitude for both the **PRF** and the **CSTR** to see what will happen. This is done by combining the two functions into one **Table** and then plotting the results as a **GraphicsArray**.

```
In[83]:= Table[{pfr1[0.05, .01, 1. 10ⁿ, 100],
          cstrstst[0.05, .01, 1. 10ⁿ, 100]}, {n, -3, 1, 1}]
         Show[
           GraphicsArray[%]
         ];

Out[83]= {{-Graphics-, -Graphics-}, {-Graphics-, -Graphics-},
          {-Graphics-, -Graphics-}, {-Graphics-, -Graphics-},
          {-Graphics-, -Graphics-}}
```

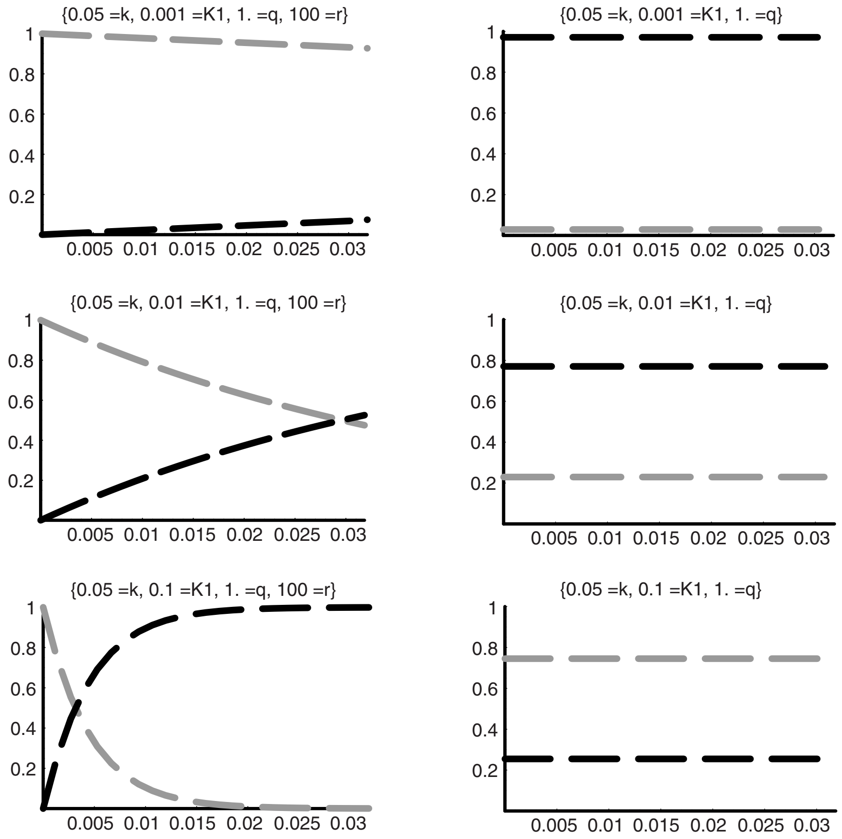


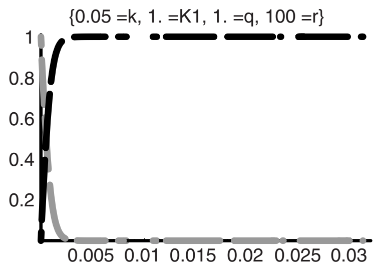


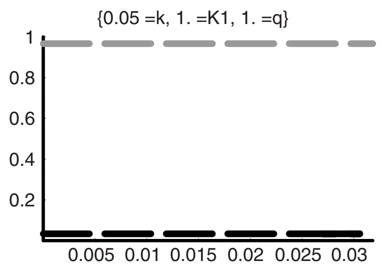
The left-most column of the graphs is for the **PFR**. Starting at the bottom and working vertically to the top of the column, we notice that the conversion rises from the low level we saw at **q** = 10 to nearly complete conversion at **q** = 0.1 and beyond. For the **CSTR** the trend is the same: as **q** drops the conversion rises, but notice that at **q** = 0.1 the conversion is ~75%, whereas at the same condition in the **PFR** it is complete, that is, 99.99%. In fact, even at **q** = 0.01, the **CSTR** has still not achieved full conversion of the feedstock. Remember: The only difference between the two cases is that the **CSTR** is well mixed throughout its volume, but the **PFR** is well mixed only radially, and not at all axially. This leads then to the oft-quoted rule-of-thumb that the **PFR** is more efficient than the **CSTR**. At the same volume of reactor one achieves higher conversion, or to achieve equivalent conversion the **PFR** volume can be smaller than the **CSTR** volume.

In the previous calculations, we assumed that **K1** was small. This forces the rate toward first-order dependence. Therefore, how does the comparison between **PFR** and **CSTR** work out if we vary **K1** over several orders of magnitude to make the kinetics range from first order to zeroth order? To do this we will fix the flow rate at $\mathbf{q} = 1$ and vary **K1** from 10^{-3} to 10^{3} as follows:

```
In[85]:= comps2 =
           Table[{pfr1[0.05, 1. 10^n, 1., 100],
            cstrstst[0.05, 1. 10^n, 1., 100]}, {n, -3, -0, 1}];
         Show[GraphicsArray[comps2]];
```







The results indicate that even if the adsorption equilibrium constant is large, the **PFR** shows better results than the **CSTR**. The only time this is violated for an isothermal reaction system is if the kinetics are of negative order overall, because then the **CSTR** will actually be more efficient than the **PFR**; otherwise the **PFR** wins.

9.7 Mixing Effects on Selectivities—Series and Series-Parallel with CSTR and PFR

In the previous chapter, we examined series and series-parallel kinetics. The extent of mixing can have an effect on the selectivity to the various products in such reaction networks. The selectivity is the percentage of the products that are any one of the products. To compute the yield we take the product of the conversion and the selectivity. Thus the yield is a fraction of a fraction.

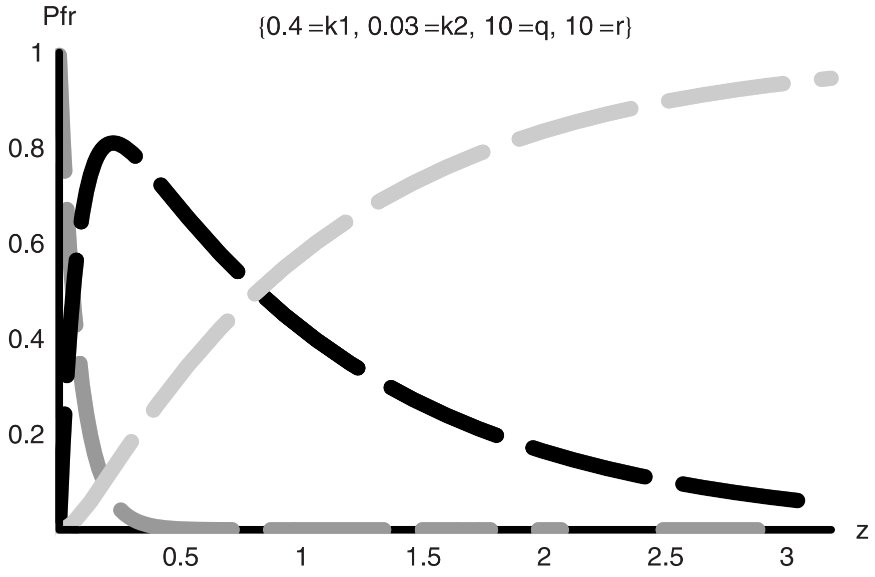
We can begin by computing the selectivities and yields for the series network in the **CSTR** versus the **PFR** first. Consider the simplest series reaction network:

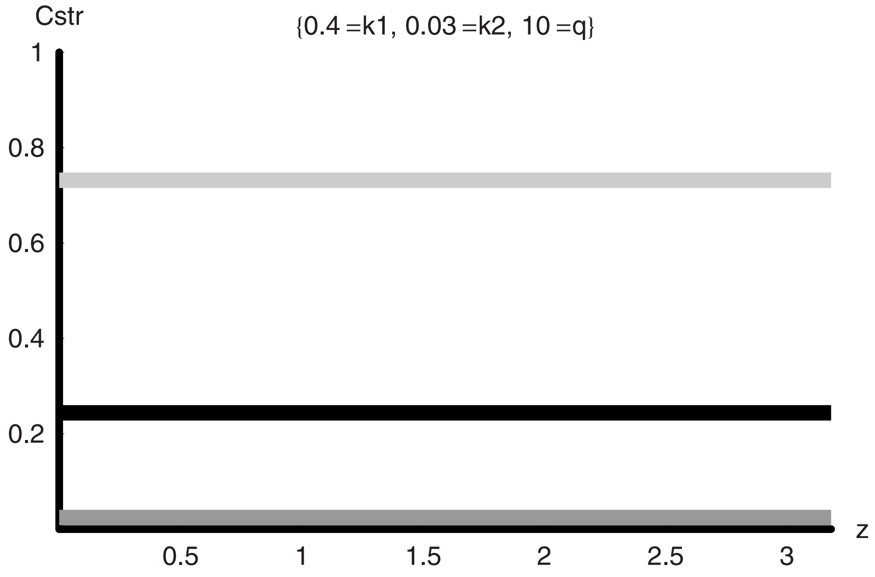
$$A \rightarrow B \rightarrow D$$
$$r_{\mathrm{ab}} = k_1 C_A; \quad r_{\mathrm{bd}} = k_2 C_B$$

The **Module** functions for the **CSTR** and **PFR** with these species and kinetics are written in what follows as "**cstrABD**" and "**pfrABD**."

```
In[87]:= Clear["Global`*"]

In[88]:= cstrABD[k1_, k2_, q_, r_]:=
          Module[
           {Caf = 1, Acr, V = 1000, ststsolns, Ca, Cb, Cd},
           Acr = N[πr²];
                   V
           zmax = ───;
                  Acr
           ststsolns = Solve[
                                 q
             {0 == (Caf - Ca)── - k1 Ca,
                                 V
                       q
              0 == -Cb── + k1 Ca - k2 Cb,
                       V
              0 == -Cd q/V + k2 Cb},
                  {Ca, Cb, Cd}];
             CA = ststsolns[[1, 1, 2]];
             CB = ststsolns[[1, 2, 2]];
             CD = ststsolns[[1, 3, 2]];
        SetOptions[Plot, DefaultFont → {"Helvetica", 10}];
          Plot[{CA, CB, CD}, {t, 0, zmax},
           PlotStyle → {{GrayLevel[0.6], Thickness[0.02]},
             {GrayLevel[.0], Thickness[0.02]}, {GrayLevel[0.8],
              Thickness[0.02]}},
           AxesLabel → {z, Cstr},
           PlotRange → {{0, zmax}, {0, Caf}},
           PlotLabel → {k1 "=k1", k2 "=k2", q "=q"},
           DisplayFunction → Identity]]
In[89]:= pfrABD[k1_, k2_, q_, r_] :=
         Module[
          {Caf = 1, V = 1000, vz, Acr, Cao = 1, Cbo = 0, Cdo = 0,
           ε = .4, pfrsolns, Ca, Cb, Cd, CA, CB, CD, zmax},
          Acr = N[πr^2];
          vz = q/Acr;
          zmax = V/Acr;
          pfrsolns = NDSolve[{
          vz Ca'[z] == -k1 Ca[z],
          vz Cb'[z] == +k1 Ca[z] - k2 Cb[z],
               vz Cd'[z] == +k2 Cb[z],
               Ca[0] == Cao, Cb[0] == Cbo, Cd[0] == Cdo},
          {Ca[z], Cb[z], Cd[z]}, {z, 0, zmax}];
          CA[z] = Evaluate[Ca[z] /. pfrsolns];
          CB[z] = Evaluate[Cb[z] /. pfrsolns];
            CD[z] = Evaluate[Cd[z] /. pfrsolns];
            SetOptions[Plot, DefaultFont → {"Helvetica", 10}];
          Plot[{CA[z], CB[z], CD[z]}, {z, 0, zmax},
            PlotStyle → {{Dashing[{0.15, 0.05}], GrayLevel[0.6],
             Thickness[0.02]},
            {Dashing[{0.15, 0.05}],
             GrayLevel[0], Thickness[0.02]},
            {Dashing[{0.15, 0.05}], GrayLevel[.8],
              Thickness[0.02]}},
             AxesLabel → {z, Pfr},
             PlotRange → {{0, zmax}, {0, Caf}},
             PlotLabel → {k1"=k1", k2"=k2", q"=q", r"=r"},
             DisplayFunction → Identity]]
```

```
In[90]:= Show[
           GraphicsArray[
            {{pfrABD[.4, .03, 10, 10]},
             {cstrABD[.4, .03, 10, 10]}}],
           DisplayFunction → $DisplayFunction];
```





For both reactors the volumes are identical, and that is critical to this comparison. If **B** (black dashed lines, upper part of the graph, and black solid, lower) were the species that we wished to produce, then for the same volume the **CSTR** with its perfect mixing produces a higher yield than does the **PFR** at the same volume and flow conditions. This is a consequence of the higher efficiency of the **PFR**. Notice, however, that if we made the **PFR** smaller by cutting it off at **z** = 0.3 for volume of $\frac{1000}{10} = 100$, then it would be better for the production of **B** than the **CSTR** with a volume of 1000 for the production of **B**.

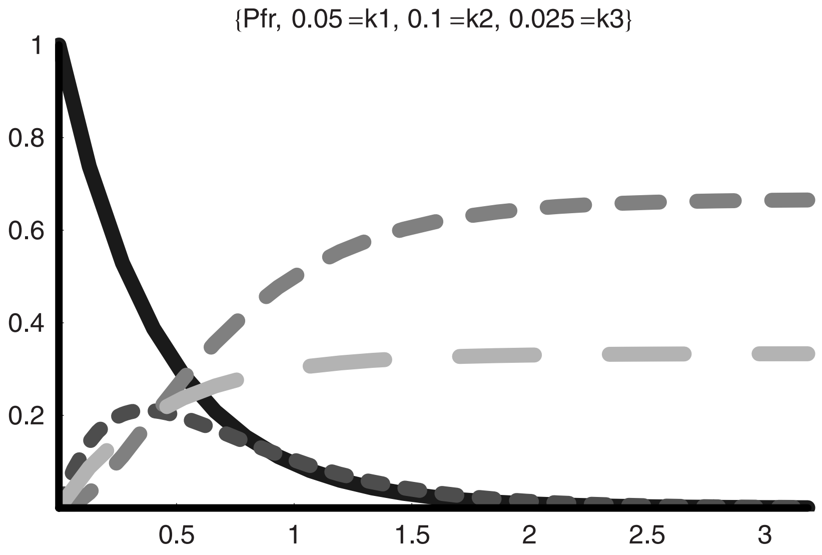
Next, we rewrite the two codes to handle another limiting case, that of parallel reactions of **A** to form either **B** or **D**:

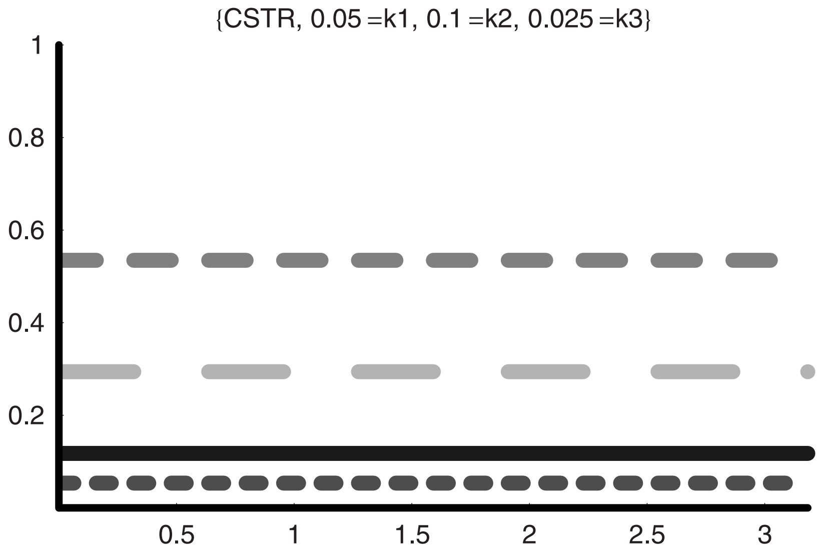
$$\begin{tikzcd} \mathrm{A} \arrow[r] \arrow[d] & \mathrm{B} \arrow[r] & \mathrm{D} \quad r_{\mathrm{ab}} = k_1 C_A; \quad r_{\mathrm{ad}} = k_2 C_B \\ \mathrm{E} \quad r_{\mathrm{ae}} = k_3 C_A \end{tikzcd}$$

We will take the magnitudes of the two rate constants to be the same as in the previous example, which makes the rate of formation of **B** a factor of two greater than the rate of formation of **D**. The codes are called "**cstrABAD**" and "**pfrABAD**."

```
In[91]:= Clear["Global`*"]

In[92]:= cstrABAD[k1_, k2_, k3_, q_, r_]:=
          Module[
           {Caf = 1, Acr, V = 1000, ststsolns, Ca, Cb, Cd, Ce,
            CA, CB, CD, CE},
           Acr = N[πr^2];
                    V
            zmax = ───;
                   Acr
            ststsolns = Solve[
                                q
              {0 == (Caf - Ca)─ - k1Ca - k3Ca,
                                V
                        q
               0 == -Cb─ + k1Ca - k2Cb,
                        V
                        q                      q
               0 == -Cd─ + k2Cb,0 == -Ce─ + k3Ca},{Ca,Cb,Cd,Ce}];
                        V                      V
               Ca = ststsolns[[1, 1, 2]];
               Cb = ststsolns[[1, 2, 2]];
            Cd = ststsolns[[1, 3, 2]];
            Ce = ststsolns[[1, 4, 2]];
            SetOptions[Plot, DefaultFont → {"Helvetica", 10}];
            Plot[{Ca, Cb, Cd, Ce}, {t, 0, zmax},
             PlotStyle → {
              {Thickness[0.02], Dashing[{0, 0}], GrayLevel[0.1]},
              {Thickness[0.02], Dashing[{0.02, 0.03}], GrayLevel[0.3]},
              {Thickness[0.02], Dashing[{0.05, 0.05}], GrayLevel[.5]},
              {Thickness[0.02], Dashing[{0.1, 0.1}], GrayLevel[.7]}
              },
      PlotRange → {{0, zmax}, {0, Caf}},
      PlotLabel → {CSTR, k1 "=k1", k2 "=k2", k3 "=k3"},
      DisplayFunction → Identity]
     ]
   General::spell1: Possible spelling error: new symbol
    name "cstrABAD" is similar to existing symbol "cstrABD".
In[93]:= pfrABAD[k1_, k2_, k3_, q_, r_]:=
    Module[
     {Caf = 1, V = 1000, vz, Acr, Cao = 1, Cbo = 0,
      Cdo = 0, Ceo = 0, ε = .4, pfrsolns, Ca, Cb, Cd, CA,
      CB, CD, zmax},
     Acr = N[πr^2];
     vz = q/Acr;
     zmax = V/Acr;
     pfrsolns = NDSolve[{
    vzCa'[z] == -k1Ca[z] - k3Ca[z],
    vzCb'[z] == +k1Ca[z] - k2Cb[z],
          vzCd'[z] == + k2Cb[z],
          vzCe'[z] == + k3Ca[z],
          Ca[0] == Cao, Cb[0] == Cbo, Cd[0] == Cdo,
           Ce[0] == Ceo},
    {Ca[z], Cb[z], Cd[z], Ce[z]}, {z, 0, zmax}];
    CA[z] = Evaluate[Ca[z] /. pfrsolns];
    CB[z] = Evaluate[Cb[z] /. pfrsolns];
      CD[z] = Evaluate[Cd[z] /. pfrsolns];
      CE[z] = Evaluate[Ce[z] /. pfrsolns];
      SetOptions[Plot, DefaultFont → {"Helvetica", 10}];
    Plot[{CA[z], CB[z], CD[z], CE[z]}, {z, 0, zmax},
    PlotStyle → {
     {Thickness[0.02], Dashing[{0, 0}], GrayLevel[0.1]},
     {Thickness[0.02], Dashing[{0.02, 0.03}], GrayLevel[0.3]},
     {Thickness[0.02], Dashing[{0.05, 0.05}], GrayLevel[.5]},
     {Thickness[0.02], Dashing[{0.1, 0.1}], GrayLevel[.7]}
    },
     PlotRange → {{0, zmax}, {0, Caf}},
     PlotLabel → {Pfr, k1 "=k1", k2 "=k2", k3 "=k3"},
     DisplayFunction → Identity]
     ]

   General::spell1: Possible spelling error: new symbol
   name "pfrABAD" is similar to existing symbol "pfrABD".
```

```
In[94]:= Show[
           GraphicsArray[
             {{pfrABAD[.05, .1, .025, 10, 10]},
              {cstrABAD[.05, .1, .025, 10, 10]}}
                          ],
           DisplayFunction  → $DisplayFunction];
```





A = Blk Sld;   B = Blk DSh;   D = Lt Gry Dsh;   E = Dk Gry Dsh.

We see once again that the overall conversion is higher in the **PFR** than in the **CSTR** and that the fractions of **D** and **E** in the products are therefore larger than in the **CSTR**. The efficiency of the **PFR** with positive order kinetics versus the perfectly mixed **CSTR** raises an interesting question—if one were to divide the volume of one **CSTR** into two **CSTR**s in series of equal volume, would there be any change in efficiency? What if there were three or even more **CSTR**s in series with equal volumes that all summed up to that of the original one—would there be a significant difference? We address this in the next section.

9.8 PFR as a Series of CSTRs

The main difference between the **PFR** and **CSTR** idealizations is the mathematical one of a spatially distributed versus homogeneous system, which leads to quite different equations. If one were to take the globally homogeneous **CSTR** and break it up into smaller homogeneous regions, then in the limit of an infinite number of these taken in series they would become equivalent to one **PFR** of equal total volume. We can see this by comparing one, two, three, and more **CSTR**s in series with one **PFR**. As the number of **CSTR**s increases the results will approach the **PFR** (see Figure 6). The simplest way to do this is to write bit of code that allows us to specify the total volume and then to vary the number of contiguous homogeneous cells that are present. We can begin by taking a look at the case of **A** $\rightarrow$ **B** with linear, irreversible kinetics, because this is simple. We also understand it well because we can solve it exactly.

The steady-state solutions for one CSTR are shown here:

```
In[95]:= ststsolns = Solve[
           {0 == (Caf - Ca) q/V - k1 Ca,
            0 ==-Cb q/V + k1 Ca},
                 {Ca, Cb}]

Out[95]= {{Cb → Caf k1 V/(q + k1 V), Ca → Caf q/(q + k1 V)}}
```

We can generalize this for any nth **CSTR** in a series as follows:

```
In[96]:= Clear[ca, cb, q, Caf, ntot, Vtot, k, n]
         Solve[
          {0 == (Ca[n - 1] - Ca[n]) q/V - k Ca[n],
           0 == (Cb[n - 1] - Cb[n]) q/V + k Ca[n]},
          {Ca[n], Cb[n]}
                 ]
Out[97]={{Cb[n] → k V Ca[-1 + n]/(q + k V) + Cb[-1 + n], Ca[n] → q Ca[-1 + n]/(q + k V)}}
```

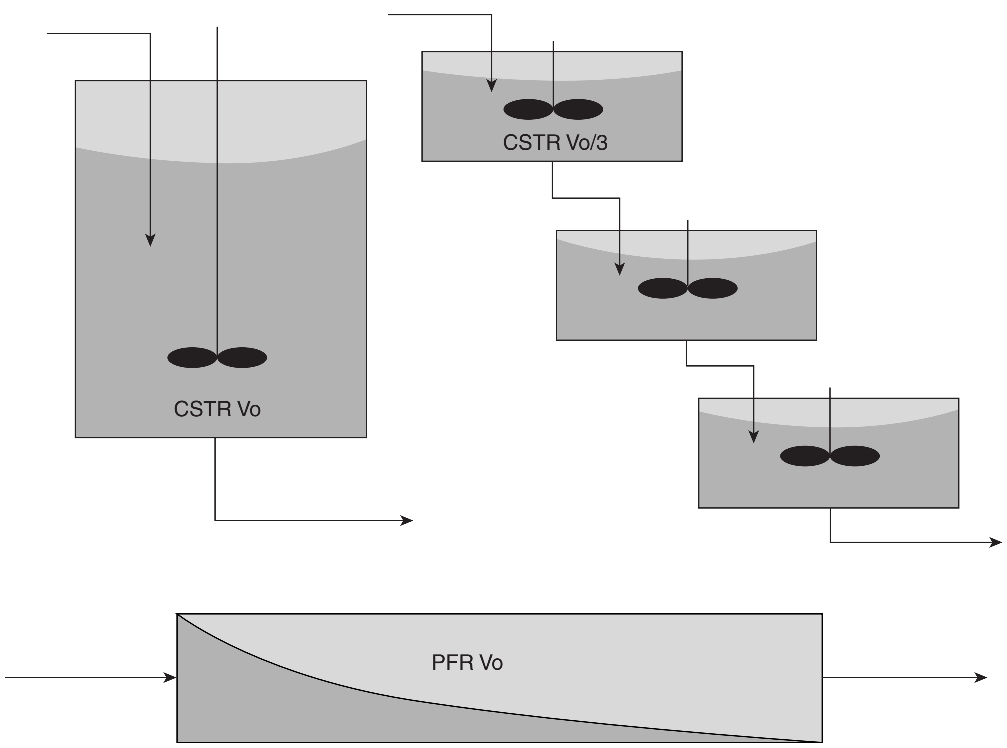


Figure 6

This is a recursion formula for the exact case. We would like to be able to apply this to any number **n** of **CSTR**s in series and find an analytical and then quantitative result for comparison to the exact **PFR** result. To do this we need recursive programming. There are three programming styles in *Mathematica*: **Rule-Based, Functional**, and **Procedural**. We will attack this problem in recursion with **Rule-Based, Functional**, and **Procedural** programming. We can begin by looking at the *rule-based* recursion codes for **Ca** and **Cb** in any **n CSTR**s.

The "seeds" for these rules are the solutions for the first **CSTR**, and then these exit concentrations become the inlet concentrations to the second **CSTR**, whose exit concentrations become the inlet concentrations for the third **CSTR**, and so it goes on through to **n CSTR**s. We can have *Mathematica* assemble the equations and the variables that we will need for the **Solve** routine. This is illustrated with $n = 3$:

```
In[98]:= n = 3
         Clear[q, V, Vo, k, Caf, Cbf, Cdf, Ca]
                                           q
         Table[{(Ca[i - 1] - Ca[i]) ─ - kCa[i] == 0,
                                           V
                                           q
                    (Cb[i - 1]-Cb[i]) ─ + kCa[i] == 0},
                                           V
               {i, 1, n}]
         vars = Table[{Ca[i], Cb[i]}, {i, 1, n}]
```

```
Out[98]=  3
          General::spell1 : Possible spelling error: new symbol
          name "Cdf" is similar to existing symbol "CDF".
```

Out[100]= $\{\{\frac{q(Ca[0]-Ca[1])}{V} - k\,Ca[1] == 0, k\,Ca[1] + \frac{q(Cb[0]-Cb[1])}{V} == 0\},$
$\{\frac{q(Ca[1]-Ca[2])}{V} - k\,Ca[2] == 0, k\,Ca[2] + \frac{q(Cb[1]-Cb[2])}{V} == 0\},$
$\{\frac{q(Ca[2]-Ca[3])}{V} - k\,Ca[3] == 0, k\,Ca[3] + \frac{q(Cb[2]-Cb[3])}{V} == 0\}\}$

```
Out[101]= {{Ca[1], Cb[1]}, {Ca[2], Cb[2]}, {Ca[3], Cb[3]}}
```

Now we place these into **Solve** as follows:

```
In[102]:= Clear[q, V, Vo, k, Caf, Cbf, Cdf, n, Ca]
          n = 4;
          V = Vo/n;
          Timing[eqns = Table[{(Ca[i-1]-Ca[i]) q/V - k Ca[i] == 0,
                       (Cb[i - 1] - Cb[i]) q/V + k Ca[i] == 0},
             {i, 1, n}];
          vars = Table[{Ca[i], Cb[i]}, {i, 1, n}];
          Ca[0] = Caf;
           Cb[0] = 0;
          solns =
            Flatten[Solve[Flatten[eqns], Flatten[vars]]]]
```

Out[105]= {0.22 Second, $\{Cb[4] \to -\frac{-256\,Caf\,k\,q^3\,Vo - 96\,Caf\,k^2 q^2\,Vo^2 - 16\,Caf\,k^3 q\,Vo^3 - Caf\,k^4\,Vo^4}{(4q + k\,Vo)^4},$

$Cb[2] \to -\frac{-8\,Caf\,k\,q\,Vo - Caf\,k^2\,Vo^2}{(4q + k\,Vo)^2},$

$Cb[3] \to -\frac{-48\,Caf\,k\,q^2 Vo - 12\,Caf\,k^2 q\,Vo^2 - Caf\,k^3 Vo^3}{(4q + k\,Vo)^3},$

$Cb[1] \to \frac{Caf\,k\,Vo}{4q + k\,Vo},\ Ca[4] \to \frac{256\,Caf\,q^4}{(4q + k\,Vo)^4},$

$Ca[3] \to \frac{64\,Caf\,q^3}{(4q + k\,Vo)^3},\ Ca[2] \to \frac{16\,Caf\,q^2}{(4q + k\,Vo)^2},$

$Ca[1] \to \frac{4\,Caf\,q}{4q + k\,Vo}\}\}$

We see that in this program the equations are first written explicitly from **n** = 1 to **n**, their output is suppressed, but then they are solved symbolically. We have enclosed the overall functions in "**Timing**" in order to obtain a report of the **CPU** time required to conduct this. By supplying the necessary parameters and changing **Solve** to **NSolve**, we can find a solution for the outlet of **n-CSTR**s. We make this into a **Module** function of **n**, the number of **CSTR**s:

```
In[106]:= << Miscellaneous`RealOnly`

In[107]:= Clear[Cstr, cstr, q, V, Vo, k, Caf, Cbf, Cdf, n]
          Clear["Global`*"]

          General::spell1 : Possible spelling error: new symbol
           name "cstr" is similar to existing symbol "Cstr".

In[109]:= cstr[n_] :=
           Module[
            {q = 10, Vo = 100, k = 0.1, eqns, vars, solns, i},
            V = Vo/n;
          eqns = Table[{(Ca[i - 1] - Ca[i]) q/V - k Ca[i] == 0,
                        (Cb[i - 1] - Cb[i]) q/V + k Ca[i] == 0},
               {i, 1, n}];
          vars = Table[{Ca[i], Cb[i]}, {i, 1, n}];
            Ca[0] = 1;
          Cb[0] = 0;
          solns = NSolve[Flatten[eqns], Flatten[vars]][[1]];
            {solns[[2 n - 1]],
            solns[[2 n]]}]
```

Now we can try out this module program with 1000 CSTRs in series and check its timing:

```
In[110]:= cstr[1000] // Timing

Out[110]= {14.22 Second, {Ca[1000] → 0.368063, Cb[1000] → 0.631937}}
```

We have called the package "**Miscellaneous'RealOnly**" to avoid the complex solutions that might otherwise be returned. We have also taken just the solutions from the last **CSTR** by using **{solns[[2n − 1]], solns[[2n]]}**. Now we can use the "listability" of this function **cstr[n]** and **Map** it down a vector of values for *n*. The *infix* form for **Map** is /@ and so we use it as follows:

```
In[111]:= n = {1, 2, 3, 10, 20, 100, 1000};
          cstr /@n // Timing

Out[112]= {16.26 Second, {{Ca[1] → 0.5, Cb[1] → 0.5},
           {Ca[2] → 0.444444, Cb[2] → 0.555556},
           {Ca[3] → 0.421875, Cb[3] → 0.578125},
           {Ca[10] → 0.385543, Cb[10] → 0.614457},
           {Ca[20] → 0.376889, Cb[20] → 0.623111},
{Ca[100] → 0.369711, Cb[100] → 0.630289},
{Ca[1000] → 0.368063, Cb[1000] → 0.631937}}}
```

This computation took ~47.75 sec of **CPU** time on an old Pentium I processor but just 2.1 sec on a newer 1-GHz Pentium chip. We added //**Timing** after the command line also in *infix* form, in order to obtain this information. The time will vary with different machines and processors. More important, what we notice is that there is a large change in **Ca** and **Cb** when we go from one **CSTR** to 2 or 3 or even to 10, but when we get beyond 10 to 20, 100, and even 1000, the increase in the number of **CSTR**s gives diminishing returns.

In the following code we solve the equations for the **PFR** at the same conditions and with the same parameter values, especially that of the volume **Vo**.

```
In[113]:= Clear["Global`*"]

In[114]:= Caf = 1;
          Vo = 100;
          q = 10;
          k1 = .1;
          zmax = 10;
                Vo
          A = ─────;
               zmax
                q
          vz = ─;
                A

          pfr = NDSolve[
               {vzCa'[z] == -k1Ca[z],
                vzCb'[z] == +k1Ca[z],
                Ca[0] == Caf, Cb[0] == 0},
                {Ca[z], Cb[z]},
               {z, 0, zmax}];

          CA[z_] := Evaluate[Ca[z] /. pfr];
          CB[z_] := Evaluate[Cb[z] /. pfr];

          {CA[zmax], CB[zmax]}// Timing

          General::spell1: Possible spelling error: new symbol
          name "pfr" is similar to existing symbol "Pfr".

Out[124]= {0. Second, {{0.367879}, {0.632121}}}
```

Here we see that the limiting values of **Ca** and **Cb** are 0.37 and 0.63, respectively, exit concentrations which were nearly identical to those obtained with the large number of **CSTR**s (>100) and closely approached by even 10 **CSTR**s. The important point we learn from this is that even with two or three **CSTR**s we begin to move toward the **PFR** limit.

Now we will return to the recursive programming part of this problem because it is a prototypical type problem we can expect to encounter often in chemical engineering analysis and computations. We can now try *functional* programming to derive the solutions that we seek. We know the solution for **Ca** at the exit of the first **CSTR** and we have a recursive relationship for the exit concentrations emerging from the next **n-CSTR**s. Therefore, we can put these two together into a functional program as follows:

```
In[125]:= Clear[ca, V, q, Caf, Vo]
```

$$\texttt{ca[1]} = \frac{q\,Caf}{q + k\,V};$$

$$\texttt{ca[n\_]} := \texttt{ca[n]} = \frac{q\,\texttt{ca[n - 1]}}{q + k\,V}$$

We can see how this works. We defined the seed for **ca[1]** first and then we created the function for any **ca[n]** as follows:

$$ca[n\_] := ca[n] = \frac{q\,ca[n-1]}{q+k\,V}$$

Taking $n = 4$ we find:

```
In[128]:= ca[4]
```

$$\texttt{Out[128]=}\quad \frac{Caf\,q^4}{(q+k\,V)^4}$$

Recall that in this recursion relation the symbol $\mathbf{V} = \frac{\mathbf{Vo}}{\mathbf{4}}$; therefore, we can redo the computation to derive:

$$\texttt{In[129]:=}\quad V = \frac{Vo}{4};$$

```
          ca[4]
          Together[%]
```

$$\texttt{Out[130]=}\quad \frac{Caf\,q^4}{\left(q+\frac{k\,Vo}{4}\right)^4}$$

$$\texttt{Out[131]=}\quad \frac{256\,Caf\,q^4}{(4\,q+k\,Vo)^4}$$

Referring back to the earlier solution that we derived for $n = 4$, we see that the two agree perfectly. The function for **ca[n]** may be used in the **Table** function to derive the first four solutions symbolically:

```
In[132]:= n = 4;
          Table[ca[x], {x, 1, n}]
          Together[%]
```

Out[133]= $\{\frac{\text{Caf } q}{q+\frac{k\,Vo}{4}}, \frac{\text{Caf } q^2}{(q+\frac{k\,Vo}{4})^2}, \frac{\text{Caf } q^3}{(q+\frac{k\,Vo}{4})^3}, \frac{\text{Caf } q^4}{(q+\frac{k\,Vo}{4})^4}\}$

Out[134]= $\{\frac{4\,\text{Caf } q}{4\,q+k\,Vo}, \frac{16\,\text{Caf } q^2}{(4q+k\,Vo)^2}, \frac{64\,\text{Caf } q^3}{(4q+k\,Vo)^3}, \frac{256\,\text{Caf } q^4}{(4q+k\,Vo)^4}\}$

This is a simple and yet very powerful use of *Mathematica*'s set-delay utility : = . The right-hand side is the whole recursive equation with an equal sign, but by placing this after the set-delay : = it becomes a pattern that will not be evaluated until **n** is specified. When **n** is specified, then it begins with the seed and evaluates the function until it gets to the value of **n**. Thus, when we set **n** = 4, we obtain the concentration expression for the exit of the fourth **CSTR**. If we place the function in a **Table**, and set **n** = 4 we obtain the exit concentrations for all four of the **CSTR**s. We follow the same procedure for **cb[n]** as shown here:

```
In[135]:= Clear[cb]
```

$$V = \frac{Vo}{s};$$

$$cb[1] = \frac{\text{Caf } kV}{q+kV};$$

$$cb[n\_] := cb[n] = \frac{k\,V\,ca[n-1]}{q+kV} + cb[n-1]$$

```
In[139]:= s = 4
          cb[s]
          Together[%]
Out[139]= 4
```

Out[140]= $\frac{\text{Caf } k\, q^3\, Vo}{4(q+\frac{k\,Vo}{4})^4} + \frac{\text{Caf } k\, q^2\, Vo}{4(q+\frac{k\,Vo}{4})^3} + \frac{\text{Caf } k\, q\, Vo}{4(q+\frac{k\,Vo}{4})^2} + \frac{\text{Caf } k\, Vo}{4(q+\frac{k\,Vo}{4})}$

Out[141]= $\frac{256\,\text{Caf } k\, q^3\, Vo + 96\,\text{Caf } k^2 q^2\, Vo^2 + 16\,\text{Caf } k^3 q\, Vo^3 + \text{Caf } k^4\, Vo^4}{(4\,q+k\,Vo)^4}$

Again, the solution for **Cb** at the exit of **CSTR** number 4 matches that previously derived. Note also that this solution for **B** was explicit in **n** because we had already defined **ca[n]**. If we had not done this, then what follows is what we would have seen:

```
In[142]:= Clear["Global`*"]
```

In[143]:= $V = \frac{Vo}{s};$

$$cb[1] = \frac{\text{Caf } k\,V}{q+k\,V};$$

$$cb[n\_] := cb[n] = \frac{k\,V\,ca[n-1]}{q+k\,V} + cb[n-1]$$

```
          s = 4
          cb[s]
          Together[%]
```

Out[146]= 4

$$\text{Out[147]=}\quad \frac{\text{Caf k Vo}}{4(\text{q}+\frac{\text{k Vo}}{4})}+\frac{\text{k Vo ca[1]}}{4(\text{q}+\frac{\text{k Vo}}{4})}+\frac{\text{k Vo ca[2]}}{4(\text{q}+\frac{\text{k Vo}}{4})}+\frac{\text{k Vo ca[3]}}{4(\text{q}+\frac{\text{k Vo}}{4})}$$

$$\text{Out[148]=}\quad \frac{\text{Caf k Vo + k Vo ca[1] + k Vo ca[2] + k Vo ca[3]}}{\text{4 q + k Vo}}$$

To avoid any such problems we simply place the two sets of functions together into one working cell as follows:

```
In[149]:= Clear["Global`*"]

In[150]:= V = Vo/s;
          ca[1] = q Caf/(q + k V);
          ca[n_] := ca[n] = q ca[n - 1]/(q + k V)
          cb[1] = Caf k V/(q + k V);
          cb[n_] := cb[n] = k V ca[n - 1]/(q + k V) + cb[n - 1]
```

Testing the result we obtain:

```
In[155]:= s = 4;
          {Together[ca[s]],Together[cb[s]]}
```

$$\text{Out[156]=}\ \left\{\frac{256\,\text{Caf}\,\text{q}^4}{(4\,\text{q}+\text{k}\,\text{Vo})^4},\frac{256\,\text{Caf}\,\text{k}\,\text{q}^3\,\text{Vo}+96\,\text{Caf}\,\text{k}^2\text{q}^2\,\text{Vo}^2+16\,\text{Caf}\,\text{k}^3\text{q}\,\text{Vo}^3+\text{Caf}\,\text{k}^4\text{Vo}^4}{(4\,\text{q}+\text{k}\,\text{Vo})^4}\right\}$$

We can apply values to the parameters and the values **n**. We use **s** to define the number of **CSTR**s rather than **n** per se in order to avoid a name clash that leads to an infinite loop. Before we try this, however, we shall set the **$RecursionLimit** to 1000. The default value of 256 is not sufficient for us to go out to numbers as large as **n** = 1000 **CSTR**s.

```
In[157]:= Clear["Global`*"]

In[158]:= $RecursionLimit = 2000;
          V = Vo/s;
          ca[1] = q Caf/(q + k V);
          ca[n_] := ca[n] = q ca[n - 1]/(q + k V)
          cb[1] = Caf k V/(q + k V);
          cb[n_] := cb[n] = k V ca[n - 1]/(q + k V) + cb[n - 1]
```

```
              Caf = 1;
              k = .1;
              q = 10;
              Vo = 100;
              s = 1000
              {ca[s], cb[s]} // Timing

Out[168]= 1000

Out[169]= {1.49 Second, {0.368063, 0.631937}}
```

We can see that this is a huge speed-up from the original code that we wrote in which we first rewrote all the equations and then did the computations. This new program required only 2.91 sec on a Pentium I to do the same and just 0.17 sec with the 1-GHz Pentium III processor.

We can also solve this recursion problem more traditionally using a *procedural* approach. The recursion in this case is buried within a "Do" loop, which is the classic structure in the procedural programming paradigm. The "Do" loop is placed within a Module function to keep all the variable names localized:

```
In[170]:= ca1[n_] :=
           Module[{m, ca},
             V = $\frac{Vo}{n}$;
             Do[ca[1] = $\frac{q\,Caf}{q+k\,V}$; ca[m] = $\frac{q\,ca[m-1]}{q+k\,V}$,{m,0,n}];
             ca[n]
           ]

In[171]:= ca1[4]
           Together[%]

Out[171]=  0.4096

Out[172]=  0.4096
```

Within the **Do** loop we have the same recursion that we had implemented by rules—we specify the seed as **ca[1]** and then the recursion relation as **ca[m]**. The set-delayed function argument of **ca1[n_]** supplies **n**, as the limit of the iterative sequence. The last statement just outside the **Do** loop writes the value of **ca[n]** and then it stops. We implement the same approach for species **B**:

```
In[173]:= Clear[ca1]

In[174]:= cb1[n_] :=
           Module[{m, cb},
            V = $\frac{Vo}{n}$;
            Do[cb[1] = $\frac{Caf\,k\,V}{q+k\,V}$;
```

```
              cb[m] = (ca1[m-1]k V)/(q+k V) +cb[m-1],{m, 0, n}];
              cb[n]
            ]

In[175]:= cb1[4]
          Together[%]

Out[175]= 0.2+0.2ca1[1]+0.2ca1[2]+0.2ca1[3]

Out[176]= 0.2+0.2ca1[1]+0.2ca1[2]+0.2ca1[3]
```

In order to evaluate the expression fully in term of just the parameters, we need the expression for **ca1[n]** to have been evaluated. We do the two in one cell to make this happen:

```
In[177]:= cacb[n_]:=
           Module[{m, ca, cb},
            V = Vo/n;
            Do[{ca[0] = Caf, ca[1] = (q Caf)/(q + kV), cb[0] = 0,
                cb[1] = (Caf k V)/(q + k V)};
             {ca[m] = (q ca[m - 1])/(q + k V), cb[m] = (ca[m - 1]k V)/(q+kV) + cb[m - 1]},
                {m, 1, n}];
            {Together[ca[n]], Together[cb[n]]}
            ]

In[178]:= cacb[4]

Out[178]= {0.4096, 0.5904}
```

This works very nicely indeed. We can apply numerical values to these solutions, solve, and even obtain the timing for a comparison to the rule-based and functional programming cases:

```
In[179]:= Caf = 1;
          k = .1;
          q = 10;
          ntot = 10;
          Vo = 100;
          cacb[1000] // Timing

Out[184]= {1.43 Second, {0.368063, 0.631937}}
```

We see here that the procedural solution is identical to the asymptotic solutions obtained earlier and the time to do **n** = 1000 is only 1.98 sec of CPU on a Pentium I and 0.22 on the

Pentium III (I GHz), which is much faster than the first, rule-based program and a little slower than the functional implementation. Once again we apply the function **cacb** that we have created in *listable* so we can **Map** it down the vector of *n* values that we had used before in the rule-based computations and we can compare timings:

```
In[185]:= n = {1, 2, 3, 10, 20, 100, 1000};
          cacb /@n // Timing

Out[186]= {1.59 Second,
           {{0.5, 0.5},{0.444444,0.555556},{0.421875,0.578125},
            {0.385543, 0.614457}, {0.376889, 0.623111},
            {0.369711, 0.630289}, {0.368063, 0.631937}}}
```

When we did this via brute force using all the equations and no recursion, we utilized 17.54 sec of CPU time while the procedural program required just 1.26 sec.

Now we can compare how the conversions grow to the PFR limit as the number of CSTRs increases. First we compute the concentrations of species A and B for 1–150 CSTRs.

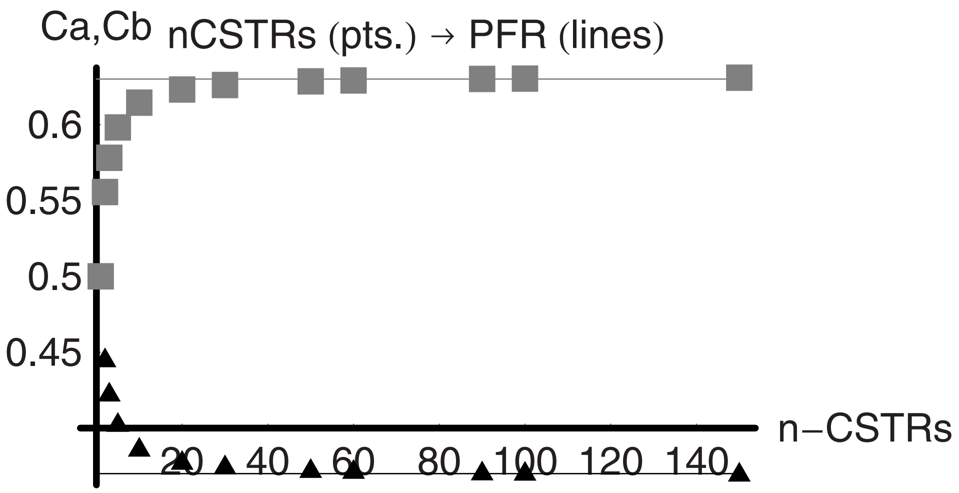
```
In[187]:= n = {1, 2, 3, 5, 10, 20, 30, 50, 60, 90, 100, 150};
          cstrconcs = cacb /@n

Out[188]= {{0.5, 0.5}, {0.444444, 0.555556}, {0.421875, 0.578125},
           {0.401878, 0.598122}, {0.385543, 0.614457},
           {0.376889, 0.623111}, {0.373927, 0.626073},
           {0.371528, 0.628472}, {0.370924, 0.629076},
           {0.369914, 0.630086}, {0.369711, 0.630289},
           {0.369102, 0.630898}}
```

```
Out[198]= {{0.5, 0.5}, {0.444444, 0.555556}, {0.421875, 0.578125},
           {0.401878, 0.598122}, {0.385543, 0.614457},
           {0.376889, 0.623111}, {0.373927, 0.626073},
           {0.371528, 0.628472}, {0.370924, 0.629076},
           {0.369914, 0.630086}, {0.369711, 0.630289},
           {0.369102, 0.630898}}
```

Next we plot these versus the PFR limiting concentrations:

```
In[189]:= << Graphics`MultipleListPlot`

In[190]:= calist = Table[{n[[x]], cstrconcs[[x, 1]]}, {x, 1, Length[n]}];
          cblist = Table[{n[[x]], cstrconcs[[x, 2]]}, {x, 1, Length[n]}];

          MultipleListPlot[
           calist, cblist, DefaultFont → {"Helvetica", 15},
           SymbolShape → {PlotSymbol[Triangle,5],PlotSymbol[Box,5]},
           SymbolStyle → {GrayLevel[0], GrayLevel[0.5]},
     Epilog → {
       {GrayLevel[0.5], Line[{{0, .63}, {150, 0.63}}]},
       {Line[{{0, 1 - .63}, {150, 1 - 0.63}}]}
               },
     AxesLabel → {"n-CSTRs", "Ca,Cb"},
     PlotLabel → "nCSTRs (pts.) → PFR (lines)"
    ];
```

```
General::spell1 : Possible spelling error: new symbol name
   "cblist" is similar to existing symbol "calist".
```



After approximately 30 CSTRs in series the result is the same as one PFR of equal volume. This makes sense mathematically in terms of our analysis and it also makes good sense intuitively because we are using the same total volume more efficiently.

9.9 Residence Time Distribution

We first encountered in Chapter 3 on mixing in multicomponent systems the problem of bypassing and less than perfect mixing. If we have two or more reactants that must mix in order for reaction to occur, then any deviations from a single-valued residence time distribution will show up as an apparent deviation from the predictions based upon perfect mixing. The spread in the residence time distribution leads to different extents of reaction for the fluid elements with these different times.

The lack of perfect mixing leads to this distribution. Recirculation zones may lead to longer than average residence times in some regions of the tank and bypassing to shorter than average time (see Figure 7). Longer or shorter times can translate into regions of higher or lower conversion. It is this type of problem we want to examine now.

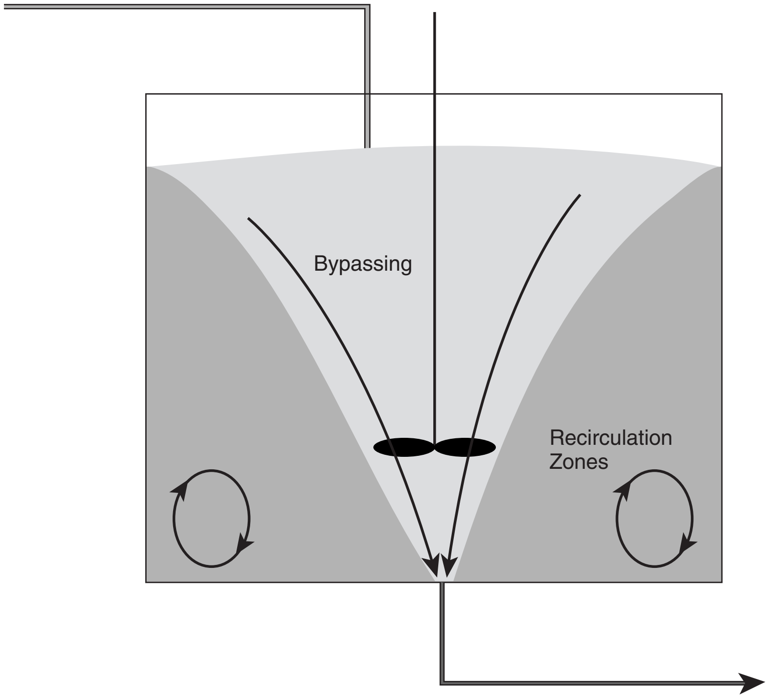


Figure 7

We have developed the equations for a steady-state CSTR in which the reversible reaction of **A** and **B** produces **D** and one mole of **D** reacts back to produce **A** and **B** but the kinetics are second order:

$$\mathrm{A} + \mathrm{B} \rightleftarrows \mathrm{D}$$

$$0 = (\mathrm{Caf} - \mathrm{Ca})\frac{q}{V} - \mathrm{kab\,Ca\,Cb} + \mathrm{kd\,Cd}^2$$

$$0 = (\mathrm{Cbf} - \mathrm{Cb})\frac{q}{V} - \mathrm{kab\,Ca\,Cb} + \mathrm{kd\,Cd}^2$$

$$0 = -\mathrm{Cd}\frac{q}{V} + \mathrm{kab\,Ca\,Cb} - \mathrm{kd\,Cd}^2$$

$$0 = \mathrm{qbf} - \mathrm{q}$$

$$0 = \frac{(\mathrm{Caf} - \mathrm{Ca})}{\theta} - \mathrm{kab\,Ca\,Cb} + \mathrm{kd\,Cd}^2$$

$$0 = \frac{(\mathrm{Cbf} - \mathrm{Cb})}{\theta} - \mathrm{kab\,Ca\,Cb} + \mathrm{kd\,Cd}^2$$

$$0 = -\frac{\mathrm{Cb}}{\theta} + \mathrm{kab\,Ca\,Cb} - \mathrm{kd\,Cd}^2$$

Of course $\frac{q}{V} = \frac{1}{\theta}$, the holding time for the fluid in the reactor. When we derived these equations we did so under the assumption of *perfect mixing*, or *complete back-mixing*, which is another term for this idealization. If there were perfect mixing, then there would be just one residence time and that would be the same as the holding time θ. When the residence time distribution is a **DiracDelta** function at one time, then we have one holding time. But what if this is not the case? What if, instead, the times spent by fluid elements are distributed about some mean value—then what? Well, then we would have to average these equations over the distribution to get the average concentration emerging from the less than perfectly mixed reactor. We begin by solving the equation in terms of θ:

```
In[193]:= Clear["Global`*"]

In[194]:= cstr = Simplify[
             Solve[
              {0 == (Caf - Ca)/θ - kab Ca Cb + kd Cd^2,
              {0 == (Cbf - Cb)/θ - kab Ca Cb + kd Cd^2,
               0 == -Cd/θ + kab Ca Cb — kd Cd^2},
              {Ca, Cb, Cd}]];

            TableForm[cstr, TableDirections → {Column, Column}]
```

Out[195]//TableForm =

$$\mathrm{Ca} \to -\frac{1+\mathrm{Cbf\,kab}\,\theta-\mathrm{Caf}\,(\mathrm{kab}-2\mathrm{kd})\,\theta+\sqrt{-4\mathrm{Caf\,Cbf\,kab}\,(\mathrm{kab}-\mathrm{kd})\,\theta^2+(1+\mathrm{Caf\,kab}\,\theta+\mathrm{Cbf\,kab}\,\theta)^2}}{2\,(\mathrm{kab}-\mathrm{kd})\,\theta}$$

$$\mathrm{Cb} \to -\frac{1+\mathrm{Caf\,kab}\,\theta-\mathrm{Cbf}\,(\mathrm{kab}-2\mathrm{kd})\,\theta+\sqrt{-4\mathrm{Caf\,Cbf\,kab}\,(\mathrm{kab}-\mathrm{kd})\,\theta^2+(1+\mathrm{Caf\,kab}\,\theta+\mathrm{Cbf\,kab}\,\theta)^2}}{2\,(\mathrm{kab}-\mathrm{kd})\,\theta}$$

$$\mathrm{Cd} \to \frac{1+\mathrm{Cbf\,kab}\,\theta+\mathrm{Caf\,kab}\,\theta+\sqrt{-4\mathrm{Caf\,Cbf\,kab}\,(\mathrm{kab}-\mathrm{kd})\,\theta^2+(1+\mathrm{Caf\,kab}\,\theta+\mathrm{Cbf\,kab}\,\theta)^2}}{2\,(\mathrm{kab}-\mathrm{kd})\,\theta}$$

$$\mathrm{Ca} \to \frac{-1-\mathrm{Cbf\,kab}\,\theta+\mathrm{Caf}\,(\mathrm{kab}-2\mathrm{kd})\,\theta+\sqrt{-4\mathrm{Caf\,Cbf\,kab}\,(\mathrm{kab}-\mathrm{kd})\,\theta^2+(1+\mathrm{Caf\,kab}\,\theta+\mathrm{Cbf\,kab}\,\theta)^2}}{2\,(\mathrm{kab}-\mathrm{kd})\,\theta}$$

$$\mathrm{Cb} \to \frac{-1-\mathrm{Caf\,kab}\,\theta+\mathrm{Cbf}\,(\mathrm{kab}-2\mathrm{kd})\,\theta+\sqrt{-4\mathrm{Caf\,Cbf\,kab}\,(\mathrm{kab}-\mathrm{kd})\,\theta^2+(1+\mathrm{Caf\,kab}\,\theta+\mathrm{Cbf\,kab}\,\theta)^2}}{2\,(\mathrm{kab}-\mathrm{kd})\,\theta}$$

$$\mathrm{Cd} \to \frac{1+\mathrm{Cbf\,kab}\,\theta+\mathrm{Caf\,kab}\,\theta-\sqrt{-4\mathrm{Caf\,Cbf\,kab}\,(\mathrm{kab}-\mathrm{kd})\,\theta^2+(1+\mathrm{Caf\,kab}\,\theta+\mathrm{Cbf\,kab}\,\theta)^2}}{2\,(\mathrm{kab}-\mathrm{kd})\,\theta}$$

The next step is to express these as functions of θ. By examination, we can find that the second set of solutions is the one that leads to physically realistic values of the concentrations; thus we use these:

```
In[196]:= Clear[Ca, Cb, Cd]
          Ca[θ_] = cstr[[2, 1, 2]];
          Cb[θ_] = cstr[[2, 2, 2]];
          Cd[θ_] = cstr[[2, 3, 2]];
```

Testing the solutions requires putting them back into the equations to see if they are valid:

In[200]:= $\{\text{Simplify}[0 == \frac{(\text{Caf} - \text{Ca}[\theta])}{\theta} - \text{kab Ca}[\theta]\,\text{Cb}[\theta] + \text{kd Cd}[\theta]^2],$
$\text{Simplify}[0 == \frac{(\text{Cbf} - \text{Cb}[\theta])}{\theta} - \text{kab Ca}[\theta]\,\text{Cb}[\theta] + \text{kd Cd}[\theta]^2],$
$\text{Simplify}[0 == -\frac{\text{Cd}[\theta]}{\theta} + \text{kab Ca}[\theta]\,\text{Cb}[\theta] - \text{kd Cd}[\theta]^2]\}$

```
Out[200]= {True, True, True}
```

This shows that the functions we are using do indeed satisfy the equations for the steady-state **CSTR**.

We will use the **NormalDistribution** to make the representations of the residence time distribution. The **Probability Density Function (PDF)** is made up of the **Normal Distribution** and the variable θ. This can be integrated in closed form:

```
In[201]:= << Statistics`NormalDistribution`

In[202]:= Clear[ndist, pdf, θ, δθ, θmin, θmax]
          ndist = NormalDistribution[θm, δθ];
          pdf = PDF[ndist, θ];
          Integrate[pdf, {θ, θmin, θmax}]

        General::spell1: Possible spelling error: new symbol
         name "pdf" is similar to existing symbol "PDF".

        General::spell1: Possible spelling error: new symbol
         name "θm" is similar to existing symbol "θ".
```

Out[205]= $$\frac{\sqrt{\frac{\pi}{2}}\,\delta\theta\,\text{Erf}[\frac{-\theta m+\theta max}{\sqrt{2}\,\delta\theta}] - \sqrt{\frac{\pi}{2}}\,\delta\theta\,\text{Erf}[\frac{-\theta m+\theta min}{\sqrt{2}\,\delta\theta}]}{\sqrt{2\pi}\,\delta\theta}$$

Because the **Normal distribution PDF** requires a mean value and a variance, we supply these and then **Plot** the result in order to visualize the distribution. We have purposefully chosen a very narrow distribution for the first case. Next the parameter values are assigned and we **Integrate** the products of the concentration functions and the **PDF** in θ over the range of θ values. As shown in the following graph, these are the residence-time-averaged values of the concentrations and the conversion of **A** and **B**:

```
In[206]:= θm = 10;
          δθ = .2;
          θmin = 0.001 θm;
          θmax = 5 θm;
          ndist = NormalDistribution[θm, δθ];
          pdf = PDF[ndist, θ];
          NIntegrate[pdf, {θ, θmin, θmax}]
          NIntegrate[θ pdf, {θ, θmin, θmax}]/
```

```
 NIntegrate[pdf, {θ, θmin, θmax}]
Plot[pdf,
  {θ, .0001 θm, 2θm},
  AxesLabel → {"θ", "PDF[θ]"},
  PlotStyle → {GrayLevel[0.4], Thickness[0.01],
   Dashing[{0.02, 0.03}]},
  PlotRange → {{0, 2θm}, {0, Max[Table[N[pdf] + .1,
   {θ, θmin, θmax}]]}},
  Epilog → {Line[{{θm - .01, 0}, {θm - .01,
   Max[Table[N[pdf] + .1, {θ, θmin, θmax}]]}}]},
  PlotLabel → "θm"];
Caf = 1;
Cbf = 1;
kd = 1.1;
kab = 2.2;
```

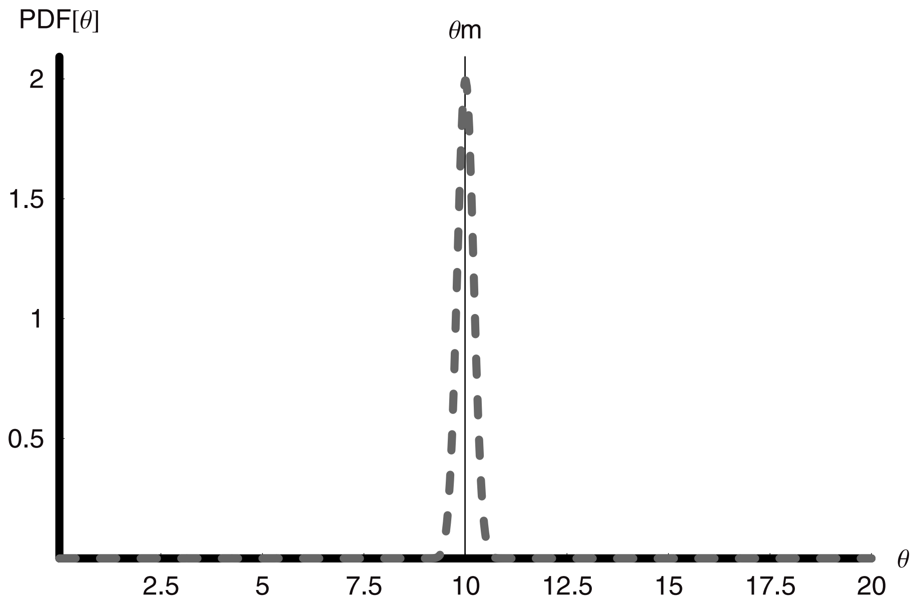
$$\text{cave} = \frac{\text{NIntegrate[Ca[}\theta\text{] pdf,\{}\theta\text{,}\theta\text{min,}\theta\text{max\}]}}{\text{NIntegrate[pdf,\{}\theta\text{,}\theta\text{min,}\theta\text{max\}]}}$$

$$\text{cdave} = \frac{\text{NIntegrate[Cb[}\theta\text{] pdf,\{}\theta\text{,}\theta\text{min,}\theta\text{max\}]}}{\text{NIntegrate[pdf,\{}\theta\text{,}\theta\text{min,}\theta\text{max\}]}}$$

$$\text{cdave} = \frac{\text{NIntegrate[Cd[}\theta\text{] pdf,\{}\theta\text{,}\theta\text{min,}\theta\text{max\}]}}{\text{NIntegrate[pdf,\{}\theta\text{,}\theta\text{min,}\theta\text{max\}]}}$$

```
100(1 - cave)
```

```
Out[212]= 1.

Out[213]= 10.
```



```
Out[219]= 0.432349

        General::spell1: Possible spelling error: new symbol
         name ″cbave″ is similar to existing symbol ″cave″.

Out[220]= 0.432349

        General::spell: Possible spelling error: new symbol
         name ″cdave″ is similar to existing symbols {cave,
         cbave}.

Out[221]= 0.567651

Out[222]= 56.7651

In[223]:= Clear["Global`*"]
```

Rather than copy this code cell multiple times and run it successively, it would be much better to create a **Module** function, but even better than a **Module** function would be a **Package**. A **Package** is a program that we can call at any time simply by loading it and then running it. This will allow us to run without name collisions and to use what we have developed outside of this notebook context. Normally, when we are running *Mathematica* we are operating in the **Global** context, which is why we have often started our codes cells with **Clear[″Global′*″]**. This means clear all the variable names that we have made or used within the **Global** context. To find out which context we are in we input:

```
In[224]:= $Context

Out[224]= Global`
```

Contexts are very much like directories and you can find much written about this topic elsewhere. We will create a package called **cstrresdist**, which we will store in a folder called **AddOns** within the **Applications** folder. The inputs will be the forward and reverse rate constants **kab** and **kd**, the mean residence time **θm**, and the variance in the residence time **$\delta\theta$**. The general form for a **Package** is:

Begin Package[“Context‘PackageName”]

PackageName::Usage=“Narrative explanation of Package...”

Begin[“‘Private‘”]

function name[variables_]: = ...

End[ ]

EndPackage [ ]

We will use this format and the functions we have defined will be implemented as a **Module Function**. For the case at hand, instead of making the Context = **AddOns’**, it has been placed directly into the **Global‘** context.

```
In[225]:= BeginPackage["Global`cstrrtd`",
                    {"Statistics`ContinuousDistributions`",
                     "Statistics`NormalDistribution`",
                     "Statistics`Common`DistributionsCommon`",
                     "Statistics`DescriptiveStatistics`"
             }]

           cstrrtd::Usage = "cstrrtd[kab,kd,θm,δθ] creates the
              r.t.d, plots it and then computes the average exit
              concentrations of A and B and the conversion of A"
           Begin["`Private`"]

           cstrrtd[kab_, kd_, θm_, δθ_] :=
            Module[
             {Caf = 1, Cbf = 1, caave, cbave, cdave, a, b, θ,
              conversion, x, y, z},
             θmin = 0.001θm;
               θmax = 5θm;
```

$$\texttt{ca[}\theta\texttt{\_] = }\frac{1}{2(\texttt{kab - kd})\theta}(-1 - \texttt{Cbf kab}\,\theta + \texttt{Caf(ka - 2 kd)}\theta$$

$$+\sqrt{(-4\,\texttt{Caf Cbf kab(kab - kd)}\theta^2 + (1+\texttt{Caf kab}\theta+\texttt{Cbf kab}\theta)^2)});$$

$$\texttt{cd[}\theta\texttt{\_] = }\frac{1}{2(\texttt{kab - kd})\theta}(1 + \texttt{Caf kab}\,\theta + \texttt{Cbf kab}\,\theta -$$

$$\sqrt{(-4\,\texttt{Caf Cbf kab(kab - kd)}\theta^2+(1+\texttt{Caf kab}\theta + \texttt{Cbf kab}\theta)^2)});$$

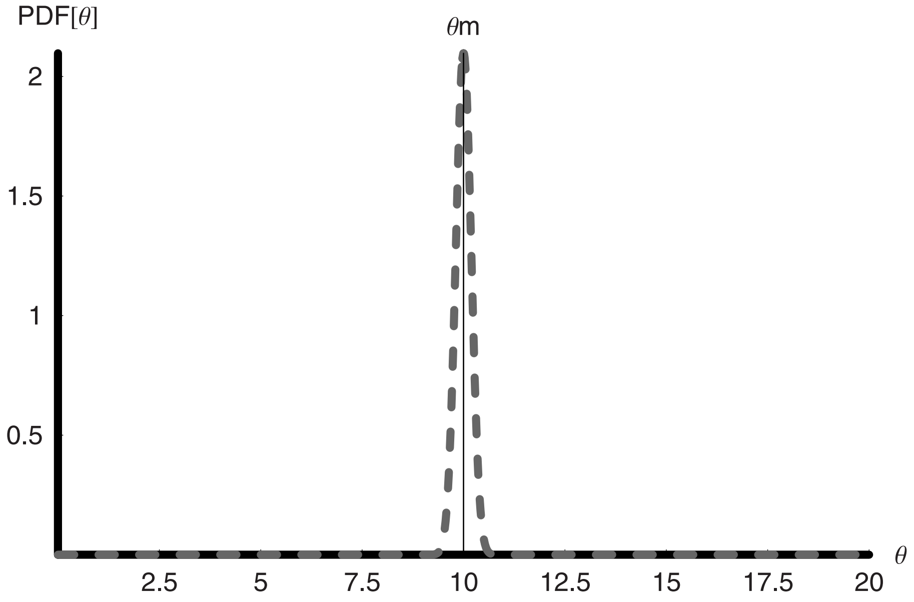
```
             distfunc[x_, y_, z_] = PDF[NormalDistribution[x,y],z];

             Plot[distfunc[θm, δθ, θ],
              {θ, .0001θm, 2θm},
             AxesLabel  → {"θ", "PDF[θ]"},
             PlotStyle  → {GrayLevel[0.4], Thickness[0.01],
               Dashing[{0.02, 0.03}]},
             PlotRange  →
              {{0, 2θm}, {0, Max[Table[N[distfunc[θm, δθ, θ]],
               {θ, θmin, θmax}]]}},
             Epilog  → {Line[{{θm, 0},
              {θm, Max[Table[N[distfunc[θm, δθ, θ]],
              {θ, θmin, θmax}]]}}]},
             PlotLabel  → "θm"];

           a = NIntegrate[distfunc[θm, δθ, θ], {θ, θmin, θmax}];
           b = NIntegrate[θ distfunc[θm, δθ, θ], {θ, θmin, θmax}]/
                NIntegrate[distfunc[θm, δθ, θ], {θ, θmin, θmax}];
     caave = NIntegrate[ca[θ] distfunc[θm, δθ, θ], {θ, θmin, θmax}]/
       NIntegrate[distfunc[θm, δθ, θ], {θ, θmin, θmax}];
```

```
    cdave = NIntegrate[cd[θ] distfunc[θm, δθ, θ], {θ, θmin, θmax}]/
      NIntegrate[distfunc[θm, δθ, θ], {θ, θmin, θmax}];
    conversion = 100(1 - caave);
    {a, b, caave, cdave, conversion}
   ]
  End[]
  EndPackage[]
```

```
Out[225]= Global`cstrrtd`

Out[226]= cstrrtd[kab,kd,θm,δθ] creates the r.t.d, plots it and
            then computes the average exit concentrations of A
         and B and the conversion of A

Out[227]= Global`cstrrtd`Private'

Out[229]= Global`cstrrtd`Private'
```

To see how this works we can run a case as follows:

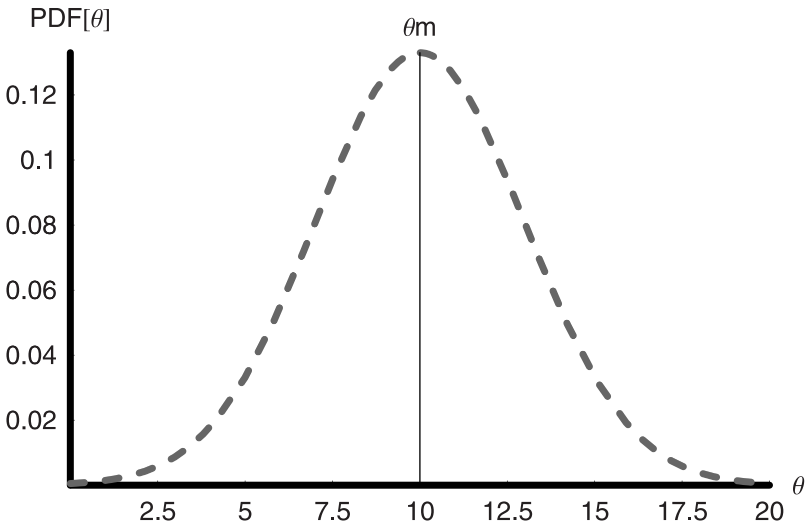
```
In[231]:= cstrrtd[2, 1, 10, .19]
```



```
Out[231]= {1., 10., 0.434089, 0.565911, 56.5911}
```

The graph is the distribution of residence times about the mean. The bottom line of output gives us the integral of the **PDF**, the mean normalized residence time, the concentration of **A** and **B**, and, finally, the conversion of **A**. We see from this graph that with the mean residence time of 100 and variance of just 2, this is a very sharp distribution. What would happen if the r.t.d. were broader—say, 30?

```
In[232]:= cstrrtd[2, 1, 10, 3]
```



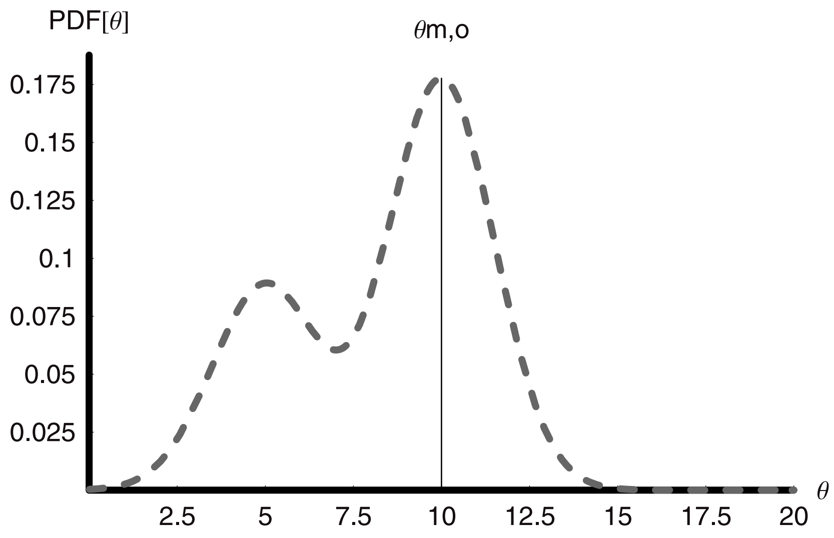
```
Out[232]= {0.999566, 10.0047, 0.436484, 0.563516, 56.3516}
```

We see from the preceding that even though the distribution has been broadened considerably, there is no evidence of an effect on the overall conversion. This makes sense because there are just as many fluid elements below the average as there are above and so we get average behavior—just as we intuitively know we should. What, however, happens when the flow is so maldistributed that the residence time is actually bimodal?

In the following package, **Global 'rtdcstr'**, two normal distributions are weighted (**wf1** and **wf2**) and added together to give the overall residence time distribution. The second Gaussian distribution is taken to be centered at half the mean value of the first, but with the same variance. The function **p** is their sum (see the following):

```
In[233]:= BeginPackage["Global`rtdcstr`",
                    {"Statistics`ContinuousDistributions`",
                    "Statistics`NormalDistribution`",
                    "Statistics`Common`DistributionsCommon`",
                    "Statistics`DescriptiveStatistics`"
                 }]
```

```
rtdcstr::usage = "testing 1,2,3..."
Begin["`Private`"]
rtdcstr[kab_, kd_, θm_, σθ_, wf1_, wf2_]:= Module[
  {Caf = 1, Cbf = 1, x, xmin, xmax},
  xmin = 0.001θm;
  xmax = 5θm;
  ndist1 = NormalDistribution[θm, σθ];
  ndist2 = NormalDistribution[.5θm, σθ];
  pdfunction1 = PDF[ndist1, x];
  pdfunction2 = PDF[ndist2, x];
  p = (wf1 pdfunction1 + wf2 pdfunction2)/(wf1 + wf2);
  ca[y_] = 1/(2(kab - kd)y) (-1-Cbf kaby+Caf(kab-2kd)y+
       √(-4Caf Cbf kab(kab-kd)y^2+(1+Caf kaby+Cbf kaby)^2));
  Plot[p, {x, 0, xmax},
   AxesLabel → {"θ", "PDF[θ]"},
   PlotStyle → {GrayLevel[0.4], Thickness[0.01],
    Dashing[{0.02, 0.03}]},
   PlotRange → {{xmin, 2θm},
    {0, .01 + Max[Table[N[p], {x, xmin, xmax}]]}},
   Epilog → {Line[{{θm, 0}, {θm, Max[Table[N[p],
    {x, xmin, xmax}]]}}]},
   PlotLabel → "θm,o"];

  totp = NIntegrate[p, {x, 0, xmax}];
  thetam = NIntegrate[xp,{x,0,xmax}]/totp;
  caave = NIntegrate[ca[x]p,{x,0,xmax}]/totp;
  conversion = 100 (1-caave);
  {totp, thetam, caave, conversion}
 ]
End[]
EndPackage[]

Out[233]= Global`rtdcstr`

Out[234]= testing 1,2,3...

Out[235]= Global`rtdcstr`Private`

Out[237]= Global`rtdcstr`Private`

In[239]:= rtdcstr[2, 1, 10, 1.5, 4, 2]
```



```
Out[239]= {0.999857, 8.33458, 0.441837, 55.8163}
```

We see that this extreme case has some effect on the conversion, but not to any significant extent.

Let us turn now to the case of series reaction:

$$A \rightarrow B \rightarrow D$$

We can once again solve the equations for the concentrations as a function of the residence time:

```
In[240]:= Clear["Global`*"]

          cstrabd2=Simplify[
            Solve[
```
$$\{0 == \frac{(Caf - Ca)}{\theta} - kabCa,$$
$$0 == \frac{-Cb}{\theta} + kabCa - kbdCb,$$
$$0 == -\frac{Cb}{\theta} + kabCb\}$$
```
             {Ca, Cb, Cd}]];

          TableForm[cstrabd2, TableDirections → {Column, Column}]
```

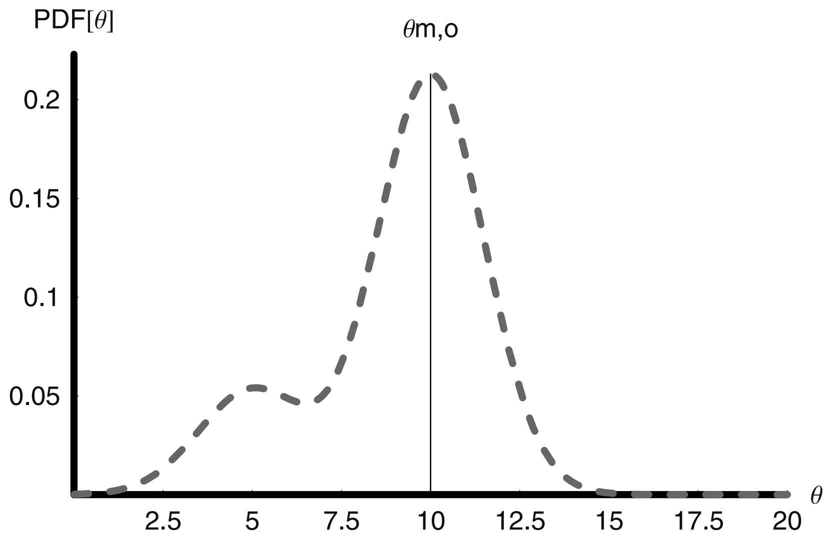
```
Out[242]//TableForm=
```
$$Cd \rightarrow \frac{Caf\,kab\,kbd\theta^2}{(1 + kab\theta)(1 + kbd\theta)}$$

$$\mathrm{Cb} \to \frac{\mathrm{Caf\,kab}\theta}{(1 + \mathrm{kab}\theta)(1 + \mathrm{kbd}\theta)}$$
$$\mathrm{Ca} \to \frac{\mathrm{Caf}}{(1 + \mathrm{kab}\theta)}$$

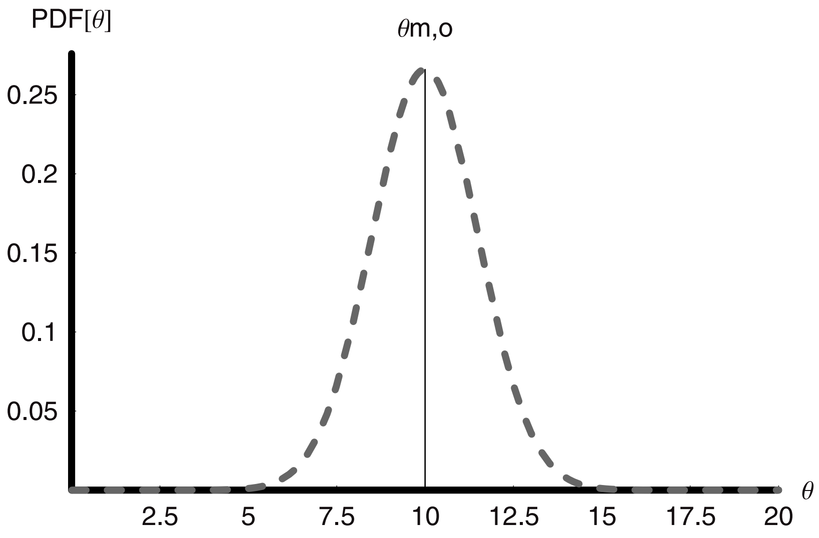
The package that we have just written for the bimodally distributed residence times can be adapted to give a new package we shall call **rtdcstrabd2**. The output of this **Package** will be the r.t.d. of the integrated **PDF** value, the average residence time, then the conversion followed by the selectivities to **A** and to **B**:

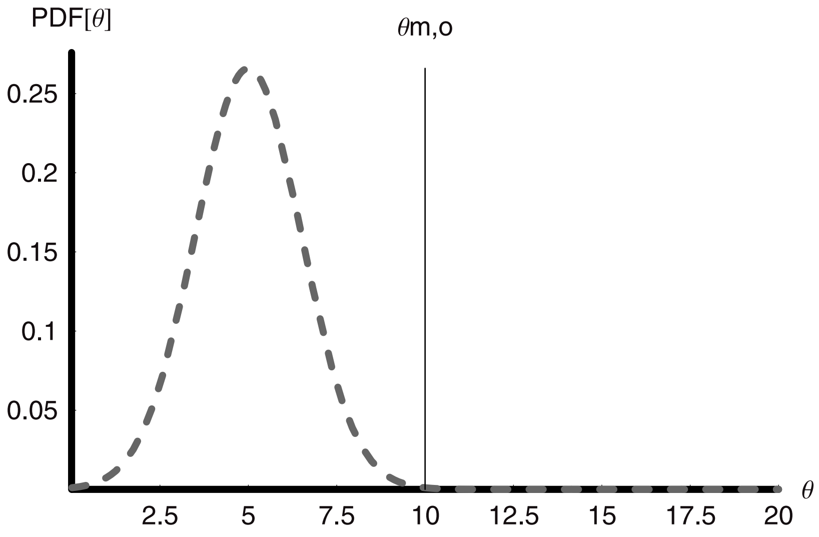
```
In[243]:= BeginPackage["Global`rtdcstrabd2`",
                 {"Statistics`ContinuousDistributions`",
                  "Statistics`NormalDistribution`",
                  "Statistics`Common`DistributionsCommon`",
                  "Statistics`DescriptiveStatistics`"
              }]
          rtdcstrabd2::usage = "testing 1,2,3..."
          Begin["`Private`"]
          rtdcstrabd2[kab_, kbd_, θm_, σθ_, wf1_, wf2_]:= Module[
            {Caf = 1, Cbf = 1, x, xmin, xmax},
            xmin = 0.001θm;
            xmax = 5θm;
            ndist1 = NormalDistribution[θm, σθ];
            ndist2 = NormalDistribution[.5θm, σθ];
            pdfunction1 = PDF[ndist1, x];
            pdfunction2 = PDF[ndist2, x];
```
$$p = \frac{\mathrm{wf1\ pdfunction1 + wf2\ pdfunction2}}{\mathrm{wf1 + wf2}};$$
$$\mathrm{ca[y\_]} = \frac{\mathrm{Caf}}{1 + \mathrm{kab\ y}};$$
$$\mathrm{cb[y\_]} = \frac{\mathrm{Caf\ kab\ y}}{(1 + \mathrm{kab\ y})(1 + \mathrm{kbd\ y})};$$
$$\mathrm{cd[y\_]} = \frac{\mathrm{Caf\ kab\ kbd\ y}^2}{(1 + \mathrm{kab\ y})(1 + \mathrm{kbd\ y})};$$
```
            Plot[p, {x, 0, xmax},
             AxesLabel → {"θ", "PDF[θ]"},
             PlotStyle → {GrayLevel[0.4], Thickness[0.01],
              Dashing[{0.02, 0.03}]},
             PlotRange → {{xmin, 2θm},
              {0, .01 + Max[Table[N[p], {x, xmin, xmax}]]}},
             Epilog → {Line[{{θm, 0}, {θm, Max[Table[N[p],
              {x, xmin, xmax}]]}}]},
             PlotLabel → "θm,o"];
```

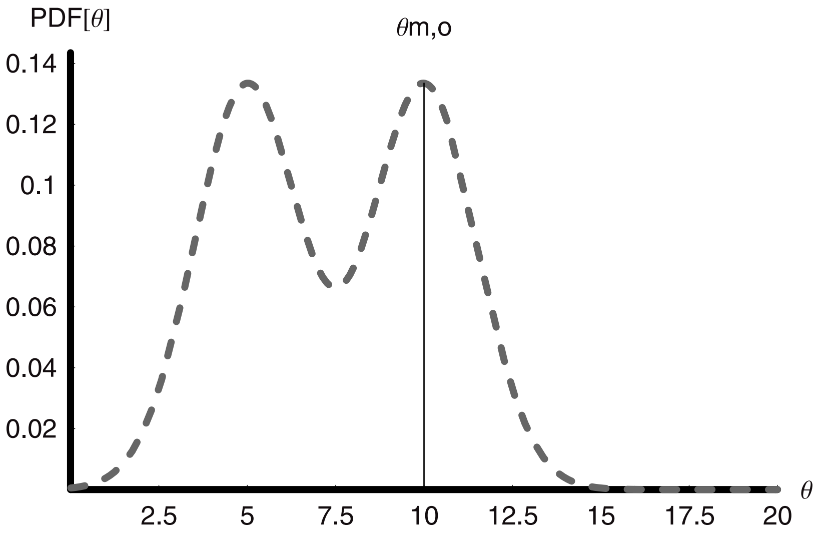
```
      totp = NIntegrate[p, {x, 0, xmax}];
      thetam = NIntegrate[xp,{x,0,xmax}] / totp;
      caave = NIntegrate[ca[x] p,{x,0,xmax}] / totp;
      cbave = NIntegrate[cb[x] p,{x,0,xmax}] / totp;
      cdave = NIntegrate[cd[x] p,{x,0,xmax}] / totp;
      conversion = 100 (1 - caave);
      selB = cbave / (caave + cbave + cdave);
      selD = cdave / (caave + cbave + cdave);
      {totp, thetam, conversion, selB, selD}
       ]
    End[]
    EndPackage[]
Out[243]= Global`rtdcstrabd2`

Out[244]= testing 1,2,3...

Out[245]= Global`rtdcstrabd2`Private`

Out[247]= Global`rtdcstrabd2`Private`

In[249]:= rtdcstrabd2[2, 1, 10, 1.5, 1, .25]
```

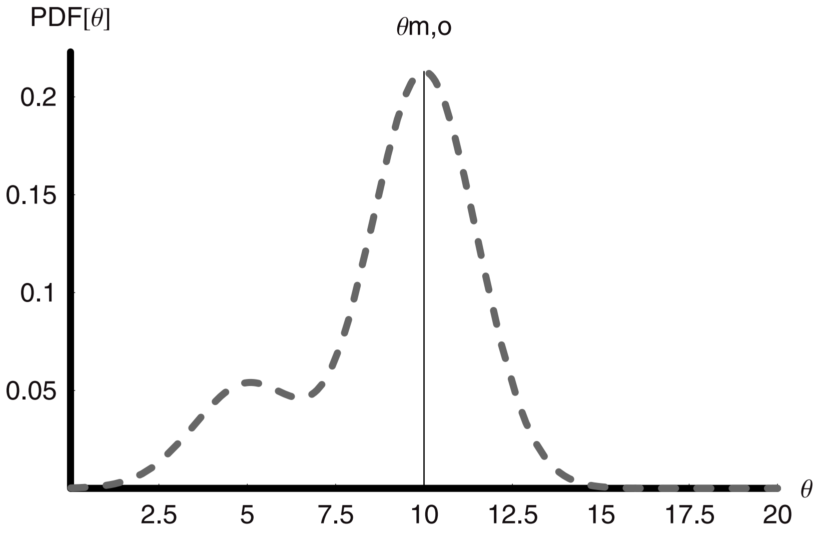


```
Out[249]= {0.999914, 9.00081, 94.1074, 0.102341, 0.838733}

In[250]:= {rtdcstrabd2[.2, .1, 10, 1.5, 1, 0],
           rtdcstrabd2[.2, .1, 10, 1.5, 0, 1],
           rtdcstrabd2[.2, .1, 10, 1.5, 1, 1],
           rtdcstrabd2[.2, .1, 10, 1.5, 1, .25]}
```









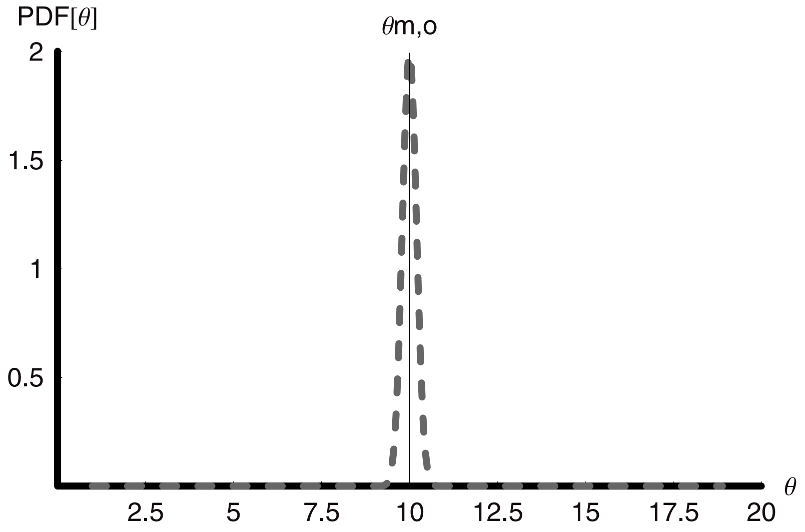
```
Out[250]= {{1., 10., 66.3228, 0.332179, 0.331049},
          {0.999571, 5.00231, 48.8138, 0.323047, 0.165091},
          {0.999785, 7.50169, 57.5702, 0.327614, 0.248088},
          {0.999914, 9.00081, 62.8222, 0.330353, 0.297869}}
```

Here we can see a much stronger effect of the **r.t.d.** on the conversions and the selectivities. The cases we have chosen to examine are purposefully extreme. With the **r.t.d.** centered at

θm = 10, the conversion of **A** is 66% and the selectivities to **A** and to **B** are virtually 1 : 1. When we push the **r.t.d.** to a shorter residence time and **θm** = 5 the whole picture changes to one of lower conversion and higher selectivity to **B** by almost 2 : 1 over **D**—as we would expect. Note, however, that the yield to **B** is now under 16%, whereas in the first case it was closer to 22%. With a bimodal distribution having nearly equal probabilities at **θm** = 5 and 10, the picture changes once again: The conversion is lower than the narrowly distributed case centered at **θm** = 10, and so too is the selectivity to **D** rather than **B**. Finally, if the maldistribution is less severe, then too is the departure from the unimodal result.

For comparison, we show in the following and compute the case of a very narrow distribution and the actual values for the base case of a single holding time, that is for Dirac-Delta Function of residence times:

```
In[251]:= rtdcstrabd2[.2, .1, 10, .2, 1, 0]
```



```
Out[251]= {1., 10., 66.6607, 0.333315, 0.333293}
```

```
In[252]:= kab = .2;
          kbd = .1;
          Caf = 1;
          θ = 10;
```

$$\text{Cd} \rightarrow \text{N}\left[\frac{\text{Caf kab kbd}\,\theta^2}{(1+\text{kab}\,\theta)(1+\text{kbd}\,\theta)}\right]$$

$$\text{Cb} \rightarrow \text{N}\left[\frac{\text{Caf kab}\,\theta}{(1+\text{kab}\,\theta)(1+\text{kbd}\,\theta)}\right]$$

$$\text{Ca} \rightarrow \text{N}\left[\frac{\text{Caf}}{1+\text{kab}\,\theta}\right]$$

```
Out[256]= {{{{Cd → 0.333333}, {Cb → 0.333333}, {Ca → 0.333333}}}}
```

9.10 Time-Dependent PFR—Complete and Numerical Solutions

The PFR equation that we derived had two partial derivatives—one in time and one in space. Recall that this is the equation for component **A** being fed to the PFR:

$$\frac{\partial \mathrm{Ca}}{\partial t} = \frac{-q}{\mathrm{Acr}}\frac{\partial \mathrm{Ca}}{\partial z} - r_{A-}$$

The time-dependent derivative of concentration is the accumulation term for the differential volume Acr dz. This is essentially the same for all the reactor types we have studied. What makes the PFR different and perhaps more interesting is the spatial derivative.

Since it was not within our ability to solve the time-dependent equation, we naturally solved the steady-state problem so that the accumulation term went to zero, which left only the spatial derivative:

$$0 = \frac{-q}{A}\frac{\partial \mathrm{Ca}}{\partial z} - r_{A-}$$
$$\frac{q}{A}\frac{\partial \mathrm{Ca}}{\partial z} = -r_{A-}$$

This problem is more soluble because it involves only this one derivative. In fact, if we recall that $\frac{q}{A}$ is the velocity v_z in the PFR, then we simplify the equation further:

$$\frac{q}{A}\frac{\partial \mathrm{Ca}}{\partial z} = -r_{A-}$$
$$v_z\frac{\partial \mathrm{Ca}}{\partial z} = -r_{A-}$$

However, $\frac{v_z}{\partial z}$ is the ratio of a constant velocity to a differential distance. This has units of reciprocal time. Formally, we can take the constant into the derivative and this gives us $\frac{1}{\partial \frac{z}{v_z}}$, which can be defined as the reciprocal of the differential holding time $\frac{1}{\partial \tau}$. The equation becomes:

$$\frac{\partial \mathrm{Ca}}{\partial \tau} = -r_{A-}$$

This result is very nice because it shows us that at the steady state the **PFR** has the same governing equation as the transient batch reactor, except that instead of the real time the differential is given in terms of the differential holding time.

9.11 Transient PFR

The question arises as to how long it will take a reactor operating in the plug flow regime to reach a steady state for a specific set of reaction kinetics, volume, and flow rate. To solve this problem we need to solve both in time and in space. If the kinetics are simple, then we can solve the problem analytically, that is, we can derive expressions for the concentrations that are functions of time and position. However, often the kinetics are not straightforward and analytical solutions must be surrendered in favor of numerical solutions.

The numerical solution will produce values of the concentration at specific times and positions. What happens then is that the equations are solved for a grid of times and positions. Starting with initial conditions, each new solution is based on the solution at the previous grid point. The differentials are approximated by differences and the problem reduces to one of solving the simultaneous difference equations. Many very elegant numerical recipes are used to do this, but none that need concern us here. Instead, we accept the work from decades of research and development in computing and applied math and simply use its powerful results.

9.12 Equations, Initial Conditions, and Boundary Conditions

Consider the following reaction and its occurrence in a transient **PFR**:

$$\mathrm{A} + \mathrm{B} \rightarrow \mathrm{D} \rightarrow \mathrm{E}$$

The first reaction takes place via second-order kinetics **k1 Ca Cb** and the second is first order in **D, k2 Cd**. The concentrations of every species will be a function of both space and time:

$$\mathrm{Ci}[\mathrm{z},\, \mathrm{t}]$$

The differential equations for the concentrations of each species are of the form shown for species **A**:

$$\frac{\partial \mathrm{Ca}}{\partial t} = \frac{-q}{A}\frac{\partial \mathrm{Ca}}{\partial z} - r_{A-}$$

We can write these as a set of equations for **A, B, D**, and **E**, each of which is coupled through their concentrations. We also need four initial conditions for these concentrations. We can set the concentrations of **A** and **B** to their inlet values as if the tube were uniformly filled with them initially. For the two products we can set each of them to zero at time zero at all **z** positions in the reactor. We do this as follows:

```
Creactant[z, 0] == Creactant, o
Cproduct[z, 0] == 0
```

For a partial differential equation we also need to have a set of *boundary conditions*. The initial conditions are for the time differential and the boundary conditions arise for the spatial differential. The boundary conditions are analogous to initial conditions. The boundary conditions must be satisfied at some position or positions for all times. The assignment of proper boundary conditions to physical problems usually becomes the most challenging part of the analysis, but it is also the most interesting! Boundary conditions for our problem can be fairly straightforward: We will let the concentrations of the reactants **A** and **B** be equal to a constant at the inlet for all times. The other concentrations will be zero. Here is how we express that:

```
Ca[0, t] == Cao
Cb[0, t] == Cbo
Cd[0, t] == 0...
```

The first part of this problem is to write the set of component equations, the initial conditions, and the boundary conditions, including the kinetics. Call this set **eqns**. Next write out the set of variables that will be solved for calling **it vars**.

```
eqns = { component equations, initial conditions, boundary conditions}
 vars = {Ci[z, t]....}
```

We make a vertical list of parameter names and values and then we use **NDSolve** with the set of equations and variables as follows:

```
solns = NDSolve[eqns, vars, {t, 0, tmax}, {z, 0, zmax}]
```

When **NDSolve** does the numerical integration it automatically fits a set of polynomials to the numerical values of each variable at each grid point in time and position. Therefore, the output will be an interpolation function. We assign these interpolation functions to function names and **patterns**. We will solve numerically and then plot the concentrations for **A, D**, and **E** in **z**- and **t**-space. As this is unlike the other problems we have done to this point, we will present it in a highly interactive step-by-step fashion. Putting these pieces together into a **Module** or package only makes sense after the computation and the implemented code are understood. Here is the code we have just described. We will let the results for each step flow to output so that we can see how this works in detail:

```
In[257]:= Clear[ca, cb, cd, ce, q, A, k1, k2, cao, cbo, cdo, ceo,
                tmax, zmax, cA, cB, cD, cE]

          General::spell1: Possible spelling error: new symbol name
           "cao" is similar to existing symbol "Cao".

          General::spell1: Possible spelling error: new symbol name
           "cbo" is similar to existing symbol "Cbo".

          General::spell1: Possible spelling error: new symbol name
           "cdo" is similar to existing symbol "Cdo".

          General::stop: Further output of General :: spell1 will be
           suppressed during this calculation.
```

Here are the equations, the initial and boundary conditions, and the variable names:

```
In[258]:= eqns = {
           D[ca[z, t], t] == -q/A D[ca[z, t], z] - k1 ca[z, t] cb[z, t],
           D[cb[z, t], t] == -q/A D[cb[z, t], z] - k1cb[z, t] cb[z, t],
           D[cd[z, t], t] ==-q/A D[cd[z, t], z] + k1ca[z, t] cb[z, t]
                              -k2cd[z, t],
           D[ce[z, t], t] == -q/A D[ce[z, t], z] + k2cd[z, t],
           ca[0, t] == cao,
           cb[0, t] == cbo,
           cd[0, t] == cdo,
           ce[0, t] == ceo,
           ca[z, 0] == cao,
           cb[z, 0] == cbo,
           cd[z, 0] == 0.0,
           ce[z, 0] == 0.0}
          vars = {ca[z, t], cb[z, t], cd[z, t], ce[z, t]};
```

Out[258]= $\{ca^{(0,1)}[z,t] == -k1\, ca[z,t]cb[z,t] - \frac{q\, ca^{(1,0)}[z,t]}{A}$,

$ca^{(0,1)}[z,t] == -k1\, cb[z,t]^2 - \frac{q\, cb^{(1,0)}[z,t]}{A}$,

$ca^{(0,1)}[z, t] == k1\, ca[z, t]\, cb[z, t] - k2\, ca[z, t] - \frac{q\, cd^{(1,0)}[z,t]}{A}$,

```
ca(0,1)[z,t] == k2 ca[z,t] - q ce(1,0)[z,t]/A ,ce[0,t] == cao,
cb[0,t] == cbo, cd[0,t] == cdo, ce[0,t] == ceo,
ca[z,0] == cao, cb[z,0] == cbo, cd[z,0] == 0.,
 ce[z,0] == 0.}
```

To solve this numerically we need the following parameter values:

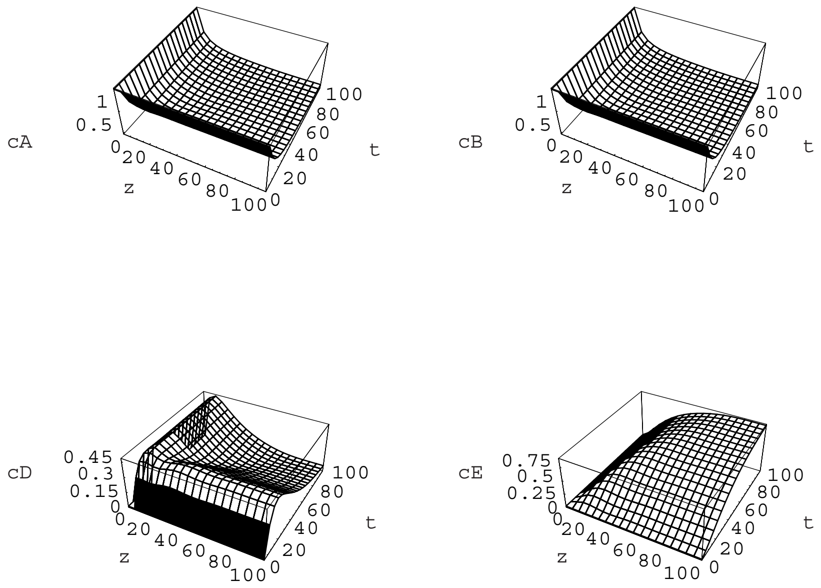
```
In[260]:= q = 7.5;
          A = 10;
          k1 = 0.15;
          k2 = 0.04;
          cao = 1;
          cbo = 1;
          cdo = 0;
          ceo = 0;
          tmax = 100;
          zmax = 100;
```

Next the equations and variables are placed within NDSolve and solved over a range of positions (*z*-values) and times. Then we assign the resultant interpolation functions to the appropriate function names:

```
In[280]:= solns = NDSolve[eqns, vars, {z, 0, zmax},{t, 0,tmax}];

In[281]:= cA[z_, t_]:= Evaluate[ca[z, t] /. solns[[1]]]
          cB[z_, t_]:= Evaluate[cb[z, t] /. solns[[1]]]
          cD[z_, t_]:= Evaluate[cd[z, t] /. solns[[1]]]
          cE[z_, t_]:= Evaluate[ce[z, t] /. solns[[1]]]
```

Finally, we use the newly defined functions in graphical routines as shown in the following, which are now three dimensional in order to provide us with surfaces of points that make up the solutions to this problem for each species:

```
In[286]:= a = Plot3D[cA[z, t], {z, 0, zmax}, {t, 0, tmax},
              ColorOutput  → GrayLevel,
              AxesLabel → {"z "," t", "cA "}, ViewPoint → {1, -2, 2},
              PlotPoints  → 20,
              Ticks  → {Automatic, Automatic, {0, 0.5, 1}},
              DisplayFunction  → Identity];

          b = Plot3D[cB[z, t], {z, 0, zmax}, {t, 0, tmax},
              ColorOutput  → GrayLevel,
              AxesLabel  → {"z "," t", "cB "},
              ViewPoint  → {1, -2, 2}, PlotPoints  → 20,
            Ticks  → {Automatic, Automatic, {0, 0.5, 1}},
            DisplayFunction  → Identity];
        d = Plot3D[cD[z, t], {z, 0, zmax}, {t, 0, tmax},
            ColorOutput  → GrayLevel,
            AxesLabel  → {"z "," t", "cD "},
            PlotPoints  → 20,
            Ticks  → {Automatic, Automatic, {0, 0.15, .30, .45}},
            DisplayFunction  → Identity];

        e = Plot3D[cE[z, t], {z, 0, zmax}, {t, 0, tmax},
            ColorOutput  → GrayLevel,
            AxesLabel  → {"z "," t", "cE "},
            PlotPoints  → 20,
            Ticks  → {Automatic, Automatic, {0, 0.25, .5, .75}},
            DisplayFunction  → Identity];

        Show[GraphicsArray[{{a, b}, {d, e}}],
         DisplayFunction  → $DisplayFunction];
```



The graphs of each of the species concentrations are plotted as a function of position along the tube **z** and time **t**. At the edges of the graphs for the concentrations of **A** and **B** we see the boundary and initial conditions. All values are unit or zero concentration as we had specified. As we move through time, we see the concentrations of both species drop monotonically at any position. Furthermore, if we take any time slice, we see that the concentrations of reactants drop exponentially with position—as we know they should. At the longer times the profiles of

A and **B** through the reactor have reached their steady-state values. Species **D** is intermediate in the network. Its concentration rises sharply as a function of time and of position, maximizing across the reactor at short time and then falling to its steady-state value. The only region of the reactor in which the concentration of **D** persists at relatively high levels is at a position approximately 10 units from the entrance. As time goes on the initially high **D** concentration beyond position $\mathbf{z \approx 10}$ falls. Species **E** is the ultimate product of this network. At the lower left corner of the CE concentration graph, the concentration is zero. Through time and position this rises to a steady-state level of nearly 0.8 in the upper right corner of the plot (which represents the reactor outlet after much time has passed).

All in all, the series network in a transient **PFR** appears to follow the trends that we would expect in evolving to the steady-state condition. The time to reach the steady state is a function of the rate constants, the inlet concentrations, and the holding time, that is, of the volume flow rate and reactor volume.

9.13 Summary

This chapter has been devoted to continuous reactors and their analyses. We have examined the powerful idealizations of the **CSTR** and **PFR**. Pseudo-steady states and steady states have been covered as well as chemical equilibrium. We must remember that the steady-state condition can be far from the equilibrium condition, and the two time-independent situations must not be confused. The time-dependent **CSTR** and **PFR** are interesting problems, but they are less often used than the steady-state solutions.

We have seen how the kinetics fit into a reactor equation as a constitutive relationship. Flow and reaction come together in these systems to affect the rate of accumulation. Hence when we refer to the "rate" we must be careful to be specific about the reactor—if it is a constant volume batch reactor, then rate means the chemical rate. If the reactor is a transient **CSTR** or **PFR**, then the rate of change of the concentration at the exit of the reactor is not the chemical rate alone. Mixing effects are important and we have seen how to begin to account for the fluid mechanics in a reactor through the empirical measure of the r.t.d. The r.t.d. does affect the outcome from the reactor, but the sensitivity to the r.t.d. depends upon the kinetics and their functional form.

Finally, we have taken our *Mathematica* skills up another level by writing not just **Module** functions, but actual **Packages**. By writing **Packages** in the **Global** context we implement them immediately. We can, however, write and save packages to another context and then call them from our own library as we need them, either as stand-alone computations or as embedded functions in another package.

CHAPTER 10

Worked Problems

The following are exercises that have been used with honors students and that seem to be both interesting and challenging. In many cases only the rudimentary coded solution is given. From this one can build much more sophisticated code and in many cases create Module functions that can be used repeatedly with different parameter values. Note that all of these separate problems were run as if the Kernel had to be restarted with each computation. Running sequentially without adding statements to clear variables will result in absurd output.

10.1 The Level-Controlled Tank

Introduction

The filling or draining of a tank is a relatively simple situation to model. If the tank has both input and output then it is a combination of the two previous cases and the differential equation for the rate of change of the level in the tank is as follows:

$$\frac{\mathrm{dh}[t]}{\mathrm{dt}} = -\frac{(\mathrm{qf} - q)}{A}$$

When the input flow rate is equal to the output flow rate, the system is at steady state and the level remains constant because $\frac{\mathrm{dh}[t]}{\mathrm{dt}} = 0$.

A tank could be operated in such a way that the inlet was always equal to the outlet when the two were at their designed magnitudes. Thus if q* was the designed flow rate in a CSTR, for example, then the system could be maintained at some specific design level h* and design volume V*. As long as the inlet or outlet rates did not change, the system would remain at this condition. From our study of kinetics, we know that maintaining V* as a constant would be desirable in order to keep the holding time $\tau = \frac{V^*}{q}$ constant and to remain at the design level of production.

Perturbation of the Inlet Flow Rate and Control

Suppose that the inlet flow rate was mostly a constant but from time to time it suffered an upset. The upset would either increase the inlet flow or decrease it. If this were to occur, then the level in the tank would either increase or decrease, unless there was some attempt made to change the outlet flow rate. A simple control algorithm would be one in which the exit flow rate is adjusted automatically when there is an upset, either above or below the design flow rate. To do this analysis we need to specify the upset and the system's response to it.

Let the upset be some additional flow rate over or below the design value. Stated in simple mathematical terms:

$$\mathrm{qf} = \mathrm{q}^* + \mathrm{qp}$$

where q* is the design magnitude and qp is either positive or negative. We will say more about the upset flow rate momentarily, but first we will describe the controlled outlet flow rate. The outlet flow rate is typically at the setpoint q* until the upset qp takes place. At that point the flow rate must be adjusted to compensate the change in the inlet flow rate. We describe this by setting the adjusted flow rate equal to the set point plus a new flow rate that is proportional to the difference between the set point level and the actual level at that time:

$$\mathrm{q} = \mathrm{q}^* - \mathrm{K}(\mathrm{h}^* - \mathrm{h(t)})$$

For this to be dimensionally consistent, it is clear that the proportionality constant must have dimensions of area per time so that its product with h* − h(t) is in dimensions of volume per time. During the upset we are discussing here, the system responds transiently, that is, it goes away from steady state. The governing equation is:

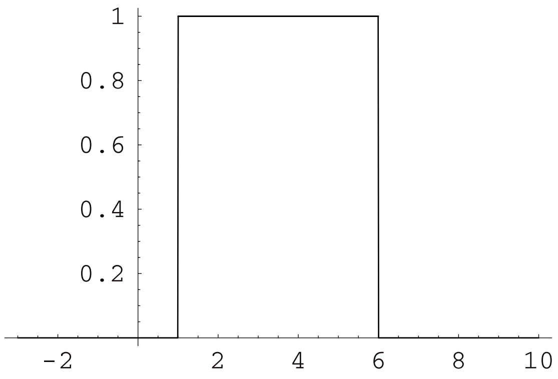
$$\frac{\mathrm{dh}(t)}{\mathrm{dt}} = \frac{q^* + \mathrm{qp} - (q^* - K(h^* - h(t)))}{A}$$
$$\frac{\mathrm{dh}(t)}{\mathrm{dt}} = \frac{\mathrm{qp} + K(h^* - h(t))}{A}$$

This equation is the one that must be integrated and to do so we must know how the upset behaves in time. Before the upset from t = 0 until some time just below t1, qp is zero and h* = h(t):

$$0 < t < t1 \quad \frac{dh(t)}{dt} = 0 \Rightarrow h(t) = h^*$$

The upset begins at time t1, and it instantaneously rises to a value of qp. The upset stays at the level qp from t1 until a time just less than t2. At time t2 it falls instantaneously back to zero. This kind of function is called a UnitStep. For the sake of making a graph of this type of disturbance, we plot the UnitStep function and 1 − UnitStep function using the parameters 1 and 6 in order to have a pulse of unit height that is five time-units wide:

```
In[2064]:= << Calculus‘DiracDelta‘
           a = Plot[UnitStep[x - 1], {x, -3, 6},
            DisplayFunction → Identity];
           b = Plot[1 - UnitStep[x - 6], {x, 10, 1},
            DisplayFunction → Identity];
           Show[a, b, DisplayFunction → $DisplayFunction];

           DiracDelta::obslt :
            All DiracDelta and UnitStep functionality is now
            autoloaded. The package Calculus`DiracDelta` is
            obsolete.
```



In general the height of the disturbance is qp and its width is t2 − t1. The question now becomes one of how to do the integrations during and after the disturbance.

Integration Through and Beyond the Disturbance

We must now integrate this equation from t1 to t:

$$\frac{\mathrm{dh2}(t)}{\mathrm{dt}} = \frac{\mathrm{qp} + K(h^* - h2(t))}{A}$$

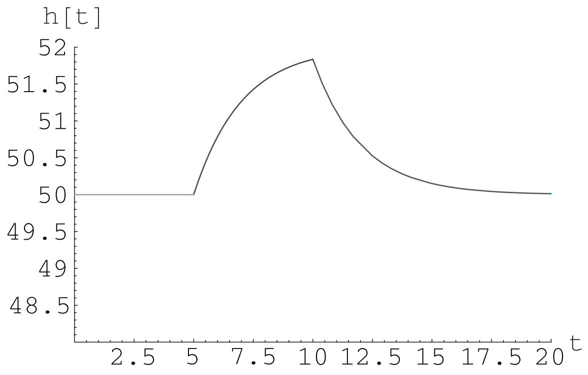
The initial condition is h2(t1) = h* and we should integrate to any t less than t2, so we integrate to t.

When this is done we have an expression for h(t) during the course of the disturbance. We must next integrate after the disturbance is complete, that is, from t2 out to any additional time t. Here we have the following equation to work with because qp is zero:

$$\frac{\mathrm{dh3}(t)}{\mathrm{dt}} = \frac{K(h^* - h3(t)}{A}$$

The initial condition for this period is h3[t2] = h2[t2]! Therefore, we must evaluate the constant of integration very carefully.

Finally, we know that if the disturbance is positive, then the inlet flow increases and the tank level should rise. If there were no control it would rise and stay at a new higher level. With control it should rise and then fall back to the control or design level. Tracking the change in level versus time we should see a sawtooth that looks like this:



Problem Statements

A) Find the expression for h2[t] by using DSolve. Evaluate the constant of integration using the initial condition that h2[0] => h2[t1] = hd:

$$\mathrm{DSolve}\left[\mathrm{h2}'[\mathrm{t}] == \frac{\mathrm{qp} + \mathrm{K}(\mathrm{hd} - \mathrm{h2}[\mathrm{t}])}{\mathrm{A}}, \mathrm{h2}[\mathrm{t}], \mathrm{t}\right]$$

B) Find the expression for h3[t] using DSolve. Evaluate the constant of integration using the initial condition that h3[0] => h3[t2] = h2[t2].

$$\text{DSolve}\left[\text{h3}'[\text{t}] == \frac{\text{K}(\text{hd} - \text{h3}[\text{t}])}{\text{A}}, \text{h3}[\text{t}], \text{t}\right]$$

C) Create functions for h2[t] and h3[t], and given the following parameter values, Plot, h1, h2, and h3 consecutively in time.

hd = 50;
K = 5;
A = 10;
qp = 10;
t1 = 5;
t2 = 10;
tmax = 30;

Solution to Part A

Find a general solution to the differential equation:

In[1]:= **Clear[h2, qp, hd, K, A, t1, t2]**

$$\textbf{DSolve}\left[\{\textbf{h2}'[\textbf{t}] == \frac{\textbf{qp + K(hd - h2[t])}}{\textbf{A}}\}, \textbf{h2[t], t}\right]$$

Out[2]= $\{\{\text{h2[t]} \rightarrow \frac{\text{hd K + qp}}{\text{K}} + e^{-\frac{\text{Kt}}{\text{A}}}\text{C[1]}\}\}$

Evaluate the constant of integration at the initial condition h2[t1] = hd using Solve:

In[3]:= $\textbf{Solve}[\textbf{hd} == \frac{\textbf{hd K + qp}}{\textbf{K}} + e^{-\frac{\textbf{Kt1}}{\textbf{A}}}\textbf{C[1], C[1]}]$

Out[3]= $\{\{\text{C[1]} \rightarrow -\frac{e^{\frac{\text{Kt1}}{\text{A}}}\text{qp}}{\text{K}}\}\}$

Define the constant C1 and then replace it and simplify:

In[4]:= $\textbf{C1} = -\frac{e^{\frac{\textbf{Kt1}}{\textbf{A}}}\textbf{qp}}{\textbf{K}};$

$\textbf{Collect}[\textbf{Simplify}[\frac{\textbf{hd K + qp}}{\textbf{K}} + e^{-\frac{\textbf{Kt}}{\textbf{A}}}\textbf{C1], qp}]$

Out[5]= $\mathrm{hd} + \frac{(1 - e^{\frac{K(-t+t1)}{A}})\mathrm{qp}}{K}$

Now define a function for h2[t] to use in parts B and C:

In[6]:= **h2[t_] := hd +** $\left(\frac{1}{K} - \frac{e^{\frac{K(-t+t1)}{A}}}{K}\right)\mathrm{qp}$

Solution to Part B

Find the general solution to the differential equation defining the rate of change in the level of the tank after t2:

In[7]:= **Clear[hd, h3]**

DSolve[{h3'[t] == $\frac{K(\mathrm{hd} - \mathrm{h3}[t])}{A}$**}, h3[t], t]**

Out[8]= {{h3[t] → hd + $e^{-\frac{Kt}{A}}$ C[1]}}

The initial condition in this case is given as h3[0] = h3[t2] = h2[t2]. We find this by substituting h2[t] into the initial condition equation and evaluating C again:

In[9]:= **h2[t2]**

Out[9]= $\mathrm{hd} + \left(\frac{1}{K} - \frac{e^{\frac{K(t1-t2)}{A}}}{K}\right)\mathrm{qp}$

In[10]:= **Solve[h2[t2] == hd +** $e^{-\frac{Kt2}{A}}$ **C[1], C[1]]**

Out[10]= {{C[1] → $-\frac{e^{\frac{Kt2}{A}}(-1 + e^{\frac{K(t1-t2)}{A}})\mathrm{qp}}{K}$}}

Replacing the expression for C into that derived for h3[t], we find the expression for h3[t]:

In[11]:= **Simplify[hd +** $e^{-\frac{Kt}{A}}\left(-\frac{e^{\frac{Kt2}{A}}(-1 + e^{\frac{K(t1-t2)}{A}})\mathrm{qp}}{K}\right)$**]**

Out[11]= $\mathrm{hd} - \frac{e^{\frac{K(-t+t2)}{A}}(-1 + e^{\frac{K(t1-t2)}{A}})\mathrm{qp}}{K}$

Anticipating the need for this function, we define it:

In[12]:= h3[t_] := $\dfrac{\text{hd}\,K + \left(-e^{\frac{K(-t+t1)}{A}} + e^{\frac{K(-t+t2)}{A}}\right)\text{qp}}{K}$

Solution to Part C — Graphs of h1[t],h2[t],h3[t]

The code that follows works with the three functions we have derived for the three intervals of time-dependent behavior. After specifying the three functions, we list the parameter values. Each has a semicolon at the end in order to prevent its value from being echoed back to the monitor. Note that t1 and t2 are constants that must be specified.

The first plot is made with the following command:

Plot[{h1[t], h2[t], h3[t]}, {t,0,50}];

The syntax involves specifying the functions to be plotted first and the time interval over which they should be plotted next. The details show that the functions are implemented as a set included within the curly brackets {} as is the time range. The disadvantage of this approach is that all functions are plotted over the whole time range, even though they only apply to separate intervals of time. To overcome this problem we use the three code lines that follow. These have the following format:

pl1 = Plot[h1[t], {t,0,t1}, DisplayFunction→Identity, PlotStyle→Hue[.4]];

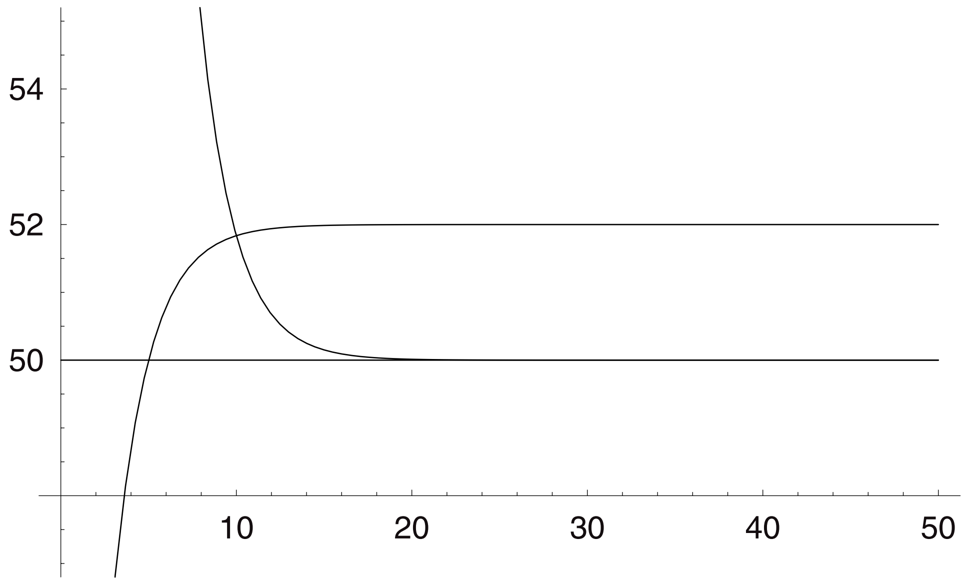
The command structure is nearly the same as that used before with some notable differences. We include only one function in each command—in this case h1[t]. Then we specify the interval we want to plot this function over; for h1 it is from t = 0 to t = t1. For h2[t] we set the interval to be from t1 to t2 and for h3[t] from t1 to t. Next, we set the DisplayFunction to Identity. This surprises the output of the graphics but saves them in the plot called "pl1." In order to distinguish between these three plots we change their color. This is done by setting PlotStyle to Hue[0.4] (Hue can have a value between 0 and 1). We use different value for the three different plots. Finally we call each of the plots in the show command pl1, pl2, and pl3, and we set DisplayFunction to $DisplayFunction. This makes one plot from the three separate plots and we have no overlapping of the functions.

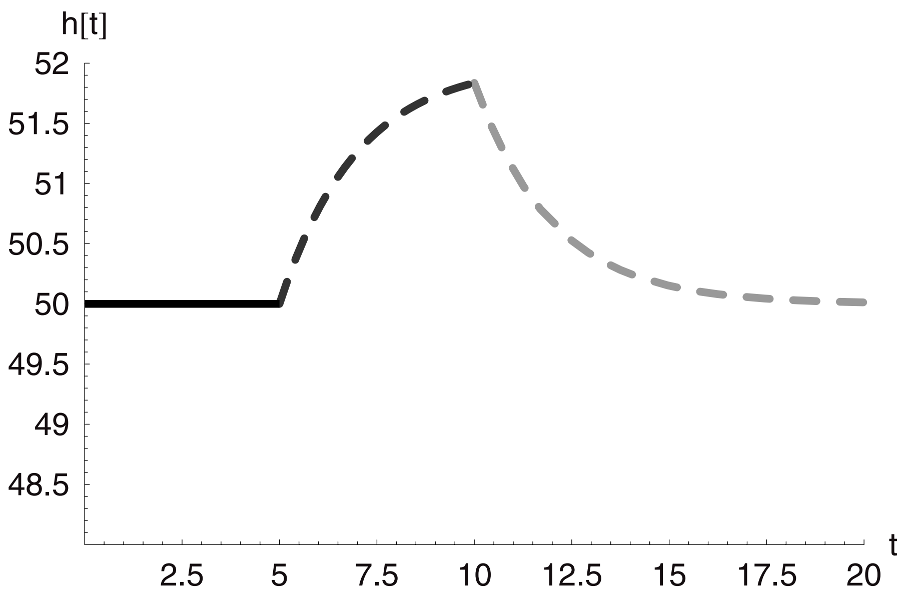
```
In[13]:= SetOptions[{Plot}, DefaultFont → {"Helvetica", 12}];
         h1[t_] := hd
```

$$\mathtt{h2[t\_]} := \mathtt{hd} + \left(\frac{1}{\mathtt{K}} - \frac{e^{\frac{\mathtt{K}(-\mathtt{t}+\mathtt{t1})}{\mathtt{A}}}}{\mathtt{K}}\right)\mathtt{qp}$$

$$\mathtt{h3[t\_]} := \frac{\mathtt{hd}\,\mathtt{K} + \left(-e^{\frac{\mathtt{K}(-\mathtt{t}+\mathtt{t1})}{\mathtt{A}}} + e^{\frac{\mathtt{K}(-\mathtt{t}+\mathtt{t2})}{\mathtt{A}}}\right)\mathtt{qp}}{\mathtt{K}}$$

```
hd = 50;
K = 5;
A = 10;
qp = 10;
t1 = 5;
t2 = 10;
tmax = 30;
Plot[{h1[t], h2[t], h3[t]}, {t, 0, 50}];
pl1 = Plot[h1[t], {t, 0, t1}, DisplayFunction → Identity,
   PlotStyle → {{Thickness[.01], GrayLevel[0.0]}}];
pl2 = Plot[h2[t], {t, t1, t2}, DisplayFunction → Identity,
   PlotStyle → {{Thickness[.01], GrayLevel[0.2],
   Dashing[{0.03, 0.03}]}}];
pl3 = Plot[h3[t], {t, t2, tmax}, DisplayFunction → Identity,
   PlotStyle → {{Thickness[.01], GrayLevel[0.6],
   Dashing[{0.03, 0.03}]}}];
Show[pl3, pl2, pl1, DisplayFunction → $DisplayFunction,
   PlotRange → {{0, 20}, {48, 52}},
   AxesLabel → {"t", "h[t]"}];
```





10.2 Batch Competitive Adsorption

Introduction

Adsorption of an impurity onto a porous solid such as activated carbon, alumina, or silica is often used to purify gases and liquids. Adsorption usually is reversible, but if the heat of adsorption is high then the tendency to desorb may be low. Typically adsorption is done in a continuous process. It also may be done in a batch process for small-scale separations or to determine the parameters that control the adsorption process for a given adsorbate (the adsorbing molecule) and a given adsorbent (the porous solid).

In this problem we will simulate a batch adsorption process that takes place with two adsorbate components. The simulation will allow us to do computational experiments with the aim of learning how the adsorption and desorption parameters affect the behavior of this process. Building the simulation will provide new experience in developing the model equations, utilizing more complex constitutive relationships, finding numerical solutions to these equations, and displaying the results graphically.

Adsorption/Desorption

Adsorption Sites

Adsorption and desorption can be considered to be analogous to a reversible chemical reaction. By way of this analogy there must be a forward rate corresponding to adsorption and a reverse rate for desorption. The net rate of adsorption is the difference between these two rates.

When a molecule descends to a solid surface and comes to "rest" we consider this an adsorption event. The time a molecule spends in this state may be very short (10^{-13} sec) or it may be long (>hours). Because molecules have real bulk, volume, and dimensions, when they rest at the surface they occupy some area. Thinking of a flat plane as the surface, then the cross-sectional area (shadow area) of the molecule is the area of the surface that is occupied. The locus of points beneath this molecule can be termed the "adsorption site." The area of the surface divided by the area of the site gives the theoretical number of sites present at the surface:

$$N_{\text{sites}} = \frac{A_{\text{Surface}}}{A_{\text{site}}}$$

Dividing this number by Avogadro's number L and the volume occupied by the surface, that is, by the volume of the high surface area solid V_s gives the concentration of adsorption sites:

$$C_{\text{Sites}} = \frac{N_{\text{sites}}}{V_s L}$$

Rates of Adsorption and Desorption

The rate of adsorption is proportional to the concentration of the adsorbate in the bulk phase (gas or solid) surrounding the solid and the difference between the total concentration of sites in the adsorbent phase (porous solid) and the number of sites already occupied by the adsorbate molecules:

$$r_{\text{adsorption}} \alpha C_A^{\text{Bulk}} \left(C_{\text{Sites}} - C_A^{\text{Surface}}\right)$$
$$r_{\text{adsorption}} = k_{A,,\text{ads}} C_A^{\text{Bulk}} \left(C_{\text{Sites}} - C_A^{\text{Surface}}\right)$$

The proportionality constant is the adsorption rate constant for the species A on this particular solid.

The rate of desorption of a given molecule is proportional to that molecule's concentration on the surface:

$$r_{\text{desorption}} \alpha C_A^{\text{Surface}}$$
$$r_{\text{desorption}} = k_{A,,\text{des}} C_A^{\text{Surface}}$$

The surface concentrations are given in a way that is analogous to the adsorption site concentration, that is, as the number of moles of the adsorbed species per volume of the adsorbent solid.

Competitive Adsorption

If two adsorbate molecules A and B compete for the same sites then the adsorption and desorption rate expressions are:

$$r_{\text{adsorption},A} = k_{A,,\text{ads}} C_A^{\text{Bulk}} \left(C_{\text{Sites}} - C_A^{\text{Surface}} - C_B^{\text{Surface}} \right)$$
$$r_{\text{adsorption},B} = k_{B,,\text{ads}} C_B^{\text{Bulk}} \left(C_{\text{Sites}} - C_A^{\text{Surface}} - C_B^{\text{Surface}} \right)$$
$$r_{\text{desorption}} = k_{A,,\text{des}} C_A^{\text{Surface}}$$
$$r_{\text{desorption}} = k_{B,,\text{des}} C_B^{\text{Surface}}$$

Note that the competition for the sites is accounted for only in the adsorption rate term. This term recognizes that the surface is occupied by two species and to the extent that this happens simultaneously, the rate of adsorption of either species is diminished by the presence of the other at the surface, just as it is diminished by its own occupancy of the surface.

Problem Statement

1. Set up the material balance equations for the competitive adsorption of A and B on an adsorbent phase. Since there are two phases and two components there must be four component equations.

2. Using the constitutive kinetics given, show explicitly that the equations are dimensionally consistent. To be consistent what must the dimensions of the adsorption and desorption rate constants be?

3. Develop a simulation for this system utilizing NDSolve to integrate the equations and Plot to display the time-dependent behavior of the four concentrations in one plot. Set up the simulation with the equations and initial conditions given first as a set. The variables also are given as a set. These are followed by the parameters in a vertical list so that they can be easily changed to test behavior. Next use NDSolve to find the numerical solutions to the equations. Assignment of the interpolation functions to a series of four functions is done next. Finally a Plot routine is implemented. The skeleton of the simulation then should be:

```
Eqns = {eqn 1, eqn 2, eqn 3, eqn 4, initial conditions (1,2,3,4)}
Parameters;
Vars = {Ci[t]...Cl[t]}
solns = NDSolve[Eqns, Vars, {t,0,tmax}]
Cin[t_] := Evaluate[Ci[t]/.solns]...Cln[t] := Evaluate[Cl[t]/.solns]
Plot[{Cin[t]...Cln[t]},{t,0,tmax},...]
```

4. Use this simulation to examine the behavior of this seemingly simple set of equations by varying the parameters according to the following matrix. After each simulation save the graphical output by copying the graph and pasting it to a new bracket in your notebook, to a new notebook, or to a Word document.

Sim. No.	$C_{A,o}^{\text{Bulk}}$	$C_{A,o}^{\text{Surface}}$	$C_{B,o}^{\text{Bulk}}$	$C_{B,o}^{\text{Surface}}$	C_{Sites}	k_{Ads}^{A}	k_{Des}^{A}	k_{Ads}^{B}	k_{Des}^{B}	tmax
1	1	0	1	0	1	1	1	1	1	10
2	1	0	1	0	1	10	10	1	1	10
3	1	0	1	0	1	100	100	1	1	5
4.a	1	0	1	0	1	1	10	.1	.001	10
4.b	″	″	″	″	″	″	″	″	″	100
5	1	0	1	0	10	.01	.001	.1	.00001	50

5. How could you write a piece of code using Module to accomplish all of this and require only the simulation parameters and number?

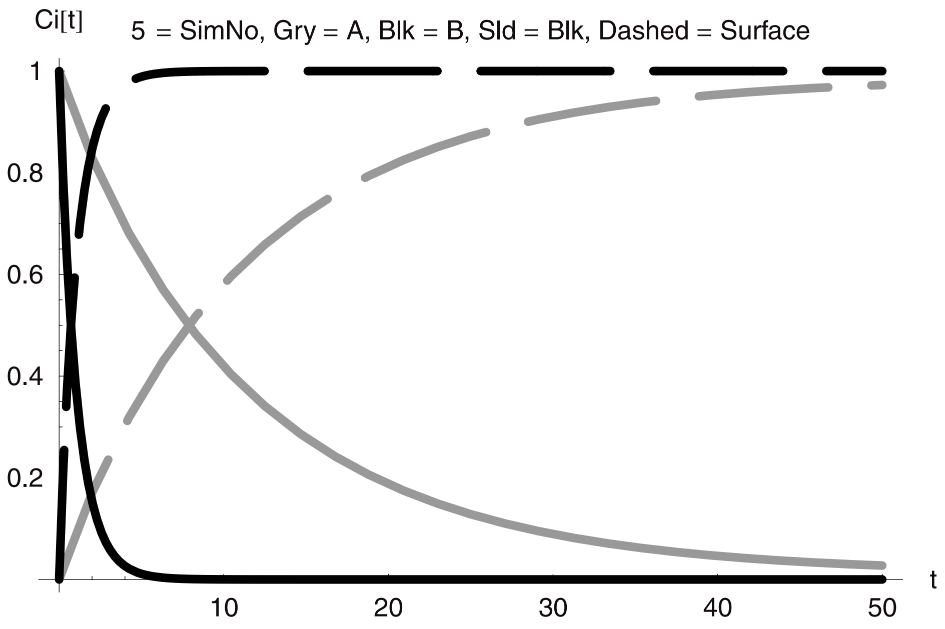
Solution

Here is an example of how this can be handled for the case of the fifth set of parameters:

```
In[1]:= SetOptions[{Plot}, DefaultFont → {"Helvetica", 10}];

        eqnsa = {Cab'[t] == -kaa Cab[t](Cs - Cas[t] - Cbs[t])
                  + kad Cas[t],
                 Cas'[t] == +kaa Cab[t] (Cs - Cas[t] - Cbs[t])
                  - kad Cas[t],
                 Cbb'[t] == -kba Cbb[t](Cs - Cas[t] - Cbs[t])
                  + kbd Cbs[t],
                 Cbs'[t] == +kba Cbb[t](Cs - Cas[t] - Cbs[t])
                  - kbd Cbs[t],
            Cab[0] == Cabo, Cas[0] == Caso, Cbb[0] == Cbbo,
             Cbs[0] == Cbso};

        vars = {Cab[t], Cas[t], Cbb[t], Cbs[t]};

        tmax = 50;

        n = 5;

        Cabo = 1;
        Cbbo = 1;
        Caso = 0;
        Cbso = 0;

        kaa = .01;
        kad = .001;
        kba = .1;
        kbd = .00001;

        Cs = 10;

        solns = NDSolve[eqnsa, vars, {t, 0, tmax}];
```

```
Cabn[t_] := Evaluate[Cab[t] /. solns]
Casn[t_] := Evaluate[Cas[t] /. solns]
Cbbn[t_] := Evaluate[Cbb[t] /. solns]
Cbsn[t_] := Evaluate[Cbs[t] /. solns];

n "= Simul. No."
tmax "= tmax"
kaa "= kaa"
kad "= kad"
kba "= kba"
kbd "= kbd"

Plot[{Cabn[t], Casn[t], Cbbn[t], Cbsn[t]}, {t, 0, tmax},
        PlotStyle → {{GrayLevel[0.6], Thickness[.01]},
        {Dashing[{0.15, 0.05}], GrayLevel[0.6],
         Thickness[.01]},
        {GrayLevel[0], Thickness[.01]},
         {Dashing[{0.15, 0.05}], GrayLevel[0],
         Thickness[.01]}}, AxesLabel → {"t", "Ci[t]"},
        PlotLabel → "n"= SimNo, Gry = A, Blk = B,
         Sld = Blk, Dashed = Surface"];

General::spell1 : Possible spelling error: new symbol
 name "Cabo" is similar to existing symbol "Cab".

General::spell : Possible spelling error: new symbol name
 "Caso" is similar to existing symbols {Cabo, Cas}.

General::spell : Possible spelling error: new symbol name
 "Cbbo" is similar to existing symbols {Cabo, Cbb}.

General::spell : Possible spelling error: new symbol name
 "Cbso" is similar to existing symbols {Caso, Cbbo, Cbs}.

General::stop : Further output of General::spell will be
 suppressed during this calculation.

General::spell : Possible spelling error: new symbol name
 "Cabn" is similar to existing symbols {Cab, Cabo}.

General::spell : Possible spelling error: new symbol name
 "Casn" is similar to existing symbols {Cabn, Cas, Caso}.

General::spell : Possible spelling error: new symbol name
 "Cbbn" is similar to existing symbols {Cabn, Cbb, Cbbo}.

General::spell : Possible spelling error: new symbol name
 "Cbsn" is similar to existing symbols {Casn, Cbbn, Cbs,
 Cbso}.
```

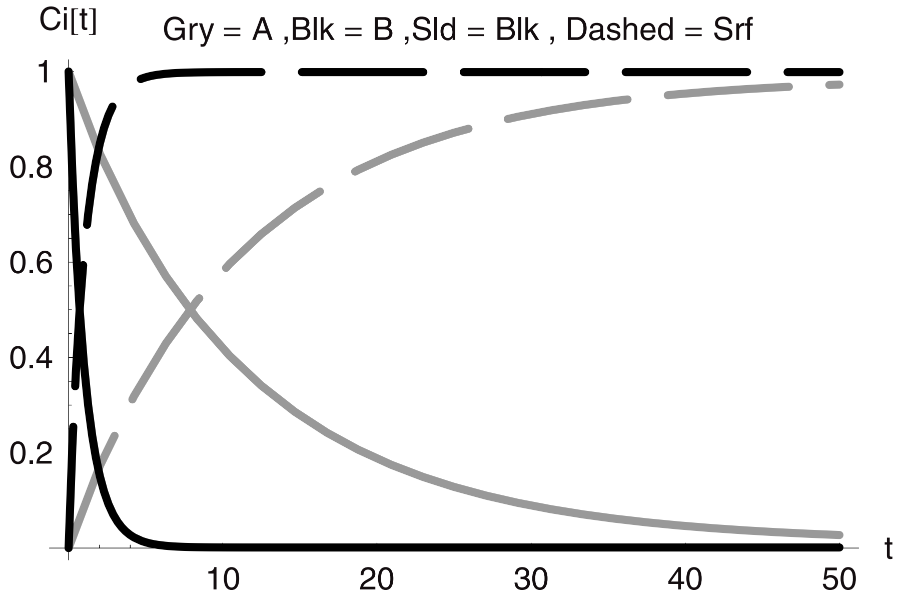
```
Out[20]= 5 = Simul. No.

Out[21]= 50 = tmax

Out[22]= 0.01 = kaa

Out[23]= 0.001 = kad

Out[24]= 0.1 = kba

Out[25]= 0.00001 = kbd
```



The following creates a Module function from the code provided in the preceding text.

```
In[27]:= adsdes[Cabo_, Cbbo_, Caso_, Cbso_, kaa_, kad_, kba_,
           kbd_, tmax_, n_] :=
          Module[{eqnsa, vars, solns, Cabn, Casn, Cbbn, Cbsn,
           Cab, Cas, Cbb, Cbs, t},
         SetOptions[{Plot}, DefaultFont → {"Helvetica", 12}];

         eqnsa = {Cab'[t] == -kaa Cab[t](Cs - Cas[t] - Cbs[t])
                    + kad Cas[t],
                   Cas'[t] == +kaa Cab[t](Cs - Cas[t] - Cbs[t])
                    - kad Cas[t],
                   Cbb'[t] == -kba Cbb[t](Cs - Cas[t] - Cbs[t])
                    + kbd Cbs[t],
                   Cbs'[t] == +kba Cbb[t](Cs - Cas[t] - Cbs[t])
                    - kbd Cbs[t],
              Cab[0] == Cabo, Cas[0] == Caso, Cbb[0] == Cbbo,
                Cbs[0] == Cbso};

     vars = {Cab[t], Cas[t], Cbb[t], Cbs[t]};

     solns = NDSolve[eqnsa, vars, {t, 0, tmax}];

     Cabn[t] = Evaluate[Cab[t] /. solns];
     Casn[t] = Evaluate[Cas[t] /. solns];
     Cbbn[t] = Evaluate[Cbb[t] /. solns];
     Cbsn[t] = Evaluate[Cbs[t] /. solns];

      Print["Simulation Number ="n];

     Plot[{Cabn[t], Casn[t], Cbbn[t], Cbsn[t]}, {t, 0, tmax},
           PlotStyle → {{GrayLevel[0.6], Thickness[.01]},
            {Dashing[{0.15, 0.05}], GrayLevel[0.6],
            Thickness[.01]}, {GrayLevel[0], Thickness[.01]},
            {Dashing[{0.15, 0.05}], GrayLevel[0],
            Thickness[.01]}},
           AxesLabel → {"t", "Ci[t]"},
         PlotLabel → "Gry = A, Blk = B, Sld = Blk,
         Dashed = Srf"]
     ]
```

```
In[28]:= adsdes[1, 1, 0, 0, .01, .001, .1, .001, 50, x]
         Simulation Number = x
```



```
Out[28]= - Graphics -
```

10.3 A Problem in Complex Kinetics

Introduction

Most reactions occur while other reactions are also taking place simultaneously. Very often the products of one reaction are the reactants for the next. Similarly, a reactant may be involved in more than one reaction. When reactions occur in a sequence, this is referred to as a series network. An example would be:

$$A \rightleftarrows B \rightleftarrows D \rightleftarrows E$$

Alternatively, reactions may take place in parallel:

$$\begin{gathered} A \rightleftarrows B \\ A \rightleftarrows D \end{gathered}$$

Additions of these two simple cases can lead to series parallel networks of chemical reaction. When we encounter a problem like this one, we have to handle the kinetics carefully. This is just what we will do in the case of this problem.

Parallel and Series Reversible Reactions

Consider the case of a reversible reaction whose products lead to another product:

$$\begin{aligned} &A + B \rightleftarrows D + E \\ &r_{\text{net},1} = \text{k1 Ca Cb} - \text{k2 Cd Ce} \\ &D + E \rightarrow F \\ &r_{\text{net},2} = \text{k3 Cd Ce} \\ &A + A \rightleftarrows G \\ &r_{\text{net},3} = \text{k4 Ca}^2 - \text{k5 Cg} \end{aligned}$$

The net rate of the first reversible reaction can be given as $r_{\text{net},1} = \text{k1 Ca Cb} - \text{k2 Cd Ce}$. The second reaction is in series with the first and we find it has kinetics that are given by $\text{r} = \text{k3 Cd Ce}$. It is irreversible. The third reaction, A to G, is parallel to that of the first reaction and it too is reversible. This reaction is second order in the forward direction and first order in the reverse direction.

Problem Statements

A) Using NDSolve, set up the model equations for each component assuming that the reactions take place in a batch reactor. The following parameters are those that you will need to solve the equations numerically. Show, using Plot, the change in each concentration as a function of time.

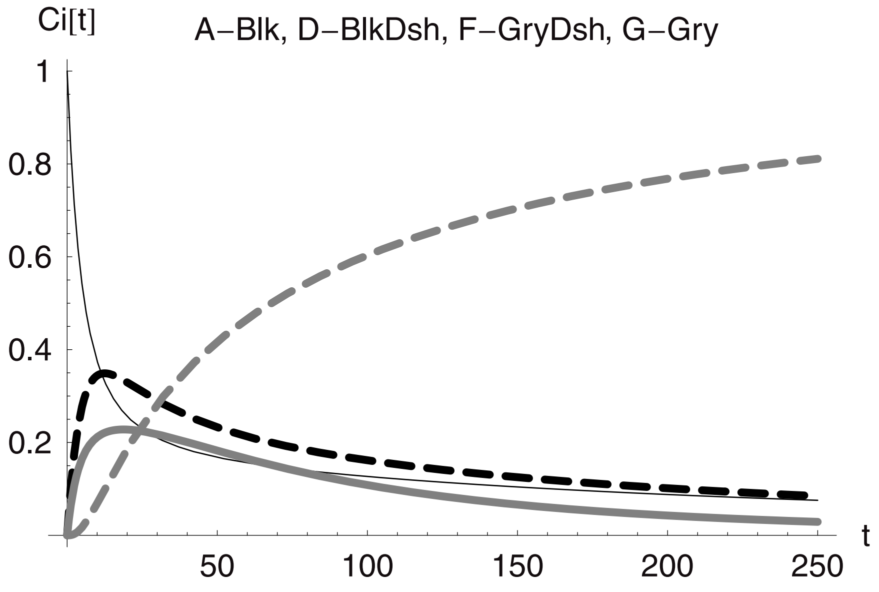
k1 = .1;
k2 = .05;
k3 = .1;
k4 = .065;
k5 = .02;

Cao = 1;
Cbo = 1;
Cdo = 0;
Ceo = 0;
Cfo = 0;
Cgo = 0;
tmax = 250;

B) Having solved the batch case, now set up the same kinetics for the case of a CSTR operated at steady state. Use NSolve to find the optimal flow rate and holding time for the production of D (and E) assuming that the inlet concentrations of A and B are each 1 and that the volume is 100.

Solution

```
In[1]:= SetOptions[{Plot}, DefaultFont → {"Helvetica", 12}];

        k1 = .1;
        k2 = .05;
        k3 = .1;
        k4 = .065;
        k5 = .02;

        Cao = 1;
        Cbo = 1;
        Cdo = 0;
        Ceo = 0;
        Cfo = 0;
        Cgo = 0;
        tmax = 250;

        solns = NDSolve[{ Ca'[t] == -k1 Ca[t] Cb[t] + k2 Cd[t] Ce[t]
                             - k4 Ca[t]^2 + k5 Cg[t],
                          Cb'[t] == -k1 Ca[t] Cb[t] + k2 Cd[t] Ce[t],
                          Cd'[t] == k1 Ca[t] Cb[t] - k2 Cd[t] Ce[t]
                             - k3 Cd[t] Ce[t],
                          Ce'[t] == k1 Ca[t] Cb[t] - k2 Cd[t] Ce[t]
                             - k3 Cd[t] Ce[t],
                    Cf'[t] == k3 Cd[t] Ce[t],
                    Cg'[t] == k4 Ca[t]^2 - k5 Cg[t],

                        Ca[0] == Cao,
                        Cb[0] == Cbo,
                        Cd[0] == Cdo,
                        Ce[0] == Ceo,
                        Cf[0] == Cfo,
                        Cg[0] == Cgo},

                    {Ca[t], Cb[t], Cd[t], Ce[t], Cf[t], Cg[t]},
                     {t, 0, tmax}];

Can[t_] := Evaluate[Ca[t] /. solns]
Cbn[t_] := Evaluate[Cb[t] /. solns]
Cdn[t_] := Evaluate[Cd[t] /. solns]
Cen[t_] := Evaluate[Ce[t] /. solns]
Cfn[t_] := Evaluate[Cf[t] /. solns]
Cgn[t_] := Evaluate[Cg[t] /. solns]

Plot[{Can[t], Cdn[t], Cfn[t], Cgn[t]}, {t, 0, tmax},
  PlotStyle → {GrayLevel[0], {Thickness[.01],
    Dashing[{0.03, 0.02}], GrayLevel[0]},
    {Thickness[.01], Dashing[{0.03, 0.02}], GrayLevel[0.5]},
    {Thickness[.01], GrayLevel[0.5]}},
PlotLabel → "A-Blk, D-BlkDsh, F-GryDsh, G-Gry",
AxesLabel → {"t", "Ci[t]"}];
```



Now we will put together a set of CSTR equations at steady state, fix the volume, and vary the flow rate in order to maximize the formation of D (and E).

```
In[22]:= Clear[Ca, Cb, Cd, Ce, Cf, Cg, t, solns, Can, Cbn,
          Cdn, Cen, Cfn, Cgn, q]
         k1 = .1;
         k2 = .05;
         k3 = .1;
         k4 = .065;
         k5 = .02;

         Cao = 1;
         Cbo = 1;
         Cdo = 0;
         Ceo = 0;
         Cfo = 0;
         Cgo = 0;
         tmax = 250;

         eqns = {(Caf - Ca) q/V - k1 Ca Cb + k2 Cd Ce - k4 Ca^2
                   + k5 Cg == 0,
                 (Cbf - Cb) q/V - k1 Ca Cb + k2 Cd Ce == 0,
                 -Cd q/V + k1 Ca Cb - k2 Cd Ce - k3 Cd Ce == 0,
                 -Ce q/V + k1 Ca Cb - k2 Cd Ce - k3 Cd Ce == 0,
                 -Cf q/V + k3 Cd Ce == 0,
                 -Cg q/V + k4 Ca^2 - k5 Cg == 0};
         vars = {Ca, Cb, Cd, Ce, Cf, Cg};
         Caf = 1
         Cbf = 1
         V = 100
         q = 5.75
         V/q
         Needs["Miscellaneous`RealOnly`"]
         NSolve[eqns, vars]

Out[37]= 1

Out[38]= 1

Out[39]= 100

Out[40]= 5.75
```

```
Out[41]= 17.3913

Nonreal::warning : Nonreal number encountered.

Out[43]= {{Cf → Nonreal, Cg → Nonreal, Cb → Nonreal,
           Cd → Nonreal, Ce → Nonreal, Ca → Nonreal},
          {Cf → Nonreal, Cg → Nonreal, Cb → Nonreal,
           Cd → Nonreal, Ce → Nonreal, Ca → Nonreal},
          {Cf → Nonreal, Cg → Nonreal, Cb → Nonreal,
           Cd → Nonreal, Ce → Nonreal, Ca → Nonreal},
          {Cf → Nonreal, Cg → Nonreal, Cb → Nonreal,
           Cd → Nonreal, Ce → Nonreal, Ca → Nonreal},
          {Cf → 0.977466, Cg → 0.238722, Cb → 0.772229,
           Cd → -0.749695, Ce → -0.749695, Ca → 0.533507},
          {Cf → 0.12714, Cg → 0.162419, Cb → 0.602479,
           Cd → 0.270381, Ce → 0.270381, Ca → 0.44006}}
```

10.4 Transient CSTR

Time Independence

A very real advantage of a CSTR or PFR is that it can be operated at steady state. This makes it very easy to analyze kinetics of a chemical reaction because the experiments are so easy to conduct. We can see this by looking at the CSTR equation once again. Assuming steady state then, the equation for species A undergoing reaction to B is:

$$\frac{(\mathrm{Caf}-\mathrm{Ca})q}{V} = -r_{A-}$$

And for species B the equation is:

$$\frac{-C_B q}{V} = +r_{A-}$$

The quantity $\frac{q}{V}$ is fixed once the flow rate is fixed because the reactor volume is a constant. We refer to $\frac{V}{q}$ as the holding time θ. Thus $\frac{q}{V}$ is the reciprocal of the holding time, or $\frac{1}{\theta}$. So the difference between the inlet concentration of A and its outlet concentration, divided by the holding time, is the rate of chemical reaction! It is startling to realize that something as nanoscopic, or molecular, as the rate of chemical reaction can be found from measurements that are so macroscopic. By varying the flow rate we can vary the holding time and find the rate of the chemical reaction. By varying the inlet concentration of A keeping all else constant we can find the dependence of the rate on the concentration of A by plotting the rate versus Ca. In this way kinetic rate expressions can be readily determined using a CSTR.

It is also interesting to note that the second equation is actually the design equation for a CSTR. If we are told the rate at which B must be produced and given that we have kinetics available for the rate of reaction, then we can substitute in the kinetics for the rate and solve for the volume. All of this assumes a steady state.

Time Dependence—The Transient Approach to Steady State

Although steady-state CSTRs are simple to operate and to analyze and even though they offer real advantages to the kineticist (scientist who studies kinetics), it is also true that these systems must start up. They do not start up and achieve steady state instantaneously. The time period in which the system moves toward a steady-state condition is called the transient, meaning that the system is in transition from one which is time dependent to one that is time independent and at steady state. We have no way of knowing how long it will take a given set of reactions to achieve a steady state in the CSTR before we either do an experiment or solve the time-dependent model equations. If we choose experiment as a way to assess this we need to be prepared to do many experiments and to make a sizeable expenditure of time and/or money. This is impractical, so we do the math instead. If we do it correctly, then it is cheap, fast, and provides us with insights that experiments cannot yield. We will consider just such a case in this problem.

Complex Catalytic Kinetics

Consider the reaction of a molecule that takes place on a solid catalyst surface. This reaction simply involves converting one form of the molecule into another; in other words, it is an isomerization reaction. However, the reaction in question takes place only on the catalyst surface and not without the catalyst.

When we analyze a reaction of this kind we find that at least two steps are involved. The first is called adsorption and is reversible. Adsorption is the transfer of a molecule from the bulk phase, either the gas or liquid, to the solid surface. The adsorption process is reversible and takes place without any change in the molecule. Like any reversible process, adsorption comes to equilibrium. Because no change occurs in the molecule, the rate of approach to equilibrium is very rapid and occurs essentially instantaneously. Once this occurs then the molecule on the surface can react to product. We can break the problem down into the adsorption equilibrium and the reaction rate of the adsorbed molecule. Take the isomerization to be first order on a surface concentration of species A and consider the reaction to be irreversible. The adsorption equilibrium steps take place by the interaction of the molecule in the bulk phase with a so-called adsorption site on the solid surface. The adsorption site is the locus of points on the surface that interact directly with the molecule.

$$A_{\text{bulk}} + \text{site} \rightleftarrows A_{\text{surface}}$$
$$A_{\text{surface}} \rightarrow B_{\text{surface}}$$
$$B_{\text{surface}} \rightleftarrows B_{\text{bulk}} + \text{site}$$

Once B is formed, it too undergoes adsorption and desorption. The desorption carries B from the surface and into the bulk fluid phase. The rate of this reaction is first order in the surface concentration of A and first order in the concentration of surface sites. It follows a simple kinetic rate law:

$$r_{A-} = k_{\text{surface}} C_{A,\text{surface}} C_{\text{sites}}$$

The surface concentration is difficult to measure, so we need to reexpress it in terms of the bulk phase concentration of species A. To do this we take advantage of the fact that the molecules adsorb and desorb so quickly that they come to equilibrium rapidly with the surface sites. Therefore we can express the surface concentration in terms of the equilibrium. The equilibrium gives rise to the following relationship for the surface concentration of A in terms of the bulk concentration of A:

$$C_{A,\text{surface}} = \frac{K_A C_A}{1 + K_A C_A}$$

We can substitute this expression into the rate expression for the reaction. This leads to this rate in terms of the bulk phase concentrations:

$$r_{A-} = k_{\text{surface}} \frac{K_A C_A}{1 + K_A C_A} C_{\text{sites}}$$

The concentration of sites can be incorporated into a rate constant by rewriting the product of the surface site concentration and the surface rate constant simply as a rate constant:

$$\mathrm{k} = k_{\text{surface}} C_{\text{sites}}$$

We can do this because the surface site concentration is also a constant. Thus the overall rate for this catalytic reaction is:

$$r_{A-} = \frac{k K_A C_A}{1 + K_A C_A}$$

Transient Response of a CSTR with Catalytic Kinetics

The time-dependent component balance equations for A and B in the CSTR are as follows:

$$\frac{\mathrm{d}C_A V}{\mathrm{dt}} = (C_{\mathrm{Af}} - C_A)\mathrm{q} - r_{A-}\mathrm{V} \qquad \frac{\mathrm{d}C_B V}{\mathrm{dt}} = -C_B \mathrm{q} + r_{A-}\mathrm{V}$$

We can substitute into these equations the kinetics we have just derived:

$$\frac{\mathrm{d}C_A V}{\mathrm{dt}} = (C_{\mathrm{Af}} - C_A)\mathrm{q} - \frac{k K_A C_A}{1 + K_A C_A}\mathrm{V} \qquad \frac{\mathrm{d}C_B V}{\mathrm{dt}} = -C_B \mathrm{q} + \frac{k K_A C_A}{1 + K_A C_A}\mathrm{V}$$

The integration of these two equations in time will show us how long it will take the reactor to achieve steady-state conversion of A and production of B.

Problem Statements

A) The first step in this problem is to set up a solution to these equations and a graphical display of the results. Using NDSolve, solve these time-dependent equations and by using Plot graph the concentrations as functions of time for the following set of parameters:

Caf = 1;
V = 1;
q = 10;
Cao = 0;
Cbo = 0;
k = 2;
K1 = .01;
tmax = 1000;

What is the level of conversion with this system volume and flow rate?

B) Increase the volume of the reactor in decade intervals and examine the steady-state conversion at each new volume. What is happening? As this happens, what happens to the time required to achieve a steady state? Show all plots.

C) Set the reactor volume to 1000 and the flow rate q to 1. Plot and record the conversion. Now repeat the calculation increasing q in decade intervals to 10,000. What happens to the conversion and why does it happen?

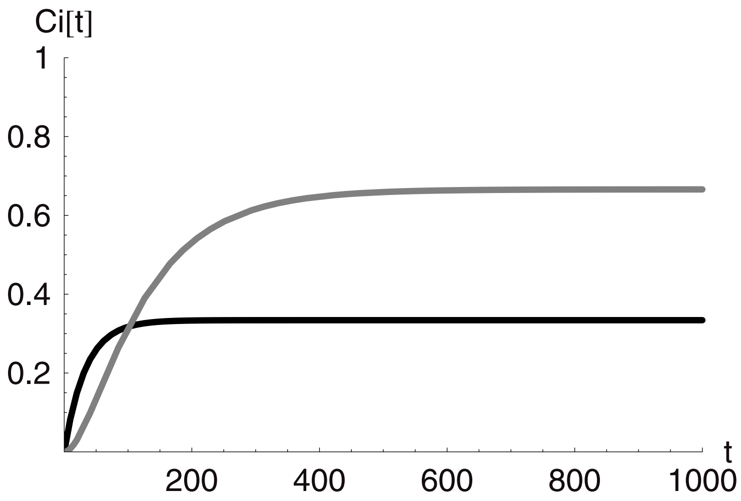
D) How could you use this reactor to evaluate the kinetics?

Solution

Here is code to get the work started.

```
In[1]:= SetOptions[{Plot}, DefaultFont → {"Helvetica", 12}];

        Caf = 1;
        V = 1000;
        q = 10;
        Cao = 0;
        Cbo = 0;
        k = 2;
        K1 = .01;
        tmax = 1000;
```

```
solns = NDSolve[{
                                   q       k K1 Ca[t]
            Ca'[t] == (Caf - Ca[t])─  -  ──────────────,
                                   V     1 + K1 Ca[t]
                              q     k K1 Ca[t]
            Cb'[t] == -Cb[t]─  +  ──────────────,
                              V   1 + K1 Ca[t]
             Ca[0] == Cao, Cb[0] == Cbo},
{Ca[t], Cb[t]}, {t, 0, tmax}];

CA[t_] := Evaluate[Ca[t] /. solns]
CB[t_] := Evaluate[Cb[t] /. solns]

Plot[{CA[t], CB[t]}, {t, 0, tmax},
  PlotStyle → {{Thickness[.01], GrayLevel[0.0]},
   {Thickness[.01], GrayLevel[0.5]}},
  PlotRange → {{0, tmax}, {0, Caf}},
  AxesLabel → {"t", "Ci[t]"}];
```



10.5 CSTR-PFR—A Problem in Comparison and Synthesis

Introduction

Reactions are often complex. By that we mean that rather than having one reaction take place the reactants and products are involved in many different chemical transformations simultaneously. For example, at the temperature required to produce B from A, we may find that A will also react to produce D. These are two parallel reactions:

$$A \rightarrow B$$
$$A \rightarrow D$$

In addition, whether more D or more B is produced depends on the values of the two rate constants for the two steps.

Another case that is often encountered is one in which the product of the reaction of A to B may itself be reactive and at the same conditions will produce D. These reactions occur in series:

$$A \rightarrow B \rightarrow D$$

We will concern ourselves with this problem using the series network of reactions. We will explore the differences between the complete back-mixed CSTR and the axially distributed but radially well-mixed PFR.

A → B → D Network

Consider this reaction network in more detail. Let us assume that D is in fact the product that we seek to produce, but that it must go through B. It is quite realistic to suppose that this is the only viable route to D, but that B is very undesired. For example, D may be a pharmaceutical or nutraceutical with special properties, whereas B is harmful when present in quantities above a given level. Impurity problems of this kind also show up in other chemical products, including specialties and materials. The presence of B above a certain threshold may deleteriously affect the performance of the product. Thus the impurity problem is one that is very real and that crops up across the industries in which chemical engineers participate.

When faced with a problem such as this there are many options that may be pursued to solve it. The crudest, but often practiced approach is to tolerate the impurity insofar as it is a component of the product mix emerging from the reactor, but to separate it downstream of the reactor in a dedicated unit. In some cases this may be the only cost-effective or efficient option. There is another approach and that is to employ reaction engineering.

If B is to be minimized we can see intuitively that at higher overall conversion of A and production of D, B will also be converted to a higher level. If the reactor is made larger, then this can be achieved. But how much larger should it be? If the reactor is to be larger, does it matter if it is back mixed or not? What if there is an existing reactor of specific size and type? Can we improve its behavior and reduce separation costs through reaction engineering?

To answer these questions we must have the relevant kinetics and a target level for this species.

Kinetics and the Objective

The kinetics for each step are given in what follows. We use relative values for the parameters and variables to keep the details simple and to maximize our learning:

$$r_1 = \text{k1}\ C_A^2(\text{A} \rightarrow \text{B});\ \text{k1} = 1 \text{ units}$$
$$r_2 = \text{k2}\ C_B^2(\text{B} \rightarrow \text{A});\ \text{k2} = .09 \text{ units}$$

Let us also assume that the concentration of B in the exit stream should be 0.01 or less to ensure the performance of the specialty chemical D in its application. The inlet or feed concentration of A, CAf, is taken as unity and those of B and D are zero. The relative volume flow rate is also taken as unity.

Problem Statements

A) Using NSolve, develop a steady-state model for a CSTR with a volume of 50. What would the relative concentrations of A, B, and D be as they emerged from this reactor?

B) Using NDSolve, develop a steady-state model for a PFR with the same volume and a cross-sectional area of 10 units. At which position in the PFR does the concentration of B maximize? What is the value of the concentration at this point? What would the relative concentrations of A, B, and D be as they emerged from this reactor? How do they compare to the CSTR with equal volume?

C) Using the two preceding simulations, link them to form a new reactor consisting of a CSTR and PFR in series. To integrate the PFR equation initial conditions are needed at the inlet. Let each initial condition needed for the PFR concentrations be given as the exit concentrations from the CSTR.

i) If the volume of the CSTR is 50, then what additional PFR volume must be used in order to bring down the concentration of B to a level of 0.01?

ii) Instead of adding a PFR to the exit of the CSTR, suppose your colleague had chosen to add simply another back-mixed reactor, that is, another CSTR to the first. (This is just the same as increasing the volume of the original CSTR.) What volume would the CSTR-CSTR require to match the performance of the CSTR-PFR, that is, to bring the concentration of B to 0.01?

CSTR Alone

```
In[1]:= q = 1;
        k1 = 1;
        k2 = .9;
        Caf = 1;
        Vcstr = 200;
        cstrsolns = NSolve[{
         (Caf - CAcstr) q/Vcstr - k1 CAcstr^2 == 0,
                   -CBcstr q/Vcstr + k1 CAcstr^2 - k2 CBcstr^2 == 0,
                      -CDcstr q/Vcstr + k2 CBcstr^2 == 0},
                   {CAcstr, CBcstr, CDcstr}]
        CApfro = Evaluate[CAcstr /. cstrsolns[[4]]]
        CBpfro = Evaluate[CBcstr /. cstrsolns[[4]]]
        CDpfro = Evaluate[CDcstr /. cstrsolns[[4]]]

        General::spell1 : Possible spelling error: new symbol
          name "CBcstr" is similar to existing symbol "CAcstr".

        General::spell : Possible spelling error: new symbol name
          "CDcstr" is similar to existing symbols {CAcstr, CBcstr}.

Out[6]= {{CDcstr → 1.1533, CAcstr → -0.0732549,
          CBcstr → -0.0800451},
         {CDcstr → 1.00652, CAcstr → 0.0682549,
          CBcstr → -0.0747783},
         {CDcstr → 0.998765, CAcstr → -0.0732549,
          CBcstr → 0.0744896},
         {CDcstr → 0.862522, CAcstr → 0.0682549,
          CBcstr → 0.0692228}}

Out[7]= 0.0682549

        General::spell1 : Possible spelling error: new symbol
          name "CBpfro" is similar to existing symbol "CApfro".

Out[8]= 0.0692228

        General::spell : Possible spelling error: new symbol name
          "CDpfro" is similar to existing symbols {CApfro, CBpfro}.

Out[9]= 0.862522
```

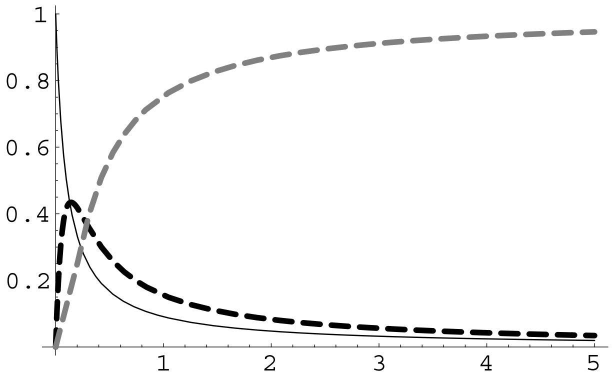
PFR Alone

```
In[10]:= Acs = 10;
         CApfro = 1;
         CBpfro = 0;
         CDpfro = 0;
         Vcstr = 50
```

```
pfrsolns = NDSolve[{CApfr'[z] == -$\frac{A_{cs}}{q}$ k1 CApfr[z]$^2$,
          CBpfr'[z] == +$\frac{A_{cs}}{q}$ (k1 CApfr[z]$^2$ - k2 CBpfr[z]$^2$),
          CDpfr'[z] == +$\frac{A_{cs}}{q}$ k2 CBpfr[z]$^2$,
          CApfr[0] == CApfro,
          CBpfr[0] == CBpfro,
          CDpfr[0] == CDpfro},
          {CApfr[z], CBpfr[z], CDpfr[z]},
          {z, 0, $\frac{V_{cstr}}{A_{cs}}$}]
Caexit[z_] := Evaluate[CApfr[z] /. pfrsolns]
Cbexit[z_] := Evaluate[CBpfr[z] /. pfrsolns]
Cdexit[z_] := Evaluate[CDpfr[z] /. pfrsolns]
Plot[{Caexit[z], Cbexit[z], Cdexit[z]}, {z, 0, $\frac{V_{cstr}}{A_{cs}}$},
    PlotStyle →
    {GrayLevel[0], {Thickness[.01], Dashing[{0.03, 0.02}],
      GrayLevel[0]},
     {Thickness[.01], Dashing[{0.03, 0.02}],
       GrayLevel[0.5]},
     {Thickness[.01], GrayLevel[0.5]}}];
zf = $\frac{V_{cstr}}{A_{cs}}$
{Caexit[zf], Cbexit[zf], Cdexit[zf]}
Table[Cbexit[z], {z, 0, $\frac{V_{cstr}}{A_{cs}}$}]
```

```
Out[14]= 50

         General::spell1 : Possible spelling error: new symbol
          name "CApfr" is similar to existing symbol "CApfro".

         General::spell : Possible spelling error: new symbol
          name "CBpfr" is similar to existing symbols
          {CApfr, CBpfro}.

         General::spell : Possible spelling error: new symbol
          name "CDpfr" is similar to existing symbols
          {CApfr, CBpfr, CDpfro}.

Out[15]= {{CApfr[z] → InterpolatingFunction[{{0., 5.}}, <>][z],
           CBpfr[z] → InterpolatingFunction[{{0., 5.}}, <>][z],
           CDpfr[z] → InterpolatingFunction[{{0., 5.}}, <>][z]}}

           General::spell1 : Possible spelling error: new symbol
            name "Cbexit" is similar to existing symbol "Caexit".
```

```
General::spell : Possible spelling error: new symbol
 name "Cdexit" is similar to existing symbols
 {Caexit, Cbexit}.
```



```
Out[20]= 5

Out[21]= {{0.0196083}, {0.0342294}, {0.946162}}

Out[22]= {{0.}, {0.155406}, {0.0827424}, {0.056225}, {0.0425573},
          {0.0342294}}
```

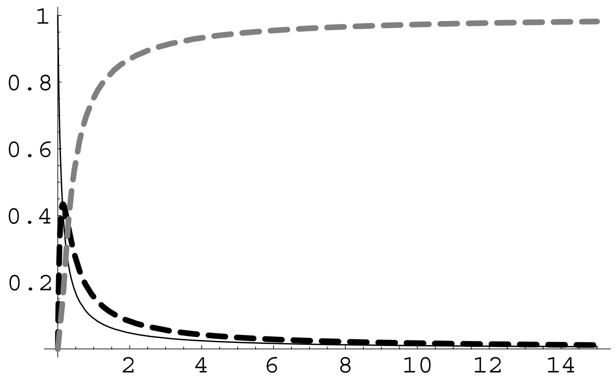
```
In[23]:= Acs = 10;
         Vcstr = 150
         pfrsolns = NDSolve[{CApfr'[z] == -Acs/q k1 CApfr[z]^2,
              CBpfr'[z] == +Acs/q (k1 CApfr[z]^2 - k2 CBpfr[z]^2),
              CDpfr'[z] == +Acs/q k2 CBpfr[z]^2,
              CApfr[0] == CApfro,
              CBpfr[0] == CBpfro,
              CDpfr[0] == CDpfro},
              {CApfr[z], CBpfr[z], CDpfr[z]},
              {z, 0, Vcstr/Acs}]
         Caexit[z_] := Evaluate[CApfr[z] /. pfrsolns]
         Cbexit[z_] := Evaluate[CBpfr[z] /. pfrsolns]
         Cdexit[z_] := Evaluate[CDpfr[z] /. pfrsolns]
         Plot[{Caexit[z], Cbexit[z], Cdexit[z]}, {z, 0, Vcstr/Acs},
             PlotStyle →
             {{GrayLevel[0]}, {Thickness[.01],
               Dashing[{0.03, 0.02}], GrayLevel[0]},
        {Thickness[.01], Dashing[{0.03, 0.02}],
         GrayLevel[0.5]}}];
```

$$zf = \frac{V_{cstr}}{A_{cs}}$$

```
{Caexit[zf], Cbexit[zf], Cdexit[zf]}
```

$$\text{Table[Cbexit[z], \{z, 0, } \frac{V_{cstr}}{A_{cs}}\text{\}]}$$

```
Out[24]= 150

Out[25]= {{CApfr[z] → InterpolatingFunction[{{0., 15.}}, <>][z],
           CBpfr[z] → InterpolatingFunction[{{0., 15.}}, <>][z],
           CDpfr[z] → InterpolatingFunction[{{0., 15.}}, <>][z]}}
```



```
Out[30]= 15

Out[31]= {{0.00662276}, {0.0115696}, {0.981808}}

Out[32]= {{0.}, {0.155406}, {0.0827424}, {0.056225}, {0.0425573},
          {0.0342294}, {0.0286256}, {0.0245975}, {0.0215631},
          {0.0191948}, {0.0172951}, {0.0157377}, {0.0144375},
          {0.0133356}, {0.01239}, {0.0115696}
```

10.6 Membrane Reactor — Overcoming Equilibrium with Simultaneous Separation

Introduction

Reversible chemical reactions given enough time will come to equilibrium. In the batch reactor equilibrium is diagnosed when the conversion of the reactant no longer changes even when the reaction time, that is, the batch holding time, is increased. When the reactor is a flow reactor, either CSTR or PFR, we find equilibrium when the conversion no longer changes with increased holding time (q/V).

Often the equilibrium position of a reversible process is such that the conversion to product is low at reasonable holding times (i.e., flow rates and reactor volumes). For example, the dehydrogenation of saturated alkanes and alkyl aromatics to produce alkenes and aryl-alkenes and hydrogen is a very important case in point:

$$CH_3CH_2CH_2CH_3 \rightleftharpoons CH_3CH_2CH = CH_2 + H_2$$

This is economically disadvantageous because it means either that rates of production will be low or that the investment in the reactor will be very high because it needs to be so large. There is a clever way around this that always has been employed on a small scale and that is now gaining currency for selected larger-scale processes.

Reaction with Separation

Chemical equilibrium responds to a "stress" by moving to the side that relieves the effect of "stress" and returns the system to equilibrium, according to Le Chatlier's Principle. If we add heat to an exothermic reaction it will shift toward the reactants on the left. If we add mass, concentration, or pressure on the reactant side of the equilibrium, the system responds by shifting toward the products. If we can remove products from the reaction zone (the system or control volume), then we also shift the reaction equilibrium to the right. In fact, even if we remove just one of the products from a set of products, the system will shift to the right.

In the case of the reaction class we are considering, that is, the dehydrogenation of alkanes to alkenes and hydrogen, continuous removal of either the alkene or hydrogen from the system will shift the conversion of the reactant alkane further to the products. This is the phenomenon we wish to examine.

Hydrogen-Selective Membranes

Palladium and its alloys as well as some new ceramic membrane materials will separate hydrogen selectively from hydrocarbons and they will do so at temperatures that are high enough for reaction to take place. Thus, one can operate a membrane separation of hydrogen in conjunction with the production of hydrogen by a dehydrogenation of alkane. The extent to which the hydrogen is removed from the reaction zone will be the extent to which the reaction proceeds to olefin at a conversion level beyond that achieved at equilibrium. In the case of palladium the hydrogen must first dissociate into atoms at the surface prior to entering the lattice to diffuse through the membrane. One of the advantages of the ceramic membranes is that they are nanoporous and so dihydrogen in molecular form will diffuse through them intact. This process requires less energy and is relatively faster. It also leads to a simple linear dependence upon the dihydrogen rather than to a square root dependence. For these reasons we will consider this type of membrane.

Consider the following reaction as representative of this type and the rates of the forward and reverse reactions:

$$\mathrm{A} \rightleftharpoons \mathrm{B} + \mathrm{H2}$$
$$r_f = \mathrm{k1\ CA}^2$$
$$r_{\mathrm{rev}} = \mathrm{k2\ CB\ CH2}$$

The rate of transport of hydrogen across the membrane of area As in units of mol/time is given by:

$$r_{\mathrm{transport}} = \mathrm{As\ Pm(CH2I} - C\mathrm{H2II)}$$

The reaction can be considered to take place in the volume above the membrane. Only hydrogen is transported through the membrane, whereupon it leaves the lower volume via convective flow. Unconverted alkane A, the product alkene B, and hydrogen are also convected out of the volume above the membrane. Consider both the volumes above and below the membrane to be well mixed.

Problem Statements

A) Construct the time-independent model equations for the change in concentration of A,B and H2 on the top side of the membrane and for H2 on the lower side of the membrane.

B) The first step is to establish whether or not the reactions have adequate holding time to reach equilibrium in the absence of permeation. Therefore, letting Pm = 0 and using the following list of parameter values, show what happens as the flow rate on the top side of the membrane drops in decade increments from 1000 au to .001 au. What happens and in which decade interval of flow rate (i.e., holding time) does the reaction attain equilibrium. What are the equilibrium levels of A,B and H2?

CAf = 1;

qI = 100;

qII = 1000;

VI = 10;

VII = 10;

Am = 1;

k1 = .01;

k2 = 1;

Pm = 0.0;

C) Using the same model and the same parameter values, set Pm = 0.01. Now increase the value of Am in decades and note the values of A,B,CH2I (top) and CH2 (bottom). What happens and why does it happen?

D) Starting at the final parameter values in part C begin to increase qII in decade increments. What effect if any does this have on the conversion of A and the production of B? Why?

Utilize NSolve to numerically solve the equations. The recommended format for doing this is as follows:

```
eqns = {set of four model equations including convection,
        reaction, and permeation}
vars = {set of four variables to be solved for in time: CA...}
Vertical list of parameters;
solns = NSolve[eqns, vars}]
```

E) In fact, the expression for hydrogenation permeation across a palladium membrane is not simply linear in the concentrations, but instead follows the square root of each hydrogen concentration:

$$r_{\text{transport}} = \text{Pm Am}\sqrt{\text{Ks}}\,(\sqrt{\text{CH2I[t]}} - \sqrt{\text{CH2II[t]}})$$

Rewrite the model equations for the time-dependent case to handle this complication and solve using the same parameter values as before.

Solution

For parts A through D the following code will be useful:

Steady-State Membrane Reaction with Separation

```
In[1]:= eqns = {(CAf - CAI) qI/VI - k1 CAI + k2 CBI CH2I == 0,
                (-CBI) qI/VI + k1 CAI - k2 CBI CH2I == 0,
             (-CH2I) qI/VI + k1 CAI - k2 CBI CH2I
               - Pm Am(CH2I - CH2II) == 0,
             (-CH2II) qII/VII + Pm Am(CH2I - CH2II) == 0}

        vars = {CAI, CBI, CH2I, CH2II}

        CAf = 1;
```

```
qI = 100;
qII = 1000;
VI = 10;
VII = 10;
Am = 1;
k1 = .01;
k2 = 1;
Pm = 0.0;

NSolve[eqns, vars]

General::spell1 : Possible spelling error: new symbol
 name "CH2II" is similar to existing symbol "CH2I".
```

$$\text{Out[1]= }\{-\text{CAI k1} + \text{CBI CH2I k2} + \frac{(\text{CAf} - \text{CAI})\text{qI}}{\text{VI}} == 0,$$

$$\text{CAI k1} - \text{CBI CH2I k2} - \frac{\text{CBI qI}}{\text{VI}} == 0,$$

$$\text{CAI k1} - \text{CBI CH2I k2} - \text{Am}(\text{CH2I} - \text{CH2II})\text{Pm} - \frac{\text{CH2I qI}}{\text{VI}} == 0,$$

$$\text{Am}(\text{CH2I} - \text{CH2II})\text{Pm} - \frac{\text{CH2II qII}}{\text{VII}} == 0\}$$

```
Out[2]= {CAI, CBI, CH2I, CH2II}

Out[12]= {{CH2II → 0., CAI → 11.011, CBI → -10.011,
            CH2I → -10.011},
           {CH2II → 0., CAI → 0.999001, CBI → 0.000998901,
            CH2I → 0.000998901}}
```

For parts D and E this code will be necessary (note that permeabilities are set to zero).

Transient Membrane Reaction with Separation

```
In[71]:= SetOptions[{Plot}, DefaultFont → {"Helvetica", 10}];
```

$$\text{eqns} = \{\text{CAI'[t]} == (\text{CAf} - \text{CAI[t]})\frac{\text{qI}}{\text{VI}} - \text{k1 CAI[t]}$$
$$+ \text{k2 CBI[t] CH2I[t]},$$

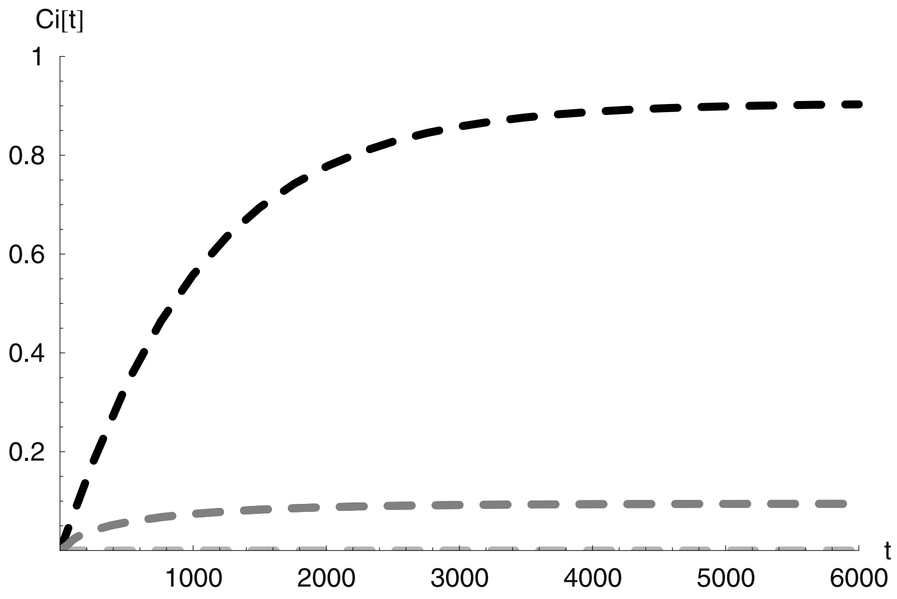
$$\text{CBI'[t]} == (-\text{CBI[t]})\frac{\text{qI}}{\text{VI}} + \text{k1 CAI[t]}$$
$$- \text{k2 CBI[t] CH2I[t]},$$

$$\text{CH2I'[t]} == (-\text{CH2I[t]})\frac{\text{qI}}{\text{VI}} + \text{k1 CAI[t]}$$
$$- \text{k2 CBI[t] CH2I[t]} - \text{Pm Am}(\text{CH2I[t]} - \text{CH2II[t]}),$$

```
        CH2II'[t] == (-CH2II[t]) qII/VII
         + Pm Am (CH2I[t] - CH2II[t]),
            CAI[0] == CAIo,
    CBI[0] == CBIo,
    CH2I[0] == CH2Io,
    CH2II[0] == CH2IIo};

vars = {CAI[t], CBI[t], CH2I[t], CH2II[t]};

Clear[CAf, qI, qII, VI, VII, Am, k1, k2, Pm, Ks, CAIo,
 CBIo, CH2Io, CH2IIo]

CAf = 1;
qI = .01;
qII = 1000;
VI = 10;
VII = 10;
Am = 1;
k1 = .01;
k2 = 1.0;
Pm = 0.0;
CAIo = 0;
CBIo = 0;
CH2Io = 0;
CH2IIo = 0;
Clear[solns]
tmax = 6000

solns = NDSolve[eqns, vars, {t, 0, tmax},
 MaxSteps → 2000]

Ca1[t_] := Evaluate[CAI[t] /. solns]
a = Plot[Ca1[t], {t, 0, tmax},
    PlotStyle → {Thickness[0.01], GrayLevel[0],
     Dashing[{.03, .03}]},
    PlotRange → {{0, tmax}, {0, CAf}},
     DisplayFunction → Identity];

Cb1[t_] := Evaluate[CBI[t] /. solns]
b = Plot[Cb1[t], {t, 0, tmax},
    PlotStyle → {Thickness[0.01], GrayLevel[0.3],
     Dashing[{.03, .03}]},
    PlotRange → {{0, tmax}, {0, CAf}},
     DisplayFunction → Identity];
```

```
CH21[t_] := Evaluate[CH2I[t] /. solns]
h21 = Plot[CH21[t], {t, 0, tmax},
    PlotStyle → {Thickness[0.01], GrayLevel[0.5],
      Dashing[{.03, .03}]},
    PlotRange → {{0, tmax}, {0, CAf}},
     DisplayFunction → Identity];

CH22[t_] := Evaluate[CH2II[t] /. solns]
h22 = Plot[CH22[t], {t, 0, tmax},
    PlotStyle → {Thickness[0.01], GrayLevel[0.7],
     Dashing[{.03, .03}]},
    PlotRange → {{0, tmax}, {0, CAf}},
     DisplayFunction → Identity];

Show[{a, b, h21, h22}, AxesLabel → {"t", "Ci[t]"},
 DisplayFunction → $DisplayFunction]
```

```
Out[89]= 6000

Out[90]= {{CAI[t] → InterpolatingFunction[{{0., 6000.}}, <>][t],
          CBI[t] → InterpolatingFunction[{{0., 6000.}}, <>][t],
          CH2I[t] → InterpolatingFunction[{{0., 6000.}}, <>][t],
          CH2II[t] → InterpolatingFunction[{{0., 6000.}}, <>][t]}}
```



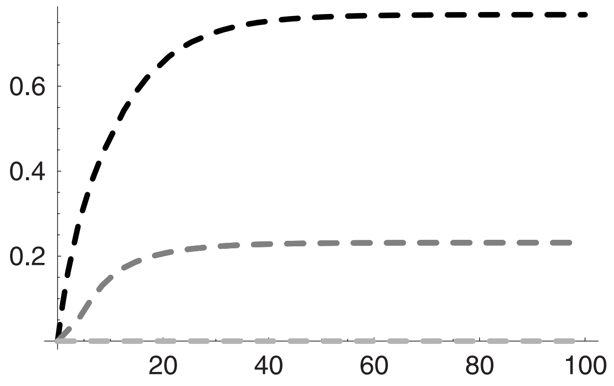
```
Out[99]= - Graphics -
```

Palladium Membrane

```
In[126]:= Clear[vars, eqns];
          SetOptions[{Plot}, DefaultFont → {"Helvetica", 10}];
                                               qI
          eqns = {CAI'[t] == (CAf - CAI[t]) ──── - k1 CAI[t]
                                               VI
                   + k2 CBI[t] CH2I[t],
                                          qI
                  CBI'[t] == (-CBI[t]) ──── + k1 CAI[t]
                                          VI
                   - k2 CBI[t] CH2I[t],
                                            qI
                  CH2I'[t] == (-CH2I[t]) ──── + k1 CAI[t]
                                            VI
                   - k2 CBI[t] CH2I[t]
                   - Pm Am √Ks (√CH2I[t] - √CH2II[t]),
                                              qII
                  CH2II'[t] == (-CH2II[t]) ─────
                                              VII
                   + Pm Am √Ks (√CH2I[t] - √CH2II[t]),
                       CAI[0] == CAIo,
                  CBI[0] == CBIo,
                  CH2I[0] == CH2Io,
                  CH2II[0] == CH2IIo};
          vars = {CAI[t], CBI[t], CH2I[t], CH2II[t]};
          CAf = 1;
          qI = 1;
          qII = 1;
          VI = 10;
          VII = 10;
          Am = 10;
          k1 = .1;
          k2 = 1;
          Pm = 0;
          Ks = .1;
          CAIo = 0;
          CBIo = 0;
          CH2Io = 0;
          CH2IIo = 0;
          Clear[solns]
          tmax = 100
          solns = NDSolve[eqns, vars, {t, 0, tmax}];
          Ca1[t_] := Evaluate[CAI[t] /. solns]
          Cb1[t_] := Evaluate[CBI[t] /. solns]
```

```
CH21[t_] := Evaluate[CH2I[t] /. solns]

CH22[t_] := Evaluate[CH2II[t] /. solns]

Plot[{Ca1[t], CH21[t], Cb1[t], CH22[t]}, {t, 0, tmax},
 PlotStyle → {{Thickness[0.01], GrayLevel[0],
  Dashing[{.03, .03}]},
   {Thickness[0.01], GrayLevel[0.3], Dashing[{.03, .03}]},
   {Thickness[0.01], GrayLevel[0.5], Dashing[{.03, .03}]},
   {Thickness[0.01], GrayLevel[0.7], Dashing[{.03, .03}]}}]
```

```
Out[145]= 100
```



```
Out[151]= - Graphics -
```

10.7 Microbial Population Dynamics

Introduction

The current explosion in biological sciences research is unprecedented. The breakthroughs in the basic sciences of genomics and related disciplines have brought us to the threshold of a new era in biological technology. Naturally, the chemical industry is involved and chemical engineers are participating in increasing numbers. Some technology pundits are predicting that this green revolution will supplant the processes and products related to the chemical industry that we have come to know in the twentieth century with new ones that are environmentally benign and biodegradable. It is a stunning and in some ways captivating vision for the future that will have obvious benefits, but unknown and unforeseeable consequences as well.

Microbes as Reactors

Paramount to this new technology is the use of microbes, that is, cellular organisms as reactors. Organisms have evolved mechanisms for dealing with environmental stress, such as the presence of a new substrate chemical in their surroundings, by rerouting their metabolic pathways. Metabolic engineers can take advantage of this through a procedure of accelerated adaptation in order to generate new microbes that consume a given substrate and produce a specific target chemical.

Microbes use enzymes as catalysts to make the desired or beneficial reaction take place, and typically under mild conditions. Brewing of beer and fermentation of fruit and vegetable mass high in starches to produce consumable ethanol are the oldest and most familiar instances of using microbial action to fulfill a desired end. But now much more has been demonstrated, ranging from the production of essential human hormones to the synthesis of specialty chemicals.

In a reactor containing a substrate a colony of microbes is inoculated and brought to maturity. As the colony grows the substrate is consumed to supply the microbes with their building blocks. Some fraction of the substrate is necessarily diverted into the formation of biomass, that is, cells—their membranes and organelles, but some other fraction is used to produce the target molecule. In a batch process, when the substrate has been consumed, the microbial colony either dies rapidly or if the process is to be stopped prior to complete substrate consumption, it is killed by a rapid change in conditions (for example, by raising the temperature as is done in the pasteurization of raw milk). From this point the problem of recovering the target molecule is one of separating it from the biomass and aqueous medium.

Kinetics

The basis of life is molecular and therefore we can describe the rates of substrate consumption, product formation, and even microbe population growth in much the same way that we would describe the rates of molecular-level chemical processes.

We will take the microbe, substrate, and product concentrations to be a[t], b[t], and c[t], respectively. The equations that describe the rates of change of each of these are shown here:

$$a'[t] = \left(\mu\text{max}\frac{b[t]}{Ks + b[t]} - k\right)a[t],$$

$$b'[t] = -\left(\frac{\mu\text{max}}{ys}\frac{b[t]}{Ks + b[t]}\right)a[t],$$

$$c'[t] = \left(\alpha + \beta\mu\text{max}\frac{b[t]}{Ks + b[t]}\right)a[t]$$

The kinetic expressions are highly nonlinear because they include a Michaelis-Menton rate term:

$$\mu\text{max}\frac{\text{b[t]}}{\text{Ks}+\text{b[t]}}\text{a[t]}$$

where μmax is a maximum rate constant, Ks is a saturation concentration, and ys is a dimensionless parameter that is similar to a stoichiometric coefficient. Similarly, α and β are dimensionless numbers that are also similar to stoichiometric coefficients—they relate the rate of production of the desired molecule to the rate of growth of microbial cell mass.

Problem Statements

A) Using NDSolve, build a simulation of the dynamics of microbial growth described by these equations.

i) Parameter values should be:

```
μ = .15; "μmax";
K = .04; "Ks";
y = 1; "ys";
α = 10^-n;
n = 2
β = .1;
tmax = 500
k = 0.1
```

ii) Plot the change in A, B, and C on one graph against time.

iii) What do the plots indicate about how the process occurs?

B) If there are two microbial species present show the dynamics of microbe growth and product formation from each and in toto. Find A1,B1,C1 and A2,B2,C2 for each microbe.

i) Parameter values for each should be:

```
μ1 = 1.1; "μmax";
K1 = .04; "Ks";
y1 = 1; "ys";
α1 = 0.02;
β1 = .5;
μ2 = .9; "μmax";
K2 = .025; "Ks";
y2 = 1; "ys";
α2 = 0.004;
β2 = .5;
tmax = 100
```

```
k1 = .3
k2 = .1
```

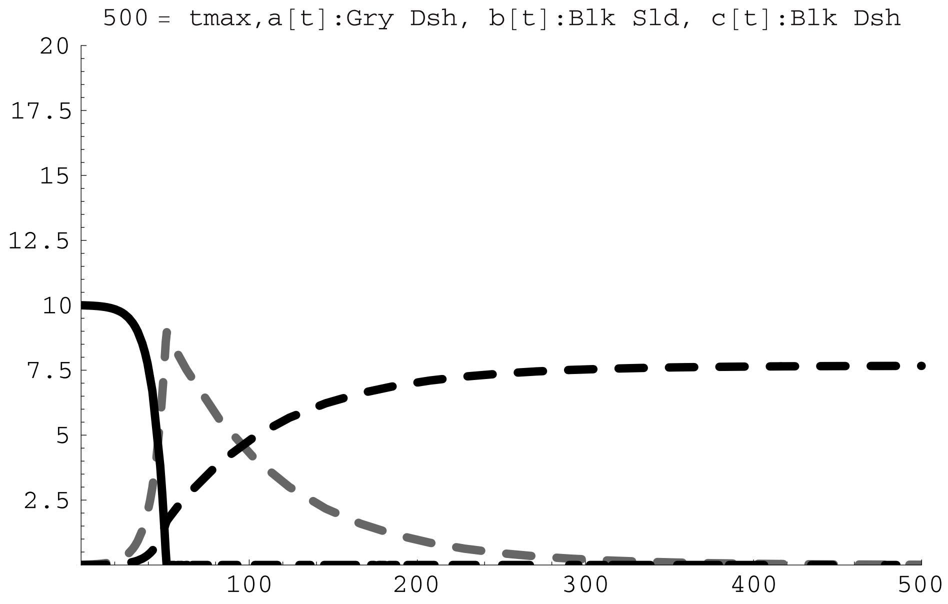
ii) Plot the change in A, B, and C on one graph against time for each microbe and then plot all six on one. Also plot the total substrate and total product concentration on a separate plot.

iii) What is happening to these microbes according to this kinetic simulation?

Solutions

For Part A:

```
In[1]:= μ = .15; "μmax";
        K = .04; "Ks";
        y = 1; "ys";
        α = 10^-n;
        n = 2
        β = .1;
        tmax = 500
        k = 0.1
        "a[t] is the change in microbial concentration;
         decreasing";
        "b[t] is the change in the substrate concentration;
         increasing";
        "c[t] is the change in product concentration; increasing";

        bugs1 = NDSolve[{
                                 b[t]
                  a'[t] == μ(─────────  - k)a[t],
                              K + b[t]
                            μ   b[t]
                  b'[t] == -─ ────────a[t],
                            y K + b[t]
                                      b[t]
                  c'[t] == (α + βμ────────)a[t],
                                  K + b[t]
                  a[0] == .01,
                  b[0] == 10,
                  c[0] == 0
                  },
               {a[t], b[t], c[t]},
               {t, 0, tmax}];
        a1[t_] := Evaluate[a[t] /. bugs1];
        b1[t_] := Evaluate[b[t] /. bugs1];
        c1[t_] := Evaluate[c[t] /. bugs1];
        pa1 = Plot[a1[t], {t, 0, tmax}, DisplayFunction → Identity,
           PlotStyle → {{Dashing[{0.03, 0.03}], Thickness[.01],
            GrayLevel[0.4]}},
           PlotRange → {{0, tmax}, {0, 20}}];
        pb1 = Plot[b1[t], {t, 0, tmax}, DisplayFunction → Identity,
           PlotStyle → {Thickness[.01], GrayLevel[0]}];
        pc1 = Plot[c1[t], {t, 0, tmax},
           PlotStyle → {{Thickness[.01], GrayLevel[0.0],
            Dashing[{0.03, 0.03}]}},
           DisplayFunction → Identity];
        Show[pa1, pb1, pc1, DisplayFunction → $DisplayFunction,
          PlotLabel → tmax "= tmax,a[t]:Gry Dsh, b[t]:Blk Sld,
           c[t]:Blk Dsh"];

Out[5]= 2

Out[7]= 500

Out[8]= 0.1
```

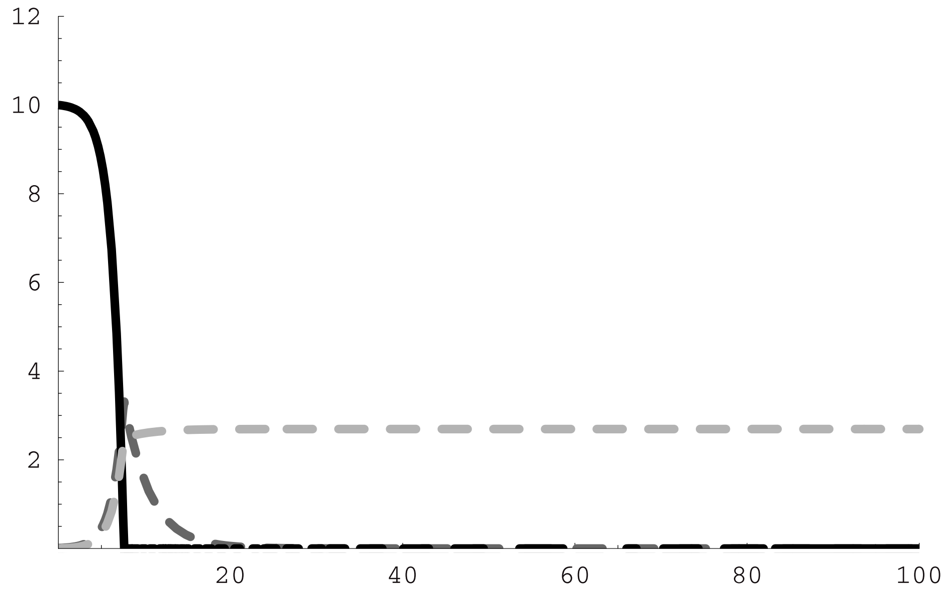


For Part B:

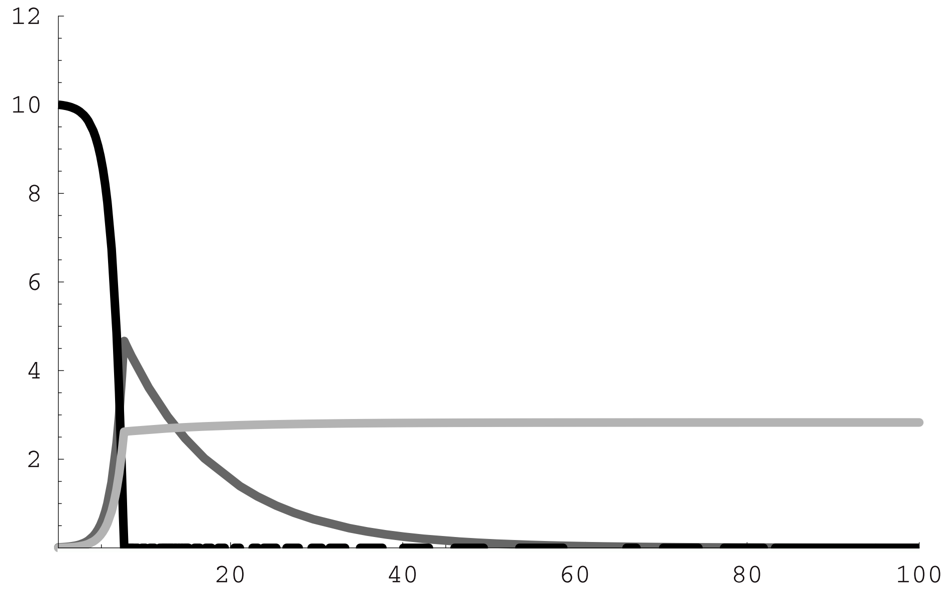
```
In[20]:= Clear[a1, b1, c1, α1, β1, K1, k1, y1, α2, β2, K2, k2,
          y2, n, t, tmax, bugs2, A1, B1, C1, A2, B2, C2, p1, p2];

         SetOptions[{Plot}, DefaultFont → {"Hevetica", 10}];
```

```
μ1 = 1.1; "μmax";
K1 = .04; "Ks";
y1 = 1; "ys";
α1 = 0.02;
β1 = .5;
μ2 = .9; "μmax";
K2 = .025; "Ks";
y2 = 1; "ys";
α2 = 0.004;
β2 = .5;
tmax = 100
k1 = .3
k2 = .1
"a[t] is the change in microbial concentration;
 decreasing";
"b[t] is the change in the substrate concentration;
 increasing";
"c[t] is the change in product concentration;
 increasing";

bugs2 = NDSolve[{
            a1'[t] == μ1(  b1[t]       - k1)a1[t],
                       ------------
                       K1 + b1[t]
            b1'[t] == -μ1    b1[t]    a1[t]
                       --  ----------
                       y1  K1 + b1[t]
             -μ2    b1[t]     a2[t],
              --  ----------
              y2  K2 + b1[t]
            c1'[t] == α1 a1[t] + β1 μ1    b1[t]     a1[t],
                                        ----------
                                        K1 + b1[t]
            a2'[t] == μ2 (   b1[t]      - k2)a2[t],
                          ----------
                          K2 + b1[t]
            c2'[t] == α2 a2[t] + β2 μ2    b1[t]     a2[t],
                                        ----------
                                        K2 + b1[t]
            a1[0] == 0.01,
            b1[0] == 10,
            c1[0] == 0,
            a2[0] == 0.01,
            c2[0] == 0
            },
        {a1[t], b1[t], c1[t], a2[t], c2[t]},
        {t, 0, tmax}];
```

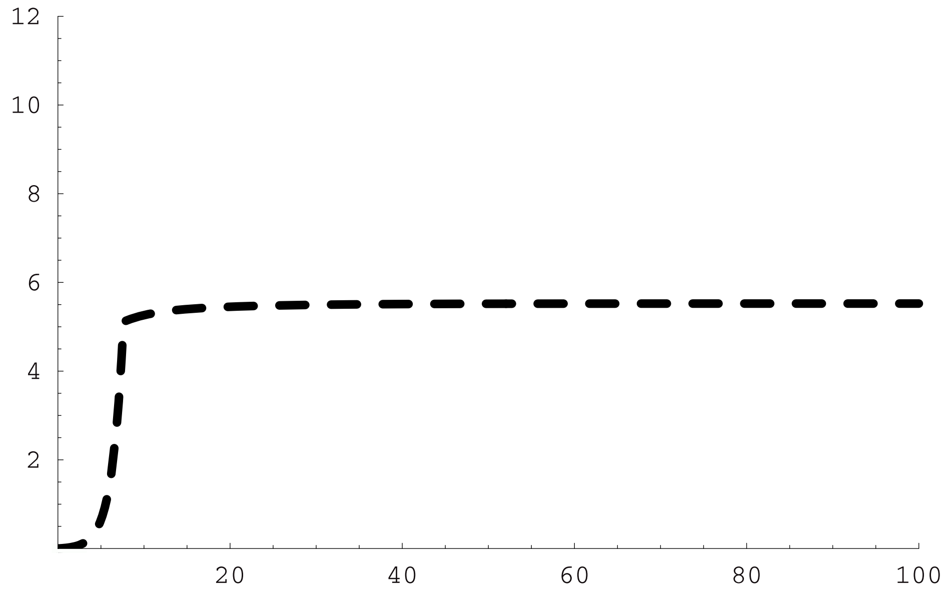
```
A1[t_] := Evaluate[a1[t] /. bugs2];
B1[t_] := Evaluate[b1[t] /. bugs2];
C1[t_] := Evaluate[c1[t] /. bugs2];
A2[t_] := Evaluate[a2[t] /. bugs2];
C2[t_] := Evaluate[c2[t] /. bugs2];

p1 = Plot[{A1[t], B1[t], C1[t]}, {t, 0, tmax},
   PlotStyle → {{Dashing[{0.03, 0.03}], Thickness[.01],
    GrayLevel[0.4]},
      {Thickness[.01], GrayLevel[0]},
      {Dashing[{0.03, 0.03}],
        Thickness[.01], GrayLevel[0.7]}},
        PlotRange → {{0, tmax}, {0, 12}}];
p2 = Plot[{A2[t], B1[t], C2[t]}, {t, 0, tmax},
   PlotStyle → {{Thickness[.01], GrayLevel[0.4]},
      {Thickness[.01], GrayLevel[0]},
      {Thickness[.01], GrayLevel[0.7]}},
   PlotRange → {{0, tmax}, {0, 12}},
   PlotRange → {{0, tmax}, {0, 4}}];
p3 = Plot[{C1[t] + C2[t]}, {t, 0, tmax},
   PlotStyle → {{Thickness[.01], GrayLevel[0.0],
    Dashing[{0.03, 0.03}]}},
   PlotRange → {{0, tmax}, {0, 12}},
   PlotRange → {{0, tmax}, {0, 4}}];
Print[tmax "= tmax"]
Show[
    p1, p2, p3, PlotLabel → "Sbstrt= Blk.Sld.;
    Tot.Prdt. = Blk.Dshd.;Bugs = Gry",
   AxesLabel → {"t", "Ci[t]"}];

General::spell1 : Possible spelling error: new symbol
 name "β1" is similar to existing symbol "α1".

General::spell1 : Possible spelling error: new symbol
 name "β2" is similar to existing symbol "α2".

General::spell : Possible spelling error: new symbol
 name "μ1" is similar to existing symbols {α1, β1}.

General::spell : Possible spelling error: new symbol
 name "μ2" is similar to existing symbols {α2, β2}.

Out[32]= 100

Out[33]= 0.3

Out[34]= 0.1
```



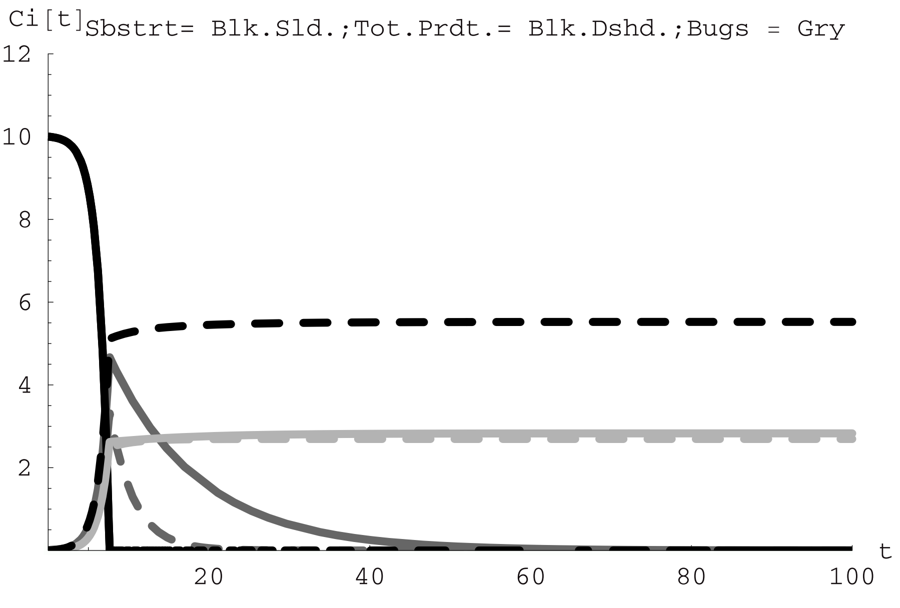
12
10
8
6
4
2
20
40
60
80
100



12
10
8
6
4
2
20
40
60
80
100


12
10
8
6
4
2
20
40
60
80
100

```
100 = tmax
```


Ci[t]Sbstrt= Blk.Sld.;Tot.Prdt.= Blk.Dshd.;Bugs = Gry
12
10
8
6
4
2
t
20
40
60
80
100

Index