(M. GREGORY)
CIVIL ENGINEERING
DEPARTMENT.

HYDRO-UNIVERSITY COMPUTING CENTRE

AMENDED PROCEDURES

for replacement in your

HYDRO-UNIVERSITY COMPUTING CENTRE
ALGOL LIBRARY PROCEDURES MANUAL

No. 37

30th September 1966

LE01	mxinvert
LE16	choleski
LE24	SYMDET
LE28	SYMDET
LL08	vecjacobi
MT03	equipol
MT06	equidydx

****

```
procedure mxinvert (a,n,eps,singular);  value n,eps;  array a;  integer n;  real eps;  label singular;
begin integer i,j,k,pivi,pivj,p,ri,ci,rk,cj,iless1;  real pivot;  integer array r,c[1:n];
            comment set row and column index vectors;
            for i:=1 step 1 until n do r[i]:=c[i]:=i;
            comment find initial pivot;  pivi:=pivj:=1;  pivot:=a[1,1];
            for i:=1 step 1 until n do for j:=1 step 1 until n do
            if abs(a[i,j])≥abs(pivot) then begin pivi:=i; pivj:=j;  pivot:=a[i,j] end;
            comment start reduction;
            for i:=1 step 1 until n do
                  begin ri:=r[pivi]; r[pivi]:=r[i]; r[i]:=ri; ci:=c[pivj]; c[pivj]:=c[i]; c[i]:=ci; iless1:=i-1;
                        if eps > abs(a[ri,ci]) then
                              begin print punch(3),sameline,digits(3),££1?MATRIX SINGULAR?,££1?i=?,i,££1?PIVOTS FOLLOW?;
                                    for i:=1 step 1 until n do
                                                 print punch(3),sameline,digits(3),££1??,r[i],££s4??,c[i];
                                    goto singular
                              end;
                        for j:=1 step 1 until iless1,i+1 step 1 until n do
                                    begin cj:=c[j];  a[ri,cj]:=a[ri,cj]/pivot end;
                                            a[ri,ci]:=1.0/pivot;  pivot :=0;
                        for k:=1 step 1 until iless1,i+1 step 1 until n do
                              begin rk:=r[k];
                                    for j:=1 step 1 until iless1,i+1 step 1 until n do
                                          begin cj:=c[j]; a[rk,cj]:=a[rk,cj]-a[ri,cj]*a[rk,ci];
                                                if k>i and j>i and abs(a[rk,cj]) ≥abs(pivot) then
                                                      begin pivi:=k;  pivj:=j; pivot:=a[rk,cj] end conditional
                                          end jloop;
                                    a[rk,ci]:=-a[ri,ci]*a[rk,ci]
                              end kloop
                  end iloop and reduction;
            comment rearrange rows;     mxperm(a[j,p],a[k,p],j,k,r,c,n,p);
            comment rearrange columns;  mxperm(a[p,j],a[p,k],j,k,c,r,n,p)
end mxinvert;
```

LE16

```
procedure choleski(a,n,fail);
value n;  integer n;  array a;  label fail;
      begin integer i,j,k,iless1;  real aii;
            for i:= 1 step 1 until n do
                  begin iless1:= i-1;
                        aii:=a[i,i]-sigma(a[i,k]*a[i,k],k,1,iless1);
                        if aii ≤ 0 then goto fail;
                        aii:=a[i,i]:= sqrt(aii);
                        for j:= i+1 step 1 until n do
                            a[j,i]:= (a[j,i]-sigma(a[i,k]*a[j,k],k,1,iless1))/aii
                  end
      end choleski;
```

LE17

```
procedure linv(a,n);
value n;  integer n;  array a;
      begin integer i,j,k,iless1;  real aii;
            for i:=1 step 1 until n do
                  begin iless1:= i-1;  aii:=a[i,i];
                        for j:= 1 step 1 until iless1 do
                            a[i,j]:= -sigma(a[i,k]*a[k,j],k,j,iless1)/aii;
                        a[i,i]:= 1.0/aii
                  end
      end linv;
```

LE18

```
procedure mxmult(a,b,c,m,n,p); value m,n,p; integer m,n,p; array a,b,c;
comment c[m p]:= a[m n]  b[n p];
begin integer i,j,k;
      for i:= 1 step 1 until m do
            for j:= 1 step 1 until p do c[i,j]:=sigma(a[i,k]*b[k,j],k,1,n)
endmxmult;
```

LE24

```
procedure SYMDET(a,n,symdet,fail);  value n;  integer n;  real symdet;  array a;  label fail;
begin       integer i,j,k,iless1,ci,ii,cj,ij;
            real det,aii,aki,aij;
            det:=1.0;
            for i:=1 step 1 until n do
                  begin iless1:=i-1;  ci:=c[i];  ii:=i+ci;  aii:=a[ii];
                        for k:=1 step 1 until iless1 do
                              begin aki:=a[k+ci];  aii:=aii-aki*aki  end;
                                    if aii≤0.0 then goto fail;
                                    det:=det*aii;
                                    aii:=a[ii]:=sqrt(aii);
                                    for j:=i+1 step 1 until n do
                                          begin cj:=c[j];  ij:=i+cj;  aij:=a[ij];
                                                for k:=1 step 1 until iless1 do aij:=aij-a[k+ci]*a[k+cj];
                                                a[ij]:=aij/aii
                                          end j
                  end i;
            symdet:=det
end SYMDET;
```

LE25

```
procedure SYMSOL(a,b,n);  value n;  integer n;  real array a,b;
begin       integer i,j,jless1,cj;  real bi,bj;
            for j:=1 step 1 until n do
                  begin bj:=b[j];  jless1:=j-1;  cj:=c[j];
                        for i:=1 step 1 until jless1 do bj:=bj-a[i+cj]*b[i];
                        b[j]:=bj/a[j+cj]
                  end;
            for i:=n step -1 until 1 do
                  begin bi:=b[i];
                        for j:=i+1 step 1 until n do bi:=bi-a[i+c[j]]*b[j];
                        b[i]:=bi/a[i+c[i]]
                  end
end USYMSOL;
```

LE28

```
procedure SYMDET(a,n,symdet,fail);  value n;  integer n;  real symdet;  array a;  label fail;
begin       integer i,j,ki,ii,k1,kiless1,ij,kj;
            real det,aii,aki,aij;
            det:=1.0;  ii:=1;
            for i:=1 step 1 until n do
                  begin aii:=a[ii];  k1:=ii-i+1;  kiless1:=ii-1;
                        for ki:=k1 step 1 until kiless1 do
                              begin aki:=a[ki];  aii:=aii-aki*aki  end;
                        if aii≤0 then goto fail;
                        det:=det*aii;
                        aii:=a[ii]:=sqrt(aii);
                        ij:=ii+i;
                        for j:=i+1 step 1 until n do
                              begin aij:=a[ij];  kj:=ij-i+1;
                                    for ki:=k1 step 1 until kiless1 do
                                          begin aij:=aij-a[ki]*a[kj];  kj:=kj+1  end;
                                                a[ij]:=aij/aii;  ij:=ij+j

                              end;
                        ii:=ii+i+1;
                  end;
            symdet:=det
end SYMDET;
```

LE29

```
procedure SYMSOL(a,b,n);  value n;  integer n;  array a,b;
begin integer i,j,k,ii;  real sum;
      ii:=1;
      for i:=1 step 1 until n do
      begin k:=ii-1;  sum:=b[i];
            for j:=i-1 step -1 until 1 do
            begin sum:=sum-b[j]*a[k];  k:=k-1  end;
            b[i]:=sum/a[ii];  ii:=ii+i+1
      end forward solution;
      k:=ii-1;  ii:=ii-n-1;
      for i:=n step -1 until 1 do
      begin sum:=b[i];
            for j:=i+1 step 1 until n do
            begin sum:=sum-b[j]*a[k];  k:=k+j  end;
            b[i]:=sum/a[ii];  k:=ii-1;  ii:=ii-i
      end back substitution
end SYMSOL;
```

LL08

```
 procedure vecjacobi(a,s,n,rho);  value n,rho;  real rho;  integer n;  array a,s;
 comment an adaptation of ACM85 to evaluate the eigenvalues and
 eigenvectors of a real symmetric matrix A[1:n,1:n].  The upper
 triangle of A should be supplied, in vector form, stored by
 columns, in a[1:n  (n+1) / 2] so that A[i,j] occupies a[i+j  (j-1)  2].
 Alternatively, by symmetry, the lower triangle of A, stored by
 rows may occupy a.  At exit the eigenvalues occupy a[1] through
 a[n], with corresponding eigenvectors in the columns of s[1:n,1:n].
 rho is the precision tolerance as used in ACM85, which,
 in practice, should not be less than the relative machine precision.
 On a test matrix of order 20, using a tolerance of $_{10}$-6 vecjacobi
 proved to be 2.5 times faster than ACM85;
 begin integer array c[1:n];  integer i,j,ci,cj,p,q,cp,cq,jless1,qless1,ip,iq;
        switch ss:=main,main1;
        real fac,aij,thr,norm1,norm2,apq,app,aqq,m,mu,lambda,cost,sint,aip,aiq,
        sip,siq,sincos,cs45;
        boolean ind;
        cs45:=1.0/sqrt(2.0);
        p:=0;  fac:=0.0;
        for i:=1 step 1 until n do
        begin s[i,i]:=1.0;  c[i]:=p;  p:=p+i;
              for j:=i+1 step 1 until n do s[i,j]:=s[j,i]:=0.0;
        end;
        for j:=2 step 1 until n do
        begin cj:=c[j];  jless1:=j-1;
              for i:=1 step 1 until jless1 do
              begin aij:=a[i+cj];  fac:=fac+2.0*aij*aij  end
        end;
        thr:=norm1:=sqrt(fac);  norm2:=rho*norm1/n;
main: thr:=thr/n;
main1: ind:=false;
       for q:=2 step 1 until n do
       begin cq:=c[q];  qless1:=q-1;
             for p:=1 step 1 until qless1 do
             begin apq:=a[p+cq];
                   if abs(apq) > thr then
                   begin cp:=c[p];  ind:= true;
                         app:=a[p+cp];  aqq:=a[q+cq];  m:=app-aqq;
                         mu:=abs(m);
                         if mu<rho then cost:=sint:=cs45
                                   else begin lambda:=sign(m)*apq;  mu:=0.5*mu;
                                              fac:=0.5/sqrt(lambda*lambda+mu*mu);
                                              cost:=sqrt(0.5+mu*fac);
                                              sint:=lambda*fac/cost
                                        end;
                         for i:=1 step 1 until n do
                         begin ci:=c[i];
                               if i < p then begin ip:=i+cp;  iq:=i+cq end
                                        else begin ip:=p+ci;
                                                   iq:=if i>q then q+ci else i+cq
                                             end;
                                    aip:=a[ip];  aiq:=a[iq];
                                    sip:=s[i,p];  siq:=s[i,q];
                                    s[i,p]:=cost*sip+sint*siq;
                                    s[i,q]:=sint*sip-cost*siq;
                                    a[ip]:=cost*aip+sint*aiq;
                                    a[iq]:=sint*aip-cost*aiq
                               end i;
                         sincos:=sint*cost;  fac:=(apq+apq)*sincos;
                         sint:=sint*sint;  cost:=cost*cost;
                         a[p+cp]:=cost*app+sint*aqq+fac;
                         a[q+cq]:=sint*app+cost*aqq-fac;
                         a[p+cq]:=0.0
                         end
                   end
             end;
             if ind then goto main1 else if thr>norm2 then goto main;
             for i:=2 step 1 until n do a[i]:=a[i+c[i]]
end vecjacobi;
```

MT03

```
real procedure equipol(xbase,y,arg,n,m,h);  value xbase,arg,m,n,h;  real xbase,arg,h;  array y;  integer m,n;
      begin integer i,j,mless1;  real jh,fi;  array f[0:m];
            if m>n then m:=n;  i:=entier((arg-xbase)/h)-m div 2;
            j:= if i<0 then 0 else if i+m>n then n-m else i;
            for i:= 0 step 1 until m do f[i]:=y[i+j];
            arg:=arg-j*h-xbase;
            mless1:=m-1;
            for i:=0 step 1 until mless1 do
                  begin fi:=f[i];  jh:=h;
                        for j:=i+1 step 1 until m do
                              begin f[j]:=fi+arg*(f[j]-fi)/jh;  jh:=jh+h  end ;
                        arg:=arg-h
                  end;
            equipol:=f[m]
      end equipol;
```

MT04

```
real procedure ait(z,f,arg,n);  value arg,n;  integer n;  real arg;  array z,f;
      begin integer i,j,nless1;  real fi,zi,u;  nless1:=n-1;
            for i:=0 step 1 until nless1 do
                  begin fi:=f[i];  zi:=z[i];  u:=arg-zi;
                        for j:=i+1 step 1 until n do
                        f[j]:=fi+u*(f[j]-fi)/(z[j]-zi)
                  end;
            ait:=f[n]
      end;
```

MT06

```
real procedure equidydx(xbase,y,arg,n,m,h,est);  value xbase,arg,m,n,h;  real xbase,arg,est,h;  array y;  integer m,n;
      begin integer i,j,mless1;  real jh,fi,diffi,fjfi;  array f,diff[0:m];
            if m>n then m:=n;  i:=entier((arg-xbase)/h)-m div 2;
            j:= if i<0 then 0 else if i+m>n then n-m else i;
            for i:= 0 step 1 until m do begin f[i]:=y[i+j];  diff[i]:=0.0 end;
            arg:=arg-j*h-xbase;
            mless1:=m-1;
            for i:=0 step 1 until mless1 do
                  begin fi:=f[i];  jh:=h;  diffi:=diff[i];
                        for j:=i+1 step 1 until m do
                              begin fjfi:=f[j]-fi;
                                    diff[j]:=diffi+(fjfi+arg*(diff[j]-diffi))/jh;
                                    f[j]:=fi+arg*fjfi/jh;  jh:=jh+h
                             end ;
                        arg:=arg-h
                  end;
            est:=f[m];
            equidydx:=diff[m]
      end equidydx;
```