Raabo.

A/S REGNECENTRALEN
SCANDINAVIAN INFORMATION PROCESSING SYSTEMS

SYSTEM LIBRARY

RCSL NO: 55-D61

TYPE : Algol 5 Procedure

AUTHOR : Peter Fleron

EDITION: September 1969 (E)

RC 4000 SOFTWARE

MATHEMATICAL PROCEDURE LIBRARY

jacobi

ABSTRACT

jacobi calculates all the eigenvalues and, if wanted, the corresponding eigenvectors of a symmetric matrix by the method of Jacobi.

1. Function and Parameters.

Jacobi calculates all the eigenvalues and, if desired, the corresponding eigenvectors of a symmetric matrix by the method of Jacobi.

Procedure head:

```
real procedure jacobi(a,lambda,x,vect,maxscan);
value vect,maxscan;
array a,lambda,x;
boolean vect;
integer maxscan;
```

Call parameters:

a	: a real array containing the given matrix.
vect	: (boolean) . If vect is true, the eigenvectors will be calculated.
maxscan	: (integer or real). If the value of maxscan is > 0, at most this number of scans are performed in the procedure. If the value is 0, no limitation is imposed on the number of scans. (see section 2. Method).

Return parameters:

jacobi	: (real). Contains the number of rotations in the last two bytes and the number of scans in the first two bytes. If the procedure exits because the maximum number of scans is reached, the negative number of scans is stored. Hence the sign of the procedure value reveals its success.
lambda	: (real array). Contains on exit the calculated eigenvalues.
x	: (real array). If the eigenvectors are wanted, they are stored as column-vectors in x. The eigenvector associated with the eigenvalue lambda(i) is stored in x(.,i).

Parameter check:

The orders of the matrix a and the vectors x and lambda are not given as parameters, but are checked by the procedure in this way:

First, the upper subscript bound of the array lambda is assigned to the order. Then it is checked that the arrays are declared

```
a(1:order,1:order)            (1)
lambda(1:order)               (2)
x(1:order,1:order)            (4)
```

(this last check is performed only if the eigenvectors are wanted; in fact, if vect=false, x can be any real array and is never touched).

In case of error, the execution is terminated by the error-message on current output

```
jacobi     <error-number>
```

where <error-number> is the sum of the numbers attached to each wrong array as stated above. Thus the declaration

```
a(1:order,0:order)
lambda(1:order)
x(1:order)
```

yields the error-number 5 if vect=true, otherwise 1.
This initial check of the parameters implies that there is no need for index-check in the procedure.

2. Method.

The method consists of a number of scans of all the super-diagonal elements of the matrix. If the element in question is greater in absolute value than a certain threshold (approximately the current root-mean-square of all super-diagonal elements), a rotation is performed, so designed that this element becomes zero.
The exit condition is that the current threshold is less than $5_{10}-13\times$initial threshold (or that the maximum number of scans is reached).
Since the procedure converges at least quadratically, little time is saved by reducing the accuracy.
Just before exit, the super- and main-diagonal elements are reestablished so that the matrix is unchanged on exit.
For further details, see [1].

3. Accuracy, Time and Storage Requirement.

Accuracy: The relative error of the eigenvalues and, if the eigenvectors are calculated, the greatest element of (xt×x-I) is unlikely to exceed n×(the relative machine accuracy, appr. 3₁₀-11). This applies also to the greatest element of (a×x-x×L)/max(lambda). However, if the magnitudes of the eigenvalues are highly different it may happen that the eigenvalues of low magnitude are determined less accurate. It can be shown (see [1]) that the absolute error of the eigenvalues is bounded by

2×||L||/sqrt(1-nI)×(nI/(1-sqrt(1-nI))+nA),

where

L = diag(lambda)

nI = ||xt×x-I|| ; xt is x transposed.

nA = ||a×x-x×L||/||L||.

Time : Generally proportional to n××3 when n is large (see section 4. Test and Discussion).

Storage requirement: 5 segments of program

18 local real variables.

4. Test and Discussion.

Several matrices have been tried by the test program (or slightly modified versions) at the end of this section.

The table below shows the type and order of the matrix in question, the number of scans, the number of rotations, the time consumed by the procedure (in sec.), and the norms nI and nA as defined in section 3. (the infinity-norm is used.)

type	n	scan	rotation	time	nI	nA
a	10	14	180	1.9	1.7₁₀-9	8.3₁₀-10
a	20	17	796	14.6	7.1₁₀-9	2.1₁₀-9
b	15	12	327	4.8	1.2₁₀-9	9.4₁₀-10
c	9	4	12	.15	2.5₁₀-10	9.2₁₀-11
d	8	11	69	.68	7.6₁₀-10	6.1₁₀-10

The types represent the following matrices:

a) HBH-matrices. The general element of a n-th order matrix is given by a(i,j)=a(j,i)=n-i+1. The eigenvalues are

```
l(i)=0.25/sin((2×i-1)×pi/(4×n+2))××2
```

b) The matrix

```
a(i,j)=a(j,i)=if i<>j then 1 else 10×(i-1)
```

c) The matrix

```
a(i,j)=a(j,i)=if i=j then 0 else 1
```

All eigenvalues are -1 except for one which is n-1.

d) The matrix and the complete output of the testprogram for this case are:

```
matrix:
  611
  196  899
 -192  113  899
  407 -192  196  611
   -8  -71   61    8  411
  -52  -43   49   44 -599  411
  -49   -8    8   59  208  208   99
   29  -44   52  -23  208  208 -911   99

scans=    11    rotations=    69    time=   0.68 sec.
nI = 7.6⏨-10     nA = 6.1⏨-10

eigenvalues:
 1.0200490189⏨  3
 1.0000000002⏨  3
 1.0000000003⏨  3
 9.8048646596⏨ -2
 2.5609435927⏨ -9
 1.0200000004⏨  3
 1.0199019515⏨  3
-1.0200490187⏨  3

end
```

Test program:

```
test1 jacobi
begin integer n;
  for underflows:=-1 while read(in,n)>0 do
  begin real x,eli,ela,maxl,normi,norma,la,layout,t0,t1,si,sa;
    array a,t(1:n,1:n),l(1:n);
    integer i,j,k,layno;

    read(in,layno);
    for i:=1 step 1 until n do
    for j:=1 step 1 until i do
    begin read(in,x);
      a(i,j):=a(j,i):=x
    end;
    t0:=systime(1,0,la);
    for i:=0,i+1 while t1<t0+2.56 do
    begin x:=jacobi(a,l,t,true,0);
      t1:=systime(1,0,la)
    end;
    t0:=(t1-t0)/i;

    maxl:=normi:=norma:=0;
    for i:=1 step 1 until n do
    begin si:=sa:=0;
      for j:=1 step 1 until n do
      begin eli:=ela:=0;
        la:=l(j);
        for k:=1 step 1 until n do
        begin eli:=eli+t(k,i)×t(k,j);
          ela:=ela+a(i,k)×t(k,j)
        end k;
        if i=j then eli:=eli-1;
        ela:=ela-t(i,j)×la;
        si:=si+abs eli;
        sa:=sa+abs ela
      end j;
      if si>normi then normi:=si;
      if sa>norma then norma:=sa;
      if abs l(i)>maxl then maxl:=abs l(i)
    end i;

    layout:=real(case layno of(<<d>,<<dd>,<<-dd>,<<-ddd>,<<-dddd>,
                 <<d>,<<d>,<< -d.dddd>,<< -d.ddddd>));
    write(out,<:<12>matrix::>);
    for i:=1 step 1 until n do
    begin k:=0;
      write(out,<:<10>:>);
      for j:=1 step 1 until i do
      begin write(out,string layout,a(i,j));
        k:=k+layno;
        if k>70 and j<i then
        begin write(out,<:<10>:>,false add 32,layno);
          k:=layno
        end
      end
    end write matrix;
    write(out,<:<10><10>scans=:>,<< -dddd>,x shift(-24) extract 24,
       <:   rotations=:>,x extract 24,
       <:   time=:>,<< ddd.00>,t0,<: sec.:>,
       <:<10>nI =:>,<< d.d⏨-dd>,normi,
       <:    nA =:>,norma/maxl,
       <:<10><10>eigenvalues:<10>:>);
    for i:=1 step 1 until n do
      write(out,<<-d.dddddddddd⏨-dd>,l(i),<:<10>:>)
  end read n
end
```

5. References

[1] Kahan, W. and Green, D. : Eigenvalues and Eigenvectors of a Real Symmetric Matrix. (Unpublished but copies of the paper are achievable on demand)

6. Algorithm

```
jacobi=set 5
jacobi=algol index.no

external
real procedure jacobi(a,lambda,x,vect,maxscan);
message jacobi, version 20.11.69, RCSL NO: 55-D61;
value vect,maxscan;
integer maxscan;
boolean vect;
array a,lambda,x;
begin real eps,t,ave,s,u,thresh,dlow,d,c,aij,ajj;
  integer i,j,ii,jj,jl,n,nrscan,nrrot;
  boolean again;

  i:=if system(3,n,lambda)<>1 then 2 else 0;
  j:=system(3,ii,a);
  if j<>n+1 or ii<>n×(n+1) then i:=i+1;
  j:=system(3,ii,x);
  if (j<>n+1 or ii<>n×(n+1)) and vect then i:=i+4;
  if i>0 then system(9,i,<:<10>jacobi  :>);

  if vect then
  for i:=1 step 1 until n do
  begin x(i,i):=1;
    for j:=i+1 step 1 until n do x(i,j):=x(j,i):=0
  end x:=identity;

  d:=0;
  for i:=1 step 1 until n do
  begin lambda(i):=a(i,i);
    for j:=i+1 step 1 until n do d:=d+a(i,j)××2
  end i;

  nrscan:=nrrot:=0;
  if d>0 then
  begin dlow:=₁₀-7×d;
    ave:=(n-1)×n×0.55;
    thresh:=sqrt(d/ave);
    eps:=5₁₀-13×thresh;

scan:again:=false;
    nrscan:=nrscan+1;
    for i:=n-1 step -1 until 1 do
    for j:=i+1 step 1 until n do
    begin comment scan;
      aij:=a(i,j);
      if abs aij >= thresh then
      begin ajj:=a(j,j);
        s:=ajj-a(i,i);
        t:=abs aij;
        if s+t<>s then
        begin comment rot<>0;
          again:=true;
          nrrot:=nrrot+1;
          if abs s<=₁₀-6×t then s:=c:=0.7071067811 8 else
          begin t:=aij/s;
            s:=0.25/sqrt(t××2+0.25);
            c:=sqrt(s+0.5);
            s:=2×t×s/c
          end rot<>pi/4;
          for ii:=1 step 1 until i do
          begin t:=a(ii,i); u:=a(ii,j);
            a(ii,i):=c×t-s×u;
            a(ii,j):=s×t+c×u
          end;
          j1:=j-1;
          for ii:=i+1 step 1 until j1 do
          begin t:=a(i,ii); u:=a(ii,j);
            a(i,ii):=c×t-s×u;
            a(ii,j):=s×t+c×u
          end;
          a(j,j):=s×aij+c×ajj;
          a(i,i):=c×a(i,i)-s×(c×aij-s×ajj);
          for ii:=j step 1 until n do
          begin t:=a(i,ii); u:=a(j,ii);
            a(i,ii):=c×t-s×u;
            a(j,ii):=s×t+c×u
          end;

          if vect then
          for ii:= 1 step 1 until n do
          begin t:=x(ii,i); u:=x(ii,j);
            x(ii,i):=c×t-s×u;
            x(ii,j):=s×t+c×u
          end;

          d:=d-aij××2;
          if d<dlow then
          begin d:=0;
            for ii:=n-1 step -1 until 1 do
            for jj:=ii+1 step 1 until n do
              d:=d+a(ii,jj)××2;
            dlow:=_n-7×d
          end;

          thresh:=sqrt(d/ave);
          if thresh<eps then goto quit
        end rotation
      end aij
    end scan;
    if again and (maxscan>0=>maxscan>nrscan) then goto scan;

    if again then nrscan:=-nrscan;

quit:for i:=1 step 1 until n do
    begin t:=a(i,i);
      a(i,i):=lambda(i);
      lambda(i):=t;
      for j:=i+1 step 1 until n do a(i,j):=a(j,i)
    end i
  end d>0;
  jacobi:=0.5 add nrscan shift 24 add nrrot
end jacobi;


comment
  Call parameters:
          a       : a real array containing the given matrix.
       vect       : (boolean) .  If vect is true, the eigenvectors
                    will be calculated.
    maxscan       : (integer or real). If the value of maxscan is > 0,
                    at most this number of scans are performed in the
                    procedure. If the value is 0, no limitation is imposed
                    on the number of scans.
```

Return parameters:

jacobi	: (real). Contains the number of rotations in the last two bytes and the number of scans in the first two bytes. If the procedure exits because the maximum number of scans is reached, the negative number of scans is stored. Hence the sign of the procedure value reveals its success.
lambda	: (real array). Contains on exit the calculated eigenvalues.
x	: (real array). If the eigenvectors are wanted, they are stored as column-vectors in x. The eigenvector associated with the eigenvalue lambda(i) is stored in x(.,i);