Numerische Mathematik 9, 1—10 (1966)

Handbook Series Linear Algebra

The Jacobi Method for Real Symmetric Matrices*

Contributed by
H. RUTISHAUSER

Background

As is well known, a real symmetric matrix can be transformed iteratively into diagonal form through a sequence of appropriately chosen *elementary orthogonal transformations* (in the following called *Jacobi rotations*):

$$A_k \to A_{k+1} = U_k^T A_k U_k \qquad (A_0 = \text{given matrix}),$$

where $U_k = U_k(p, q, \varphi)$ is an orthogonal matrix which deviates from the unit matrix only in the elements

$$u_{pp} = u_{qq} = \cos(\varphi) \quad \text{and} \quad u_{pq} = -u_{qp} = \sin(\varphi).$$

On the whole the Jacobi method performs (approximately) the operation

$$A \to D = V^T A V,$$

where D is diagonal and V is orthogonal, D being the limit of A_k for $k \to \infty$, while V is the product of all Jacobi rotations which were used for achieving the diagonalisation:

$$V = U_0 U_1 U_2 U_3 \ldots .$$

The main virtues of the Jacobi method stem from the fact that the Jacobi rotations produce a systematic decrease of the sum of the squares of the off-diagonal elements; the convergence of this sum to zero with increasing k guarantees the convergence of the diagonalisation process. Various schemes have been developed:

JACOBI [4] inspected the matrix A_k for the largest off-diagonal element a_{pq}[1] and then chose the rotation angle φ such that in the matrix A_{k+1} the p, q-element vanished. After the rediscovery of the method in 1952 [2], when for the first time it was used in automatic computing, it was applied in such a way that p, q ran row-wise through all superdiagonal positions of the matrix, and again the rotation angle φ was chosen every time so as to annihilate the p, q-element in the matrix A_{k+1}. Later POPE and TOMPKINS [5] suggested a strategy which tended to avoid inefficient rotations and thus achieved diagonalisation with less effort.

* *Editor's note.* In this fascicle, prepublication of algorithms from the Linear Algebra series of the Handbook for Automatic Computation is continued. Algorithms are published in ALGOL 60 reference language as approved by the IFIP. Contributions in this series should be styled after the most recently published ones.

[1] Because of the symmetry of the matrices involved, only the matrix elements in and above the diagonal need be taken into consideration.

Proofs as well as estimates for the convergence properties of the Jacobi method have been given by HENRICI [3], SCHOENHAGE [7] and WILKINSON [8]. As these papers show, both the original Jacobi method as well as the more computer-oriented method with row-wise scanning of the upper triangle have *quadratic convergence*, and this is what makes the Jacobi method interesting.

Applicability

The present algorithm — which essentially uses row-wise scanning of the upper triangle — can be used and is absolutely foolproof for all real symmetric matrices. It is designed such that it requires no tolerance limit for termination but proceeds just as long as it is both necessary and meaningful on the computer which performs the calculation[2]. Usually this will take of the order of 6 to 10 sweeps, i.e. from $3n^2$ to $5n^2$ Jacobi rotations.

On the other hand it should be recognized that while the Jacobi process is more compact and elegant and also produces eigenvectors without difficulties, it is also more time-consuming than the combination of the Householder-transformation with a method for computing eigenvalues of tridiagonal matrices (e.g. the combination of the procedures *householder tridiagonalisation* [9], *tridibisection1* [10] or *bisect* [1], *tridiinverse* [11] and *backsubstitution* [9] as published already in this Handbook). For large-order matrices, therefore, the use of the latter procedures in place of procedure *jacobi* may be more appropriate. Moreover, the Jacobi method is comparatively uneconomical for symmetric bandmatrices whose bandwidth is small compared to n; for matrices of this kind the LR-transformation [6] will be more efficient.

Formal Parameters

a) quantities to be given:

n the order of the matrix A.

eivec **true**, if eigenvectors are desired, otherwise **false**.

a **array** $a[1:n, 1:n]$ containing the elements of the matrix A. To be precise, only the diagonal and superdiagonal elements (the $a[i, k]$ with $k \geqq i$) are actually used.

b) results produced by the procedure:

a The superdiagonal elements of the array *a* are destroyed by the process, but the diagonal and subdiagonal elements are unchanged (therefore full information on the given matrix A is still contained in array *a*, if the subdiagonal elements had also been filled in).

d The diagonal elements $d[i]$ of the matrix D, i.e. the approximate eigenvalues of A.

[2] Well-timed termination of the present algorithm requires that the Boolean expression $a + eps = a$ with *a* and *eps* non-negative have the value **true** provided *eps* is small compared to *a* within the computer accuracy. For computers in which this condition is not fulfilled the termination of the algorithm may be be delayed. Furthermore, procedure *jacobi* may produce erroneous results with computers which upon underflow produce values other than zero.

v **array** $v[1:n, 1:n]$ containing (provided *eivec* = **true**) the elements of the matrix V, the k.th column of V being the normalized eigenvector to the eigenvalue $d[k]$.[3]

rot The number of Jacobi rotations which have been needed to achieve the diagonalisation.

c) An example of application:

Since procedure *jacobi* does not order the computed eigenvalues, such ordering, if desired, must be done by the user of the procedure, e.g.

```
comment n, a[i, k] are assumed as being given;
begin
    integer array r[1:n]; integer aux;
    jacobi(n, true, a, d, v, rot);
    for k := 1 step 1 until n do r[k] := k;
    for k := 1 step 1 until n - 1 do
        for l := k+1 step 1 until n do
            if d[r[k]] < d[r[l]] then
              begin aux := r[k]; r[k] := r[l]; r[l] := aux end;
comment: now d[r[k]] is the k.th eigenvalue in descending order and v[j, r[k]]
         is the j.th component of the corresponding eigenvector;
```

The ALGOL-Program

```
procedure jacobi(n, eivec) trans: (a) res: (d, v, rot);
  value n, eivec;
  integer n, rot; boolean eivec; array a, d, v;
  begin
      real sm, c, s, t, h, g, tau, theta, tresh;
      integer p, q, i, j;
      array b, z[1:n];
program:
      if eivec then
        for p := 1 step 1 until n do
            for q := 1 step 1 until n do
                v[p, q] := if p=q then 1.0 else 0.0;
      for p := 1 step 1 until n do
            begin b[p] := d[p] := a[p, p]; z[p] := 0 end;
      rot := 0;
      for i := 1 step 1 until 50 do
swp:
      begin
        sm := 0;
```

[3] Note that even if no eigenvectors are desired (*eivec* = **false**), the rules of ALGOL all the same require that v be declared, e.g. as **array** $v[1:1, 1:1]$.

```
            for p := 1 step 1 until n − 1 do
                for q := p + 1 step 1 until n do
                    sm := sm + abs(a[p, q]);
            if sm = 0 then goto out;
            tresh := if i < 4 then 0.2 × sm/n↑2 else 0.0;
            for p := 1 step 1 until n − 1 do
                for q := p + 1 step 1 until n do
                begin
                    g := 100 × abs(a[p, q]);
                    if i > 4 ∧ abs(d[p]) + g = abs(d[p]) ∧
                      abs(d[q]) + g = abs(d[q]) then a[p, q] := 0
                    else
                      if abs(a[p, q]) > tresh then
rotate:               begin
                        h := d[q] − d[p];
                        if abs(h) + g = abs(h) then t := a[p, q]/h
                        else
                          begin
                            theta := 0.5 × h/a[p, q];
                            t := 1/(abs(theta) + sqrt(1 + theta↑2));
                            if theta < 0 then t := −t
                          end computing tan of rotation angle;
                        c := 1/sqrt(1 + t↑2);
                        s := t × c;
                        tau := s/(1 + c);
                        h := t × a[p, q];
                        z[p] := z[p] − h;
                        z[q] := z[q] + h;
                        d[p] := d[p] − h;
                        d[q] := d[q] + h;
                        a[p, q] := 0;
                        for j := 1 step 1 until p − 1 do
                        begin
                          g := a[j, p]; h := a[j, q];
                          a[j, p] := g − s × (h + g × tau);
                          a[j, q] := h + s × (g − h × tau)
                        end of case 1 ≦ j < p;
                        for j := p + 1 step 1 until q − 1 do
                        begin
                          g := a[p, j]; h := a[j, q];
                          a[p, j] := g − s × (h + g × tau);
                          a[j, q] := h + s × (g − h × tau)
                        end of case p < j < q;
                        for j := q + 1 step 1 until n do
                        begin
                          g := a[p, j]; h := a[q, j];
                          a[p, j] := g − s × (h + g × tau);
```

```
                    a[q, j] := h+s×(g−h×tau)
                  end of case q<j≦n;
                  if eivec then
                    for j := 1 step 1 until n do
                    begin
                      g := v[j, p];  h := v[j, q];
                      v[j, p] := g−s×(h+g×tau);
                      v[j, q] := h+s×(g−h×tau)
                    end of case v;
                  rot := rot+1;
                end rotate;
            end;
          for p := 1 step 1 until n do
          begin
            d[p] := b[p] := b[p]+z[p];
            z[p] := 0
          end p
        end swp;
out:
    end jacobi;
```

Organisational and Notational Details

The pivots of the Jacobi rotations, i.e. the elements $a[p, q]$ which are annihilated by the rotations, are chosen in a row-wise pattern, namely

```
for p := 1 step 1 until n−1 do
  for q := i+1 step 1 until n do
    rotate in rows and columns p and q;
```

Such a sequence of $\binom{n}{2}$ Jacobi rotations is called one *sweep*; immediately after completion of one sweep, the next sweep is begun, etc., until finally the diagonalisation is completed.

However, the present program contains the following additional features:

a) During the first three sweeps it performs only those rotations for which

$$abs(a[p, q]) > tresh \quad (=0.2 \times sm/n\uparrow 2),$$

where *sm* is the sum of the moduli of all superdiagonal elements. In the later sweeps *tresh* is set to zero.

b) If before the p, q-rotation the element $a[p, q]$ is small compared to $a[p, p]$ *and* small compared to $a[q, q]$, then $a[p, q]$ is set to zero and the p, q-rotation is skipped.

This is certainly meaningful since it produces no larger error than would be produced anyhow if the rotation had been performed; however, in order that the procedure can be used for computing eigenvectors of perturbed diagonal[4] matrices, this device is suppressed during the first four sweeps.

[4] It should be recognized that this feature does not work for perturbed non-diagonal matrices.

c) In order to annihilate the p, q-element, the rotation parameters c, s are computed as follows: Compute first the quantity

$$theta = cot(2\varphi) = \frac{a[q,q]-a[p,p]}{2a[p,q]};$$

then $tan(\varphi)$ as the smaller root (in modulus) of the equation

$$t^2+2t\times theta = 1 \quad (\text{or } t=0.5/theta, \text{ if } theta \text{ is large}),$$

and thereof $c=cos(\varphi)$, $s=sin(\varphi)$ and $tau=tan(\varphi/2)$, these latter entering into the rotation formulae.

d) With these features, the procedure will sooner or later make all superdiagonal elements zero (machine representation of zero), whereupon the process is discontinued.

e) *jacobi* does not actually operate on the diagonal elements of the array a but transfers them at the beginning into an array $d[1:n]$ and then performs all operations on $d[x]$ instead of upon $a[x, x]$.

Numerical Properties

The present program attempts to diminish the accumulation of roundoff-errors as produced by the many Jacobi rotations needed for diagonalisation. This is achieved by the following measures:

a) It does not use the usual formulae

$$a_{pp}^{(new)} = c^2\times a_{pp} - 2cs\times a_{pq} + s^2\times a_{qq},$$
$$a_{qq}^{(new)} = s^2\times a_{pp} + 2cs\times a_{pq} + c^2\times a_{qq},$$
$$a_{pq}^{(new)} = (c^2-s^2)\times a_{pq} + cs\times(a_{pp}-a_{qq})$$

for computing the p, p-, q, q- and p, q-elements of the matrix A_{k+1}, but the equivalent formulae

$$a_{pp}^{(new)} = a_{pp} - t\times a_{pq},$$
$$a_{qq}^{(new)} = a_{qq} + t\times a_{pq},$$
$$a_{pq}^{(new)} = 0.$$

b) Likewise also the formulae for the off-diagonal elements (and similarly for the components of the matrix V) are modified, e.g.

$$a_{pj}^{(new)} = c\times a_{pj} - s\times a_{qj},$$
$$a_{qj}^{(new)} = s\times a_{pj} + c\times a_{qj}$$

into the equivalent formulae

$$a_{pj}^{(new)} = a_{pj} - s\times(a_{qj} + tau\times a_{pj}),$$
$$a_{qj}^{(new)} = a_{qj} + s\times(a_{pj} - tau\times a_{qj}),$$

where $tau = tan(\varphi/2) = s/(1+c)$.

c) The terms $t \times a_{pq}$, by which the diagonal elements are changed, are accumulated separately (in an array $z[1:n]$), and only at the end of every sweep are the accumulated increments of all diagonal elements used to compute new (and better) values of the latter.

Now, according to WILKINSON [12], the total error incurred in the diagonalisation of a symmetric matrix is for every eigenvalue at most $E = 18.2 n^{1.5} r \|A_0\| \Theta$ ($\| \| =$ Schur Norm), where r is the number of sweeps required to perform the diagonalisation, and Θ is the smallest positive number such that in the computer $1 + \Theta \neq 1$.

This error estimate is influenced by the measures described above only insofar as the contributions of the roundoff-errors become insignificant as soon as the off-diagonal elements of the iterated matrix A_k become small as compared to the diagonal elements. Thus with the present algorithm we can in fact substitute a smaller r, 3 say, in the above formula for E.

Results of Tests

The following tests have been performed with procedure *jacobi* on the CDC 1604-A computer[5] of the Swiss Federal Institute of Technology, Zurich:

a) Matrix A with $a[i, k] = max(i, k)$. The following table gives some of the eigenvalues found for $n = 30$:

1) with the present procedure *jacobi*, requiring a total of 2339 Jacobi rotations.

2) With another procedure for the Jacobi method, not containing any special measures against the accumulation of roundoff-errors.

3) With the qd-algorithm, using the fact that $-A^{-1}$ is a tridiagonal matrix corresponding to the qd-line

$$q_k = e_k = 1 \qquad (k = 1, 2, \ldots, n-1),$$

$$q_n = -1/n, \qquad e_n = 0.$$

The eigenvalues corresponding to this qd-line have been computed with the ALCOR multiple precision procedures and rounded correctly to the number of digits given:

	λ_1	λ_2	λ_3
1)	639.629 434 44	−0.250 687 020 21	−0.252 763 251 41
2)	639.629 435 87	−0.250 687 021 37	−0.252 763 252 02
3)	639.629 434 44	−0.250 687 020 23	−0.252 763 251 51

	λ_{16}	λ_{29}	λ_{30}
1)	−0.500 273 498 39	−24.077 530 173	−114.511 176 46
2)	−0.500 273 500 09	−24.077 530 237	−114.511 176 74
3)	−0.500 273 498 45	−24.077 530 172	−114.511 176 46.

Thus the present procedure *jacobi* is for all eigenvalues superior to the older Jacobi-program, and the actual errors are far below the estimate given in the

[5] The 1604-A has a 36-bit mantissa and a binary exponent ranging from −1024 to 1023.

foregoing section (numerical properties), which would be $E=9_{10}-5$ in this example.

b) The matrix $B=8J-5J^2+J^3$ of order 44, where J denotes the tridiagonal matrix with $j[i,i]=2$, $j[i,i+1]=1$, i.e.

$$B=\begin{pmatrix}
5 & 2 & 1 & 1 & & & & & & & \\
2 & 6 & 3 & 1 & 1 & & & & & & \\
1 & 3 & 6 & 3 & 1 & 1 & & & & & \\
1 & 1 & 3 & 6 & 3 & 1 & 1 & & & & \\
 & 1 & 1 & 3 & 6 & 3 & 1 & 1 & & & \\
 & & & & & & 1 & 1 & 3 & 6 & 2 \\
 & & & & & & & 1 & 1 & 2 & 5
\end{pmatrix}$$

Eleven of the eigenvalues of this matrix are in the interval $4 \leqq \lambda \leqq 4.163$, the length of which is 1% of the total length of the spectrum:

4.162 503 824 429 765
4.145 898 033 750 315
4.140 771 754 112 427
4.120 834 653 401 856
4.094 745 370 081 830
4.078 725 692 426 601
4.052 841 589 388 415
4.034 568 007 676 363
4.005 211 953 160 050
4.004 531 845 800 653
4

The following table contains for 6 eigenvalues and the first five components of the corresponding eigenvectors:

1) The values computed by procedure *jacobi*, requiring a total of 6260 Jacobi rotations.

2) The values computed by another procedure for the Jacobi method, as in the previous example.

3) The precise values, correctly rounded to the given number of digits.

Eigenvalues:

	λ_1	λ_{15}	λ_{28}
1)	15.922 215 640	6.000 000 0001	4.005 211 9528
2)	15.922 215 715	6.000 000 0270	4.005 211 9740
3)	15.922 215 641	6	4.005 211 9532

	λ_{29}	λ_{30}	λ_{44}
1)	4.004 531 8456	4.000 000 0000	0.038 856 634 495
2)	4.004 531 8652	4.000 000 0193	0.038 856 634 615
3)	4.004 531 8458	4	0.038 856 634 457

Eigenvectors (components 1 through 5):

	v_1	v_{15}	v_{28}
1)	.014 705 95592	−.182 574 18584	−.210 690 17490
	.029 340 26590	−.182 574 18584	−.014 705 96163
	.043 831 63305	.000 000 00000	.209 663 71574
	.058 109 45688	.182 574 18579	.029 340 25522
	.072 104 17727	.182 574 18580	−.207 615 79602
2)	.014 705 95557	−.182 574 18615	−.210 690 00855
	.029 340 26517	−.182 574 18611	−.014 705 94683
	.043 831 63200	.000 000 00004	.209 663 53291
	.058 109 45555	.182 574 18622	.029 340 26251
	.072 104 17569	.182 574 18617	−.207 615 59730

	v_{29}	v_{30}	v_{44}
1)	.210 689 99816	−.182 574 18407	.014 705 95593
	−.014 705 96619	.182 574 18523	−.029 340 26589
	−.209 663 53377	.000 000 00164	.043 831 63301
	.029 340 28251	−.182 574 18485	−.058 109 45685
	.207 615 60979	.182 574 18745	.072 104 17724
2)	.210 690 16797	−.182 574 18989	.014 705 95596
	−.014 705 93265	.182 574 18996	−.029 340 26597
	−.209 663 72355	.000 000 00282	.043 831 63316
	.029 340 23626	−.182 574 19290	−.058 109 45703
	.207 615 81820	.182 574 18454	.072 104 17751

3) The precise values of the moduli of the numbers occurring among these components are (correctly rounded to the number of digits given)

.014 705 95590	.182 574 18584
.029 340 26587	.207 615 70379
.043 831 63301	.209 663 62482
.058 109 45684	.210 690 08574
.072 104 17724	

As these values indicate, procedure *jacobi* also produces better eigenvectors; only the vectors v_{28} and v_{29} corresponding to a very close pair of eigenvalues are slightly worse than with the older Jacobi process. However, the two vectors v_{28} and v_{29} computed by *jacobi* define the plane better than the corresponding vectors as produced by the earlier Jacobi procedure.

c) In order to check the performance of procedure *jacobi* when applied to perturbed diagonal matrices, the following matrix C of order 10 was tested, and satisfactory results were obtained:

$$c_{kk} = 1 - 10\uparrow(1-k),$$

$$c_{ik} = 10^{-12}, \quad \text{if} \quad i \neq k, \qquad i-k = \text{even},$$

$$c_{ik} = 10^{-15}, \quad \text{if} \quad i-k = \text{odd}.$$

A second test of procedure *jacobi* was performed by Dr. W. BARTH, Rechenzentrum der Technischen Hochschule, Darmstadt. The present author wishes to thank Dr. BARTH for this service.

References

[1] BARTH, W., and J. H. WILKINSON: The bisection method. Numer. Math. (to appear).

[2] GREGORY, R. T.: Computing eigenvalues and eigenvectors of a symmetric matrix on the ILLIAC. Math. Tab. and other Aids to Comp. **7**, 215–220 (1953).

[3] HENRICI, P.: On the speed of convergence of cyclic and quasicyclic Jacobi methods for computing eigenvalues of Hermitian matrices. J. Soc. Ind. Appl. Math. **6**, 144–162 (1958).

[4] JACOBI, C. G. J.: Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen. Crelle's Journal **30**, 51–94 (1846).

[5] POPE, D. A., and C. TOMPKINS: Maximizing functions of rotations-experiments concerning speed of diagonalisation of symmetric matrices using Jacobi's method. J. Ass. Comp. Mach. **4**, 459–466 (1957).

[6] RUTISHAUSER, H., and H. R. SCHWARZ: The LR-transformation method for symmetric matrices. Numer. Math. **5**, 273–289 (1963).

[7] SCHOENHAGE, A.: Zur Konvergenz des Jacobi-Verfahrens. Num. Math. **3**, 374–380 (1961).

[8] WILKINSON, J. H.: Note on the quadratic convergence of the cyclic Jacobi process. Numer. Math. **4**, 296–300 (1962).

[9] — Householder's method for symmetric matrices. Numer. Math. **4**, 354–361 (1962).

[10] — Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection. Numer. Math. **4**, 362–367 (1962).

[11] — Calculation of the eigenvectors of a symmetric tridiagonal matrix by inverse iteration. Numer. Math. **4**, 368–372 (1962).

[12] — The algebraic eigenvalue problem, 662 p. Oxford: Clarendon Press 1965.

Institut für Angewandte Mathematik
Eidgen. Technische Hochschule
Zürich/Schweiz