WIKIPEDIA

Jacobi eigenvalue algorithm

In numerical linear algebra, the **Jacobi eigenvalue algorithm** is an iterative method for the calculation of the eigenvalues and eigenvectors of a real symmetric matrix (a process known as diagonalization). It is named after Carl Gustav Jacob Jacobi, who first proposed the method in 1846,[1] but only became widely used in the 1950s with the advent of computers.[2]

Contents

Description

Convergence

Cost

Algorithm

- Notes
- Example

Applications for real symmetric matrices

Generalizations

References

Further reading

External links

Description

Let S be a symmetric matrix, and $G = G(i, j, \theta)$ be a Givens rotation matrix. Then:

$$S' = GSG^\top$$

is symmetric and similar to S.

Furthermore, S' has entries:

$$\begin{aligned}
S'_{ii} &= c^2\, S_{ii} - 2\, sc\, S_{ij} + s^2\, S_{jj} \\
S'_{jj} &= s^2\, S_{ii} + 2sc\, S_{ij} + c^2\, S_{jj} \\
S'_{ij} &= S'_{ji} = (c^2 - s^2)\, S_{ij} + sc\, (S_{ii} - S_{jj}) \\
S'_{ik} &= S'_{ki} = c\, S_{ik} - s\, S_{jk} && k \neq i, j \\
S'_{jk} &= S'_{kj} = s\, S_{ik} + c\, S_{jk} && k \neq i, j \\
S'_{kl} &= S_{kl} && k, l \neq i, j
\end{aligned}$$

where $s = \sin(\theta)$ and $c = \cos(\theta)$.

Since G is orthogonal, S and S' have the same Frobenius norm $||\cdot||_F$ (the square-root sum of

squares of all components), however we can choose θ such that $S'_{ij} = 0$, in which case S' has a larger sum of squares on the diagonal:

$$S'_{ij} = \cos(2\theta)S_{ij} + \tfrac{1}{2}\sin(2\theta)(S_{ii} - S_{jj})$$

Set this equal to 0, and rearrange:

$$\tan(2\theta) = \frac{2S_{ij}}{S_{jj} - S_{ii}}$$

if $S_{jj} = S_{ii}$

$$\theta = \frac{\pi}{4}$$

In order to optimize this effect, S_{ij} should be the off-diagonal element with the largest absolute value, called the *pivot*.

The Jacobi eigenvalue method repeatedly performs rotations until the matrix becomes almost diagonal. Then the elements in the diagonal are approximations of the (real) eigenvalues of *S*.

Convergence

If $p = S_{kl}$ is a pivot element, then by definition $|S_{ij}| \leq |p|$ for $1 \leq i, j \leq n, i \neq j$. Let $\Gamma(S)^2$ denote the sum of squares of all off-diagonal entries of S. Since S has exactly $2N := n(n-1)$ off-diagonal elements, we have $p^2 \leq \Gamma(S)^2 \leq 2Np^2$ or $2p^2 \geq \Gamma(S)^2/N$. Now $\Gamma(S^J)^2 = \Gamma(S)^2 - 2p^2$. This implies $\Gamma(S^J)^2 \leq (1 - 1/N)\Gamma(S)^2$ or $\Gamma(S^J) \leq (1 - 1/N)^{1/2}\Gamma(S)$, i.e. the sequence of Jacobi rotations converges at least linearly by a factor $(1 - 1/N)^{1/2}$ to a diagonal matrix.

A number of N Jacobi rotations is called a sweep; let S^σ denote the result. The previous estimate yields

$$\Gamma(S^\sigma) \leq \left(1 - \frac{1}{N}\right)^{N/2} \Gamma(S),$$

i.e. the sequence of sweeps converges at least linearly with a factor $\approx e^{1/2}$.

However the following result of Schönhage[3] yields locally quadratic convergence. To this end let S have m distinct eigenvalues $\lambda_1, \ldots, \lambda_m$ with multiplicities $\nu_1, \ldots, \nu_m$ and let $d > 0$ be the smallest distance of two different eigenvalues. Let us call a number of

$$N_S := \frac{n(n-1)}{2} - \sum_{\mu=1}^{m} \frac{1}{2}\nu_\mu(\nu_\mu - 1) \leq N$$

Jacobi rotations a Schönhage-sweep. If S^s denotes the result then

$$\Gamma(S^s) \leq \sqrt{\frac{n}{2} - 1}\left(\frac{\gamma^2}{d - 2\gamma}\right), \quad \gamma := \Gamma(S) .$$

Thus convergence becomes quadratic as soon as $\Gamma(S) < \frac{d}{2+\sqrt{\frac{n}{2}-1}}$

Cost

Each Jacobi rotation can be done in O(n) steps when the pivot element p is known. However the search for p requires inspection of all $N \approx ½\ n^2$ off-diagonal elements. We can reduce this to O(n) complexity too if we introduce an additional index array $m_1, \ldots, m_{n-1}$ with the property that m_i is the index of the largest element in row i, (i = 1, ..., n − 1) of the current S. Then the indices of the pivot (k, l) must be one of the pairs (i, m_i). Also the updating of the index array can be done in O(n) average-case complexity: First, the maximum entry in the updated rows k and l can be found in O(n) steps. In the other rows i, only the entries in columns k and l change. Looping over these rows, if m_i is neither k nor l, it suffices to compare the old maximum at m_i to the new entries and update m_i if necessary. If m_i should be equal to k or l and the corresponding entry decreased during the update, the maximum over row i has to be found from scratch in O(n) complexity. However, this will happen on average only once per rotation. Thus, each rotation has O(n) and one sweep O(n^3) average-case complexity, which is equivalent to one matrix multiplication. Additionally the m_i must be initialized before the process starts, which can be done in n^2 steps.

Typically the Jacobi method converges within numerical precision after a small number of sweeps. Note that multiple eigenvalues reduce the number of iterations since $N_S < N$.

Algorithm

The following algorithm is a description of the Jacobi method in math-like notation. It calculates a vector e which contains the eigenvalues and a matrix E which contains the corresponding eigenvectors, i.e. e_i is an eigenvalue and the column E_i an orthonormal eigenvector for e_i, i = 1, ..., n.

```
procedure jacobi(S ∈ R^(n×n); out e ∈ R^n; out E ∈ R^(n×n))
  var
    i, k, l, m, state ∈ N
    s, c, t, p, y, d, r ∈ R
    ind ∈ N^n
    changed ∈ L^n

  function maxind(k ∈ N) ∈ N ! index of largest off-diagonal element in row k
    m := k+1
    for i := k+2 to n do
      if |S_ki| > |S_km| then m := i endif
    endfor
    return m
  endfunc

  procedure update(k ∈ N; t ∈ R) ! update e_k and its status
    y := e_k; e_k := y+t
    if changed_k and (y=e_k) then changed_k := false; state := state-1
    elsif (not changed_k) and (y≠e_k) then changed_k := true; state := state+1
    endif
  endproc

  procedure rotate(k,l,i,j ∈ N) ! perform rotation of S_ij, S_kl
    [ S_kl ]     [ c  -s ] [ S_kl ]
    [      ] := [       ] [      ]
    [ S_ij ]     [ s   c ] [ S_ij ]
  endproc

  ! init e, E, and arrays ind, changed
```

```
  E := I; state := n
  for k := 1 to n do ind_k := maxind(k); e_k := S_kk; changed_k := true endfor
  while state≠0 do ! next rotation
    m := 1 ! find index (k,l) of pivot p
    for k := 2 to n−1 do
      if |S_k ind_k| > |S_m ind_m| then m := k endif
    endfor
    k := m; l := ind_m; p := S_kl
    ! calculate c = cos φ, s = sin φ
    y := (e_l−e_k)/2; d := |y|+√(p²+y²)
    r := √(p²+d²); c := d/r; s := p/r; t := p²/d
    if y<0 then s := −s; t := −t endif
    S_kl := 0.0; update(k,−t); update(l,t)
    ! rotate rows and columns k and l
    for i := 1 to k−1 do rotate(i,k,i,l) endfor
    for i := k+1 to l−1 do rotate(k,i,i,l) endfor
    for i := l+1 to n do rotate(k,i,l,i) endfor
    ! rotate eigenvectors
    for i := 1 to n do
      ⎡   ⎤     ⎡     ⎤⎡   ⎤
      ⎢E_ik⎥     ⎢c  −s⎥⎢E_ik⎥
      ⎢   ⎥ := ⎢     ⎥⎢   ⎥
      ⎣E_il⎦     ⎣s   c⎦⎣E_il⎦
    endfor
    ! rows k, l have changed, update rows ind_k, ind_l
    ind_k := maxind(k); ind_l := maxind(l)
  loop
endproc
```

Notes

1. The logical array *changed* holds the status of each eigenvalue. If the numerical value of e_k or e_l changes during an iteration, the corresponding component of *changed* is set to *true*, otherwise to *false*. The integer *state* counts the number of components of *changed* which have the value *true*. Iteration stops as soon as *state* = 0. This means that none of the approximations $e_1, \ldots, e_n$ has recently changed its value and thus it is not very likely that this will happen if iteration continues. Here it is assumed that floating point operations are optimally rounded to the nearest floating point number.

2. The upper triangle of the matrix *S* is destroyed while the lower triangle and the diagonal are unchanged. Thus it is possible to restore *S* if necessary according to

```
for k := 1 to n−1 do ! restore matrix S
    for l := k+1 to n do
        S_kl := S_lk
    endfor
endfor
```

3. The eigenvalues are not necessarily in descending order. This can be achieved by a simple sorting algorithm.

```
for k := 1 to n−1 do
    m := k
    for l := k+1 to n do
        if e_l > e_m then
            m := l
        endif
    endfor
    if k ≠ m then
        swap e_m,e_k
        swap E_m,E_k
    endif
endfor
```

4. The algorithm is written using matrix notation (1 based arrays instead of 0 based).

5. When implementing the algorithm, the part specified using matrix notation must be performed simultaneously.

6. This implementation does not correctly account for the case in which one dimension is an independent subspace. For example, if given a diagonal matrix, the above implementation will never terminate, as none of the eigenvalues will change. Hence, in real implementations, extra logic must be added to account for this case.

Example

Let $S = \begin{pmatrix} 4 & -30 & 60 & -35 \\ -30 & 300 & -675 & 420 \\ 60 & -675 & 1620 & -1050 \\ -35 & 420 & -1050 & 700 \end{pmatrix}$

Then *jacobi* produces the following eigenvalues and eigenvectors after 3 sweeps (19 iterations) :

$e_1 = 2585.25381092892231$

$E_1 = \begin{pmatrix} 0.0291933231647860588 \\ -0.328712055763188997 \\ 0.791411145833126331 \\ -0.514552749997152907 \end{pmatrix}$

$e_2 = 37.1014913651276582$

$E_2 = \begin{pmatrix} -0.179186290535454826 \\ 0.741917790628453435 \\ -0.100228136947192199 \\ -0.638282528193614892 \end{pmatrix}$

$e_3 = 1.4780548447781369$

$E_3 = \begin{pmatrix} -0.582075699497237650 \\ 0.370502185067093058 \\ 0.509578634501799626 \\ 0.514048272222164294 \end{pmatrix}$

$e_4 = 0.1666428611718905$

$E_4 = \begin{pmatrix} 0.792608291163763585 \\ 0.451923120901599794 \\ 0.322416398581824992 \\ 0.252161169688241933 \end{pmatrix}$

Applications for real symmetric matrices

When the eigenvalues (and eigenvectors) of a symmetric matrix are known, the following values

are easily calculated.

Singular values

The singular values of a (square) matrix *A* are the square roots of the (non-negative) eigenvalues of $A^T A$. In case of a symmetric matrix *S* we have of $S^T S = S^2$, hence the singular values of *S* are the absolute values of the eigenvalues of *S*

2-norm and spectral radius

The 2-norm of a matrix *A* is the norm based on the Euclidean vectornorm, i.e. the largest value $\|Ax\|_2$ when x runs through all vectors with $\|x\|_2 = 1$. It is the largest singular value of *A*. In case of a symmetric matrix it is the largest absolute value of its eigenvectors and thus equal to its spectral radius.

Condition number

The condition number of a nonsingular matrix *A* is defined as $\mathbf{cond}(A) = \|A\|_2 \|A^{-1}\|_2$. In case of a symmetric matrix it is the absolute value of the quotient of the largest and smallest eigenvalue. Matrices with large condition numbers can cause numerically unstable results: small perturbation can result in large errors. Hilbert matrices are the most famous ill-conditioned matrices. For example, the fourth-order Hilbert matrix has a condition of 15514, while for order 8 it is 2.7×10^8.

Rank

A matrix *A* has rank *r* if it has *r* columns that are linearly independent while the remaining columns are linearly dependent on these. Equivalently, *r* is the dimension of the range of *A*. Furthermore it is the number of nonzero singular values.
In case of a symmetric matrix r is the number of nonzero eigenvalues. Unfortunately because of rounding errors numerical approximations of zero eigenvalues may not be zero (it may also happen that a numerical approximation is zero while the true value is not). Thus one can only calculate the *numerical* rank by making a decision which of the eigenvalues are close enough to zero.

Pseudo-inverse

The pseudo inverse of a matrix *A* is the unique matrix $X = A^+$ for which *AX* and *XA* are symmetric and for which *AXA = A, XAX = X* holds. If *A* is nonsingular, then '$A^+ = A^{-1}$. When procedure jacobi (S, e, E) is called, then the relation $S = E^T \mathbf{Diag}(e) E$ holds where Diag(*e*) denotes the diagonal matrix with vector *e* on the diagonal. Let e^+ denote the vector where e_i is replaced by $1/e_i$ if $e_i \leq 0$ and by 0 if e_i is (numerically close to) zero. Since matrix *E* is orthogonal, it follows that the pseudo-inverse of S is given by $S^+ = E^T \mathbf{Diag}(e^+) E$.

Least squares solution

If matrix *A* does not have full rank, there may not be a solution of the linear system *Ax = b*. However one can look for a vector x for which $\|Ax - b\|_2$ is minimal. The solution is $x = A^+ b$. In case of a symmetric matrix *S* as before, one has $x = S^+ b = E^T \mathbf{Diag}(e^+) E b$.

Matrix exponential

From $S = E^T \mathbf{Diag}(e) E$ one finds $\exp S = E^T \mathbf{Diag}(\exp e) E$ where exp *e* is the vector where e_i is replaced by $\exp e_i$. In the same way, *f*(*S*) can be calculated in an obvious way for any (analytic) function *f*.

Linear differential equations

The differential equation *x′ = Ax*, *x*(0) = *a* has the solution *x*(*t*) = exp(*t A*) *a*. For a symmetric

matrix *S*, it follows that $x(t) = E^T \mathrm{Diag}(\exp te) Ea$. If $a = \sum_{i=1}^{n} a_i E_i$ is the expansion of *a* by the eigenvectors of *S*, then $x(t) = \sum_{i=1}^{n} a_i \exp(te_i) E_i$.

Let W^s be the vector space spanned by the eigenvectors of *S* which correspond to a negative eigenvalue and W^u analogously for the positive eigenvalues. If $a \in W^s$ then $\lim_{t\,\infty} x(t) = 0$ i.e. the equilibrium point 0 is attractive to *x*(*t*). If $a \in W^u$ then $\lim_{t\,\infty} x(t) = \infty$, i.e. 0 is repulsive to *x*(*t*). W^s and W^u are called *stable* and *unstable* manifolds for *S*. If *a* has components in both manifolds, then one component is attracted and one component is repelled. Hence *x*(*t*) approaches W^u as $t \to \infty$.

Generalizations

The Jacobi Method has been generalized to complex Hermitian matrices, general nonsymmetric real and complex matrices as well as block matrices.

Since singular values of a real matrix are the square roots of the eigenvalues of the symmetric matrix $S = A^T A$ it can also be used for the calculation of these values. For this case, the method is modified in such a way that *S* must not be explicitly calculated which reduces the danger of round-off errors. Note that $JSJ^T = JA^T AJ^T = JA^T J^T JAJ^T = B^T B$ with $B := JAJ^T$.

The Jacobi Method is also well suited for parallelism.

References

1. Jacobi, C.G.J. (1846). "Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen" (http://gdz.sub.uni-goettingen.de/dms/load/img/?PID=GDZPPN002144522). *Crelle's Journal* (in German). **1846** (30): 51–94. doi:10.1515/crll.1846.30.51 (https://doi.org/10.1515%2Fcrll.1846.30.51).
2. Golub, G.H.; van der Vorst, H.A. (2000). "Eigenvalue computation in the 20th century" (https://doi.org/10.1016%2FS0377-0427%2800%2900413-1). *Journal of Computational and Applied Mathematics*. **123** (1–2): 35–65. doi:10.1016/S0377-0427(00)00413-1 (https://doi.org/10.1016%2FS0377-0427%2800%2900413-1).
3. Schönhage, A. (1964). "Zur quadratischen Konvergenz des Jacobi-Verfahrens". *Numerische Mathematik* (in German). **6** (1): 410–412. doi:10.1007/BF01386091 (https://doi.org/10.1007%2FBF01386091). MR 0174171 (https://www.ams.org/mathscinet-getitem?mr=0174171).

Further reading

- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), "Section 11.1. Jacobi Transformations of a Symmetric Matrix" (http://apps.nrbook.com/empanel/index.html#pg=570), *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press, ISBN 978-0-521-88068-8
- Rutishauser, H. (1966). "Handbook Series Linear Algebra: The Jacobi method for real symmetric matrices". *Numerische Mathematik*. **9** (1): 1–10. doi:10.1007/BF02165223 (https://doi.org/10.1007%2FBF02165223). MR 1553948 (https://www.ams.org/mathscinet-getitem?mr=1553948).
- Sameh, A.H. (1971). "On Jacobi and Jacobi-like algorithms for a parallel computer" (https://doi.org/10.1090%2Fs0025-5718-1971-0297131-6). *Mathematics of Computation*. **25** (115):

579–590. doi:10.1090/s0025-5718-1971-0297131-6 (https://doi.org/10.1090%2Fs0025-5718-1971-0297131-6). JSTOR 2005221 (https://www.jstor.org/stable/2005221). MR 0297131 (https://www.ams.org/mathscinet-getitem?mr=0297131).

- Shroff, Gautam M. (1991). "A parallel algorithm for the eigenvalues and eigenvectors of a general complex matrix". *Numerische Mathematik*. **58** (1): 779–805. CiteSeerX 10.1.1.134.3566 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.134.3566). doi:10.1007/BF01385654 (https://doi.org/10.1007%2FBF01385654). MR 1098865 (https://www.ams.org/mathscinet-getitem?mr=1098865).
- Veselić, K. (1979). "On a class of Jacobi-like procedures for diagonalising arbitrary real matrices". *Numerische Mathematik*. **33** (2): 157–172. doi:10.1007/BF01399551 (https://doi.org/10.1007%2FBF01399551). MR 0549446 (https://www.ams.org/mathscinet-getitem?mr=0549446).
- Veselić, K.; Wenzel, H. J. (1979). "A quadratically convergent Jacobi-like method for real matrices with complex eigenvalues". *Numerische Mathematik*. **33** (4): 425–435. doi:10.1007/BF01399324 (https://doi.org/10.1007%2FBF01399324). MR 0553351 (https://www.ams.org/mathscinet-getitem?mr=0553351).

External links

- Matlab implementation of Jacobi algorithm that avoids trigonometric functions (https://groups.google.com/group/sci.math.num-analysis/msg/8282d0d412f72d2e)
- C++11 implementation (https://github.com/jewettaij/jacobi_pd)