Jacobi Method

Iterative Method for Computing Eigenvalues and Eigenvectors

Outline

- Similarity transformation & diagonalization
- Given’s rotation matrix
- Overview of Jacobi method
- Naïve algorithm
- Improvement
- Revised algorithm
- Convergence analysis

Review: Similarity Transformation

- Theorem: If matrix P is invertible, the following transformation preserves the eigenvalues of A.

  $B = P^{-1}AP$. (similarity transformation)

- Theorem: Let $B = P^{-1}AP$ and x be an eigenvector of A. Then $y = P^{-1}x$ is an eigenvector of B.
- Proof: see lecture_70 EigenSystem

Diagonalization by Similarity Transformations

- Let D be a diagonal matrix, which is obtained by using a sequence of similarity transformations:

$$D = {P_k}^{-1} \dots {P_1}^{-1} {P_0}^{-1} A P_0 P_1 \dots P_k = P^{-1} A P,$$
$$P = P_0 P_1 \dots P_k.$$

- And we have,

$$A = PDP^{-1}.$$

Review: Diagonalization

- Theorem:

  If matrix ***A*** can be diagonalized into $A = PDP^{-1}$. Then matrix ***D*** is the diagonal matrix of the eigenvalues and matrix ***P*** is the column matrix of the eigenvectors of A.

Proof:

$$D = P^{-1}AP$$

$$AP = (PDP^{-1})P = P * D.$$

- ***D*** is a basic column-operation matrix.
- It multiplies the columns of ***P*** by its diagonal entries.
- Let *p* be the *i*th column of ***P***.

$$A * p = \lambda_i * p$$

Basic Ideas of Jacobi Method

- Precondition: Assume matrix $\boldsymbol{A}$ is symmetric.
- Basic ideas of Jacobi method:
  - By using a series of similarity transformations, we convert $\boldsymbol{A}$ into a diagonal matrix $\boldsymbol{D}$:

$$D = {P_k}^{-1} \dots {P_1}^{-1} {P_0}^{-1} A P_0 P_1 \dots P_k = P^T A P.$$

  - Then, we can compute all eigenvalues and eigenvectors in one process.

Given's Rotational Matrix

- Basic terms:
  - θ: rotational angle in radians
  - c: cos(θ), s: sin(θ)

- Definition: a Given's rotational matrix is defined as:

$$R(p,q,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- Construction method:

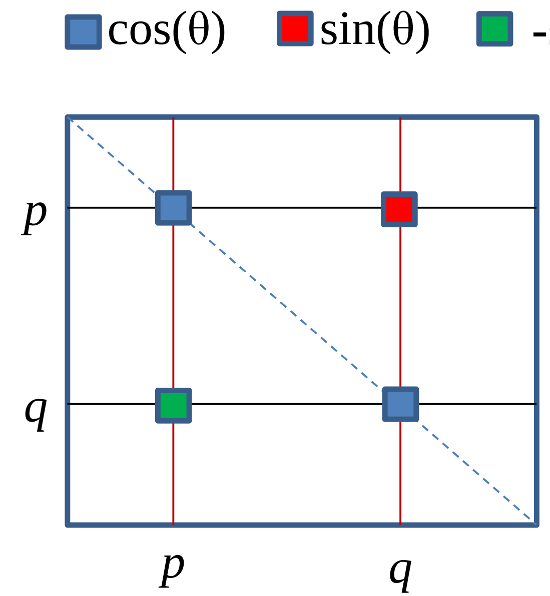
  R = I; //an *n* by *n* identity matrix

  R[p][p] = R[q][q] = c;

  R[p][q] = -s;

  R[q][p] = s;

Format of Given's rotational matrix



Given's matrix R(p, q, θ)

Some Notes

- The Given's rotational matrix is post-multiplied with the target matrix. $B = R^{-1}AR$.
- Therefore, it is a <u>column-operation</u> matrix.
- It is the transport matrix of a rotational matrix, used in Computer Graphics.

$$R(p,q,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ vs } R^T(p,q,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & s & 0 \\ 0 & -s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Basic Properties

- P1: the Given's matrix ***R*** is an orthonormal matrix.
  - All columns are unit vectors.
  - All rows are unit vectors.
  - The columns are mutually orthogonal.
  - The rows are mutually orthogonal.
  - The inverse matrix = the transpose matrix.
    $$R^{-1} = R^T$$
  - The determinant of ***R*** = 1.
- Given's rotational matrix

$$R(p,q,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Basic Properties

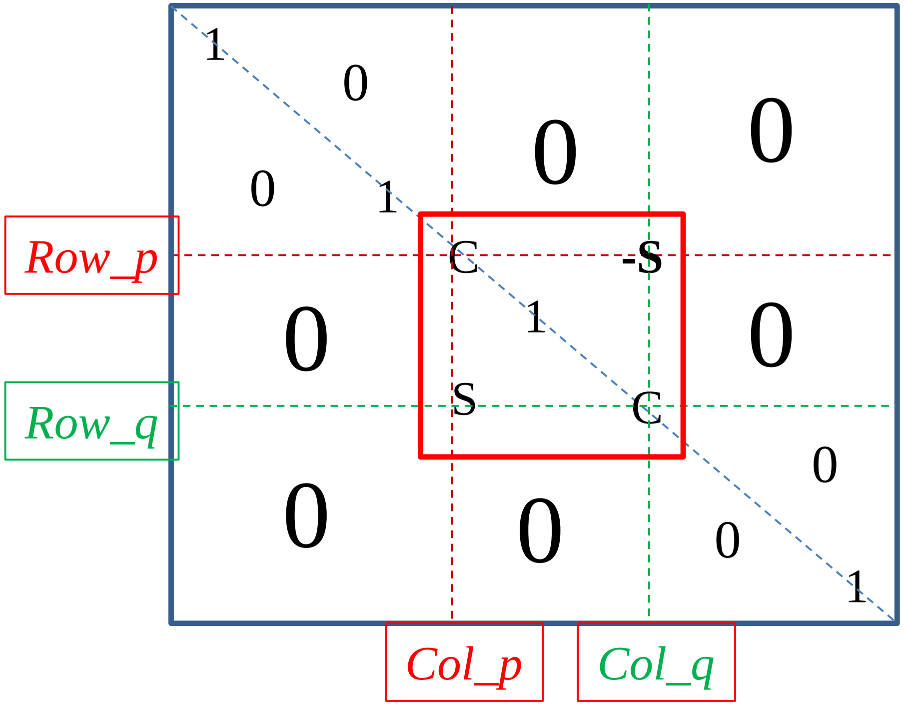
- P2. When a vector is transformed by a rotational matrix, its direction is changed but its magnitude remains the same.

- P3. The 2-norm of the Given's rotational matrix is 1.
  - Let $\boldsymbol{x}$ be a unit vector. Consider $\|Ax\| = \|x\| = 1$.
  - The max eigenvalue of Given's matrix = 1.

Similarity Transformation Matrix for Diagonalization

- The diagonalization process eliminates the off-diagonal entries gradually, $D = P^{-1}AP$.
- The inverse of the transformation matrix must be easy to compute.
- Good candidate: Given's rotational matrix.

- A Given's matrix



Similarity Transformation using Given's Matrix

- Assume $\boldsymbol{R}$ is a Given's matrix

  $R = R(p, q, \theta), R^{-1} = R^T.$
- Consider the following similarity transformation

  $B = R^{-1}AR = R^TAR$

  [note] $\boldsymbol{B}$ is a symmetric matrix.

- After the transformation,

  $B_{pq} = \boxed{(A_{qq} - A_{pp})} sin(\theta) cos(\theta) + A_{pq}(cos^2(\theta) - sin^2(\theta)),$

  $B_{qp} = B_{pq}$,//B is symmetric.

Rotational Angle of Given's Matrix

$$B_{pq} = \left(A_{qq} - A_{pp}\right) sin(\theta)cos(\theta) + A_{pq}(cos^2(\theta) - sin^2(\theta)),$$

- To eliminate this entry, we should have

$$0 = \left(A_{qq} - A_{pp}\right) sin(\theta)cos(\theta) + A_{pq}(cos^2(\theta) - sin^2(\theta)),$$

$$\frac{1}{2}\left(A_{qq} - A_{pp}\right)sin(2\theta) + A_{pq}\, cos(2\theta) = 0,$$

$$\frac{\sin(2\theta)}{\cos(2\theta)} = \frac{2A_{pq}}{A_{pp} - A_{qq}},$$

$$\tan(2\theta) = \frac{2A_{pq}}{A_{pp} - A_{qq}}.$$

$$\theta = \frac{1}{2}tan^{-1}\left(\frac{2A_{pq}}{A_{pp} - A_{qq}}\right).$$

Why do we not consider other entries of ***B***?
Answer: we just want to eliminate ***A****[p][q]* and ***A****[q][p]* in one transformation.

Review: Basic Ideas of Jacobi Method

- Assume matrix $\boldsymbol{A}$ is symmetric.
- Jacobi method:
  - Using a series of similarity transformations, we convert $\boldsymbol{A}$ into a diagonal matrix $\boldsymbol{D}$:

$$D = {R_k}^T \ldots {R_1}^T R_0^T A R_0 R_1 \ldots R_k = P^T A P.$$

D = diagonal matrix of the eigenvalues,

P = the column matrix of the eigenvectors.

The Primitive Algorithm

select $P = I$;

$A_{pq} = max\_off\_diag\_entry(A, \&p, \&q)$;

while($|A_{pq}| > \varepsilon$) do{

$\theta = \frac{1}{2}tan^{-1}(\frac{2A_{pq}}{A_{pp}-A_{qq}})$;

$make\_rotate\_mtx(R, \theta, p, q)$;

$P = P*R$; $//P^{(k+1)} = P^{(k)}R_k$

$B = A*R$;

$A = R^T*B$; $//A^{(k+1)} = R_k^T A^{(k)} R_k$

$A_{pq} = max\_off\_diag\_entry(A, \&p, \&q)$;

}

Improvement (1)

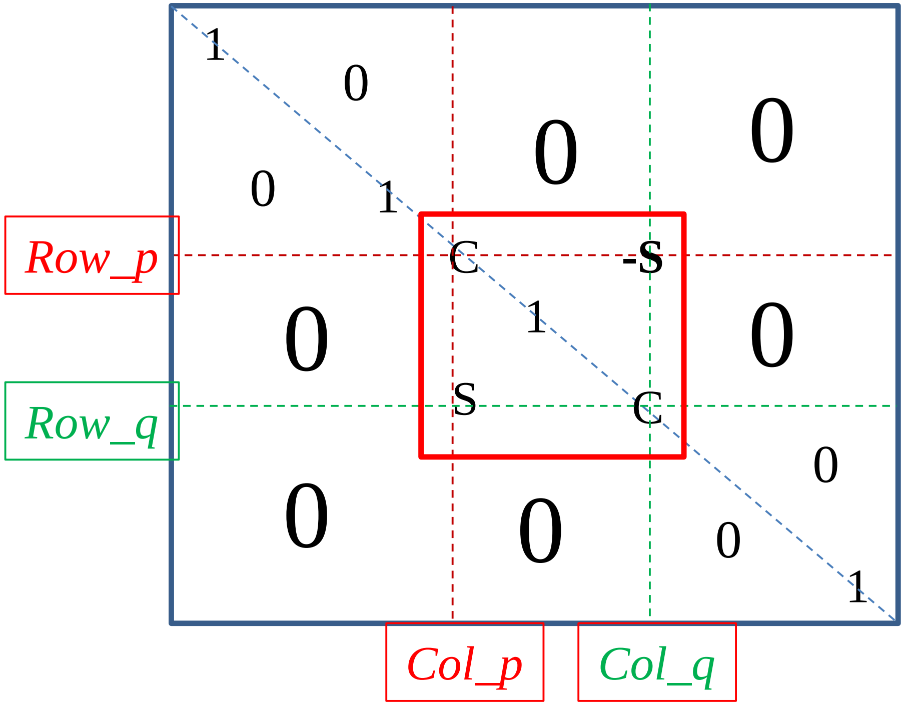
- In each iteration, we have to perform 3 matrix-matrix multiplication.
  - $O(3n^3)$ time steps.
- Consider ***B* = *A**R**,
  - *R* is a column-operation matrix.

  new_col_p = c*old_col_p $\boxed{+}$ s*old_col_q;

  new_col_q = $\boxed{-s*}$old_col_p + c*old_col_q;

  - Other columns are unhanged.
- The multiplication can be achieved in O(n) steps.

Given's matrix ***R*** =



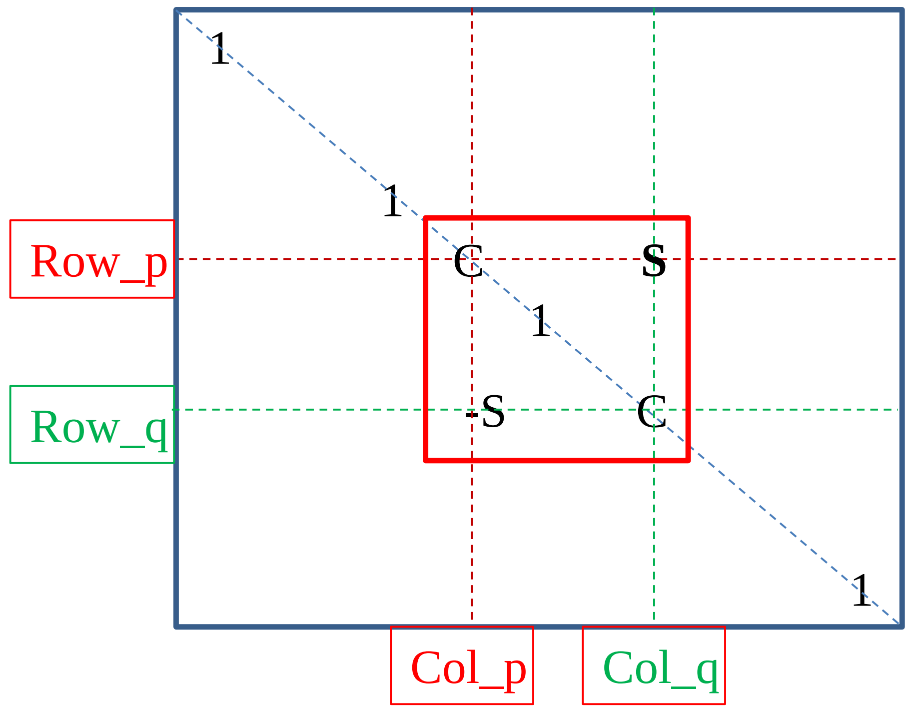
Improvement (2)

- Consider $A = R^T * B$,
  - R^T is a row-operation matrix.

    *new_row_p = c*old_row_p + s*old_row_q;*

    *new_row_q = -s*old_row_p + c*old_row_q;*
  - Other rows are unhanged.
- The multiplication can be achieved in O(n) steps.

$\boldsymbol{R^T} =$



The Updating Equation (1)

- Similarity transformation
  $R(p,q,\theta)$: Given's rotation matrix
  $B = R^T A R$, similar to A.
- Analytic algorithm for computing the entries:
  $B_{pp} = c^2 A_{pp} + 2sc A_{pq} + s^2 A_{qq},$
  $B_{qq} = s^2 A_{pp} - 2sc A_{pq} + c^2 A_{qq},$
  $B_{pq} = B_{qp} = 0.0,$ //max off-diagonal entries
  //rows & columns p & q.
  $B_{pk} = B_{kp} = cA_{pk} + sA_{qk}, k \neq p, q\ , 0 \leq k \leq n-1,$
  $B_{qk} = B_{kq} = -sA_{pk} + cA_{qk}, k \neq p, q\ , 0 \leq k \leq n-1,$
  – Other entries $B_{ij} = A_{ij}$.

The Updating Equation (2)

- Accumulation of Given's matrices

  $Q = P*R$,   //column operations

- Analytic algorithm

  $Q_{kp} = cP_{kp} + sP_{kq}, 0 \leq k \leq n-1,$ //column p

  $Q_{kq} = -sP_{kp} + cP_{kq}, 0 \leq k \leq n-1,$ //column q.

  – Other entries $Q_{ij} = P_{ij}$.

Improvements

- The following matrix multiplications take only O(n) steps:

  $P = P*R;$

  $B = A*R;$

  $A = R^T*B;$

- The most expensive operation in each iteration is

  $A_{pq} = max\_off\_diag\_entry(A, \&p, \&q);$

  – It takes $O(n^2/2)$ steps.

  – But, it can be speed-up by using a *max-heap* structure O(*2nlog n*).

The Revised Algorithm (1/3)

select P = I;
A_{pq} *= max_off_diag_entry(A, &p, &q);*
while($|A_{pq}|>\varepsilon$*) do{*

$\theta = \frac{1}{2}tan^{-1}(\frac{2A_{pq}}{A_{pp}-A_{qq}});$

c = cos(θ);
s = sin(θ);
*update_P_mtx(P, p, q, c, s); // P = P*R;*
update_A_mtx(A, p, q, c, s); //B = R^T*A*R;*
A_{pq} *= max_off_diag_entry(A, &p, &q);*
}

The Revised Algorithm (2/3)

```
update_A_mtx(A, p, q, c, s) {
Bp[p] = c*c*A[p][p] + 2.0*s*c*A[p][q] +
s*s*A[q][q];
   Bq[q] = s*s*A[p][p] - 2.0*s*c*A[p][q] +
c*c*A[q][q];
for(k=0;k<n;k++){
    if(k≠p&&k≠q) Bp[k] = c*A[k][p] + s*A[k][q];
    if(k≠p&&k≠q) Bq[k] = -s*A[k][p]+c*A[k][q];
  }

 for(k=0;k<n;k++){
   A[p][k] = A[k][p] = Bp[k];
   A[q][k] = A[k][q] = Bq[k];
 }
A[p][q] = A[q][p] = 0.0;
}
```

Equations for updating matrix ***A***:

$$B_{pp} = c^2 A_{pp} + 2scA_{pq} + s^2 A_{qq},$$
$$B_{qq} = s^2 A_{pp} - 2scA_{pq} + c^2 A_{qq},$$
$$B_{pq} = B_{qp} = 0.0,$$
$$B_{pk} = B_{kp} = cA_{pk} + sA_{qk},$$
$$k \neq p, q, 0 \leq k \leq n-1,$$
$$B_{qk} = B_{kq} = -sA_{pk} + cA_{qk},$$
$$k \neq p, q, 0 \leq k \leq n-1,$$

$$\boldsymbol{B = R^T A R}$$

The Revised Algorithm (3/3)

```
update_P_mtx(P, p, q, c, s){
  for(k=0;k<n;k++){
    Qp[k] = c*P[k][p] + s*P[k][q];
    Qq[k] = -s*P[k][p] + c*P[k][q];
  }
  for(k=0;k<n;k++){
    P[k][p] = Qp[k];
    P[k][q] = Qq[k];
  }
}
```

The analytic equation for updating matrix ***P***:

$$Q_{kp} = cP_{kp} + sP_{kq},\quad 0 \le k \le n-1,$$

$$Q_{kq} = -sP_{kp} + cP_{kq},\quad 0 \le k \le n-1,$$

– Other entries $Q_{ij} = P_{ij}$.

P = P*R;

Time Complexity Analysis

- Each iteration needs $O(n^2/2)$ steps.
  - For finding the entry with the max magnitude.

- Theorem: Jacobi method always converges

  Proof:
  - Compute the sum of the squares of the lower off-diagonal entries after the *k*-th iteration:

    $\delta_{k+1} = \sum \sum {A_{ij}}^2, i < j.$
  - It can be shown that $\delta_{k+1} = \delta_k - 2{A_{pq}}^2$.
  - Thus $\delta_{k+1} < \delta_k$.

- Jacobi method converges faster, if the multiplicities of some eigenvalues are greater than 1. (duplicated eigenvalues)

Converge Rate

- Theorem: Jacobi method converges, at least, linearly.
  - Time complexity = $O(k \cdot \log\frac{1}{\varepsilon} \cdot \frac{n^2}{2})$.
- Proof: omitted.

- Revised method
  - Computing the arctangent, cosine and sine values causing numerical errors.
  - Can we compute c and s by using a Newton method?
    $(A_{qq} - A_{pp})\,s * c + A_{pq}(c^2 - s^2) = 0,$
    $s^2 + c^2 = 1.0$

Conclusion

- We use Given’s rotation to perform similarity transformation.
- After a sequence of transformations, matrix A is diagonalized.
- Jacobi method computes all eigenvalues and eigenvector in one process.
- Jacobi method always converges for symmetric matrices.

Reading Assignment

- Review our Algorithms & Data Structures textbooks for the following topics:
  - Max-heap, priority queues
  - Heap-sort

- In power method, we learned that:

  $B = (A - \mu I)$ has the same eigenvectors as A and its eigenvalues are $\{\lambda_i - \mu\}$.

  Can we use this shifting method to speed-up Jacobi method?