Algebraic Eigenvalue Problem

Computers are useless. They can only give answers.

Pablo Picasso

Fall 2010

Topics to Be Discussed

- **This unit requires the knowledge of eigenvalues and eigenvectors in linear algebra.**
- **The following topics will be presented:**
  - **The Power method for finding the largest eigenvalue and its corresponding eigenvector**
  - **Coordinate rotation**
  - **Rotating a symmetric matrix**
  - **Classic Jacobi method (1846) for finding *all* eigenvalues and eigenvectors of a *symmetric* matrix**

Eigenvalues & Eigenvectors: 1/3

- **Given a square matrix $\mathbf{A}$, if one can find a number (real or complex) λ and a vector $\mathbf{x}$ such that $\mathbf{A}\cdot\mathbf{x} = \lambda\mathbf{x}$ holds, λ is an eigenvalue and $\mathbf{x}$ an eigenvector corresponding to λ (of matrix $\mathbf{A}$).**
- **Since the right-hand side of $\mathbf{A}\cdot\mathbf{x} = \lambda\mathbf{x}$ can be rewritten as $\lambda\mathbf{I}\cdot\mathbf{x}$, where $\mathbf{I}$ is the identity matrix, we have $\mathbf{A}\cdot\mathbf{x} = \lambda\mathbf{I}\cdot\mathbf{x}$ and $(\mathbf{A}-\lambda\mathbf{I})\mathbf{x} = \mathbf{0}$.**
- **Solving for λ from equation $\det(\mathbf{A}-\lambda\mathbf{I}) = 0$ yields *all* eigenvalues of $\mathbf{A}$, where $\det()$ is the determinant of a matrix.**

Eigenvalues & Eigenvectors: 2/3

- **If A is a $n \times n$ matrix, $\det(A-\lambda I) = 0$ is a polynomial of degree n in λ, and has n roots (*i.e.*, n possible values for λ), some of which may be complex conjugates (*i.e.*, $a+bi$ and $a-bi$).**
- **However, people rarely use this method to find eigenvalues because (1) directly expanding $\det(A-\lambda I) = 0$ to a polynomial is tedious, and (2) there is no close-form solution if $n > 4$.**
- **Many methods transform A to simpler forms so that $\det(A-\lambda I) = 0$ can be obtained easily.**

Eigenvalues & Eigenvectors: 3/3

- **The eigenvalues of a diagonal matrix are its diagonal entries.**
- **For example, if we have a diagonal matrix:**

$$A = \begin{bmatrix} d_1 & & & & \\ & d_2 & & & \\ & & \ddots & & \\ & & & d_{n-1} & \\ & & & & d_n \end{bmatrix}$$

- **Then, det(A-λI)=0 is**

$$(d_1 - \lambda)(d_2 - \lambda)\ldots(d_{n-1} - \lambda)(d_n - \lambda) = 0$$

- **Hence, the roots of det(A-λI)=0 are the *d_i*'s.**

Power Method: 1/11

- **What if we take a guess $\mathbf{z}$ and compute $\mathbf{A}\cdot\mathbf{z}$?**
- **If $\mathbf{z}$ is actually an eigenvector, then $\mathbf{A}\cdot\mathbf{z} = \lambda\mathbf{z}$.**
- **Let $\mathbf{w} = \mathbf{A}\cdot\mathbf{z} = \lambda\mathbf{z}$. Since for every entry of $\mathbf{w}$ and $\mathbf{z}$ we have $w_i = \lambda z_i$ and $\lambda = w_i/z_i$.**
- **If $\mathbf{z}$ is not an eigenvector, then $\mathbf{w}$ may be a vector closer to an eigenvector than $\mathbf{z}$ is.**
- **Therefore, we may use $\mathbf{w}$ in the next iteration to find an even better approximation.**
- **From $\mathbf{w}$, we have $\mathbf{u} = \mathbf{A}\cdot\mathbf{w}$; from $\mathbf{u}$ we have $\mathbf{v} = \mathbf{A}\cdot\mathbf{u}$; etc. Hopefully, some vector $\mathbf{x}$ will satisfy $\mathbf{A}\cdot\mathbf{x} = \lambda\mathbf{x}$.**

Power Method: 2/11

- **Note that: if $\mathbf{x}$ is an eigenvector, $\alpha\mathbf{x}$ is also an eigenvector because $\alpha(\mathbf{A}\cdot\mathbf{x})=\alpha(\lambda\mathbf{x})$ and $\mathbf{A}\cdot(\alpha\mathbf{x}) = \lambda(\alpha\mathbf{x})$!**
- **Therefore, we may scale an eigenvector. The simplest way is to scale the vector by the component with maximum absolute value. After scaling, the value of each component is in $[-1,1]$.**
- Example: **Let $\mathbf{x}$ be $[15, -20, -8]$. Since $|-20|$ is the largest, the scaling factor is -20 and the scaled $\mathbf{x}$ is $[-15/20, 1, 8/20]$.**

Power Method: 3/11

- **This scaling has an advantage.**
- **Given a vector $\mathbf{z}$, we compute $\mathbf{w} = \mathbf{A}\cdot\mathbf{z}$.**
- **If $\mathbf{w}$ is a good approximate of $\lambda\mathbf{z}$, we have $\mathbf{w} \approx \lambda\mathbf{z} = \mathbf{A}\cdot\mathbf{z}$.**
- **Therefore, we should have $w_i \approx \lambda z_i$ for every i.**
- **If vector $\mathbf{z}$ is scaled so that its largest entry, say z_k, is 1, then $w_k \approx \lambda z_k = \lambda$!**
- **In other words, the scaling factor is an *approximation* of an eigenvalue!**

Power Method: 4/11

- We may start with a **z** and compute $\mathbf{w} = \mathbf{A} \cdot \mathbf{z}$.
- The largest component w_k of **w** is an approximation of an eigenvalue λ (*i.e.*, $w_k \approx \lambda$).
- Then, **w** is scaled with its largest component w_k and used as a new **z** (*i.e.*, $\mathbf{z} = \mathbf{w}/w_k$).
- This process is applied iteratively until we have $|\mathbf{A} \cdot \mathbf{z} - w_k \mathbf{z}| < \varepsilon$, where ε is a tolerance value.

Power Method: 5/11

- **Suppose this process starts with vector $\mathbf{x}_0$.**
- **The computation of $\mathbf{x}_i$ is $\mathbf{x}_i = \mathbf{w}_i/w_{i,k} = (\mathbf{A}\cdot\mathbf{x}_{i-1})/w_{i,k}$, where $w_{i,k}$ is the maximum component of $\mathbf{w}_i$.**
- **Since $\mathbf{x}_{i-1} = \mathbf{w}_{i-1}/w_{i-1,k} = (\mathbf{A}\cdot\mathbf{x}_{i-2})/w_{i-1,k}$, we may rewrite the $\mathbf{x}_i$ as follows for some c, d and g:**

$$\mathbf{x}_i = c(\mathbf{A}\cdot\mathbf{x}_{i-1}) = c(d\mathbf{A}(\mathbf{A}\mathbf{x}_{i-2})) = g\mathbf{A}^2\mathbf{x}_{i-2}$$

- **Continuing this process, we have the following for some p:**

$$\mathbf{x}_i = p\mathbf{A}^i\mathbf{x}_0$$

- **Hence, $\mathbf{x}_i$ is obtained by some power of $\mathbf{A}$, and, hence, the "power" method.**

Power Method: 6/11

- Example: **Consider the following 2×2 matrix**

$$A = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$$

- **This matrix has eigenvalues 5 and 1 and corresponding eigenvectors [1,1] and [-3,1]**
- **Let us start with z=[1/2,1] . Since the maximum entry of z is 1, no scaling is needed.**
- **Compute w=A·z = [4,9/2].**

Power Method: 7/11

- **Since $w=[4,9/2]$ and its largest entry is $9/2$,**
  - **The approximate eigenvalue is $9/2$**
  - **The scaled $z = w/(9/2) = [8/9,1]$**
- **Compute $w = A \cdot z =[43/9,44/9]$. Now, we have**
  - **The approximate eigenvalue is $44/9$**
  - **The new $z = [43/44,1]$**
- **Compute $w = A \cdot z = [109/22,219/44]$ and we have**
  - **The approximate eigenvalue is $219/44$**
  - **The new $z = [218/219,1]$**
- **After 3 iterations, we have an approximate eigenvalue $219/44 = 4.977 \approx 5$ and eigenvector $[218/219,1] = [0.9954,1] \approx [1,1]$.**

Power Method: 8/11

- **A** is the input matrix, **z** an approx. eigenvector

```
z = random and scaled vector ! initialize
DO                           ! loop until done
  w   = A*z
  max = 1                    ! find the |max| entry
  DO i = 2, n
    IF (ABS(w(i)) > ABS(w(max))) max = i
  END DO
  eigen_value = w(max)       ! ABS(w(max)) the largest
  DO i = 1, n                ! Scale w(*) to z(*)
    z(i) = w(i)/eigen_value
  END DO
  IF (ABS(A*z - eigen_value*z)) < Tol) EXIT
END DO
```

Power Method: 9/11

- **Example:** **Find an eigenvalue and its corresponding eigenvector of A, x_0 = [1,1,1,1]:**

$$\begin{bmatrix} 11 & -26 & 3 & -12 \\ 3 & -12 & 3 & -6 \\ 31 & -99 & 15 & -44 \\ 9 & -10 & -3 & -4 \end{bmatrix}$$

- **Iter 1: w=[-24,-12,-97,-8], approx. λ = -97 and new z=w/(-97)=[0.247423,0.123711,1,0.0824742].**
- **Iter 2: w = [1.51546,1.76289,6.79381,-2.34021], approx. λ = 6.79381, z=w/(6.79381) = [0.223065,0.259484,1,-0.344461]**

Power Method: 10/11

- **Iter 3:** **w = [2.84067,2.62216,11.3824,-2.20941], approx. λ = 11.3824, new z=w/λ=[0.249567, 0.230369,1,-0.194107]**
- **Iter 4:** **w=[2.08492,2.14891,8.47074,-2.28116], approx λ = 8.47074, new z = w/λ = [0.246132,0.253687,1,-0.269299]**
- 15 more iterations ....
- **Iter 19:** **approx. λ = 9 and corresponding eigenvector (*i.e.*, z) = [0.25, 0.25,1,-0.25]**

Power Method: 11/11

- What does power method do?
- **It finds the largest eigenvalue (*i.e.*, *dominating eigenvalue*) and its corresponding eigenvector.**
- **If vector z is *perpendicular* to the eigenvector corresponding to the largest eigenvalue, power method will not converge in *exact arithmetic*.**
- **Thus, z may be a random vector, initially.**
- **Convergence rate is $|\lambda_2/\lambda_1|$, where λ_1 and λ_2 are the largest and second largest eigenvalues.**
- **If rate is << 1, faster convergence is possible. *If it is close to 1, convergence will be very slow.***

Jacobi Method: Basic Idea

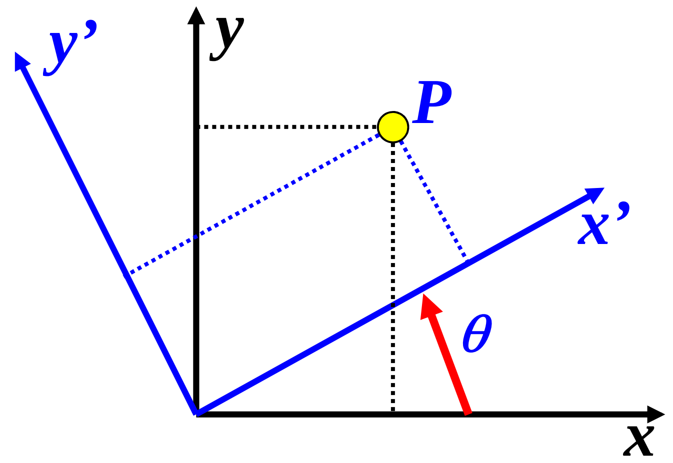
- **Finding *all* eigenvalues and their corresponding eigenvectors is not an easy task.**
- **However, in 1846 Jacobi found a relatively easy way to find *all* eigenvalues and eigenvectors of a *symmetric* matrix.**
- **Jacobi suggested that a symmetric matrix would be diagonal after being transformed repeatedly with appropriate "rotations."**
- **In what follows, we shall talk about coordinate rotation, rotations applied to a symmetric matrix, and Jacobi's method.**

Coordinate Rotation: 1/2

- **Suppose rotating system (x,y) an angle of θ yields (x',y'). The relationship between (x',y') and (x,y) is**

$$x' = \cos(\theta)x + \sin(\theta)y$$

$$y' = -\sin(\theta)x + \cos(\theta)y$$



- **This can be represented in a matrix form:**

rotation matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Coordinate Rotation: 2/2

- An n-dimensional rotation matrix is $n \times n$.
- If rotation is on the x_p-x_q plane with an angle θ, the (p,q)-rotation matrix $\mathbf{R}_{p,q}(\theta)$ is:

$$R_{p,q}(\theta) = \begin{array}{cc}
\begin{bmatrix}
1 & & & & & & & & \\
 & \ddots & & & & & & \mathbf{0} & \\
 & & 1 & & & & & & \\
 & & & \cos(\theta) & & \sin(\theta) & & & \\
 & & & & 1 & & & & \\
 & & & -\sin(\theta) & & \cos(\theta) & & & \\
 & & & & & & 1 & & \\
 & \mathbf{0} & & & & & & \ddots & \\
 & & & & & & & & 1
\end{bmatrix} & \begin{array}{l} \\ \\ \\ \text{row } p \\ \\ \text{row } q \\ \\ \\ \\ \end{array} \\
\begin{array}{ccccccccc} & & & \text{col } p & & \text{col } q & & & \end{array} &
\end{array}$$

Symmetric Matrix Rotation: 1/11

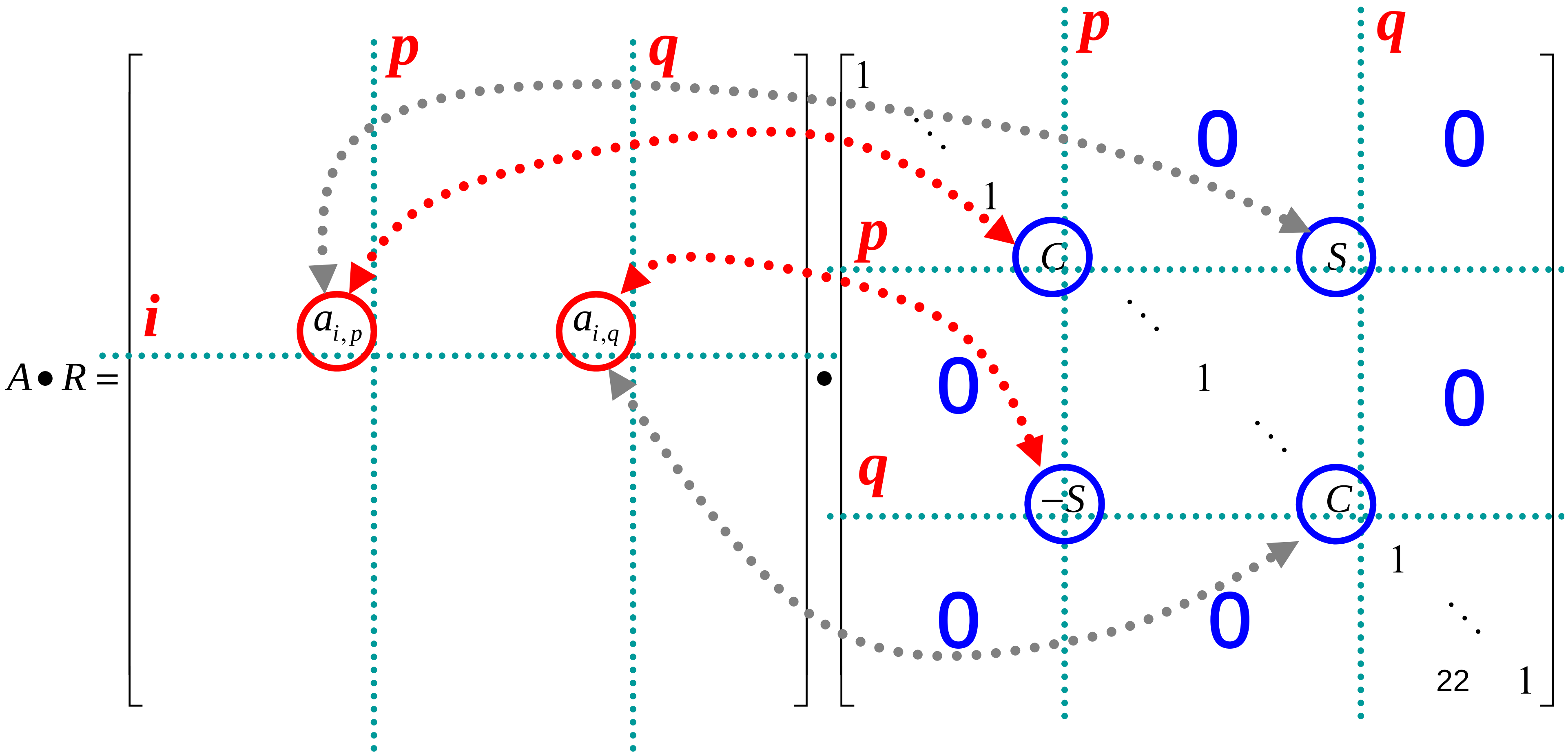
- **A symmetric matrix $\mathbf{A} = [a_{i,j}]_{n \times n}$ is a matrix satisfying $a_{i,j} = a_{j,i}$, where $1 \leq i < j \leq n$.**
- **In other words, a symmetric matrix is "*symmetric*" about its diagonal.**
- **The transpose of matrix $\mathbf{B}$ is $\mathbf{B}^T$.**
- **Rotation matrix $\mathbf{R}_{p,q}(\theta)$ is not symmetric.**
- **Rotating a matrix $\mathbf{A}$ with rotation matrix $\mathbf{R}$ is computed as $\mathbf{A}' = \mathbf{R}^T \bullet \mathbf{A} \bullet \mathbf{R}$**
- If $\mathbf{A}$ is symmetric, $\mathbf{A}'$ is also symmetric.

Symmetric Matrix Rotation: 2/11

- **Given a symmetric matrix $\mathbf{A} = [a_{i,j}]_{n\times n}$ and a rotation matrix $\mathbf{R}_{p,q}(\theta)$, written as $\mathbf{R}$ for simplicity, find $\mathbf{A'} = \mathbf{R}^T \bullet \mathbf{A} \bullet \mathbf{R}$.**
- **This is an easy task: we compute $\mathbf{H}=\mathbf{A}\bullet\mathbf{R}$, followed by $\mathbf{A'} = \mathbf{R}^T\bullet\mathbf{H}$.**
- **Do we have to use matrix multiplication?**
- No**, it is not necessary due to the very simple form of the rotation matrix $\mathbf{R}$ and $\mathbf{R}^T$.**

Symmetric Matrix Rotation: 3/11

● **Observation: A•R is identical to A except for column *p* and column *q* (*C*=cos(θ) and *S*=sin(θ))**



Symmetric Matrix Rotation: 4/11

- **A•R is computed as follows, where *C* and *S* are cos(θ) and sin(θ), respectively.**
- **Other than column *p* and column *q*, all entries are identical to those of A.**

$$A \bullet R = \begin{bmatrix} & a_{1,p}C - a_{1,q}S & & a_{1,p}S + a_{1,q}C & \\ & a_{2,p}C - a_{2,q}S & a_{i,j} & a_{2,p}S + a_{2,q}C & \\ & \vdots & & \vdots & \\ & a_{p,p}C - a_{p,q}S & & a_{p,p}S + a_{p,q}C & \\ a_{i,j} & \vdots & \ddots & \vdots & a_{i,j} \\ & a_{q,p}C - a_{q,q}S & & a_{q,p}S + a_{q,q}C & \\ & \vdots & & \vdots & \\ & a_{n-1,p}C - a_{n-1,q}S & & a_{n-1,p}S + a_{n-1,q}C & \\ & a_{n,p}C - a_{n,q}S & a_{i,j} & a_{n,p}S + a_{n,q}C & \end{bmatrix}$$

$$\text{col } p \qquad\qquad\qquad\qquad \text{col } q$$

Symmetric Matrix Rotation: 5/11

- **Suppose we have computed $\mathbf{H=A\bullet R}$, how do we compute $\mathbf{A'=R^T\bullet A\bullet R=R^T\bullet H}$?**
- **The transpose of $\mathbf{R}$, $\mathbf{R^T}$, is very similar to $\mathbf{R}$**

$$R^T = \begin{bmatrix} 1 &&&&&&&& \\ & \ddots &&&&&&& \\ && 1 &&&&&& \\ &&& C && S &&& \\ &&&& 1 &&&& \\ &&& -S && C &&& \\ &&&&&& 1 && \\ &&&&&&& \ddots & \\ &&&&&&&& 1 \end{bmatrix}^T = \begin{bmatrix} 1 &&&&&&&& \\ & \ddots &&&&&&& \\ && 1 &&&&&& \\ &&& C && -S &&& \\ &&&& 1 &&&& \\ &&& S && C &&& \\ &&&&&& 1 && \\ &&&&&&& \ddots & \\ &&&&&&&& 1 \end{bmatrix}$$

Symmetric Matrix Rotation: 6/11

- **Computing A’=R^T•H is very similar to computing A•R.**
- **The only difference is row *p* and row *q*.**

$$
R^T \bullet H =
\begin{array}{cc}
 & \begin{array}{ccccccccccc} & & & p & & & & q & & & \end{array} \\
\begin{array}{c} \\ \\ \\ p \\ \\ \\ \\ q \\ \\ \\ \\ \end{array} &
\begin{bmatrix}
1 & & & & & & & & & & \\
 & \ddots & & & & & & & & & \\
 & & 1 & & & & & & & & \\
 & & & C & & & & -S & & & \\
 & & & & \ddots & & & & & & \\
 & & & & & 1 & & & & & \\
 & & & & & & \ddots & & & & \\
 & & & S & & & & C & & & \\
 & & & & & & & & 1 & & \\
 & & & & & & & & & \ddots & \\
 & & & & & & & & & & 1
\end{bmatrix}
\end{array}
\bullet
\begin{array}{cc}
 & \begin{array}{cc} i & j \end{array} \\
\begin{array}{c} \\ p \\ \\ q \\ \\ \end{array} &
\begin{bmatrix}
 & \\
a_{p,i} & a_{p,j} \\
 & \\
a_{q,i} & a_{q,j} \\
 &
\end{bmatrix}
\end{array}
$$

Symmetric Matrix Rotation: 7/11

- **Here is the result of A'=R^T•A•R=R^T•H:**

$$
R^T \bullet A \bullet R =
\begin{array}{c}
\\ \\ \\ p \\ q \\ \\ \\
\end{array}
\begin{array}{c}
\begin{array}{cccccccc}
 & & & p & & q & & \\
\end{array}
\\
\left[
\begin{array}{cccccccc}
 & a_{i,j} & & a_{1,p}C - a_{1,q}S & & a_{1,p}S + a_{1,q}C & a_{i,j} & \\
 & & & a_{2,p}C - a_{2,q}S & & a_{2,p}S + a_{2,q}C & & \\
 & & & \vdots & & \vdots & & \\
a_{p,1}C - a_{q,1}S & a_{p,2}C - a_{q,2}S & \cdots & A'_{p,p} & \cdots & A'_{p,q} & \cdots & a_{p,n}C - a_{q,n}S \\
 & & & \vdots & \ddots & \vdots & & \\
a_{p,1}S + a_{q,1}C & a_{p,2}S + a_{q,2}C & \cdots & A'_{q,p} & & A'_{q,q} & \cdots & a_{p,n}S + a_{q,n}C \\
 & & & \vdots & & \vdots & & \\
 & a_{i,j} & & a_{n-1,p}C - a_{n-1,q}S & & a_{n-1,p}S + a_{n-1,q}C & a_{i,j} & \\
 & & & a_{n,p}C - a_{n,q}S & & a_{n,p}S + a_{n,q}C & & \\
\end{array}
\right]
\end{array}
$$

Because of symmetry, we only update the upper triangular part.

$$A'_{p,p} = a_{p,p}C^2 - 2a_{p,q}C \times S + a_{q,q}S^2$$

$$A'_{q,q} = a_{p,p}S^2 + 2a_{p,q}C \times S + a_{q,q}C^2$$

$$A'_{p,q} = A'_{q,p} = \left(a_{p,p} - a_{q,q}\right)C \times S + a_{p,q}\left(C^2 - S^2\right)$$

Symmetric Matrix Rotation: 8/11

- **Part I:** **Update $a_{p,p}$, $a_{p,q}$ and $a_{q,q}$, where $p < q$.**

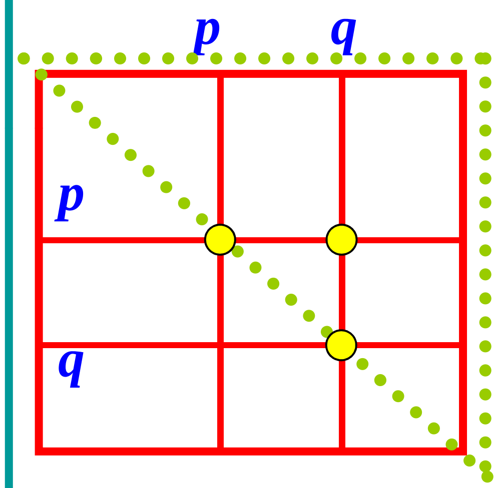
$$A'_{p,p} = a_{p,p}C^2 - 2a_{p.q}C \times S + a_{p,q}S^2$$

$$A'_{q,q} = a_{p,p}S^2 + 2a_{p.q}C \times S + a_{p,q}C^2$$

$$A'_{p,q} = A'_{q,p} = (a_{p,p} - a_{q,q})C \times S + a_{p,q}(C^2 - S^2)$$

```
! PART I:  Update a(p,p), a(q,q), a(p,q)
!     C = cos(θ)   and   S = sin(θ)
!     a(*,*) is an n×n symmetric matrix
!     p, q : for (p,q)-rotation, where p < q

App = C*C*a(p,p) - 2*C*S*a(p,q) + S*S*a(q,q)
Aqq = S*S*a(p,p) + 2*C*S*a(p,q) + C*C*a(q,q)
Apq = C*S*(a(p,p)-a(q,q)) + (C*C-S*S)*a(p,q)
a(p,p) = App
a(q,q) = Aqq
a(p,q) = Apq
```



NOTE: **only the upper triangular portion is updated!**

Symmetric Matrix Rotation: 9/11

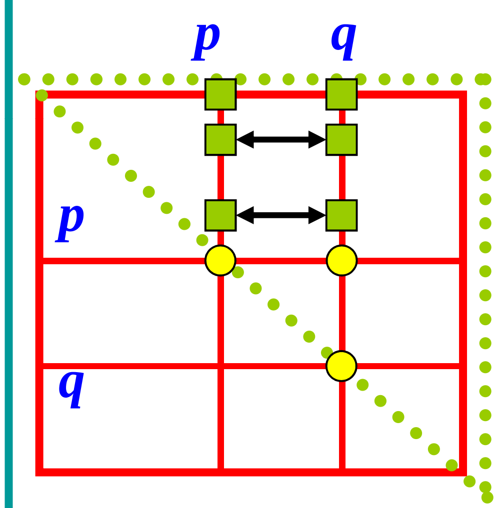
- **Part II: Update Row 1 to Row *p*-1.**

$$A'_{i,p} = a_{i,p}C - a_{i,q}S$$

$$A'_{i,q} = a_{i,p}S + a_{i,q}C$$

```
! PART II: Update column p and column q from
!          row 1 to row p-1.
!
! h is used to save the new value of a(i,p)
!    since a(i,p) is used to compute a(i,q)
!    and cannot be destroyed right away!

DO i = 1, p-1
  h      = C*a(i,p) - S*a(i,q)
  a(i,q) = S*a(i,p) + C*a(i,q)
  a(i,p) = h
END DO
```



NOTE: **only the upper triangular portion is updated!**

Symmetric Matrix Rotation: 10/11

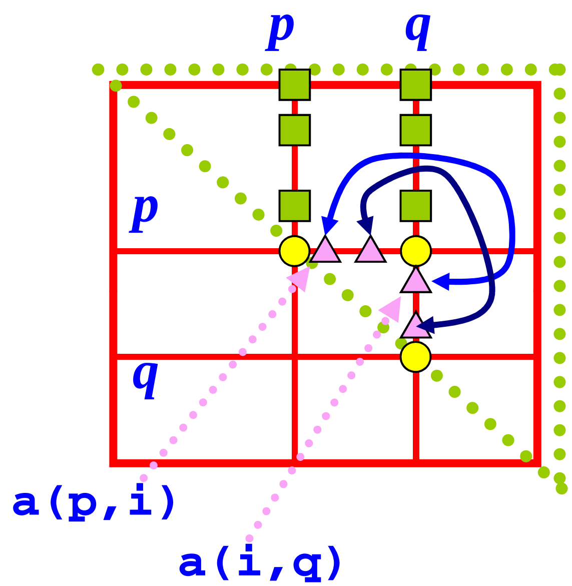
- Part III: **Update Row *p*+1 to Row *q*-1.**

$$A'_{i,p} = a_{i,p}C - a_{i,q}S$$

$$A'_{i,q} = a_{i,p}S + a_{i,q}C$$

```
! PART III: Update the portion between
!           row p+1 and row q-1
!
! h is used to save the new value
!    of a(p,i) because a(p,i) is used
!    to compute a(i,q) and cannot be
!    destroyed right away!

DO i = p+1, q-1
  h      = C*a(p,i) - S*a(i,q)
  a(i,q) = S*a(p,i) + C*a(i,q)
  a(p,i) = h
END DO
```



Note the symmetry in the update!
Note also that `a(p,i) = a(i,p)` **and** `a(q,i) = a(i,q)`

Symmetric Matrix Rotation: 11/11

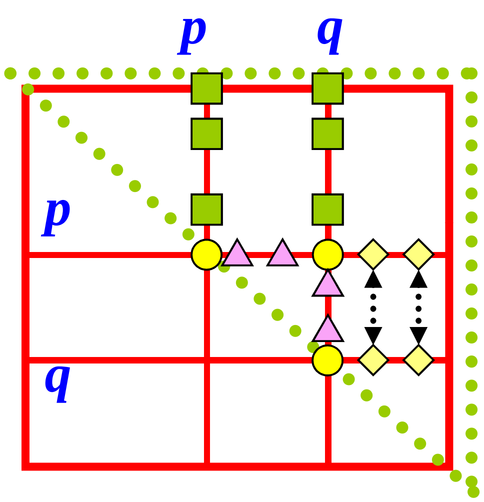
- Part IV: **Update Row *q*+1 to Row *n* .**

```
! PART IV: Update column q+1 to column n
!
! h is used to save the new value
!    of a(p,i) because a(p,i) is used to
!    compute a(q,i) and cannot be
!    destroyed right away!
! Due to symmetry, this part actually
!    updates the last sections of row p
!    and Row q

DO i = q+1, n
  h      = C*a(p,i) - S*a(q,i)
  a(q,i) = S*a(p,i) + C*a(q,i)
  a(p,i) = h
END DO
```

$$A'_{i,p} = a_{i,p}C - a_{i,q}S$$

$$A'_{i,q} = a_{i,p}S + a_{i,q}C$$



Note the symmetry in the update!

Eigenvalues of 2×2 Symmetric Matrices: 1/4

- **Consider a 2×2 symmetric matrix A:**

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \quad \textbf{where } a_{1,2} = a_{2,1}$$

- **Applying a rotation in the *xy*-plane yields the following symmetric matrix A' for some angle θ, where *C*=cos(θ) and *S*=sin(θ) :**

$$A' = R^T \cdot A \cdot R = \begin{bmatrix} a_{1,1}C^2 - 2a_{1,2}C \times S + a_{2,2}S^2 & (a_{1,1} - a_{2,2})C \times S + a_{1,2}(C^2 - S^2) \\ & a_{1,1}S^2 + 2a_{1,2}C \times S + a_{2,2}C^2 \end{bmatrix}$$

Eigenvalues of 2×2 Symmetric Matrices: 2/4

- **The off-diagonal element is**

$$\left(a_{1,1}-a_{2,2}\right)C\times S+a_{1,2}\left(C^2-S^2\right)$$

- **If a θ can be chosen so that the off-diagonal elements $a_{1,2}$ and $a_{2,1}$ are 0, matrix A is diagonal and the diagonal entries are eigenvalues!**

$$\left(a_{1,1}-a_{2,2}\right)C\times S+a_{1,2}\left(C^2-S^2\right)=0$$

$$\Downarrow$$

$$\frac{-a_{1,2}}{a_{1,1}-a_{2,2}}=\frac{C\times S}{C^2-S^2}=\frac{\cos(\theta)\sin(\theta)}{\cos^2(\theta)-\sin^2(\theta)}$$

$$\Downarrow$$

$$\frac{a_{1,2}}{a_{2,2}-a_{1,1}}=\frac{2}{2}\times\frac{\cos(\theta)\sin(\theta)}{\cos^2(\theta)-\sin^2(\theta)}=\frac{1}{2}\frac{\sin(2\theta)}{\cos(2\theta)}=\frac{\tan(2\theta)}{2}$$

$$\Downarrow$$

$$\tan(2\theta)=\frac{2a_{1,2}}{a_{2,2}-a_{1,1}}$$

$$\boxed{\theta=\frac{1}{2}\tan^{-1}\left(\frac{2a_{1,2}}{a_{2,2}-a_{1,1}}\right)}$$

if $a_{1,1}\neq a_{2,2}$

otherwise, $\theta=\pi/4$

simple facts from trigonometry

$$\boxed{\begin{aligned}\sin(2\theta)&=2\sin(\theta)\cos(\theta)\\ \cos(2\theta)&=\cos^2(\theta)-\sin^2(\theta)\end{aligned}}$$

Eigenvalues of 2×2 Symmetric Matrices: 3/4

- **Consider this A:**

$$A = \begin{bmatrix} 2 & \sqrt{3} \\ \sqrt{3} & 4 \end{bmatrix}$$

- **From matrix A, we have $a_{1,1} = 2$, $a_{2,2} = 4$ and $a_{1,2} = a_{2,1} = \sqrt{3}$.**
- **Since $\tan(2\theta) = 2a_{1,2}/(a_{2,2} - a_{1,1}) = \sqrt{3}$, we have $2\theta = \pi/3$, $\theta = \pi/6$, $S = \sin(\theta) = ½$, $C = \cos(\theta) = (\sqrt{3})/2$.**
- **The new $a_{1,1}$ is $a_{1,1}C^2 - a_{1,2}C\times S + a_{2,2}S^2 = 1$, the new $a_{2,2}$ is $a_{1,1}S^2 + a_{1,2}C\times S + a_{2,2}C^2 = 5$, and the new $a_{1,2} = a_{2,1} = 0$.**
- **Therefore, eigenvalues of A are +1 and +5!**

Eigenvalues of 2×2 Symmetric Matrices: 4/4

- **Let us verify the result. Since $S = \sin(\theta) = ½$ and $C = \cos(\theta) = (\sqrt{3})/2$, the rotation matrix R is:**

$$R = \begin{bmatrix} C & S \\ -S & C \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

- **The rotated A is $A' = R^T \cdot A \cdot R$:**

$$A' = R^T \cdot A \cdot R = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 2 & \sqrt{3} \\ \sqrt{3} & 4 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$$

- **A' is diagonal and eigenvalues of A are 1 and 5.**

Classic Jacobi Method: 1/13

- **Jacobi published a method in 1846 capable of finding all eigenvalues and eigenvectors of a symmetric matrix with repeated rotations.**
- **Find an off-diagonal entry with maximum absolute value, say $a_{p,q}$, where $p < q$.**
- **If $|a_{p,q}| < \varepsilon$, where ε is a given tolerance, stop.**
- **Apply a (p,q)-rotation to eliminate $a_{p,q}$ and $a_{q,p}$.**
- **Repeat this process until all off-diagonal elements become very small (*i.e.*, absolute value $< \varepsilon$).**
- **The diagonal entries are eigenvalues.**
- Rotations do not alter eigenvalues!

Classic Jacobi Method: 2/13

- **Basically, Jacobi method starts with a symmetric matrix $\mathbf{A_0 = A}$.**
- **Find a rotation matrix $\mathbf{R_1}$ so that an off-diagonal entry of $\mathbf{A_1 = R_1^T \cdot A_0 \cdot R_1}$ becomes $\mathbf{0}$.**
- **Then, find a rotation matrix $\mathbf{R_2}$ so that an off-diagonal entry of $\mathbf{A_2 = R_2^T \cdot A_1 \cdot R_2}$ becomes $\mathbf{0}$.**
- **The entry of $\mathbf{A_1}$ eliminated by $\mathbf{R_1}$ can become non-zero in $\mathbf{A_2}$; however, it would be smaller.**
- **Note the following fact:**

$$A_2 = R_2^T \cdot A_1 \cdot R_2 = R_2^T \cdot \left(R_1^T \cdot A_0 \cdot R_1\right) \cdot R_2$$

$$A_2 = R_2^T \cdot R_1^T \cdot A_0 \cdot R_1 \cdot R_2$$

Classic Jacobi Method: 3/13

- **Repeating this process, for iteration i, a rotation matrix $\mathbf{R}_i$ is found to eliminate one off-diagonal entry of $\mathbf{A}_i = \mathbf{R}_i^{\mathbf{T}} \cdot \mathbf{A}_{i\text{-}1} \cdot \mathbf{R}_i$.**
- **Thus, $\mathbf{A}_i$ is computed as follows:**

$$A_i = R_i^T \cdot R_{i-1}^T \cdot \cdots \cdot R_2^T \cdot R_1^T \cdot A_0 \cdot R_1 \cdot R_2 \cdot \cdots \cdot R_{i-1} \cdot R_i$$

- **Jacobi showed that after some number of iterations, all off-diagonal entries are small and the resulting matrix $\mathbf{A}_m$ is diagonal.**
- **Therefore, the diagonal entries of $\mathbf{A}_m$ are the eigenvalues of $\mathbf{A}$.**

Classic Jacobi Method: 4/13

- **Here is a template of the Jacobi method.**

find max off-diagonal entry

```
DO
  p = 1
  q = 2
  DO i = 1, n
    DO j = i+1, n
      IF (ABS(a(p,q)) < ABS(a(i,j)) THEN
        p = i
        q = j
      END IF
    END DO
  END DO
  IF (ABS(a(p,q)) < Tol)  EXIT
  Apply a (p,q)-rotation to matrix a(*,*)
END DO
```

Classic Jacobi Method: 5/13

- **The only remaining part is an efficient and accurate way of "applying a (p,q)-rotation."**
- **We saw the 2×2 case earlier: find an appropriate rotation angle θ, compute $C=\cos(\theta)$ and $S=\sin(\theta)$, and update matrix A.**
- **This approach requires $\tan^{-1}(\theta)$, which can be time consuming and may lose significant digits.**
- **Therefore, we need a faster and more accurate method.**

Classic Jacobi Method: 6/13

- The following shows the new $a_{p,p}$, $a_{q,q}$ and $a_{p,q}$ after rotation.

$$A'_{p,p} = a_{p,p}C^2 - 2a_{p,q}C \times S + a_{q,q}S^2$$

$$A'_{q,q} = a_{p,p}S^2 + 2a_{p,q}C \times S + a_{q,q}C^2$$

$$A'_{p,q} = A'_{q,p} = \left(a_{p,p} - a_{q,q}\right)C \times S + a_{p,q}\left(C^2 - S^2\right)$$

- Setting the new $A_{p,q}$ to 0 yields an angle θ that can eliminate $A_{p,q}$ and $A_{q,p}$.
- We shall use a different way to find tan(θ), from which sin(θ) and cos(θ) can be computed easily without the use of the $\tan^{-1}()$ function.

Classic Jacobi Method: 7/13

- **We follow the 2×2 case.**

$$A_{p,q} = \left(a_{p,p} - a_{q,q}\right) C \times S + a_{p,q}\left(C^2 - S^2\right) = 0$$

$$\Downarrow$$

$$\frac{a_{p,q}}{a_{q,q} - a_{p,p}} = \frac{C \times S}{C^2 - S^2} = \frac{\cos(\theta)\sin(\theta)}{\cos^2(\theta) - \sin^2(\theta)} = \frac{2}{2} \times \frac{\cos(\theta)\sin(\theta)}{\cos^2(\theta) - \sin^2(\theta)}$$

$$\Downarrow$$

$$\frac{a_{p,q}}{a_{q,q} - a_{p,p}} = \frac{1}{2} \times \frac{\sin(2\theta)}{\cos(2\theta)} = \frac{1}{2}\tan(2\theta)$$

$$\Downarrow$$

rewrite to use cot()

$$\tan(2\theta) = \frac{2a_{p,q}}{a_{q,q} - a_{p,p}} \Rightarrow \boxed{\cot(2\theta) = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}}$$

Classic Jacobi Method: 8/13

- **But, what we really need is tan(θ)!**
- **Let t = tan(θ), and we have t=S/C.**
- **From cot(2θ), we have the following:**

$$\cot(2\theta) = \frac{\cos(2\theta)}{\sin(2\theta)} = \frac{\cos^2(\theta) - \sin^2(\theta)}{2\sin(\theta)\cos(\theta)} = \frac{C^2 - S^2}{2S \times C}$$

- **Divide the numerator and denominator with C^2:**

$$\cot(2\theta) = \frac{(C^2 - S^2)/C^2}{(2S \times C)/C^2} = \frac{1 - S^2/C^2}{2(S/C)} = \frac{1-t^2}{2t}$$

- **Therefore, we have:**

$$\Delta = \frac{1-t^2}{2t} \quad \textbf{where } \Delta = \cot(2\theta) = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}$$

Classic Jacobi Method: 9/13

- **From $\Delta=(1-t^2)/(2t)$, we have $t^2+2\Delta t-1 = 0$.**
- **This means the desired $t = \tan(\theta)$ is one of the two roots of $t^2+2\Delta t-1 = 0$.**
- **The roots of $t^2+2\Delta t-1 = 0$ are**

$$t = -\Delta \pm \sqrt{\Delta^2 + 1}$$

- Which root is better?
- Important Fact: **If x_1 and x_2 are roots of $x^2+bx+c=0$, then $x_1+x_2=-b$ and $x_1 \times x_2=c$.**
- **Since the product of the roots of $t^2+2\Delta t-1 = 0$ is -1, the *smaller* (or *desired*) one must be in [-1,1].**

Classic Jacobi Method: 10/13

- **Consider the following manipulation:**

avoid cancellation

$$\left(-\Delta \pm \sqrt{\Delta^2+1}\right) \times \frac{-\Delta \mp \sqrt{\Delta^2+1}}{-\Delta \mp \sqrt{\Delta^2+1}} = \frac{1}{\Delta \pm \sqrt{\Delta^2+1}}$$

- **We have to avoid cancellation when Δ is large.**
- **If $\Delta > 0$, use $+$. The denominator is $\Delta+(\Delta^2+1)^{1/2} > 1$ and the positive root is less than 1.**
- **If $\Delta < 0$, use $-$. The denominator is $\Delta-(\Delta^2+1)^{1/2} < -1$ and the negative root is greater than -1.**

Classic Jacobi Method: 11/13

- If $\Delta > 0$ (*resp*., $\Delta < 0$), the desired "smaller" root is $1/(\Delta+(\Delta^2+1)^{1/2})$ (*resp*., $1/(\Delta-(\Delta^2+1)^{1/2})$).
- This root can be rewritten as follows:

$$t = \frac{sign(\Delta)}{|\Delta| + \sqrt{\Delta^2+1}} \quad \textbf{and} \quad |t| \le 1$$

- Since $|t| \le 1$, the angle of rotation is in $[-\pi/4,\pi/4]$.
- After t (*i.e.*, $\tan(\theta)$) is computed, $C=\cos(\theta)$ and $S=\sin(\theta)$ are the following

$$C = \cos(\theta) = \frac{1}{\sqrt{1+t^2}} \quad \textbf{and} \quad S = \sin(\theta) = C \times t$$

Classic Jacobi Method: 12/13

Compute Δ and *t*, and obtain *C*=cos(θ) and *S*=sin(θ)

```
! This section computes C and S
!   From a(p,p), a(q,q) and a(p,q)
t = 1.0
IF (a(p,p) != a(q,q)) THEN
  D = (a(q,q)-a(p,p))/(2*a(p,q))
  t = SIGN(1/(ABS(D)+SQRT(D*D+1)),D)
END IF
C = 1/SQRT(1+t*t)
S = C*t
```

In Fortran 90, `SIGN(a,b)` means using the sign of `b` with the absolute value of `a`. Thus, `SIGN(10,-1)` and `SIGN(-15,1)` yield `-10` and `15`, respectively.

Classic Jacobi Method: 13/13

- Finally, the classic Jacobi method is shown below.
- Scan the upper triangular portion for max |`a(p,q)`|, where `p < q`.
- A *(p,q)*-rotation based on the values of *C* and *S* sets `a(p,q)` and `a(q,p)` to zero.

Classic Jacobi Method

```
DO
  Find the max |a(p,q)| entry, p < q
  IF (|a(p,q)| < Tol)  EXIT
  From a(p,p), a(q,q) and a(p,q) compute t
  From t compute C and S
  Perform a (p,q)-rotation with a(p,q)=a(q,p)=0
END DO
```

Computation Example: 1/5

- **Consider the following symmetric matrix:**

$$A = \begin{bmatrix} 12 & 6 & -6 \\ 6 & 16 & 2 \\ -6 & 2 & 16 \end{bmatrix}$$

- **The largest element is on row 1 and column 2.**
- **Since $a_{1,1}=12$, $a_{1,2}=6$ and $a_{2,2}=16$, we have $\Delta = (a_{2,2}-a_{1,1})/(2a_{1,2})=0.33333334$, and $t = 0.7207582$.**
- **From $t = 0.7207582$, we have $C = 0.8112422$ and $S = 0.5847103$.**

$$R_{1,2} = \begin{bmatrix} 0.8112422 & 0.5847103 & \\ -0.5847103 & 0.8112422 & \\ & & 1 \end{bmatrix}$$

Computation Example: 2/5

- **The new $\mathbf{A = R_{1,2}^T \cdot A \cdot R_{1,2}}$ is**

eliminated

$$A = \begin{bmatrix} 7.6754445 & 0.0 & -6.036874 \\ 0.0 & 20.32456 & -1.885777 \\ -6.036874 & -1.885777 & 16.0 \end{bmatrix}$$

- **The off-diagonal entry with the largest absolute value is $a_{1,3}$ = -6.036874.**
- **Since $a_{1,1}$=7.6754445, $a_{1,3}$=-6.036874 and $a_{3,3}$=16, $\Delta = (a_{3,3}-a_{1,1})/(2a_{1,3})$ = -0.68947533, t= -0.5251753, C=0.885334, and S=-0.4645553.**

Computation Example: 3/5

- **The rotation matrix $R_{1,3}$ is:**

$$R_{1,3} = \begin{bmatrix} 0.885334 & & -0.46495553 \\ & 1 & \\ 0.46495553 & & 0.885334 \end{bmatrix}$$

- **The new matrix $A = R_{1,3}^T \cdot A \cdot R_{1,3}$ is**

was 0!

eliminated

$$A = \begin{bmatrix} 4.505028 & \boxed{-0.8768026} & 0.0 \\ & 20.32456 & -1.669543 \\ & & 19.17042 \end{bmatrix}$$

eliminate this one in the next iteration

Computation Example: 4/5

- **The largest entry is $a_{2,3}$=-1.669543.**
- **Since $a_{2,2}$=20.32456, $a_{2,3}$=-1.669543, $a_{3,3}$=19.17042, $\Delta = (a_{3,3}-a_{2,2})/(2a_{2,3})$=0.34564623, and t = 0.71240422.**
- **Therefore, C=0.81445753 and S=0.58022314.**
- **The new rotation matrix $\mathbf{R}_{2,3}$ is:**

$$R_{2,3} = \begin{bmatrix} 1 & & \\ & 0.81445753 & 0.58022314 \\ & -0.58022314 & 0.81445753 \end{bmatrix}$$

Computation Example: 5/5

- **The new matrix $A = R_{2,3}^T \cdot A \cdot R_{2,3}$ is:**

They were 0!

$$A = \begin{bmatrix} 4.505028 & \boxed{-0.7141185 \quad -0.5087411} \\ & 21.51395 & 0.0 \\ & & 17.98103 \end{bmatrix}$$

eliminated

- **With 5 more iterations, the new matrix A becomes**

$$A = \begin{bmatrix} 4.455996 & & \\ & 21.54401 & \\ & & 18.0 \end{bmatrix}$$

- **The eigenvalues are 4.455996, 21.54401, 18.0**
- **In hand calculation of small matrices, direct matrix multiplication may be more convenient!**

Where Are the Eigenvectors: 1/6

- An important fact: **If R is a rotation matrix, then $\mathbf{R}^{-1}=\mathbf{R}^{T}$! So, R's inverse is R's transpose. Note that $C^2 + S^2 = 1$!**

identity matrix

$$R \bullet R^T = \begin{bmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & \mathbf{0} & & & & & \\ & & 1 & & & & & & & & \\ & & & C & & & S & & & & \\ & & & & \ddots & & & & & & \\ & \mathbf{0} & & & & 1 & & & & \mathbf{0} & \\ & & & & & & \ddots & & & & \\ & & & -S & & & C & & & & \\ & & & & & & & 1 & & & \\ & & & & & \mathbf{0} & & & \ddots & & \\ & & & & & & & & & & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & \mathbf{0} & & & & & \\ & & 1 & & & & & & & & \\ & & & C & & & -S & & & & \\ & & & & \ddots & & & & & & \\ & \mathbf{0} & & & & 1 & & & & \mathbf{0} & \\ & & & & & & \ddots & & & & \\ & & & S & & & C & & & & \\ & & & & & & & 1 & & & \\ & & & & & \mathbf{0} & & & \ddots & & \\ & & & & & & & & & & 1 \end{bmatrix} = I$$

Where Are the Eigenvectors: 2/6

- Two more simple facts: $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$ and $(A \cdot B)^T = B^T \cdot A^T$.
- Jacobi method uses a sequence of rotation matrices R_1, R_2, …, R_m to transform the given matrix A to a diagonal form D:

$$R_m^T \cdot \left(R_{m-1}^T \cdot \left(\cdots \left(R_2^T \cdot \left(R_1^T \cdot A \cdot R_1 \right) \cdot R_2 \right) \cdots \right) \cdot R_{m-1} \right) \cdot R_m = D$$

- The above is equivalent to:

$$\left(R_m^T \cdot R_{m-1}^T \cdot \cdots \cdot R_2^T \cdot R_1^T \right) \cdot A \cdot \left(R_1 \cdot R_2 \cdot \cdots \cdot R_{m-1} \cdot R_m \right) = D$$

- Since $(A \cdot B)^T = B^T \cdot A^T$, we have the following:

$$(R_1 \cdot R_2 \cdot \ldots \cdot R_{m-1} \cdot R_m)^T \cdot A \cdot (R_1 \cdot R_2 \cdot \ldots \cdot R_{m-1} \cdot R_m) = D$$

Where Are the Eigenvectors: 3/6

- **Let $V = R_1 \cdot R_2 \cdot \ldots \cdot R_m$. Then, we have $\boxed{V^T} \cdot A \cdot V = D$.**
- **We shall show $V^{-1} = V^T$. Since $R^{-1} = R^T$ and**

$$V^{-1} = (R_1 \cdot R_2 \cdots R_m)^{-1} = R_m^{-1} \cdots R_2^{-1} \cdot R_1^{-1}$$

we have

$$V^{-1} = R_m^{-1} \cdots R_2^{-1} \cdot R_1^{-1} = R_m^T \cdots R_2^T \cdot R_1^T = \left(R_1 \cdot R_2 \cdots R_m\right)^T = V^T$$

- **Therefore, $\boxed{V^{-1}} \cdot A \cdot V = D$ holds.**
- **Multiplying both sides by V yields $A \cdot V = V \cdot D$.**

$$V^{-1} \cdot A \cdot V = D$$

$$V \cdot \left(V^{-1} \cdot A \cdot V\right) = V \cdot D$$

$$A \cdot V = V \cdot D$$

Where Are the Eigenvectors: 4/6

- **Let the column vectors of V be $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ (*i.e.*, $\mathbf{V} = [\mathbf{v}_1|\mathbf{v}_2|\mathbf{v}_3|\ldots|\mathbf{v}_n]$).**
- **Then, $\mathbf{V}\cdot\mathbf{D} = [d_1\mathbf{v}_1|d_2\mathbf{v}_2|\ldots|d_n\mathbf{v}_n]$ and $\mathbf{A}\cdot\mathbf{v}_i = d_i\mathbf{v}_i$, and the eigenvectors are the columns of V!**

$$\underbrace{\begin{array}{ccccc} \mathbf{v}_1 & \mathbf{v}_2 & & \mathbf{v}_{n\text{-}1} & \mathbf{v}_n \\ \end{array}}_{} $$

$$\underbrace{\left[\begin{array}{c|c|c|c|c} v_{1,1} & v_{1,2} & \cdots & v_{1,n-1} & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,n-1} & v_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{n-1,1} & v_{n-1,2} & \cdots & v_{n-1,n-1} & v_{n-1,n} \\ v_{n,1} & v_{n,2} & \cdots & v_{n,n-1} & v_{n,n} \end{array}\right]}_{\mathbf{V}} \cdot \underbrace{\begin{bmatrix} d_1 & & & & \\ & d_2 & & \mathbf{0} & \\ & & \ddots & & \\ & \mathbf{0} & & d_{n-1} & \\ & & & & d_n \end{bmatrix}}_{\mathbf{D}} = [d_1 \cdot \mathrm{v}_1 \,|\, d_2 \cdot \mathrm{v}_2 \,|\cdots|\, d_n \cdot \mathrm{v}_n]$$

Where Are the Eigenvectors: 5/6

- **The following shows an *inefficient* way using matrix multiplication.**

```
! A is the input n×n symmetric matrix
V = the identify matrix
DO
  find the largest off-diagonal entry |a(p,q)|
  IF (|a(p,q)| < Tol) EXIT
  compute Δ, t, S and C
  update matrix a(*,*)
  V = V*R ! eigenvectors
END DO
! Eigenvalues are the diagonal entries of A
! Eigenvectors are the columns of V
```

Where Are the Eigenvectors: 6/6

- **The computation of V=V·R is similar to A·R!**
- **V = [$v_{i,j}$] is not symmetric, and two *complete* columns (*i.e.*, columns *p* and *q*) must be updated.**

$$
V \bullet R = \begin{bmatrix}
 & v_{1,p}C - v_{1,q}S & & v_{1,p}S + v_{1,q}C & \\
 & v_{2,p}C - v_{2,q}S & v_{i,j} & v_{2,p}S + v_{2,q}C & \\
 & \vdots & & \vdots & \\
 & v_{p,p}C - v_{p,q}S & & v_{p,p}S + v_{p,q}C & \\
v_{i,j} & \vdots & \ddots & \vdots & v_{i,j} \\
 & v_{q,p}C - v_{q,q}S & & v_{q,p}S + v_{q,q}C & \\
 & \vdots & & \vdots & \\
 & v_{n-1,p}C - v_{n-1,q}S & & v_{n-1,p}S + v_{n-1,q}C & \\
 & v_{n,p}C - v_{n,q}S & v_{i,j} & v_{n,p}S + v_{n,q}C &
\end{bmatrix}
$$

col *p* col *q*

Example-Continued: 1/4

- **The input matrix is:**

$$A = \begin{bmatrix} 12 & 6 & -6 \\ 6 & 16 & 2 \\ -6 & 2 & 16 \end{bmatrix}$$

- **Since $a_{1,2}$ is the largest, rotation matrix $R_{1,2}$ is:**

$$R_{1,2} = \begin{bmatrix} 0.8112422 & 0.5847103 & \\ -0.5847103 & 0.8112422 & \\ & & 1 \end{bmatrix}$$

- **Matrix V, approx. eigenvectors, is $I \cdot R_{1,2}$**

$$V = I \cdot R_{1,2} = \begin{bmatrix} 0.8112422 & 0.5847103 & \\ -0.5847103 & 0.8112422 & \\ & & 1 \end{bmatrix}$$

Example-Continued: 2/4

- **Now, the new matrix A is:**

$$A = \begin{bmatrix} 7.6754445 & 0.0 & -6.036874 \\ 0 & 20.32456 & -1.885777 \\ -6.036874 & -1.885777 & 16.0 \end{bmatrix}$$

- **The largest off-diagonal is $a_{1,3}$ and $R_{1,3}$ is**

$$R_{1,3} = \begin{bmatrix} 0.885334 & & -0.46495553 \\ & 1 & \\ 0.46495553 & & 0.885334 \end{bmatrix}$$

- **Therefore, new approx. eigenvectors matrix V is**

$$V = V \cdot R_{1,3} = \begin{bmatrix} 0.7182204 & 0.5847103 & -0.3771916 \\ -0.5176639 & 0.8112422 & 0.27186423 \\ 0.4649555 & 0 & 0.885334 \end{bmatrix}$$

Example-Continued: 3/4

- **For iteration 3, matrix A is:**

$$A = \begin{bmatrix} 4.505028 & -0.8768026 & 0.0 \\ & 20.32456 & -1.669543 \\ & & 19.17042 \end{bmatrix}$$

- **Since $a_{2,3}$ is the largest, $R_{2,3}$ is:**

$$R_{2,3} = \begin{bmatrix} 1 & & \\ & 0.81445753 & 0.58022314 \\ & -0.58022314 & 0.81445753 \end{bmatrix}$$

- **The approx. eigenvector matrix V is:**

$$V = V \cdot R_{2,3} = \begin{bmatrix} 0.7182204 & 0.6950770 & 0.03205594 \\ -0.5176639 & 0.5029804 & 0.6921234 \\ 0.4649555 & -0.5136913 & 0.7210670 \end{bmatrix}$$

Example-Continued: 4/4

- **Five more iterations yields the new matrix A:**

$$A = \begin{bmatrix} 4.455996 & & \\ & 21.54401 & \\ & & 18.0 \end{bmatrix}$$

- **The approx. eigenvector matrix V is:**

$$V = \begin{bmatrix} 0.7473423 & 0.6644393 & -0.3606413\text{E-}6 \\ -0.4698294 & 0.5284512 & 0.7071065 \\ 0.4698295 & -0.5284505 & 0.7071069 \end{bmatrix}$$

Normally eigenvalues are sorted and eigenvectors are normalized

Convergence of Jacobi Method

- **The classic Jacobi method always converges.**
- **Let $S(A)$ be the sum of squares of all off-diagonal entries, where $A=[a_{i,j}]$ is a $n \times n$ symmetric matrix:**

$$S(A) = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} a_{i,j}^2 = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{i,j}^2$$

- **Then, the sequence of values $S(A_k)$ *decreases monotonically to zero*, where A_k is the result of the k-th rotation.**
- **This means eventually all off-diagonal entries will become zeros (*i.e.*, diagonal).**

Two Improvements: 1/4

- The following two useful improvements are due to H. Rutishauser.
- The following formula was used to compute $t = \tan(\theta)$:

$$t = \frac{sign(\Delta)}{|\Delta| + \sqrt{\Delta^2 + 1}} \quad \textbf{where} \quad \Delta = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}$$

- If Δ is large, Δ^2 may cause overflow.
- To avoid overflow, one may set t to $1/(2\Delta)$ if Δ is large because Δ^2+1 is close to Δ^2.
- How large is large enough so that Δ^2 will not overflow? Exercise!

Two Improvements: 2/4

- **The updating formulas for the new $a_{p,p}$ and $a_{q,q}$, and the formulas for rows p and q and columns p and q of matrix A can be modified so that they are computationally more stable than the original.**
- Reminder: **The following formulas were used:**

$$\frac{C^2 - S^2}{2C \times S} = \cot(2\theta) = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}$$

$$C = \cos(\theta) \qquad S = \sin(\theta) \qquad t = \frac{S}{C} = \tan(\theta)$$

Two Improvements: 3/4

- **Simplify the formula for the new *$a_{p,p}$*:**

$$\begin{aligned}
A'_{p,p} &= a_{p,p}C^2 - 2a_{p,q}C \times S + a_{q,q}S^2 \\
&= a_{p,p}(1 - S^2) - 2a_{p,q}C \times S + a_{q,q}S^2 \\
&= a_{p,p} + S^2\left(a_{q,q} - a_{p,p}\right) - 2a_{p,q}C \times S \\
&= a_{p,p} + S^2\left(2a_{p,q}\frac{C^2 - S^2}{2C \times S}\right) - 2a_{p,q}C \times S \\
&= a_{p,p} - ta_{p,q}
\end{aligned}$$

- **The one for *$a_{q,q}$* is similar:**

$$A'_{q,q} = a_{q,q} + ta_{p,q}$$

$$\boxed{\frac{C^2 - S^2}{2C \times S} = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}}$$

Two Improvements: 4/4

- **Columns p and q of A' = A·R and V' = V·R (eigenvector matrix) were updated as follows:**

$$A'_{i,p} = C\times a_{i,p} - S\times a_{i,q} \quad \textbf{and} \quad A'_{i,q} = S\times a_{i,p} + C\times a_{i,q}$$

- **Let $\tau = \tan(\theta/2) = S/(1+C)$.**
- **Note the following trigonometry identity:**

$$\tau = \tan(\theta/2) = \frac{S}{1+C} = \frac{1-C}{S} \qquad \Rightarrow \qquad C = 1 - S\times\tau$$

- **Now, we have**

$$\begin{aligned} A'_{i,p} &= C\times a_{i,p} - S\times a_{i,q} & A'_{i,q} &= S\times a_{i,p} + C\times a_{i,q} \\ &= (1-S\times\tau)a_{i,p} - S\times a_{i,q} & &= S\times a_{i,p} + (1-S\times\tau)\times a_{i,q} \\ &= a_{i,p} - S\left(a_{i,q} + \tau\times a_{i,p}\right) & &= a_{i,q} + S\left(a_{i,p} - \tau\times a_{i,q}\right) \end{aligned}$$

Cyclic Jacobi Methods: 1/5

- **To find the max entry, the upper diagonal $n(n-1)/2$ entries must be scanned.**
- **However, performing a rotation only requires $4n$ multiplication (*i.e.*, updating two columns).**
- **Is this "search" worthwhile? In other word, would this search for the max entry requires more time than updating the matrix?**
- **What if we forget about the search and just perform rotations in some order?**
- ***Cyclic Jacobi methods* just does that.**

Cyclic Jacobi Methods: 2/5

- A version of cyclic Jacobi methods scans the matrix in row order:

$$
\begin{array}{ccccc}
(1,2) & (1,3) & (1,4) & \cdots & (1,n) \\
 & (2,3) & (2,4) & \cdots & (2,n) \\
 & & (3,4) & \cdots & (3,n) \\
 & & & \cdots & \vdots \\
 & & & & (n-1,n)
\end{array}
$$

one sweep

- If the encountered entry $|a_{p,q}| \geq \varepsilon$, do a (p,q)-rotation to eliminate it.
- A complete round is called a ***sweep***.
- If a sweep does not eliminate any entry, all entries are small enough and stop!

Cyclic Jacobi Methods: 3/5

- Here is a template of this special cyclic method:

one sweep

```
DO
  NO_change = .TRUE.
  DO p = 1, n-1
    DO q = p+1, n
      IF (ABS(a(p,q)) >= Tol) THEN
        NO_change = .FALASE.
        perform a (p,q)-rotation
        update eigenvectors
      END IF
    END DO
  END DO
  IF (NO_change) EXIT
END DO
```

Cyclic Jacobi Methods: 4/5

- **This cyclic Jacobi method converges, and the sum of squares of off-diagonal entries $S(A_k)$ is a monotonic, *non-increasing* sequence.**
- Observation: **Since $S(A_k)$ is the sum of squares of all off-diagonal entries, if it decreases to zero all off-diagonal entries should be even smaller!**
- **Therefore, $S(A_k)$ can be used as a tolerance.**
- **Instead of recomputing $S(A_k)$ for each (p,q)–rotation, we may update it after each sweep!**

Cyclic Jacobi Methods: 5/5

- **Here is another, better version:**

```
DO
   S = 0
   DO p = 1, n-1
     DO q = p+1, n
       S = S + a(p,q)**2
     END DO
   END DO
   IF (2*S < Tol**2) EXIT
   perform a sweep without
     checking tolerance
END DO
```

Since this double DO only goes through the upper triangular part, S should be doubled to compute *S*(A).

This actually means

$$\sqrt{S(A)} = \sqrt{\sum_{i=1}^{n-1} \sum_{j=1, j \neq i}^{n} a_{i,j}^2} < \varepsilon$$

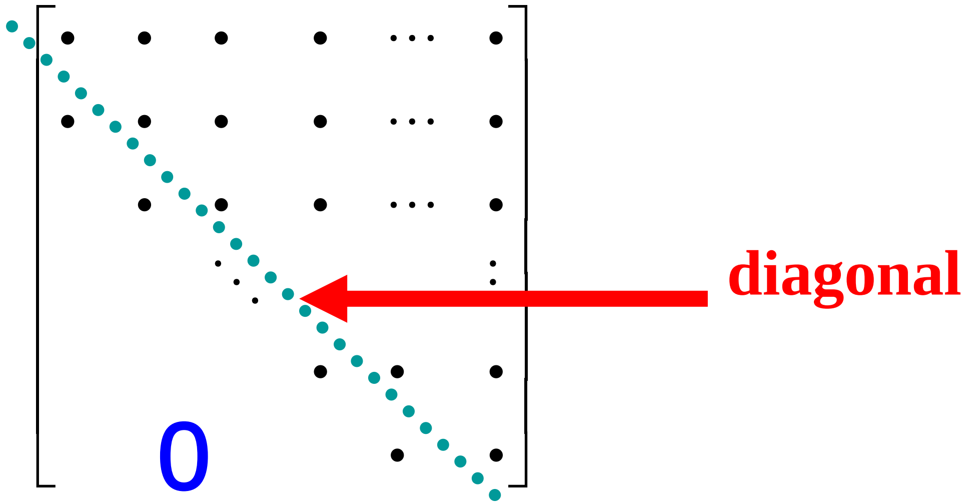
A Few Notes: 1/3

- **Computing eigenvalues and eigenvectors is not easy for general matrices.**
- **Methods (*e.g.*, Givens and Householder) are available to reduce a symmetric matrix to the *tridiagonal* form from which eigenvalues and eigenvectors can be computed efficiently.**

$$\begin{bmatrix} \alpha_1 & \gamma_2 & & & & & \\ \gamma_2 & \alpha_2 & \gamma_3 & & & \mathbf{0} & \\ & \gamma_3 & \alpha_3 & \gamma_4 & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & \mathbf{0} & & & \alpha_{n-2} & \gamma_{n-1} & \\ & & & & \gamma_{n-1} & \alpha_{n-1} & \gamma_n \\ & & & & & \gamma_n & \alpha_n \end{bmatrix}$$

A Few Notes: 2/3

- Methods are available to reduce a general matrix to the Hessenberg form.



- Then, other methods are used to find eigenvalues and eigenvectors of a matrix in Hessenberg form.

A Few Notes: 3/3

- **One of the most powerful and recommended methods is the QR algorithm, which can be used with tridiagonal and Hessenberg forms.**
- **However, Jacobi's method is more accurate than QR![1]**
- **Check `http://www.netlib.org/lapack/` for free linear algebra Fortran programs.**
- **You may also find useful programs in *Numerical Recipes* by Press, at el. There are commercial products such as the IMSL and NAG libraries, and Matlab.**

[1] J. Demmel and K. Veselić, Jacobi's method is more accurate than QR, *SIAM J. Matrix Anal. Appl.*, Vol. 13(1992), No. 4(Oct), pp. 1204-1245.

The End