

16. Computing eigenvalues and eigenvectors

- power iteration
- shift and inverse techniques
- simultaneous (subspace) iteration
- QR iteration

Eigenvalue problem

given $A \in \mathbb{R}^{n \times n}$, find eigenvector x and eigenvalue λ :

$$Ax = \lambda x$$

- we assume that $A \in \mathbb{R}^{n \times n}$ has only real eigenvalues/eigenvectors
- we discuss methods for finding eigenvalues and eigenvectors

Need for iterative methods

- polynomial $\det(\lambda I - A) = \lambda^n + a_{n-1}\lambda^{n-1} + \dots + a_1\lambda + a_0$ roots are eig. values of

$$A = \begin{bmatrix} -a_{n-1} & -a_{n-2} & -a_{n-3} & \cdots & -a_1 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

- no closed-form formula exists for roots of a general polynomial of degree $n \geq 5$
- hence, no finite algorithm exists for eigenvalues of general matrix of order $n \geq 5$

The power iteration

given $A \in \mathbb{R}^{n \times n}$ and nonzero $v^{(0)} \in \mathbb{R}^n$

for $k = 1, 2, \dots$

1. $\tilde{v} = Av^{(k-1)}$
 2. $v^{(k)} = \tilde{v}/\|\tilde{v}\|$ (or $v^{(k)} = \tilde{v}/\|\tilde{v}\|_\infty$)
 3. $\lambda^{(k)} = v^{(k)T}Av^{(k)}$
-

- if $v \in \mathbb{C}^n$ and $\lambda \in \mathbb{C}$, we replace v^T by the conjugate transpose v^H
- $\lambda^{(k)}$ converges to the *largest* eigenvalue in magnitude (dominant eigenvalue)
- $v^{(k)}$ converges to eigenvector corresponding to largest eigenvalue
- reason for normalization is to keep the iterate magnitude in check

Convergence of the power method

- let the eigenvalues and eigenvectors of A be $\{\lambda_j, x_j\}$ for $j = 1, \dots, n$
- assume $\lambda_1, \lambda_2, \dots, \lambda_n$ are sorted in decreasing order in terms of their magnitude

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$$

Dominant eigenvector

- output at the k th step is $v^{(k)} = \gamma_k A^k v^{(0)}$, where γ_k guarantees $\|v^{(k)}\| = 1$
- assume A has lin. indep. eigenvectors such that $v^{(0)} = \sum_{j=1}^n \beta_j x_j$ for some β_j
- multiplying $v^{(0)}$ by A we obtain

$$Av^{(0)} = \sum_{j=1}^n \beta_j Ax_j = \sum_{j=1}^n \beta_j \lambda_j x_j$$

- continuing, for any positive integer k we have

$$A^k v^{(0)} = \sum_{j=1}^n \beta_j \lambda_j^k x_j$$

- suppose that $\beta_1 \neq 0$, then

$$v^{(k)} = \gamma_k \lambda_1^k \sum_{j=1}^n \beta_j (\lambda_j / \lambda_1)^k x_j = \gamma_k \lambda_1^k \left(\beta_1 x_1 + \sum_{j=2}^n \beta_j (\lambda_j / \lambda_1)^k x_j \right)$$

- it follows that for $j \geq 2$ we have $|\lambda_j / \lambda_1|^k \rightarrow 0$ as $k \rightarrow \infty$
- the power method converges linearly with rate $|\lambda_2 / \lambda_1|$:
in the limit we obtain a unit vector in the direction of x_1

Dominant eigenvalue

- suppose v is an approximate eigenvector for real A
- the “best” estimate of λ is the least squares solution of $\|v\lambda - Av\|^2$
- solution is the *Rayleigh quotient* $\mu(v) = v^T Av / v^T v$
- if v were an eigenvector, then $\mu(v)$ gives the associated eigenvalue
- note that by the normalization of $v^{(k)}$, we have $\mu(v^{(k)}) = v^{(k)T} A v^{(k)}$

Limitations of the power method

- initialization may have no component in the dominant eigenvector x_1 : $\beta_1 = 0$
 - this is extremely unlikely if $v^{(0)}$ is chosen randomly
 - in practice $\beta_1 \neq 0$ due to roundoff errors
- there may be more than one eigenvalue having the same (maximum) modulus
 - iteration may converge to a linear combination of the corresponding eigenvectors
- for real matrix and real initialization, iteration cannot converge to complex vector
- we have assumed that all the eigenvectors of the matrix are linearly independent:
power method can still be applied, but convergence may be very slow

Example

- we consider two diagonal matrices

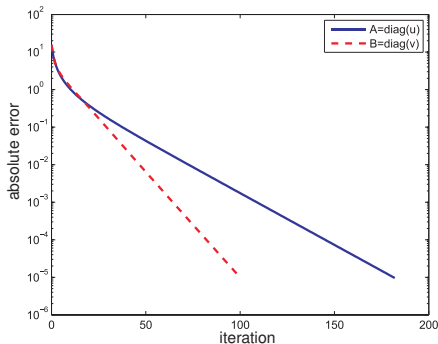
$$A = \text{diag}(1, 2, \dots, 32) \quad \text{and} \quad B = \text{diag}(1, 2, \dots, 30, 30, 32)$$

- in MATLAB syntax the matrices are

$$u = [1:32]; \quad v = [1:30, 30, 32];$$

$$A = \text{diag}(u); \quad B = \text{diag}(v);$$

- plot depicts the absolute error $|\lambda_1^{(k)} - \lambda_1| = |\lambda_1^{(k)} - 32|$



Example: PageRank

Problem: rank relevant webpages and determine their importance

- models internet as directed network with n nodes (webpages)
- PageRank computes the importance or *rank* $x_i \geq 0$ of a web site a_i by counting the number of pages pointing to a_i
- locations or indices of pages related to x_i are given by the set B_i
- if page $j \in B_i$ points to N_j pages including page i , then we set

$$x_i = \sum_{j \in B_i} \frac{1}{N_j} x_j, \quad i = 1, \dots, n$$

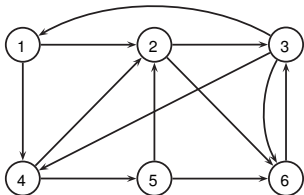
- an eigenvalue problem: $x = Ax$, where $A_{ij} = 1/N_j$ if $j \in B_i$ and zero otherwise
- if $v^{(0)} = \frac{1}{n} \mathbf{1}$, then, for $k = 0, 1, \dots$, the iteration is defined by

$$v_i^{(k+1)} = \sum_{j \in B_i} \frac{1}{N_j} v_j^{(k)}, \quad i = 1, \dots, n$$

(without normalization and eigenvalue estimation-operations)

Example

webpages are nodes in the graph, numbered 1 through 6



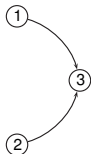
$$A = \begin{bmatrix} 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 1 \\ \frac{1}{2} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{3} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

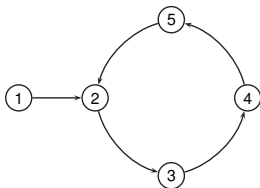
- j th column represents the outlinks from page j ; rows indicate the inlinks
- columns sum up to 1; such a matrix is called *column stochastic*
- matrix A is elementwise nonnegative
- Perron–Frobenius theorem ensures for this matrix that
 - a unique, simple eigenvalue 1 exists
 - all other eigenvalues of A are smaller in magnitude
 - the associated eigenvector of the dominant eigenvalue has real entries

- to compute PageRank, we start with $v^{(0)} = (1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$
- eventually, the method converges to the desired PageRank vector

$$x = (0.0994, 0.1615, 0.2981, 0.1491, 0.0745, 0.2174)$$

- this shows that node 3 is the top-ranked entry
- the ranking according to the values of x is (3, 6, 2, 4, 1, 5)
- in this example, since $\|v^{(0)}\|_1 = 1$, all subsequent iterates satisfy $\|v^{(k)}\|_1 = 1$
- interpretation: gives probability of a surfer being at a given webpage after a long (“infinite”) time, regardless of where they started their journey
- when there are no outlinks from a certain webpage we can replace the whole column from zeros to entries all equal to $1/n$ (in our example, $1/6$)





- entering a “dead end,” as shown in diagram can be fixed by forming:

$$\alpha A + (1 - \alpha)u\mathbf{1}^T$$

where α is a damping factor and u is a personalization vector

- interpretation: if $\alpha = 0.85$ and $u = \frac{1}{n}\mathbf{1}$, then with probability 0.85 a surfer follows links and 0.15 probability jump randomly to anywhere
- $\alpha A + (1 - \alpha)u\mathbf{1}^T$ has dominant eigenvalue is 1, the rest of its eigenvalues (which are generally complex) are bounded in magnitude by α

Outline

- power iteration
- **shift and inverse techniques**
- simultaneous (subspace) iteration
- QR iteration

Shifted power iteration

- recall that the eigenvalues of $A - \alpha I$ are $\lambda_i - \alpha$ with same eigenvectors of A
- using shifts convergence maybe enhanced to $|\lambda_2 - \alpha|/|\lambda_1 - \alpha| < |\lambda_2/\lambda_n|$

Power iteration with shifts

given $A \in \mathbb{R}^{n \times n}$ and nonzero $v^{(0)} \in \mathbb{R}^n$

for $k = 1, 2, \dots$

1. $\tilde{v} = (A - \alpha I)v^{(k-1)}$
 2. $v^{(k)} = \tilde{v}/\|\tilde{v}\|$ (or $v^{(k)} = \tilde{v}/\|\tilde{v}\|_\infty$)
 3. $\lambda^{(k)} = v^{(k)T}Av^{(k)}$
-

Two extreme eigenvalues: shift allows us to find extreme eigenvectors λ_1 or λ_n

- converges to λ_1 if $|\lambda_1 - \alpha| > |\lambda_n - \alpha|$
- converges to λ_n if $|\lambda_n - \alpha| > |\lambda_1 - \alpha|$

The inverse iteration

Inverse Iteration: power iteration applied to $(A - \alpha I)^{-1}$

given $A \in \mathbb{R}^{n \times n}$, nonzero $v^{(0)} \in \mathbb{R}^n$, and shift α

for $k = 1, 2, \dots$

1. solve $(A - \alpha I)\tilde{v} = v^{(k-1)}$
 2. $v^{(k)} = \tilde{v}/\|\tilde{v}\|$
 3. $\lambda^{(k)} = v^{(k)T}Av^{(k)}$
-

- requires solving linear eq. in each iteration
- eigenvalues of $(A - \alpha I)^{-1}$ are $1/(\lambda_j - \alpha)$ with same eigenvectors as A
- converges to λ_j with largest $1/|\lambda_j - \alpha|$, i.e., λ_j is the eigenvalue closest to α
 - this can allow us to find *any* eigenvalue
 - to converge to λ_1 taking $\alpha = \|A\|_1$ may be a reasonable since $\rho(A) \leq \|A\|$
- useful to find eigenvector corresponding to a known approximate eigenvalue

Selecting shift dynamically

Rayleigh Quotient iteration

given $A \in \mathbb{R}^{n \times n}$ and nonzero normalized $v^{(0)} \in \mathbb{R}^n$; set $\lambda^{(0)} = v^{(0)T} A v^{(0)}$

for $k = 1, 2, \dots$

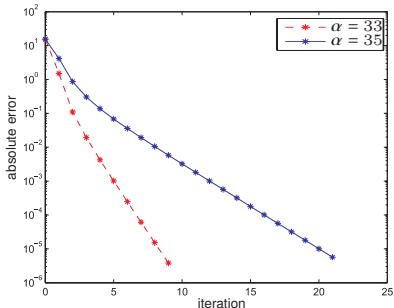
1. solve $(A - \lambda^{(k-1)} I) \tilde{v} = v^{(k-1)}$
 2. $v^{(k)} = \tilde{v} / \|\tilde{v}\|$
 3. $\lambda^{(k)} = v^{(k)T} A v^{(k)}$
-

- selecting $\alpha = \alpha^{(k)} = v^{(k)T} A v^{(k)}$ to be the Rayleigh quotient
- given an approximate eigenvector, Rayleigh quotient provides a good estimate for the corresponding eigenvalue
- improves convergence rate as we get closer to the desired eigenvalue
- requires refactoring $A - \lambda^{(k-1)} I$ each iteration for the solution step 1

Example

$$A = \text{diag}(1, 2, \dots, 32) \quad \text{and} \quad B = \text{diag}(1, 2, \dots, 30, 30, 32)$$

- we run the inverse iteration with random initialization for $\alpha = 33$ and $\alpha = 35$



- a shift closer to the dominant eigenvalue of 32 yields much faster convergence
- running the Rayleigh quotient iteration, things are even faster; a typical sequence of is $3.71\text{e-}1$, $9.46\text{e-}2$, $2.34\text{e-}4$, $2.16\text{e-}11$

Outline

- power iteration
- shift and inverse techniques
- **simultaneous (subspace) iteration**
- QR iteration

Subspace

a nonempty set \mathcal{S} of \mathbb{R}^n is a *subspace* of \mathbb{R}^n if for all $\alpha, \beta \in \mathbb{R}$ and $x, y \in \mathcal{S}$

$$\alpha x + \beta y \in \mathcal{S}$$

(closed under vector addition and scalar multiplication)

- all linear combination of elements of \mathcal{S} are in \mathcal{S}
- every subspace includes the zero vector 0
- geometrically, a subspace is a flat (plane) that passes through the origin

Examples

- $\{0\}$ and \mathbb{R}^n are subspaces
- for $m \in \mathbb{R}$, the line $\{(x, mx) \mid x \in \mathbb{R}\}$ is a subspace of \mathbb{R}^2
- $\text{span}(v_1, \dots, v_k)$ is a subspace where $v_i \in \mathbb{R}^n$
- $\mathbb{R}_+^2 = \{(x_1, x_2) \mid x_1, x_2 \geq 0\}$ is not a subspace
 - for instance, $(1, 1) \in \mathbb{R}_+^2$ but $-1(1, 1) \notin \mathbb{R}_+^2$

Basis

given a subspace \mathcal{S} , the set of vectors $\{v_1, v_2, \dots, v_k\} \in \mathcal{S}$ is a *basis for \mathcal{S}* if

1. the set $\{v_1, v_2, \dots, v_k\}$ is linearly independent
2. $\mathcal{S} = \text{span}(v_1, \dots, v_k)$

- every $x \in \mathcal{S}$ can be expressed uniquely as

$$x = \alpha_1 v_1 + \dots + \alpha_k v_k$$

for some coefficients $\alpha_1, \dots, \alpha_k$ called *coordinates* or *components*

- any set of n -linearly independent vectors $v_1, v_2, \dots, v_n \in \mathbb{R}^n$ is a *basis for \mathbb{R}^n*

Eigenspace

Invariant subspace: for $A \in \mathbb{R}^{n \times n}$ a subspace \mathcal{S} is an *invariant subspace* if

$$x \in \mathcal{S} \implies Ax \in \mathcal{S}$$

Eigenspace: for $A \in \mathbb{R}^{n \times n}$, the *eigenspace* is the set

$$\mathcal{S}_\lambda = \{v \mid Av = \lambda v\}$$

- an eigenspace is an invariant subspace of \mathbb{R}^n (or \mathbb{C}^n)
- if x_1, \dots, x_p are eigenvectors, then $\text{span}(x_1, \dots, x_p)$ is an invariant subspace

Simultaneous iteration

Simultaneous iteration

given $A \in \mathbb{R}^{n \times n}$ and $V_0 = [v_1^{(0)} \ \cdots \ v_p^{(0)}] \in \mathbb{R}^{n \times p}$ with lin. indep. columns

for $k = 1, 2, \dots$

$$V_k = AV_{k-1}$$

- power iteration with several starting points
- columns of V_k can be normalized to avoid blow up
- columns of $V_k = A^k V_0$ converge to a basis for $\text{span}(x_1, \dots, x_p)$ provided

$$|\lambda_1| > \cdots > |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n|$$

Convergence discussion

- assume A has lin. indep. eigenvectors such that $v_i^{(0)} = \sum_{j=1}^n \beta_{i,j} x_j$ for some $\beta_{i,j}$
- we have for $i = 1, 2, \dots, p$

$$\begin{aligned} v_i^{(k)} &= \lambda_1^k \beta_{i,1} x_1 + \dots + \lambda_n^k \beta_{i,n} x_n \\ &= \lambda_p^k \left(\sum_{j=1}^p (\lambda_j / \lambda_p)^k \beta_{i,j} x_j + \sum_{j=p+1}^n (\lambda_j / \lambda_p)^k \beta_{i,j} x_j \right) \end{aligned}$$

- columns of $V_k = A^k V_0$ converge to basis for $\text{span}(x_1, \dots, x_p)$ if $|\lambda_p| > |\lambda_{p+1}|$
- columns of V_k become increasingly ill-conditioned basis:
 - each column will be very close to v_1
 - so columns become almost linearly dependent

Outline

- power iteration
- shift and inverse techniques
- simultaneous (subspace) iteration
- **QR iteration**

Orthogonal iteration

Orthogonal (simultaneous) iteration

given $A \in \mathbb{R}^{n \times n}$ and an $n \times p$ matrix U_0 with orthonormal columns

for $k = 1, 2, \dots$

1. compute: $V_k = AU_{k-1}$
 2. QR factorization: $V_k = U_k R_k$
-

- V_k and U_k span the same space
- columns of U_k converge to an orthonormal basis for $\text{span}(x_1, \dots, x_p)$
- orthogonal transformation $U_k^T A U_k$ preserves eigenvalues of A
- for $U_0 = I$, $U_k^T A U_k$ converges to an upper triangular matrix with eigenvalues on diagonal if
 - the eigenvalues of A are all real and distinct
 - all the principal submatrices of A are nonsingular

Interpretation as QR factorization of powers of A

for $p = n$, we have

$$U_0 = I, \quad AU_{k-1} = U_k R_k \quad (\text{for } k \geq 1)$$

with U_k orthogonal, R_k upper triangular

- repeated substitution gives:

$$A = U_1 R_1, \quad A^2 = AU_1 R_1 = U_2 R_2 R_1, \quad A^3 = AU_2 R_2 R_1 = U_3 R_3 R_2 R_1, \quad \dots$$

- after k steps,

$$A^k = U_k S_k \quad \text{where } S_k = R_k R_{k-1} \cdots R_1$$

- the product $S_k = R_k R_{k-1} \cdots R_1$ is upper triangular

Reorganization of orthogonal iteration

- assume $p = n$ so that U_k is orthogonal and let $U_0 = I$
- from $AU_{k-1} = U_k R_k$, we have

$$U_k^T AU_{k-1} = R_k$$

and

$$A_k = U_{k-1}^T AU_{k-1} = U_{k-1}^T U_k R_k = Q_k R_k$$

last equation is QR factorization with $Q_k = U_{k-1}^T U_k$

- from this it follows that

$$A_{k+1} = U_k^T AU_k = (U_k^T AU_{k-1})(U_{k-1}^T U_k) = R_k Q_k$$

Relation to orthogonal iteration

$$A_0 = A, \quad A_k = Q_k R_k, \quad A_{k+1} = R_k Q_k \quad (\text{for } k \geq 0)$$

- the matrices A_{k+1} and A_k are orthogonally similar

$$A_{k+1} = R_k Q_k = Q_k^T A_k Q_k$$

- continuing recursively, we see that an orthogonal similarity relates A_k and A :

$$\begin{aligned} A_{k+1} &= (Q_1 Q_2 \cdots Q_k)^T A (Q_1 Q_2 \cdots Q_k) \\ &= U_k^T A U_k \quad \text{where } U_k = Q_1 Q_2 \cdots Q_k \end{aligned}$$

therefore the matrices A_k all have the same eigenvalues as A

- the orthogonal matrices $U_k = Q_1 Q_2 \cdots Q_k$ and the upper triangular R_k satisfy

$$A U_{k-1} = U_{k-1} A_k = U_{k-1} Q_k R_k = U_k R_k$$

hence, equivalence to orthogonal iteration is $U_k = Q_1 Q_2 \cdots Q_k$

QR algorithm

most popular method for finding all eigenvalues of a matrix

QR iteration

given $A \in \mathbb{R}^{n \times n}$; set $A_0 = A$

for $k = 0, 1, 2, \dots$

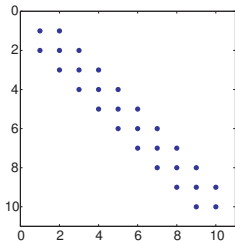
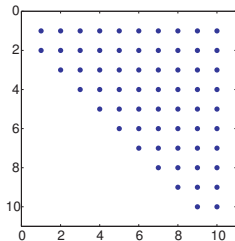
1. QR factorization: $A_k = Q_k R_k$
 2. compute: $A_{k+1} = R_k Q_k$
-

- A_k converges to upper (block) triangular matrix with eigenvalues in diagonal
- $U_k = Q_1 Q_2 \cdots Q_k$ converges to matrix of eigenvectors

Two stages

Stage 1: orthogonal similarity transformation

- idea: orthogonally transforming the matrix into upper Hessenberg form
- for symmetric A into tridiagonal matrix



Stage 2

- apply QR iteration
- computing the QR decomposition of an upper Hessenberg matrix requires order n^2 operations compared to the order n^3 for a full matrix

Example

$$A = \begin{bmatrix} .5 & -.1 & -.5 & .4 \\ -.1 & .3 & -.2 & -.3 \\ -.3 & -.2 & .6 & .3 \\ .1 & -.3 & .3 & 1 \end{bmatrix}$$

The first stage

- use Householder reflections to form $H^T A H$, which is upper Hessenberg
- denote elementary transformation matrices for the first two steps by H_1 and H_2
- $k = 1$, apply Householder reflection that zeros last two elements in first column: find a reflector u_1 such that $(-.1, -.3, .1)$ turns into $(\alpha, 0, 0)$

- the vector is $u_1 = (-.8067, -.5606, .1869)$, and we have an orthogonal 3×3 matrix of the form $P^{(1)} = I_3 - 2u_1u_1^T$; we then define

$$H_1^T = \begin{bmatrix} 1 & \\ & P^{(1)} \end{bmatrix}$$

- $H_1^T A$ now has two zeros in the (3, 1) and (4, 1) positions
- multiplying by H_1^T on the left does not touch the first row of A
- multiplying by H_1 on the right does not touch the first column of $H_1^T A$
- the first similarity transformation gives

$$H_1^T A H_1 = \begin{bmatrix} .5 & .6030 & -.0114 & .2371 \\ .3317 & .3909 & -.1203 & .0300 \\ 0 & -.1203 & .6669 & .5255 \\ 0 & .03 & .5255 & .8422 \end{bmatrix}$$

and by construction the eigenvalues of A are preserved

- similarly for $k = 2$; the reflector is $u_2 = (-.9925, .1221)$, and

$$H_2^T = \begin{bmatrix} I_2 & 0 \\ 0 & P^{(2)} \end{bmatrix}$$

where $P^{(2)} = I_2 - 2u_2u_2^T$

- we get an upper Hessenberg form, as desired, given by

$$H_2^T H_1^T A H_1 H_2 = \begin{bmatrix} .5 & .6030 & .0685 & .2273 \\ .3317 & .3909 & .1240 & 0 \\ 0 & .1240 & .4301 & -.4226 \\ 0 & 0 & -.4226 & 1.0790 \end{bmatrix}$$

- only $n - 1$ nonzeros out of $\frac{(n-1)n}{2}$ are left in the entire strictly lower left triangle

Matlab code

```
function A = houseeig(A)
%
% function A = houseeig(A)
%
% reduce A to upper Hessenberg form using Householder reflections
n = size(A,1);
for k = 1:n-2
z=A(k+1:n,k);
e1=[1; zeros(n-k-1,1)];
u=z+sign(z(1))*norm(z)*e1;
u = u/norm(u);
% multiply from left and from right by Q = eye(n-k)-2*u*u';
A(k+1:n,k:n) = A(k+1:n,k:n) - 2*u*(u'*A(k+1:n,k:n));
A(1:n,k+1:n) = A(1:n,k+1:n) - 2*(A(1:n,k+1:n)*u)*u';
end
```

QR algorithm with shifts

QR iteration with shifts

given A_0 transformed into upper Hessenberg form

for $k = 1, 2, \dots$

1. choose shift α_k
 2. QR factorization: $A_k - \alpha_k I = Q_k R_k$
 3. compute: $A_{k+1} = R_k Q_k + \alpha_k I$
-

- it often suffices to take α_k as a value along the diagonal
- common practice: use last diagonal entry of the matrix as a shift
- with properly chosen shifts, the iteration always converges

MATLAB implementation

```
function [lambda,itn] = qreig (A,tol)
% First stage, bring to upper Hessenberg form
A = houseeig(A);
% second stage: deflation loop
n = size(A,1); lambda = []; itn = [];
for j = n:-1:1
% find jth eigenvalue
[lambda(j),itn(j),A] = qrshift (A(1:j,1:j),tol);
end
function [lam,iter,A] = qrshift (A,tol)
m = size(A,1); lam = A(m,m); iter=0; I = eye(m);
if m == 1, return, end
while (iter < 100) % max number of iterations
if (abs(A(m,m-1)) < tol), return, end % check convergence
iter=iter+1;
[Q,R]=qr(A-lam*I); % compute the QR decomposition
A=R*Q+lam*I; % find the next iterate
lam = A(m,m); % next shift
end
```

Example

find eigenvalues λ and corresponding eigenfunctions $u(t)$ such that

$$u''(t) - u'(t) = \lambda u(t), \quad 0 < t < L, \quad u(0) = u(L) = 0$$

- we regard L as a parameter and seek nontrivial solutions
- eigenvalues for this differential problem are given by

$$\lambda_j^{\text{de}} = -\frac{1}{4} - \left(\frac{j\pi}{L}\right)^2, \quad j = 1, 2, \dots$$

- discretization: for small value h , we look for $\lambda, u = (u_1, u_2, \dots, u_{N-1})$:

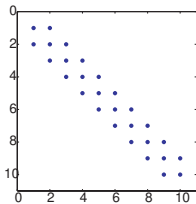
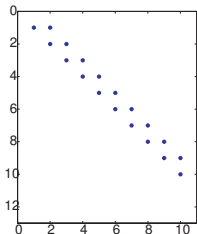
$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} - \frac{u_{i+1} - u_{i-1}}{2h} = \lambda u_i, \quad i = 1, 2, \dots, N-1,$$

setting $u_0 = u_N = 0$, for $N = L/h$

- writing this as $Au = \lambda u$, we have a nonsymmetric, potentially large tridiagonal matrix A of size $n = N - 1$ and hopefully real eigenvalues
- we have applied `qreig` to this problem for $L = 10$ using $h = .1$, *i.e.*, the matrix size is $n = 99$, with tolerance $1e-4$
- sorting the eigenvalues in descending order, the maximum absolute difference between the first six eigenvalues λ_j and their corresponding continuous comrades λ_j^{de} is $.015$; the discrepancy arises because of the discretization error
- applying `qreig` to $L = 80$ with $h = .1$ ($n = 799$) does not converge
- applying MATLAB function `eig` for $L = 80$ results in complex eigenvalues even though λ_j^{de} stay real
- the reason for the sudden appearance of complex eigenvalues has to do with ill-conditioning of the differential problem and is not our focus here

Computing the SVD

- before, we applied same orthogonal transformation on left and right (transposed)
- for SVD $A = U\Sigma V^T$, there is no need to perform the same operations
- we search for a procedure that would reduce A into bidiagonal form, using different orthogonal transformations on the left and on the right
- note that the eigenvalues of $A^T A$ are the squares of the singular values of A , and for the former we have a technique of reducing the matrix into tridiagonal form



the result of the first stage of the computation of the SVD is a bi-diagonal matrix C (left); the corresponding tridiagonal matrix $C^T C$ is given on the right

Reduction to bidiagonal

- to illustrate the idea, suppose the nonzero structure of a 5×4 matrix A is given by

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

- applying U_1^T on the left using Householder transformations, we have

$$U_1^T A = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

- now, we can apply a different orthogonal transformation on the right, but since we do not want to touch the first column, we settle for zeroing out the entries to the right of the (1, 2) element, namely,

$$U_1^T A V_1 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

- another step gives

$$U_2^T U_1^T A V_1 V_2 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

- this continues until we get a bidiagonal form
- traditional methods involve an adaptation of the QR to zero out off-diagonal entries

References and further readings

- U. M. Ascher. *A First Course on Numerical Methods*. Society for Industrial and Applied Mathematics, 2011.
- M. T. Heath. *Scientific Computing: An Introductory Survey* (revised second edition). Society for Industrial and Applied Mathematics, 2018.
- L. Vandenberghe. *EE133B lecture notes*, Univ. of California, Los Angeles. (<https://www.seas.ucla.edu/~vandenbe/ece133b.html>)