Thursday, 10/17/13 Math 471, Fall 2013, Section 001 Topic 5

1 Computing eigenvalues

Question 1: Given a matrix A, can we find λ and $x \neq 0$ such that $Ax = \lambda x$?

λ : e-value (e.g. frequency, growth rate, energy level)
x : e-vector (e.g. normal mode, principal component, bound state)

Definition 1. An orthonormal basis is a set of vectors q_i, $i = 1, \ldots, n$ such that $q_i^T q_j = 0$ for $i \neq j$, $\|q_i\|_2 = 1$ for all $i \in [1, n]$ and any vector x can be written as a linear combination of q_i,

$$x = \alpha_1 q_1 + \alpha_2 q_2 + \cdots \alpha_n q_n,$$

where the scalars α_i are the coordinates of x with respect to the basis $\{q_i\}_{i=1}^n$.

Theorem 2. *If an $n \times n$ matrix A is real and symmetric, then the eigenvalues λ_i, $i = 1, \ldots, n$ are real and the eigenvectors form an orthonormal basis of $\mathbb{R}^n$.*

$\Rightarrow$ *Real and symmetric matrices are orthogonally diagonalizable; there exists and orthogonal matrix Q such that $AQ = QD$, where D is a diagonal matrix whose nonzero elements are the eigenvalues of A. The columns of Q are the eigenvectors of A.*

Notes:

- Diagonalization separates coupled linear systems into a collection of independent scalar equations
- Matrix-vector multiplication becomes scalar multiplication for eigenvectors: $Ax = \lambda x$
- Suppose mathematical model yields a matrix; the eigenvalues determine the stability of the system's equilibria
- Eigenvalues are particularly useful for analyzing the long-time behavior of systems governed by linear equations

Example 1: $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$

$$f_A(\lambda) = \det \begin{pmatrix} 2-\lambda & -1 \\ -1 & 2-\lambda \end{pmatrix} = (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = (\lambda - 3)(\lambda - 1)$$

$$\lambda_1 = 3 : Ax = 3x \Rightarrow \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 3 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

choose $x_1 = 1$, then $2 - x_2 = 3, \quad -1 + 2x_2 = 3x_2 \Rightarrow x_2 = -1$

$$q_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \Rightarrow \|q_1\|_2 = 1$$

$$\lambda_2 = 1 : Ax = x \Rightarrow \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

choose $x_1 = 1$, then $2 - x_2 = 1, \quad -1 + 2x_2 = x_2 \Rightarrow x_2 = 1$

$$q_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \Rightarrow \|q_2\|_2 = 1$$

$q_1^T q_2 = 0 \Rightarrow \{q_1, q_2\}$ are an orthonormal basis of $\mathbb{R}^2$.

Define $Q = [\ q_1 \ \ q_2\ ]$, the matrix with vectors q as its columns, $Q = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\-1 & 1\end{pmatrix}$

$$AQ = \frac{1}{\sqrt{2}}\begin{pmatrix}2 & -1\\-1 & 2\end{pmatrix}\begin{pmatrix}1 & 1\\-1 & 1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}3 & 1\\-3 & 1\end{pmatrix}$$

Define $D = \mathrm{diag}(\lambda_1, \lambda_2)$, the matrix with eigenvalues on its main diagonal, $D = \begin{pmatrix}3 & 0\\0 & 1\end{pmatrix}$

$$QD = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\-1 & 1\end{pmatrix}\begin{pmatrix}3 & 0\\0 & 1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}3 & 1\\-3 & 1\end{pmatrix}$$

The equation $D = Q^T A Q$ is called the eigenvalue decomposition of A, and the symbol Λ is sometimes used instead of D to convey that it is the diagonal matrix of eigenvalues λ.

△

1.1 Obvious method for computing eigenvalues

Use the characteristic polynomial and the fact that $\det(A - \lambda I) = 0$.

1. Find $f_A(\lambda) = \det(A - \lambda I)$
2. Solve $f_A(\lambda) = 0$ using a rootfinding algorithm from chapter 2 (apart from 2×2 matrices, whose characteristic polynomials are quadratic, there are no easy formulas for the general roots of polynomials).

Example 2: $A = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$, $\tilde{A} = \begin{pmatrix}1+\epsilon & 0\\0 & 1-\epsilon\end{pmatrix}$: perturbed matrix

$f_A(\lambda) = (1-\lambda)^2 \Rightarrow \lambda_1 = \lambda_2 = 1$

$f_{\tilde{A}}(\lambda) = (1+\epsilon-\lambda)(1-\epsilon-\lambda) = \lambda^2 - 2\lambda + 1 - \epsilon^2 \Rightarrow \lambda_{1,2} = 1 \pm \epsilon$

1. A change in the matrix A of size ϵ results in a change in λ of size ϵ.
2. A change in the coefficients of the characteristic polynomial $f_A(\lambda)$ of size ϵ^2 results in a change in λ of size ϵ.
3. Hence, the roots of $f_A(\lambda)$ exhibit sensitive dependents on the coefficients, and this implies that this obvious method for finding eigenvalues is unstable. Even if double precision arithmetic is used, where $\epsilon_{\text{machine}} \approx 10^{-16}$, you can have errors in the eigenvalues of $e = O(10^{-8})$. You just lost 8 digits of accuracy!

△

Example 3: (Wilkinson) Consider the diagonal 20×20 matrix $A = \mathrm{diag}(1, 2, \ldots, 20)$. Since the matrix is diagonal, we know the eigenvalues are simply the integers $\lambda_p = p$, for $p = 1, 2, \ldots, 20$.

Let's examine what happens if we try to compute them using the characteristic polynomial in Matlab.

The characteristic polynomial is $f_A(\lambda) = (1-\lambda)(2-\lambda)\cdots(20-\lambda) = \sum_{j=0}^{20} a_j \lambda^j$.

The Matlab command `poly(1:20)` will expand the factored form into the expanded form and give us the coefficients a_j.

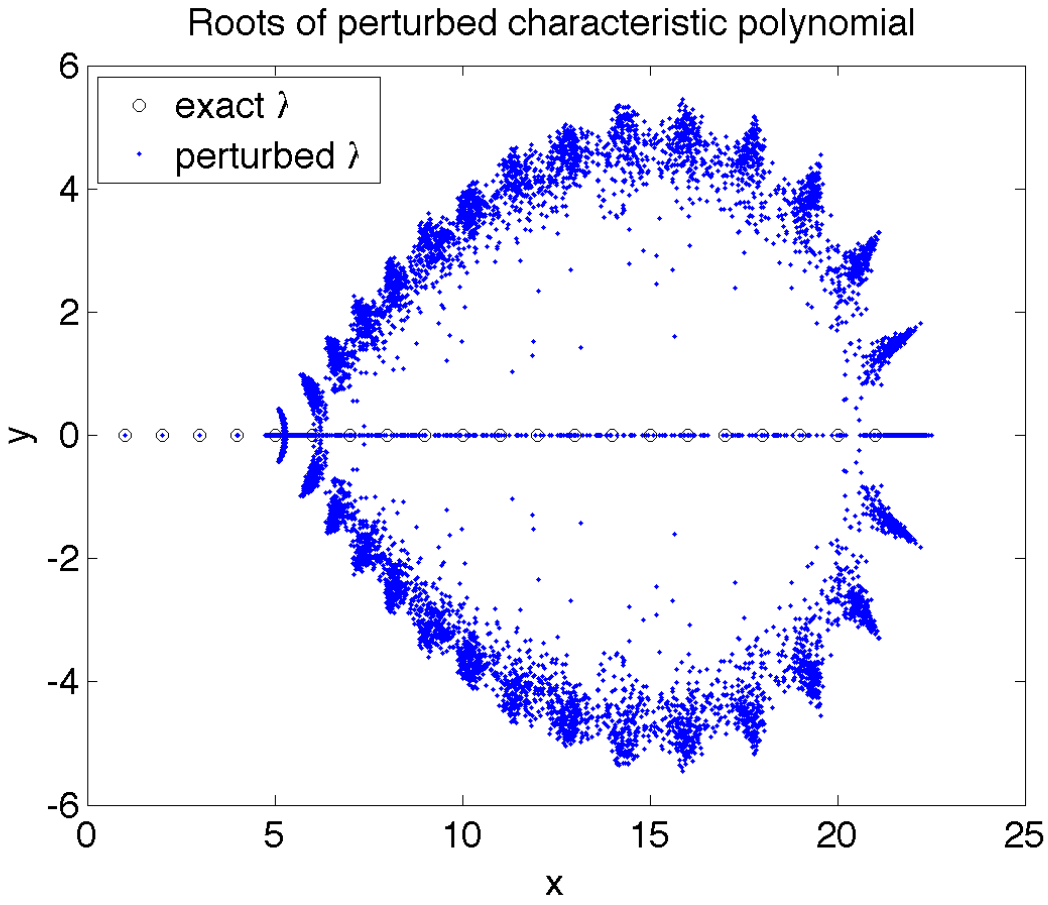
Suppose the coefficients a_j are perturbed by a small, say $O(10^{-10})$ random amount... The command `randn(1,21)` will generate a vector of 21 numbers randomly selected from the normal distribution. Therefore, `1e-10*rand(1,21)` will be 21 normally distributed random numbers, each of size $O(10^{-10})$, and the perturbed coefficients $\tilde{a}_j = a_j(1 + 10^{-10}\epsilon_j)$ can be generated by `poly(1:20).*(ones(1,21) + 1e-10*randn(1,21))`.

We then find the roots of the (slightly) perturbed characteristic polynomial...

```
clear
%% Wilkinson's example

figure(1);clf;
% plot the exact eigenvalues
plot(1:21,zeros(1,21),'ko');hold on;
set(gca,'FontSize',18);
xlabel('x');
ylabel('y');
% plot the roots of the perturbed characteristic polynomial
for i=1:500
    r = roots(poly(1:20).*(ones(1,21) + 1e-10*randn(1,21)));
    plot(real(r),imag(r),'b.');
end
hold off
legend('exact \lambda','perturbed \lambda','Location','NorthWest');
title('Roots of perturbed characteristic polynomial');

saveas(1,'wilkinson.png');
```



We see that perturbations on the order of 10^{-10} to the polynomial's coefficients lead to errors of $e = O(1)$ for some of the eigenvalues. Thus, the roots of the characteristic polynomial are sensitive to perturbations in the coefficients; this is an example of instability, and this algorithm should not be used for computing eigenvalues, in general.

△

Definition 3. The Rayleigh quotient associated with a matrix A and vector $x \neq 0$ is

$$R_A(x) = \frac{x^T A x}{x^T x}.$$

Notes:

- Suppose A is symmetric and λ_i is the eigenvalue corresponding to the orthonormal basis vector $x = q_i$. Then $R_A(q_i) = \dfrac{q_i^T A q_i}{q_i^T q_i} = \dfrac{q_i^T \lambda_i q_i}{q_i^T q_i} = \lambda_i$.
- For $x \approx q_i$, the Rayleigh quotient $R_A(x)$ approximate λ_i. So if we can approximate eigen*vectors*, we can approximate eigen*values*. We begin be deriving an error estimate using Taylor series.

Recall: multivariate Taylor series expansions

$$f(x_1, x_2) = f(a_1, a_2) + \frac{\partial f}{\partial x_1}(a_1, a_2)(x_1 - a_1) + \frac{\partial f}{\partial x_2}(x_2 - a_2) + \cdots$$

$$f(x) = f(a) + \nabla f(a)(x-a) + O(\|x-a\|^2), \qquad \nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^T$$

$$\Rightarrow R_A(x) = R_A(q_i) + \nabla R_A(q_i) \cdot (x - q_i) + O(\|x - q_i\|^2)$$

$$\nabla R_A(x) = \nabla\left(\frac{x^T A x}{x^T x}\right) = \frac{x^T x \cdot \nabla(x^T A x) - x^T A x \cdot \nabla(x^T x)}{(x^T x)^2}$$

2×2 case:

$$\nabla(x^T x) = \nabla(x_1^2 + x_2^2) = [2x_1, 2x_2]^T = 2x^T$$

$$x^T A x = (x_1, x_2)\begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = a_{11}x_1^2 + 2a_{12}x_1x_2 + a_{22}x_2^2$$

$$\nabla(x^T A x) = [2a_{11}x_1 + 2a_{12}x_2, 2a_{12}x_1 + 2a_{22}x_2]^T = 2(Ax)^T$$

$$\nabla R_A(x) = \frac{x^T x \cdot 2(Ax)^T - x^T A x \cdot 2x^T}{(x^T x)^2} = \frac{2}{x^T x}\left((Ax)^T - R_A(x)x^T\right)$$

$$\nabla R_A(q_i) = \frac{2}{q_i^T q_i}\left((Aq_i)^T - R_A(q_i)q_i^T\right) = 2\left(\lambda_i q_i^T - \lambda_i q_i^T\right) = 0$$

$\Rightarrow R_A(x) = \lambda_i + O(\|x - q_i\|^2)$: quadratic approximation

1.2 Power method

Text : section 4.1

Tuesday, 10/22/13

You may recall from linear algebra that eigenvalues are related to matrix exponentials...
Idea : Take an initial guess for an eigenvector, x, and keep reapplying the matrix A...

$$x,\ Ax,\ A^2x,\ A^3x,\ \ldots$$

Question 2: Where does this sequence of vectors lead?

Suppose the symmetric matrix A has n linearly independent eigenvectors and its largest magnitude eigenvalue is unique; i.e., we can write the eigenvalues of A in order of descending magnitude as

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|\,.$$

We write x in terms of the vectors of the orthogonal eigenvector basis (even though we don't know what they are yet)

$$x = \alpha_1 q_1 + \alpha_2 q_2 + \cdots + \alpha_n q_n$$

Then the sequence of approximations we generate are

$$\begin{aligned} x_1 &= A(\alpha_1 q_1 + \alpha_2 q_2 + \cdots + \alpha_n q_n) = \alpha_1\lambda_1 q_1 + \alpha_2\lambda_2 q_2 + \cdots + \alpha_n\lambda_n q_n \\ x_2 &= A\left(\alpha_1\lambda_1 q_1 + \alpha_2\lambda_2 q_2 + \cdots + \alpha_n\lambda_n q_n\right) = \alpha_1\lambda_1^2 q_1 + \alpha_2\lambda_2^2 q_2 + \cdots + \alpha_n\lambda_n^2 q_n \\ &\vdots \qquad\qquad\qquad \vdots \\ x_k &= \alpha_1\lambda_1^k q_1 + \alpha_2\lambda_2^k q_2 + \cdots + \alpha_n\lambda_n^k q_n \end{aligned}$$

$$\Rightarrow x_k = \sum_{j=1}^{n} \lambda_j^k \alpha_j q_j = \lambda_1^k \left(\alpha_1 q_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^k \alpha_j q_j \right)$$

Since $\frac{|\lambda_j|}{|\lambda_1|} < 1$ for $j \geq 2$, the vectors x_k approach q_1 as $k \to \infty$.

Theorem 4. *Assume that* $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$ *and* $q_1^T x_0 \neq 0$. *Then* $\|x_k - (\pm q_1)\| = O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right)$ *and* $|\lambda_k - \lambda_1| = O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^{2k} \right)$.

The $\pm$ depends on the sign of λ_1. If $q_1^T x_0 = 0$, the scheme converges to λ_2, $\pm q_2$.

Algorithm 1: Power method

Input : Symmetric matrix A, initial guess for eigenvector $q_1 = x_0$ with $\|x_0\|_2 = 1$.
Output: scalar λ_k and vector x_k such that $\lambda_k \approx \lambda_1$ and $x_k \approx q_1$.

1 **for** $k = 1, 2, \ldots$ **do**
2 $\quad y = Ax_{k-1}$ `% if A is sparse, this can be done efficiently`
3 $\quad x_k = y / \|y\|_2$ `% this is done to avoid overflow/underflow`
4 $\quad \lambda_k = x_k^T A x_k$
5 **end**

Notes: Suppose that $A = A_h$ where $(A_h x)_i = -D_+ D_- x_i = \frac{1}{h^2}(-x_{i-1} + 2x_i - x_{i+1})$, assuming $h = 1/(n+1)$, $x_0 = x_{n+1} = 0$. Then line 2 in the algorithm, $y = Ax$ can be coded as a small loop (not an n-sized loop).

```
for i = 1:n
    newx(i) = (-x(i-1) + 2x(i) - x(i+1))/h^2;
end
x = newx;
```

This is more efficient than constructing the (mostly zero) matrix A_h and computing $y = Ax$ by matrix-vector multiplication. This is also relevant for computing project 2, where $x_{n+1} = Bx_n + c$.

The power method has some limitations.

- It only gives the largest e-value, λ_1.
- Convergence toward the e-vector is only linear, and the convergence factor $\left| \frac{\lambda_2}{\lambda_1} \right|$ may not be small.

  Recall: Linear convergence means $\|x_k - (\pm q_1)\| \leq C \|x_{k-1} - (\pm q_1)\|$.

Question 3: The power method gives us a way to find the largest eigenvalue of a matrix A. Can we modify the method to find the other eigenvalues?

The largest eigenvalue of A^{-1} is λ_n^{-1}. The power method may be applied to A^{-1} to find the smallest eigenvalue of A, provided that $|\lambda_n| < |\lambda_{n-1}| \leq \cdots \leq |\lambda_1|$. In this case the vectors x_k converge to $\pm q_n$ and $1/(x_k^T A x_k)$ converges to λ_n.

1.3 Inverse iteration

Text : section 4.2

We have seen that the power method applied to A and A^{-1} to give estimates of the largest and smallest eigenvalues of A. Now we modify the algorithm to find the other eigenvalues of A.

Idea: Apply the power method to the matrix $(A-\mu I)^{-1}$, μ : shift, $\mu \neq \lambda_i$, $1 \leq i \leq n$

Let q_i be an eigenvector of A with eigenvalue λ_i. Then $1/\lambda_i$ is an eigenvalue of A^{-1} with eigenvector q_i (hw).

$$(A-\mu I)q_i = Aq_i - \mu q_i = (\lambda_i - \mu)q_i \Rightarrow (A-\mu I)^{-1}q_i = \frac{1}{\lambda_i - \mu}q_i$$

$\Rightarrow q_i$ is also an eigenvector of $A-\mu I$ with eigenvalue $\lambda_i - \mu$.

And q_i is an eigenvector of $(A-\mu I)^{-1}$ with eigenvalue $(\lambda_i - \mu)^{-1}$.

The largest eigenvalue of $(A-\mu I)^{-1}$ is $|\lambda_J - \mu|^{-1}$ where λ_J is the eigenvalue of A closest to μ. Thus the vectors $v^{(k)}$ converge to $\pm q_J$.

Algorithm 2: Inverse Power method

Input : Matrix A, shift μ, initial guess for eigenvector $q_J = x_0$ with $\|x_0\|_2 = 1$.
Output: scalar λ_k and vector x_k such that $\lambda_k \approx \lambda_J$ and $x_k \approx q_J$.

1 **for** $k = 1, 2, \ldots$ **do**
2 | solve $(A-\mu I)w = x_{k-1}$ `% e.g. L U factorization`
3 | $x_k = w/\|w\|_2$
4 | $\lambda_k = x_k^T A x_k$
5 **end**

Thursday, 10/24/13

Theorem 5. *Assume that λ_J is the eigenvalue of A closest to μ and that λ_K is the next closest, i.e.,*

$$|\lambda_J - \mu| < |\lambda_K - \mu| < |\lambda_i - \mu| \text{ for } i \neq K, J \text{ and } q_J^T v^{(0)} \neq 0.$$

Then $\left\| v^{(k)} - (\pm q_J) \right\| = O\left(\left| \frac{\lambda_J - \mu}{\lambda_K - \mu} \right|^k \right)$ *and* $\left| \lambda_J^{(k)} - \mu \right| = O\left(\left| \frac{\lambda_J - \mu}{\lambda_K - \mu} \right|^{2k} \right)$

Proof. By assumption, the eigenvalues of $(A-\mu I)^{-1}$ may be written in order as $\lambda_1 \to \frac{1}{\lambda_J - \mu}$, $\lambda_2 \to \frac{1}{\lambda_K - \mu}, \lambda_2, \ldots, \lambda_n$.

As before, the power method converges with coefficient $\left| \frac{\lambda_2}{\lambda_1} \right| \to \left| \frac{\lambda_J - \mu}{\lambda_K - \mu} \right|$. □

Notes:

- Using a suitable shift μ, any eigenvalue of A can be found and the convergence factor $\left| \frac{\lambda_J - \mu}{\lambda_K - \mu} \right|$ can be made arbitrarily small.

- Both the power method and inverse iteration converge linearly toward eigenvectors
- Unlike the power method, in the inverse power method we can choose which eigenvector we want by choosing a suitable shift μ.
- Line 2 of the algorithm can be accelerated by computing L and U (the $O(n^3)$ cost) outside of the loop (only forward and backward substitution, both $O(n^2)$, are needed within the loop since the matrix does not change).

Question 4: Now we have a method for estimating eigenvalues, given an estimated eigenvector (the Rayleigh quotient), and we have a method for estimating eigenvectors, given an estimated eigenvalue (the inverse power method). Can we combine these two ideas to accelerate convergence?

Idea: Update the matrix $(A - \mu I)^{-1}$ with the most recent estimates of λ_J at each step

Algorithm 3: Rayleigh Quotient Iteration

Input : Symmetric Matrix A, initial guess for eigenvector $q_J = x_0$ with $\|x_0\|_2 = 1$, corresponding Rayleigh quotient λ_0.

Output: scalar λ_k and vector x_k such that $\lambda_k \approx \lambda_J$ and $x_k \approx q_J$.

1 **for** $k = 1, 2, \ldots$ **do**
2 | solve $(A - \lambda_{k-1} I)w = x_{k-1}$
3 | $x_k = w / \|w\|_2$
4 | $\lambda_k = x_k^T A x_k$
5 **end**

Theorem 6. *Suppose x_0 is sufficiently close to an eigenvector q_J. Then the Rayeigh Quotient Iteration converges to an eigenvalue/eigenvector pair cubically, i.e.,*

$$\|x_k - (\pm q_J)\| = O(\left\|x_{k-1} - (\pm q_J)^3\right\|), \qquad |\lambda_k - \lambda_J| = O(|\lambda_{k-1} - \lambda_J|^3).$$

Proof. Omitted.

Example 4: Let $A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 4 \end{pmatrix}$. Find the largest eigenvalue using the power method, the shifted inverse method, and Rayleigh quotient iteration.

First, let's take a look at Matlab's results:

```
format long
a = [2 1 1; 1 3 1; 1 1 4];
lam = eigs(A);
lam(1) = 5.214319743377535
```

Now let's see how long it takes us to duplicate that answer, starting with the vector $x_0 = \frac{1}{\sqrt{3}}(1, 1, 1)^T$ as our initial guess.

k	power method	inverse iteration, $\mu = 5$	Rayleigh quotient iteration
0	5.0	5.0	5.0
1	5.181818	5.21314	5.213114
2	5.208192	5.21431267	5.21439743184
correct digits	2	6	10

△

1.4 Other techniques

Question 5: In this section we have considered only symmetric matrices; what about a the eigenvalues of a general $n \times n$ matrix?

Theorem 7 (Gerschgorin Circle Theorem)**.** *Suppose an $n \times n$ matrix A has n eigenvalues, λ_i, $i = 1, \ldots, n$. Then each eigenvalue λ_i lies in the union of circles*

$$|z - a_{ii}| \leq r_i, \quad r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| .$$

- The ith circle is centered at the ith diagonal element of A in the complex plane
- The radius of each circle is the sum of the magnitudes of the off-diagonal elements of each row

Proof. Let λ and $x \neq 0$ be an e-value, e-vector pair. Then $Ax = \lambda x \Rightarrow (A - \lambda I)x = 0$ or,

$$(a_{ii} - \lambda)x_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}x_j = 0 \Rightarrow (\lambda - a_{ii})x_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}x_j, \qquad i = 1, 2, \ldots, n.$$

We take the norm of the above equation and apply the triangle inequality,

$$|\lambda - a_{ii}| \, |x_i| = \left| \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}x_j \right| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \, |x_j| .$$

Choose i so that $\|x\|_\infty = |x_i|$; since $x \neq 0$, $|x_i| = \|x\|_\infty > 0$ and we can divide by $|x_i|$.

$$|\lambda - a_{ii}| \leq \frac{1}{|x_i|} \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \, |x_j|$$

Due to our choice of i, $\dfrac{|x_j|}{|x_i|} = \dfrac{x_j}{\|x\|_\infty} \leq 1$ for $j \neq i$,

$$\Rightarrow |\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_{ij}| \, |x_j|}{|x_i|} \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| = r_i$$

□

Notes

- In the above proof, we never said that λ was the ith eigenvalue (in fact, we cannot make that assertion). The choice of i served to separate the entries of A from the entries of x.
- Corollary: If one circle lies apart from all other circles, then that circle contains exactly one eigenvalue.
- Corollary: If one circle connects with another circle, it may contain zero, one, or two eigenvalues.
- Since the eigenvalues of A are the same as the eigenvalues of A^T, we can improve our estimates by creating the intersection of the Gerschgorin circles of A and A^T.

Example 5: Use the Gerschgorin circle theorem to estimate the eigenvalues of the following matrices:

$$A = \begin{pmatrix} 5 & 0.6 & 0.1 \\ -1 & 6 & -0.1 \\ 1 & 0 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 5 & 1 & 1 \\ 0 & 6 & 1 \\ 1 & 0 & -5 \end{pmatrix}, \quad C = \begin{pmatrix} 8 & 1 & 0 \\ 1 & 4 & \epsilon \\ 0 & \epsilon & 1 \end{pmatrix}$$

These 3×3 matrices have 3 circles, or disks, each; plus another 3 for the transpose :

$A : D_1 = \{z \in \mathbb{C} : |z - 5| \leq 0.7\}, \quad D_2 = \{z \in \mathbb{C} : |z - 6| \leq 1.1\} \quad D_3 = \{z \in \mathbb{C} : |z - 2| \leq 1\}$

$A^T : D_4 = \{z \in \mathbb{C} : |z - 5| \leq 2\}, \quad D_5 = \{z \in \mathbb{C} : |z - 6| \leq 0.6\} \quad D_6 = \{z \in \mathbb{C} : |z - 2| \leq 0.2\}$

```
clear;
%% Gerschgorin circles

A = [ 5 0.6 0.1; -1 6 -0.1; 1 0 2];
lam = eig(A);

figure(1);clf;hold on;axis equal;set(gca,'FontSize',18);
theta = 0:0.001:2*pi;
x = 0.7*cos(theta) + 5;
y = 0.7*sin(theta);
d1 = fill(x,y,'b');
set(d1,'FaceAlpha',0.2)

x = 2*cos(theta) + 5;
y = 2*sin(theta);
d4 = fill(x,y,'r');
set(d4,'FaceAlpha',0.2);

plot(lam,'k.','MarkerSize',24);

x = 1.1*cos(theta) + 6;
y = 1.1*sin(theta);
d2 = fill(x,y,'b');
set(d2,'FaceAlpha',0.2);

x = cos(theta) + 2;
y = sin(theta);
```

```
d3 = fill(x,y,'b');
set(d3,'FaceAlpha',0.2);



x = 0.6*cos(theta) + 6;
y = 0.6*sin(theta);
d5 = fill(x,y,'r');
set(d5,'FaceAlpha',0.2);

x = 0.2*cos(theta) + 2;
y = 0.2*sin(theta);
d6 = fill(x,y,'r');
set(d6,'FaceAlpha',0.2);

title('Gerschgorin circles');
legend('A','A^T','\lambda')
```

△

Question 6: What about defective matrices (matrices without a complete eigenbasis)?

Defective matrices rarely come up in applications connected to physical models.

Definition 8. A Schur factorization of a matrix A is a factorization

$$A = QTQ^*$$

where Q is unitary (orthogonal) and T is upper-triangular.
Note: This factorization says that A is similar to an upper-triangular matrix T, whose eigenvalues lie along its diagonal. Since similar matrices have the same eigenvalues, the eigenvalues of A also lie along the diagonal of T.

Theorem 9 (Schur's theorem)**.** *Every square matrix A has a Schur factorization.*

⇒ We can approximate the eigenvalues of A by iteratively finding approximations of the Schur factorization. This may be ill-conditioned for defective matrices; rather than proceed, it's often better to check the model that is associated with the matrix to see if it can be reformulated in a way that produces a nondefective matrix.

Final note: Matlab uses much more sophisticated algorithms than these to compute eigenvalues via `eig` and `eigs`. Some of these (i.e., Lanczos iteration, Arnoldi iteration, and QR algorithm) are covered in Math 571.