Numerical Methods - Numerical Linear Algebra

Y. K. Goh

Universiti Tunku Abdul Rahman

2013

Outline

Problem setting

Two major problems:

1. Solving linear system $\mathbf{Ax} = \mathbf{b}$
   - Direct methods (preferred method)
     - Gaussian Elimination, matrix factorisation
   - Iterative methods (normally for sparse matrix) - Jacobi, Gauss-Seidel
     - Jacobi, Gauss-Seidel
2. Eigenvalue problem $\mathbf{Av} = \lambda\mathbf{v}$
   - Deflation methods
   - Similarity transform methods

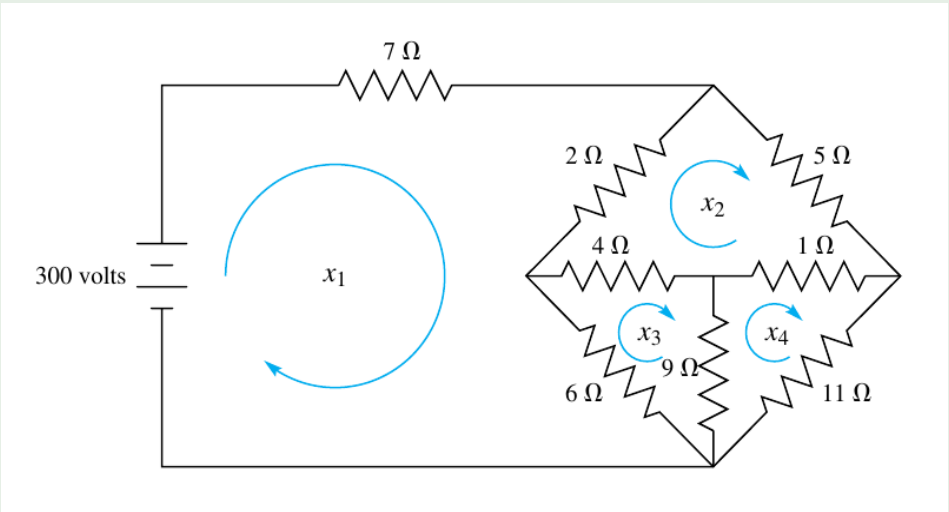
Outline

Engineering Example: Kirchhoff's Law

Example

The currents at different parts of an electrical network with many resistances and a battery can be studied with Kirchhoff's Law. Usually, the results of Kirchhoff's Law are several linear equations. Let denotes x_1, x_2, x_3 and x_4 are currents at different loops, then the equations are

$$\begin{aligned} 15x_1 - 2x_2 - 6x_3 &= 300 \\ -2x_1 + 12x_2 - 4x_3 - x_4 &= 0 \\ -6x_1 - 4x_2 + 19x_3 + 9x_4 &= 0 \\ -x_2 - 9x_3 + 21x_4 &= 0 \end{aligned}$$



Outline

Naive Gaussian Elimination

Normally the linear system is written as augmented matrix $[\mathbf{A}|\mathbf{b}]$. Naive Gaussian Elimination is a two step process:

- systematically applying row operations to get echelon form

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} \times & \times & \times & \mathrm{b} \\ 0 & \times & \times & \mathrm{b} \\ 0 & \times & \times & \mathrm{b} \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} \times & \times & \times & \mathrm{b} \\ 0 & \times & \times & \mathrm{b} \\ 0 & 0 & \times & \mathrm{b} \end{array}\right]$$

- backward substitution to obtain $\mathbf{x}$.

$$\begin{aligned} x_3 &= b_3/a_{33} \\ x_2 &= (b_2 - a_{23}x_3)/a_{22} \\ x_1 &= (b_1 - a_{13}x_3 - a_{12}x_2)/a_{11} \end{aligned}$$

Naive Gaussian Elimination Algorithm

Example

In the previous circuit example, we get the matrix equation:

$$\left[\begin{array}{cccc|c} 15 & 02 & 06 & 0 & 300 \\ -2 & 12 & -4 & -1 & 0 \\ -6 & -4 & 19 & 9 & 0 \\ 0 & -1 & -9 & 21 & 0 \end{array}\right]$$

Answer: [`MATLAB code: nm03_naive_gaussian.m`]

- systematically applying row operations to get echelon form

$$\left[\begin{array}{cccc|c} 15 & -2 & -6 & 0 & 300 \\ -2 & 12 & -4 & -1 & 0 \\ -6 & -4 & 19 & 9 & 0 \\ 0 & -1 & -9 & 21 & 0 \end{array}\right] \rightarrow \left[\begin{array}{cccc|c} 15 & -2 & -6 & 0 & 300 \\ 0 & 11.73 & -4 & -1 & 40 \\ 0 & 0 & 14.64 & 9 & 136.36 \\ 0 & 0 & 0 & 26.43 & 91.07 \end{array}\right]$$

- backward substitution $x_4 = 3.44, x_3 = 7.29, x_2 = 6.69$ and $x_1 = 23.91$.

Caveat

In the followings, we assume that all systems that we are dealing are having solutions. We exclude the cases where the systems are inconsistent or having infinite solutions. Consider the following two systems

Example

$$\begin{aligned} x_1 + x_2 + x_3 &= 4 \\ 2x_1 + 2x_2 + x_3 &= 4 \\ x_1 + x_2 + 2x_3 &= 6 \\ &\text{inconsistent} \end{aligned} \qquad \begin{aligned} x_1 + x_2 + x_3 &= 4 \\ 2x_1 + 2x_2 + x_3 &= 6 \\ x_1 + x_2 + 2x_3 &= 6 \\ &\text{infinite solutions} \end{aligned}$$

when their row-reduced matrices

$$\left[\begin{array}{ccc|cc} 1 & 1 & 1 & 4 & 7 \\ 2 & 2 & 1 & 4 & 6 \\ 1 & 1 & 2 & 6 & 6 \end{array}\right] \rightarrow \left[\begin{array}{ccc|cc} 1 & 1 & 1 & 4 & 7 \\ 0 & 0 & -1 & -4 & -2 \\ 0 & 0 & 1 & 2 & 2 \end{array}\right]$$

Naive Gaussian Elimination can fail in certain cases

Example (Zero pivot element)

$$\begin{aligned} 0x_1 + x_2 &= 1 \\ x_1 + x_2 &= 2 \end{aligned}$$

Remedy: pivoting / swapping rows.

Example (Ill-conditioned)

$$\begin{aligned} \epsilon x_1 + x_2 &= 1 \\ x_1 + x_2 &= 2 \end{aligned}$$

when ϵ is small, solutions

$$x_2 = \frac{2 - \epsilon^{-1}}{1 - \epsilon^{-1}} \approx 1, \quad x_1 = \epsilon^{-1}(1 - x_2) \approx 0$$

becomes unreliable. Remedy: Pivoting.

Partial Pivoting

- Partial Pivoting - exchange pivot row with the row with maximum leading element.
- Apply the partial pivoting when the pivoting element is zero or smaller than the leading element in the candidate row.

Example

Swap the equations in the previous slide:

$$\begin{aligned} x_1 + x_2 &= 2 \\ \epsilon x_1 + x_2 &= 1 \end{aligned}$$

Answer:

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 1-\epsilon & 1-2\epsilon \end{array}\right] \implies x_2 = \frac{1-2\epsilon}{1-\epsilon} \approx 1, \quad x_1 = \frac{1+\epsilon}{1-\epsilon} \approx 1.$$

Scaled Partial Pivoting

- It is used when the leading element in the pivoting row is much smaller than other element in the same row.

1. Define a scale factor $s_i = \max\limits_{1 \le j \le n} |a_{ij}|$.
2. Exchange pivoting row with row p where

$$\frac{|a_{p1}|}{s_p} = \max_{1 \le k \le n} \frac{|a_{k1}|}{s_k}.$$

Example (4-digit floating-point arithmetic)

$$\begin{aligned} 30.00x_1 + 591400x_2 &= 591700 \\ 5.291x_1 + 6.130x_2 &= 46.78 \end{aligned}$$

will gives the wrong answer $x_2 = 1.001$ and $x_1 = -10.00$ in a 4-digit floating-point operations.
However, with scaled partial pivoting, $a_{11}/s_1 = 0.5073 \times 10^{-4}$ and $a_{21}/s_2 = 0.8631$, we should swap the two row. The answer after the pivoting gives more accurate $x_2 = 1.001$ and $x_1 = 10.00$.

Operations count for Gaussian Elimination

- The number of arithmetic operations involve in Gaussian Elimination depends on the size $n \times n$ of the matrix considered.
- $+/-$ operations always take less time to perform compared to $\times/\div$.
- Consider the elimination step for row-i, that requires to perform $a_{ij} \leftarrow a_{ij} - \left(\frac{a_{ik}}{a_{kk}}\right) a_{kj}$ for $(n-i)$ rows.

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ \vdots & \vdots & & & \vdots \\ 0 & \times & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \times & \times & \times \end{bmatrix}$$

- For each of the $(n-i)$ rows,
  - need $(n-i)$ $\div$ operations for $a_{ik}/a_{kk} = m_{ik}$.
  - need $(n-i)(n-i+1)$ $\times/\div$ operations for $a_{ij} - m_{ik}a_{kj}$.
  - need $(n-i)(n-i+1)$ $+/-$ operations for $a_{ij} - m_{ik}a_{kj}$.
  - thus, total $+/-$ ops $= (n-i)(n0i+1)$, total $\times/\div$ ops $= (n-i)(n-i+2)$.

Operations count for Gaussian Elimination (Cont.)

- Repeating for all rows, then in the elimination stage:
  - Total $+/-$ ops is: $\sum_{i=1}^{n-1}(n-i)(n-i+1) = \frac{n^3-n}{3} \sim O(n^3)$.
  - Total $\times/\div$ ops is: $\sum_{i=1}^{n-1}(n-i)(n-i+2) = \frac{1}{6}(2n^3+3n^2-5n) \sim O(n^3)$.
- As for the backward substitution stage:
  - Total $+/-$ ops is: $1+\sum_{i=1}^{n-1}(n-i+1) = \frac{n^2-n}{2} \sim O(n^2)$.
  - Total $\times/\div$ ops is: $\sum_{i=1}^{n-1}(n-i-1+1) = \frac{n^2+n}{2} \sim O(n^2)$.
- Total operations count for Gaussian Elimination:
  - Total $+/-$ ops is: $\frac{n^3}{3}+\frac{n^2}{2}-\frac{5n}{6}$.
  - Total $\times/\div$ ops is: $\frac{n^3}{3}+n^2-\frac{n}{3}$.

Operations count for Gaussian Elimination (Cont.)

n	$+/-$	$\times/\div$
3	17	11
10	375	430
50	42,875	44,150
100	338,250	343,300

Outline

Matrix Factorisation (motivation)

- From the previous slide, we know that in Gaussian Elimination:
  - operations count for elimination stage $\sim O(n^3)$.
  - operations count for substitution stage $\sim O(n^2)$.
- Suppose the matrix $\mathbf{A}$ can be factorised into two triangular form $\mathbf{A} = \mathbf{LU}$ hen we could solve the $\mathbf{Ax} = \mathbf{b}$ by just substitution.
- The idea:
  1. Write $\mathbf{A} = \mathbf{LU}$, thus $\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}$.
  2. Define $\mathbf{y} = \mathbf{Ux}$, now $\mathbf{Ax} = \mathbf{b}$ are into two equations $\mathbf{Ly} = \mathbf{b}$ and $\mathbf{Ux} = \mathbf{y}$.
  3. Since both $\mathbf{L}$ and $\mathbf{U}$ are triangular, both equation could be solved by substitution operations.
- Since substitution has the op count $\sim O(n^2)$, it is expected to speed up the process of solving $\mathbf{Ax} = \mathbf{b}$.

n	$n^3/3$	$2n^2$	% reduction
10	$3.\bar{3} \times 10^2$	2×10^2	40
100	$3.\bar{3} \times 10^5$	2×10^4	94
1000	$3.\bar{3} \times 10^8$	2×10^6	99.4

Matrix Factorisation (motivation)

- However, matrix factorisation method will not have much advantage over naive Gaussian Elimination for one value of $\mathbf{b}$.
- This is because of the process required for factorising $\mathbf{A}$ into $\mathbf{LU}$ requires $O(n^3/3)$ op count.
- However, the factorisation method will be preferred if we are solving $\mathbf{Ax} = \mathbf{b}_i$ for different values of $\mathbf{b}_i$, as one only need to factorise $\mathbf{A}$ once and the same $\mathbf{L}$ and $\mathbf{U}$ could be applied to the different values of $\mathbf{b}_i$ repeatedly.

LU-Factorisation

- Let $\mathbf{A}$ be a 3×3 matrix and we would like to factorise into $\mathbf{LU}$.
- Let $\mathbf{M}_1$ and $\mathbf{M}_2$ be the transformation matrices that perform the elimination operations:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}$$

$$\mathbf{A} \rightarrow \mathbf{M}_1\mathbf{A} \rightarrow \mathbf{M}_2\mathbf{M}_1\mathbf{A}$$

- Note that, $\mathbf{M}_1 = \begin{bmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & 0 & 1 \end{bmatrix}$ and $\mathbf{M}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -m_{32} & 1 \end{bmatrix}$ where $m_{ij} = \frac{a_{ij}}{a_{jj}}$.
- Now, $\mathbf{M}_2\mathbf{M}_1\mathbf{A} = \mathbf{U}$, then $\mathbf{A} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\mathbf{U}$.
- i.e. $\mathbf{L} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1} = \mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix}$.

LU Factorisation (example)

Example

Factorise $\mathbf{A}$ into $\mathbf{LU}$ and solve for $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{b} = [1, 1, -3, 4]^T$.

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

ANSWER:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ -1 & -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 3 \\ 0 & -1 & -1 & -5 \\ 0 & 0 & 3 & 13 \\ 0 & 0 & 0 & -13 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 1 \\ -1 \\ -2 \\ 2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} -0.30769 \\ 1.76923 \\ 0 \\ 0.15385 \end{bmatrix}$$

$P^T LU$-Factorisation

- LU-factorisation only work if $\mathbf{A}$ can be reduced by Gaussian Elimination without pivoting.
- If partial pivoting is required for controlling the round-off error, then we need to have extra permutation matrix $\mathbf{P}$ to carry out the row exchange.
- Thus, for $\mathbf{Ax} = \mathbf{b}$ requires pivoting, we multiply $\mathbf{P}$ on both side: $\mathbf{PAx} = \mathbf{Pb}$.
- Then, factorise $\mathbf{PA}$, ie. $\mathbf{PA} = \mathbf{P^{-1}LU}$.
- Since $\mathbf{P}$ is symmetric, $\mathbf{P}^{-1} = \mathbf{P}^T$, thus $\mathbf{A} = \mathbf{P}^T\mathbf{LU}$.

Example

Find the permutation matrix $\mathbf{P}$ for exchanging row 1 and row 2, then factorise the matrix $\mathbf{A}$.

ANSWER:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 3 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 3 \\ 0 & 0 & 2.5 \end{bmatrix}$$

Outline

Special Matrices

Definition (Diagonally Dominant Matrix)

A $n \times n$ matrix $\mathbf{A}$ is said to be diagonally dominant when

$$|a_{ii}| \geq \sum_{\substack{i=1 \\ j \neq i}} |a_{ij}|$$

holds for each of $i = 1, 2, \ldots, n$.

Definition (Strictly Diagonally Dominant Matrix)

A matrix $\mathbf{A}$ is strictly diagonally dominant matrix if

$$|a_{ii}| > \sum_{\substack{i=1 \\ j \neq i}} |a_{ij}|$$

for all $i = 1, 2, \ldots, n$.

Special Matrices (Cont.)

Theorem

A strictly diagonally dominant matrix $\mathbf{A}$ *is nonsingular. Moreover, Gaussian Elimination can be performed on a strictly diagonally dominant matrix without row interchange.*

Example

$\begin{bmatrix} 15 & -2 & -6 & 0 \\ -2 & 12 & -4 & -1 \\ -6 & -4 & 19 & 9 \\ 0 & -1 & -9 & 21 \end{bmatrix}$ is diagonally dominant, but not strictly diagonally dominant.

$\begin{bmatrix} 15 & -2 & -6 & 0 \\ -2 & 12 & -4 & -1 \\ -6 & -4 & 19 & 7 \\ 0 & -1 & -9 & 21 \end{bmatrix}$ is strictly diagonally dominant, and determinant is non-zero.

Positive Definite Matrix

Definition (Positive Definite Matrix)

A matrix $\mathbf{A}$ is (strictly) positive definite if it is symmetric and if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for every non-zero n-dimensional vector $\mathbf{x}$.

Definition (Leading Principal Submatrix)

A leading principal submatrix of a $n \times n$ matrix $\mathbf{A}$ is a matrix of form

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \dots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{bmatrix}$$

for some $1 \leq k \leq n$.

Positive Definite Matrix (Cont.)

Theorem

A symmetric matrix $\mathbf{A}$ is positive definite if and only if each of its leading principal submatrices have positive determinants.

Example

$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$ is positive definite since

- $\mathbf{A_1} = [2]$ and $\det(\mathbf{A_1}) = 2 > 0$
- $\mathbf{A_2} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ and $\det(\mathbf{A_2}) = 3 > 0$
- $\mathbf{A_3} = \mathbf{A}$ and $\det(\mathbf{A_3}) = 4 > 0$
- since the determinant for all leading principal submatrices is positive, thus $\mathbf{A}$ is positive definite.
- Check, $\mathbf{x}^T\mathbf{A}\mathbf{x} = x_1^2 + (x_1 + x_2)^2 + (x_2 - x_3)^2 + x_3^2 \geq 0$.

LDL^T and LL^T Factorisations

Theorem

A matrix $\mathbf{A}$ is positive definite if and only if Gaussian Elimination without row interchange can be performed on the corresponding linear system $\mathbf{Ax} = \mathbf{b}$ with all pivot elements positive.

Corollary

If $\mathbf{A}$ is symmetric and strictly diagonally dominant, then $\mathbf{A}$ can be factorised into the form of $\mathbf{LDL}^T$.

Corollary (Cholesky Factorisation)

If $\mathbf{A}$ is positive definite if and only if $\mathbf{A}$ can be factorised into the form $\mathbf{LL}^T$.

LDL^T and LL^T Factorisations

Example (LDL^T Factorisation)

$$\mathbf{A} = \begin{bmatrix} 4 & 3 & 2 \\ 3 & 3 & 2 \\ 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \begin{bmatrix} 1 & l_{21} & l_{31} \\ 0 & 1 & l_{32} \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} d_1 & d_1 l_{21} & d_1 l_{31} \\ d_1 l_{21} & d_2 + d_1 l_{21}^2 & d_2 l_{32} + d_1 l_{21} l_{31} \\ d_1 l_{31} & d_1 l_{21} l_{31} + d_2 l_{32} & d_1 l_{31}^2 + d_2 l_{32}^2 + d_3 \end{bmatrix}$$

Example (LL^T Factorisation)

$$\mathbf{A} = \begin{bmatrix} 4 & 3 & 2 \\ 3 & 3 & 2 \\ 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} =$$

$$\begin{bmatrix} l_{11}^2 & l_{11} l_{21} & l_{11} l_{31} \\ l_{11} l_{21} & l_{21}^2 + l_{22}^2 & l_{21} l_{31} + l_{22} l_{32} \\ l_{11} l_{31} & l_{21} l_{31} + l_{22} l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix}$$

Outline

Iterative Methods

Stationary Methods

- Jacobi method
- Gauss-Seidel method
- Successive over-relaxation (SOR) method

Krylov subspace methods

- Conjugate gradient
- Generalized minimal residual

The idea

- Like fixed-point iteration, setup the recurrence relation

$$\mathbf{Q}\mathbf{x}^{(k+1)} = (\mathbf{Q} - \mathbf{A})\mathbf{x}^{(k)} + \mathbf{b},$$

  initial value $\mathbf{x}^{(0)}$ and generates $\{\mathbf{x}^{(k)}\}_{k=1}$ such that $\mathbf{x}^{(k)}$ converges to the solution $\mathbf{x}^*$ for $\mathbf{A}\mathbf{x}^* = \mathbf{b}$.
- The can be different choices for $\mathbf{Q}$ depending on
  - the ease to solve or iteration
  - ensure convergence and rate of convergence

Outline

Jacobi Method

- Choose $\mathbf{Q}$ as the diagonal of $\mathbf{A}$.
- Let $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U} \implies \mathbf{D}\mathbf{x}^{(k+1)} = (\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{b}$
- The iterative scheme:

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b} \\ \text{or} & \\ x_i^{(k+1)} &= -\sum^{n}(a_{ij}/a_{ii})x_j^{(k)} + (b_i/a_{ii}), 1 \leq i \leq n \end{aligned}$$

- The iterations will converge if $\mathbf{A}$ is strictly diagonally dominant.

Example

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 1 \\ 8 \\ -5 \end{bmatrix}$$

Gauss-Seidel Method

- Choose $\mathbf{Q}$ as the lower triangular of $\mathbf{A}$.
- Let $\mathbf{A} = \mathbf{D} + \mathbf{L} - \mathbf{U} \implies (\mathbf{D} + \mathbf{L})\mathbf{x}^{(k+1)} = \mathbf{U}\mathbf{x}^{(k)} + \mathbf{b}$
- The iterative scheme:

$$\mathbf{x}^{(k+1)} = (\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}\mathbf{x}^{(k)} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b}$$

or

$$x_i^{(k+1)} = -\sum^{n}(a_{ij}/a_{ii})x_j^{(k+1)} - \sum^{n}(a_{ij}/a_{ii})x_j^{(k)} + (b_i/a_{ii}), 1 \leq i \leq n$$

Example

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 1 \\ 8 \\ -5 \end{bmatrix}$$

Successive Over-relaxation (SOR) Method

- Choose $\mathbf{Q} = \frac{1}{\omega}(\mathbf{D} - \omega\mathbf{L})$ where $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$ and ω is a parameter.
- The iterative scheme:

$$\mathbf{x}^{(k+1)} = \mathbf{Q}^{-1}(\mathbf{Q} - \mathbf{A})\mathbf{x}^{(k)} + \mathbf{Q}^{-1}\mathbf{b}$$

- The parameter ω needs to choose in between $0 < \omega < 2$ to ensure convergence.

Example

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 8 \\ -5 \end{bmatrix} \text{ and } \omega = 1.1.$$

Convergence Theorem

Definition (Spectral Radius)

The spectral radius $\rho(\mathbf{K})$ of a matrix $\mathbf{K}$ is the magnitude of the maximum eigenvalues

$$\rho(\mathbf{K}) = \max_{1 \leq i \leq n} |\lambda_i|.$$

Theorem (Spectral Radius Theorem)

In order that the sequence generated by

$$\mathbf{x}^{(k+1)} = \mathbf{K}\mathbf{x}^{(k)} + \mathbf{c}$$

to converge, regardless of initial point $\mathbf{x}^{(0)}$*, the necessary and sufficient condition is that all eigenvalues of* $\mathbf{K}$ *lie in the open unit disc* $|z| < 1$ *or the spectral radius* $\rho(\mathbf{K}) < 1$.

Convergence Theorem (Cont.)

Theorem (Convergence theorem for Jacobi and Gauss-Seidel methods)

If $\mathbf{A}$ is diagonally dominant, the Jacobi and Gauss-Seidel methods converge for any $\mathbf{x}^{(0)}$.

Theorem (Convergence theorem for SOR method)

Suppose that $\mathbf{A}$ has positive diagonal elements and that $0 < \omega < 2$, the SOR method converges for any $\mathbf{x}^{(0)}$ if and only if $\mathbf{A}$ is positive definite.

Outline

$\mathbf{A}$-Conjugacy

Definition

If $\mathbf{u}$ and $\mathbf{v}$ are mutually orthogonal, if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.

Definition

$\mathbf{A}$-inner product of $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbf{A}} = \langle \mathbf{A}\mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{A}^T \mathbf{v}.$$

Definition

$\mathbf{u}$ and $\mathbf{v}$ are $\mathbf{A}$-conjugate if $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbf{A}} = 0$.

Definition

A matrix $\mathbf{A}$ is positive definite if $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathbf{A}} \geq 0$.

Conjugate Gradient Method

- This is a popular iterative method for solving sparse system.
- Consider a quadratic form

$$f(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, \mathbf{x} \rangle_{\mathbf{A}} - \langle \mathbf{b}, \mathbf{x} \rangle + \mathbf{c}$$

  and its gradient

$$\nabla f(\mathbf{x}) = \frac{1}{2}(\mathbf{A}^T\mathbf{x} + \mathbf{A}\mathbf{x}) - \mathbf{b}.$$

- If $\mathbf{A}$ is symmetric, then $\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$.
- Therefore the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is a critical point for $f(\mathbf{x})$.
- Thus, if $\mathbf{A}$ is a positive definite and symmetric, then we can solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ by minimising $f(\mathbf{x})$.

Conjugate Gradient Method (Cont.)

- Suppose $\mathbf{x}^* = t_1\mathbf{v}^{(1)} + t_2\mathbf{v}^{(2)} + \cdots + t_n\mathbf{v}^{(n)}$ is the true solution for $\mathbf{Ax} = \mathbf{b}$, where:
  - $\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(n)}\}$ forms a mutually $\mathbf{A}$-conjugate basis of $\mathbb{R}^n$.
  - the coefficient is given by $t_k = \dfrac{\langle \mathbf{v}^{(k)}, \mathbf{b}\rangle}{\langle \mathbf{v}^{(k)}, \mathbf{v}^{(k)}\rangle_{\mathbf{A}}}$
- Instead of compute directly the exact solution, We will construct $\mathbf{v}^{(k)}$ iteratively to approximate $\mathbf{x}^*$.
- Given a reference point, $\mathbf{x}^{(k)}$, we would like to know
  - a *residual vector* $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{Ax}^{(k)}$ associated with $\mathbf{x}^{(k)}$.
  - a *search direction* $\mathbf{v}^{(k+1)}$ that move away from $\mathbf{x}^{(k)}$ and minimises f.
  - a *update rule* $\mathbf{x}^{(k+1)} = \mathbf{r}^{(k)} + t_k\mathbf{v}^{(k)}$.
- A natural search direction will be in the direction of *steepest descent*, ie. in the direction of $-\nabla f(\mathbf{x}^{(k)}) = \mathbf{r}^{(k)}$, however the convergence will be slow for linear system.
- Instead we use the $\mathbf{A}$-conjugate set $\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(n)}\}$ of direction vectors, and require that vectors $\{\mathbf{r}^{(k)}\}$ are mutually orthogonal.

Conjugate Gradient Method (Cont.)

Theorem

The residual vectors $\mathbf{r}^{(k)}, k = 1, 2, \ldots n$, *for a conjugate gradient method, satisfy the conditions*

$$\langle \mathbf{r}^{(k)}, \mathbf{v}^{(j)} \rangle = 0, \quad \text{for each } j = 1, 2, \ldots, k.$$

The algorithm

1. Given an initial point, $\mathbf{x}^{(0)}$, compute
   - residual vector, $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$
   - first search direction $\mathbf{v}^{(1)}$ is the steepest descent direction, $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}$
2. Compute the new approximate $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + t_k \mathbf{v}^{(k)}$, where
   $$t_k = \frac{\langle \mathbf{v}^{(k)}, \mathbf{r}^{(k-1)} \rangle}{\langle \mathbf{v}^{(k)}, \mathbf{A}\mathbf{v}^{(k)} \rangle} = \frac{\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \rangle}{\langle \mathbf{v}^{(k)}, \mathbf{v}^{(k)} \rangle_{\mathbf{A}}}$$
3. Prepare for the next iteration:
   - new residual vector $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - t_k \mathbf{A}\mathbf{v}^{(k)}$
   - new search direction, $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + s_k \mathbf{v}^{(k)}$, where $s_k = \dfrac{\langle \mathbf{r}^{(k)}, \mathbf{r}^{(k)} \rangle}{\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \rangle}$
4. Repeat Step 2 and 3 until desirable accuracy.

Conjugate Gradient Method (Example)

Example

$$\mathbf{A} = \begin{bmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 24 \\ 30 \\ -24 \end{bmatrix}, \text{ and } \mathbf{x}^{(0)} = [0, 0, 0]^T.$$

Outline

Review on Eigenvalue Problem

Eigenvalue Problem

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

- The eigenvalues are zeros of characteristic polynomial, $p(\lambda) = \det(\mathbf{A} - \lambda\mathbf{x})$.
- The algebraic multiplicity of an eigenvalue is the multiplicity of the corresponding root of the characteristic polynomial.
- The geometric multiplicity of an eigenvalue is the dimension of the associated eigenspace.
- If λ is eigenvalue of $\mathbf{A}$, then
  - $p(\lambda)$ is an eigenvalue of $p(\mathbf{A})$ for any polynomial p.
  - then $p(1/\lambda)$ is an eigenvalue for $p(\mathbf{A}^{-1})$, where $\mathbf{A}$ is nonsingular.
  - λ^k is an eigenvalue of $\mathbf{A}^k$.
  - $1/\lambda$ is an eigenvalue of $\mathbf{A}^{-1}$
- If $\mathbf{A}$ is real and symmetric, then its eigenvalues are real, and eigenvectors are mutually orthogonal.
- If $\mathbf{A}$ is complex and Hermitian, then its eigenvalues are real, and eigenvectors are mutually orthogonal (inner product with conjugate transpose).
- If $\mathbf{A}$ is Hermitian and positive definite, then itst eigenvalues are positive.

Outline

Schur's decomposition

Definition (Similar Matrices)

Two matrices $\mathbf{A}$ and $\mathbf{B}$ are similar to each other, if there exist a nonsingular matrix $\mathbf{P}$ such that $\mathbf{A} = \mathbf{P}^{-1}\mathbf{B}\mathbf{P}$.

Theorem

Similar matrices have the same characteristic polynomial and eigenvalues.

Theorem (Schur's Theorem)

Every square matrix is unitarily similar to a triangular matrix. $\mathbf{U}^{\dagger}\mathbf{A}\mathbf{U} = \mathbf{T}$.

Theorem

Every Hermitian (symmetric) matrix is unitarily (orthogonally) similar to a diagonal matrix.

Schur's Decomposition (Cont.)

Example

Given a matrix $\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$ Write the matrix into its Schur decomposition form, $\mathbf{T} = \mathbf{U}^{\dagger}\mathbf{A}\mathbf{U}$.

ANSWER: [MATLAB code: `nm03_schur.m`]

- `[V, D] = eig(A)`
- The eigenvector columns $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ in $\mathbf{V}$ are not mutually orthogonal.
- Use Gram-Schmidt orthogonalisation to find a new set of orthonormal basis, which gives the required unitary matrix $\mathbf{U} = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$, ie.
  $$\begin{bmatrix} 0.246474 & -0.962242 & -0.115507 \\ 0.147691 & -0.080498 & 0.985752 \\ 0.957830 & 0.260021 & -0.122274 \end{bmatrix}$$
- The triangular matrix is given by
  $$\mathbf{T} = \mathbf{U}^{\dagger}\mathbf{A}\mathbf{U} = \begin{bmatrix} 8.4853 & 3.1950 & -0.46072 \\ 0 & 4.6318 & -0.76123 \\ 0 & 0 & 1.8828 \end{bmatrix}$$

Geršgorin's Theorem

Theorem (Geršgorin Circle)

Let $\mathbf{A}$ be an $n \times n$ matrix and R_i denote the circle in the complex plane with centre a_{ii} and radius $\sum_{j=1, j\neq i}^{n} |a_{ij}|$; that is,

$$R_i = \left\{ z \in \mathbb{C} \,\middle|\, |z - a_{ii}| \leq \sum_{j=1, j\neq i}^{n} |a_{ij}| \right\},$$

where $\mathbb{C}$ denotes the complex plane. The eigenvalues of $\mathbf{A}$ are contained within the union of these circles, $R = \bigcup_{i=1}^{n} R_i$. Moreover, the union of any k of the circles that do not intersect the remaining $(n-k)$ contains precisely k (counting multiplicity) of the eigenvalues.

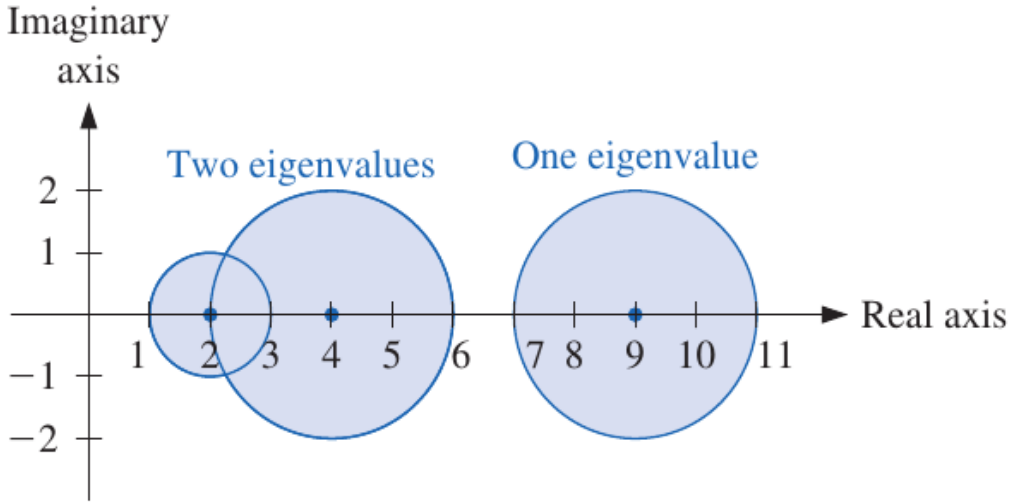
Geršgorin's Theorem (Cont.)

Example

Let $\mathbf{A} = \begin{bmatrix} 4 & 1 & 1 \\ 0 & 2 & 1 \\ -2 & 0 & 9 \end{bmatrix}$. Determine the Geršgorin's circles for $\mathbf{A}$ and find bounds of the spectral radius of $\mathbf{A}$.

ANSWER

- Eigenvalues: 8.4853, 4.6318, 1.8828.



Outline

Power Method

Let $\mathbf{A}$ be $n \times n$ matrix and its eigenvalues $|\lambda_i| > |\lambda_2| \geq |\lambda_3| \geq \ldots |\lambda_n|$ and associated eigenvectors $\mathbf{u}^{(i)}$, where $\mathbf{A}\mathbf{u}^{(i)} = \lambda_i \mathbf{u}^{(i)}, i = 1, 2, \ldots, n$.

- Power method will computes the largest eigenvalue and its associated eigenvector.
- Choose an arbitrary starting vector $\mathbf{x}^{(0)} \in \mathbb{C}^n$ and $\mathbf{x}^{(0)} = c_1\mathbf{u}^{(1)} + c_2\mathbf{u}^{(2)} + \cdots + c_n\mathbf{u}^{(n)} = \sum_{i=1}^{n} c_i\mathbf{u}^{(i)}$.
- Then, generate the sequence $\{\mathbf{x}^{(k)}\}$ from the recurrence relation

$$\mathbf{x}^{(k+1)} = \mathbf{A}\mathbf{x}^{(k)}.$$

- Note that,

$$\mathbf{x}^{(k)} = \mathbf{A}^k\mathbf{x}^{(0)} = \mathbf{A}^k \sum_{i=1}^{n} c_i\mathbf{u}^{(i)} = \sum_{i=1}^{n} \lambda_i^k c_i\mathbf{u}^{(i)} = \lambda_1^k \sum_{i=1}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^k c_i\mathbf{u}^{(i)}.$$

- Thus, when $k \to \infty$, $\lim_{k\to\infty} \left(\frac{\lambda_i}{\lambda_1}\right)^k = 0$, and $\lim_{k\to\infty} \mathbf{A}^k\mathbf{x}^{(0)} = \lim_{k\to\infty} \lambda_1^k c_1\mathbf{u}^{(1)}$.
- However, the last term will diverges if $|\lambda_1| > 1$ and converges to $\mathbf{0}$ if $|\lambda_1| < 1$, thus we need to scale $\mathbf{A}^k\mathbf{x}$ appropriately to ensure the limit is finite.

Power Method

- Let denote $\boldsymbol{\epsilon}^{(k)} = \sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^k c_i \mathbf{u}^{(i)}$, where $\lim_{k\to\infty} \boldsymbol{\epsilon}^{(k)} = \mathbf{0}$. Then, $\mathbf{x}^{(k)} = \lambda_1^k [c_1 \mathbf{u}^{(1)} + \boldsymbol{\epsilon}^{(k)}]$.
- Let ϕ be any linear functional on $\mathbb{C}^n$ such that $\phi(\mathbf{u}^{(1)}) \neq 0$ and then, $\phi(\mathbf{x}^{(k)}) = \lambda_1^k \left[c_1 \phi(\mathbf{u}^{(1)}) + \phi(\boldsymbol{\epsilon}^{(k)})\right]$.
- Then generate the sequence of ratios $\{r_k\}$:

$$r_k = \frac{\phi(\mathbf{x}^{(k+1)})}{\phi(\mathbf{x}^{(k)})} = \lambda_1 \left[\frac{c_1 \phi(\mathbf{u}^{(1)}) + \phi(\boldsymbol{\epsilon}^{(k+1)})}{c_1 \phi(\mathbf{u}^{(1)}) + \phi(\boldsymbol{\epsilon}^{(k)})}\right],$$

  where when $k \to \infty, r_k \to \lambda_1$.

The algorithm $(\mathbf{Ax} = \lambda \mathbf{x})$:

1. choose a starting point $\mathbf{x}^{(0)}$, for example $[1, 1, 1]^T$
2. Repeat for N times:
   1. $\mathbf{y} \leftarrow \mathbf{Ax}$
   2. $r \leftarrow \phi(\mathbf{y})/\phi(\mathbf{x})$ (a common choise is $\phi(\mathbf{x}) = x_1$.)
   3. $\mathbf{x} \leftarrow \mathbf{y}/||\mathbf{y}||_\infty$

Power Method (Cont.)

Example

Find the largest eigenvalue for $\mathbf{A} = \begin{bmatrix} 4 & 1 & 1 \\ 0 & 2 & 1 \\ -2 & 0 & 9 \end{bmatrix}$.

ANSWER: [MATLAB code: `nm03_power.m`]

- Choose a starting point, $\mathbf{x}^{(0)} = [1, 1, 1]^T$ and $\phi(\mathbf{x}) = x_1$.
- After 22 iteration r converges to the largest eigenvalue 8.4853.
- Note that the power method could be accelerated by applying the Aitken's Δ^2 method.

Outline

Inverse Power Method

Several modification could be done on the power method in order to compute the other eigenvalues.

- Case I : Smallest Eigenvalue
  - $\mathbf{Ax} = \lambda\mathbf{x} \Leftrightarrow \frac{1}{\lambda}\mathbf{x} = \mathbf{A}^{-1}\mathbf{x}$.
  - Since the smallest eigenvalue of $\mathbf{A}$ is the largest eigenvalue of $\mathbf{A}^{-1}$, thus we could apply the power method $\mathbf{x}^{(k+1)} = \mathbf{A}\mathbf{x}^{(k)}$.
  - To avoid calculating $\mathbf{A}^{-1}$, we could solve $\mathbf{A}\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ by LU-decomposition instead.

Example

Find the smallest eigenvalue for $\mathbf{A} = \begin{bmatrix} 4 & 1 & 1 \\ 0 & 2 & 1 \\ -2 & 0 & 9 \end{bmatrix}$.

ANSWER: [MATLAB code: `nm03_ipower_1.m`]

- Choose a starting point, $\mathbf{x}^{(0)} = [1, 1, 1]^T$ and $\phi(\mathbf{x}) = x_1$.
- Use MATLAB command `A\x` to compute $\mathbf{A}^{-1}\mathbf{x}$.
- After 14 iterations $1/r$ converges to the smallest eigenvalue 1.8828.

(Shifted) Inverse Power Method

- Case II : Shifted Inverse Power Method
  - $\mathbf{Ax} = \lambda\mathbf{x} \Leftrightarrow (\mathbf{A} - \mu\mathbf{I})\mathbf{x} = (\lambda - \mu)\mathbf{x} \Leftrightarrow \frac{1}{\lambda-\mu}\mathbf{x} = (\mathbf{A} - \mu\mathbf{I})^{-1}\mathbf{x}$.
  - Thus, we could apply inverse power method to find the smallest eigenvalue of $(\mathbf{A} - \mu\mathbf{I})$.
  - With appropriate choice of μ, then we could find the eigenvalue of $\mathbf{A}$ that is closest to μ.

Example

Find an eigenvalue for $\mathbf{A} = \begin{bmatrix} 4 & 1 & 1 \\ 0 & 2 & 1 \\ -2 & 0 & 9 \end{bmatrix}$ that is close to 4.

ANSWER: [MATLAB code: `nm03_ipower_2.m`]

- Choose a starting point, $\mathbf{x}^{(0)} = [1, 1, 1]^T$ and $\phi(\mathbf{x}) = x_1$.
- Let $\mathbf{B} = \mathbf{A} - \mu\mathbf{I}$ and use MATLAB command `B\x` to compute $\mathbf{B}^{-1}\mathbf{x}$.
- After 14 iterations $1/r + \mu$ converges to the smallest eigenvalue 1.8828.

Power Method with Rayleigh Quotient

Definition

Let $\mathbf{A}$ be a real symmetric matrix, then the Rayleigh quotient of a vector $\mathbf{x} \in \mathbb{R}^n$ is defined as $r(\mathbf{x}) = \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$.

- If $\mathbf{u}$ is an eigenvector of $\mathbf{A}$, then $r(\mathbf{u}) = \lambda$ is the corresponding eigenvalue.
- If $\mathbf{x}$ is not an eigenvector, then $r(\mathbf{x}) = \alpha$ minimises $f(\alpha) = ||\mathbf{A}\mathbf{x} - \alpha\mathbf{x}||_2$.
- Thus, $r(\mathbf{x})$ is a natural eigenvalue estimate if $\mathbf{x}$ is close, but not equal, to an eigenvector.
- In fact, $r(\mathbf{x})$ is quadratically accurate as eigenvalue estimate when $\mathbf{x} \to \mathbf{u}$.
- We could modify the power method to incorporate Rayleigh quotient to speed up the convergence.
  1. Repeat for N times:
     1. $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$
     2. $\mathbf{x} \leftarrow \mathbf{y}/||\mathbf{y}||_\infty$
     3. $r \leftarrow \mathbf{x}^T\mathbf{A}\mathbf{x}$.

Outline

Householder Transformation

A useful technique to find eigenvalues that are not suffer from too much round-off errors is QR method. However, the QR method requires tridiagonal matrix as input.

Definition (Householder Transformation)

Let $\mathbf{w} \in \mathbb{R}^n$ with $\mathbf{w}^T\mathbf{w} = 1$. The $n \times n$ matrix

$$\mathbf{P} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T$$

is called a Householder transformation.

Theorem

A Householder transformation, $\mathbf{P} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T$*, is symmetric and orthogonal, ie* $\mathbf{P}^{-1} = \mathbf{P}$.

The aim of the Householder transform is to reduce an arbitrary symmetric matrix to a tridiagonal matrix.

Householder Method

Given a $n \times n$ matrix $\mathbf{A}$, the Householder algorithm:

1. For $k = 1 : n-2$, do
   1. $\alpha = -\text{sgn}(a^{(k)}_{k_1,k}) \left(\sum_{j=k+1}^{n} (a^k_{jk})^2 \right)^{1/2}$,
   2. $r = \left(\frac{1}{2}\alpha^2 - \frac{1}{2}\alpha a^{(k)}_{k+1,k} \right)^2$,
   3. $w^{(k)}_1 = w^{(2)}_2 = \cdots = w^{(k)}_3 = 0$,
   4. $w^{(k)}_{k+1} = \frac{a^{(k)}_{k+1,k} - \alpha}{2r}$,
   5. $w^{(k)}_j = \frac{a^{(k)}_{jk}}{2r}$, for each $j = k+2, \ldots, n$,
   6. $\mathbf{P}^{(k)} = \mathbf{I} - 2\mathbf{w}^{(k)} \cdot (\mathbf{w}^{(k)})^T$,
   7. $\mathbf{A}^{(k+1)} = \mathbf{P}^{(k)} \mathbf{A}^{(k)} \mathbf{P}^{(k)}$.

QR-Algorithm

Let $\mathbf{A}$ denote a symmetric tridiagonal matrix. The aim of QR-algorithm is to decrease the off-diagnoal elements iteratively via similarity transforms.

1. Set $\mathbf{A}^{(0)} = \mathbf{A}$.
2. Repeat for $k = 1, 2, \ldots$
   1. $\mathbf{Q}^{(k)}\mathbf{R}^{(k)} = \mathbf{A}^{(k-1)}$, where $\mathbf{Q}$ is orthogonal and $\mathbf{R}$ is upper triangular.
   2. $\mathbf{A}^{(k)} = \mathbf{R}^{(k)}\mathbf{Q}^{(k)}$

- To construct the factoring matrices $\mathbf{Q}$ and $\mathbf{R}$ uses a product of $n-1$ rotation matrices. $\mathbf{R}^{(k)} = \mathbf{P}_n\mathbf{P}_{n-1}\ldots\mathbf{P}_{k+1}\mathbf{A}^{(k)}$.
- Each of these rotation matrices reset the lower off-diagonal elements to zero one-by-one.
- The performance of the QR-algorithm can be improved by introducing a shift on each step, ie work on the shifted matrix $\mathbf{A} - \mu\mathbf{I}$ instead of $\mathbf{A}$.

THE END