Linear Systems

IPCST
Seoul National University

System of Linear Equations

- General problem
  - *n* equations, *n* unknowns

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots a_{nn}x_n &= b_n
\end{aligned}
$$

$$\mathbf{A} = \{a_{ij}\} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \qquad \mathbf{Ax = b} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Naïve Gaussian Elimination

1. Forward elimination
   - Combining row operations,
   - To make an upper triangular form
     - Row operations
       1. Multiplying a row by a scalar number
       2. Adding one row to another row
       3. Swapping two rows (not necessary in Naïve Gaussian Elimination)

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right] \qquad \left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 2 & 1 & 5 \end{array}\right] \qquad \left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$$

Figures from Wikipedia

Naïve Gaussian Elimination

1. Forward elimination
   - Wen Shen p. 119
   - To make an upper triangular form: $O(n^3)$

$$
\begin{aligned}
&\text{for } j = 1, 2, 3, \cdots, n-1 \\
&\qquad \text{for } i = j+1, j+2, \cdots, n \\
&\qquad\quad (i) \leftarrow (i) - (j) \times \frac{a_{ij}}{a_{jj}}, \\
&\qquad \text{end} \\
&\text{end}
\end{aligned}
$$

$$
\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]
\quad
\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 2 & 1 & 5 \end{array}\right]
\quad
\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]
$$

Figures from Wikipedia

Naïve Gaussian Elimination

2. Backward substitution
   - Wen Shen p. 119

$$x_n = \frac{b_n}{a_{nn}},$$

for $i = n-1, n-2, \cdots, 1$

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j\right)$$

end

➢ $O(n^2)$

$$\left[\begin{array}{ccc|c} 2 & 1 & 0 & 7 \\ 0 & \frac{1}{2} & 0 & \frac{3}{2} \\ 0 & 0 & -1 & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} 2 & 1 & 0 & 7 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right] \quad \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$$

Figures from Wikipedia

參考: Partial Pivoting

- In Gaussian Elimination, $a_{jj} \approx 0$ causes errors.
- To avoid this, one should swap two rows to maximize $|a_{jj}|$
  - Find the maximum $|a_{ij}|$ in the column j
  - Swapping the row j and the row k if $|a_{kj}|$ is the largest in the column j

❖ Partial pivoting with scaling is more effective. See Wen Shen 6.3.4.

Sparse Matrices & Band Matrices

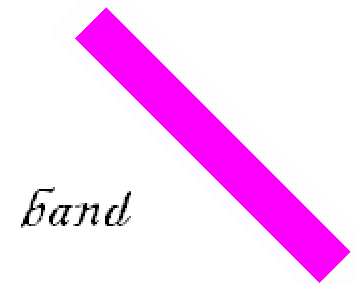
- Sparse matrix:

$$\text{density} = \frac{\text{\# of nonzero elements}}{\text{Total \# of elements}} \approx 0$$

  - But density $\neq 0$

- Band matrix:

  small bandwidth



  - Bandwidth: Max. width from the diagonal

  (the maximum distance of the nonzero elements from the main diagonal)

Tridiagonal Matrices

- Matrices with the (upper and lower) band width equal to 1
  - Cf.) bandwidth(diagonal matrix) = 0

$$\mathbf{A} = \begin{pmatrix} d_1 & c_1 & 0 & \cdots & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & \cdots & 0 & 0 & 0 \\ 0 & a_2 & d_3 & \cdots & 0 & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & d_{n-2} & c_{n-2} & 0 \\ 0 & 0 & 0 & \cdots & a_{n-2} & d_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & a_{n-1} & d_n \end{pmatrix}$$

  - Appropriate application of Gaussian elimination efficiently solves a tridiagonal system of linear equations

Tridiagonal Matrices

- Algorithm
  - Wen Shen p. 128

1. Forward elimination

$$\text{for } i = 2, 3, \cdots, n$$
$$d_i \leftarrow d_i - \frac{a_{i-1}}{d_{i-1}} c_{i-1}$$
$$b_i \leftarrow b_i - \frac{a_{i-1}}{d_{i-1}} b_{i-1}$$
$$\text{end}$$

2. Backward substitution

$$x_n \leftarrow b_n / d_n$$
$$\text{for } i = n-1, n-2, \cdots, 1$$
$$x_i \leftarrow \frac{1}{d_i}(b_i - c_i x_{i+1})$$
$$\text{end}$$

$$\mathbf{A} = \begin{pmatrix} d_1 & c_1 & 0 & \cdots & 0 & 0 & 0 \\ a_1 & d_2 & c_2 & \cdots & 0 & 0 & 0 \\ 0 & a_2 & d_3 & \cdots & 0 & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & d_{n-2} & c_{n-2} & 0 \\ 0 & 0 & 0 & \cdots & a_{n-2} & d_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & a_{n-1} & d_n \end{pmatrix}$$

$$O(n)$$

Do It Yourself

- Make your naïve Gaussian elimination code for trigonal matrices and test it with a system

$$\mathbf{A} = \begin{pmatrix} 20 & 5 & 0 & 0 & 0 \\ 5 & 15 & 5 & 0 & 0 \\ 0 & 5 & 15 & 5 & 0 \\ 0 & 0 & 5 & 15 & 5 \\ 0 & 0 & 0 & 5 & 10 \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} 1100 & 100 & 100 & 100 & 100 \end{pmatrix}^{t}$$

  – Matlab version is shown on Wen Shen p. 56. You may copy and edit its part.

Review of Linear Algebra

– Wen Shen pp. 130~133

- Diagonally dominant

$$|a_{ii}| \geq \sum_{j=1, j \neq i}^{n} \left|a_{ij}\right|, \qquad i = 1, 2, \cdots, n$$

  – Strictly diagonally dominant: > instead of ≥
  – Properties of ‘Strictly diagonally dominant’
    - Non-singular, invertible matrix
    - You don’t need pivoting in Gaussian Elimination

Review of Linear Algebra

– Wen Shen pp. 130~133

- Norm
  - Size measure of a vector or a matrix

$$\|\mathbf{x}\| \qquad\qquad\qquad \|\mathbf{A}\|$$

- Properties of a norm
  ① $\|\mathbf{x}\| \geq 0$, with $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = 0$
  ② $\|a\mathbf{x}\| = |a| \cdot \|\mathbf{x}\|$ where a is a scalar constant
  ③ $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$: triangle inequality

Review of Linear Algebra

- Wen Shen pp. 130~133

- Vector norms

1. $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|,$ l_1-norm
2. $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2},$ l_2-norm
3. $\|\mathbf{x}\|_\infty = \max_{1 \le i \le n} |x_i|,$ l_∞-norm

Review of Linear Algebra

– Wen Shen pp. 130~133

- Matrix norms

$$\|\mathbf{A}\| = \max_{\mathbf{x}\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}$$

– Properties

$$\|\mathbf{A}\| \geq \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \Longrightarrow \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|$$

$$\|\mathbf{I}\| = 1, \qquad \|\mathbf{A}\mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$$

Review of Linear Algebra

– Wen Shen pp. 130~133

- Matrix norms:
  - Examples

$$l_1 - \text{norm} \quad : \quad \|\mathbf{A}\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|$$

$$l_2 - \text{norm} \quad : \quad \|\mathbf{A}\|_2 = \max_{i} |\lambda_i| \,, \qquad \lambda_i : \ \text{eigenvalues of } \mathbf{A}$$

$$l_\infty - \text{norm} \quad : \quad \|\mathbf{A}\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$$

Review of Linear Algebra

– Wen Shen pp. 130~133

- Eigenvalues

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}, \quad \lambda\text{: eigenvalue, } \mathbf{v}\text{:eigenvector}$$

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = 0 \implies \det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

  - Property: if $\mathbf{A}$ is invertible and symmetric,

$$\lambda_i(\mathbf{A}^{-1}) = [\lambda_i(\mathbf{A})]^{-1}$$

$$\Downarrow$$

$$\|\mathbf{A}^{-1}\|_2 = \max_i |\lambda_i(\mathbf{A}^{-1})| = \max_i \frac{1}{|\lambda_i(\mathbf{A})|} = \frac{1}{\min_i |\lambda_i(\mathbf{A})|}$$

Review of Linear Algebra

– Wen Shen pp. 130~133

- Condition number
  - What to solve: $\mathbf{Ax} = \mathbf{b}$
  - ➢ Perturbed: $\mathbf{A\bar{x}} = \mathbf{b} + \mathbf{p}$
    - Relative errors

$$e_b = \frac{\|\mathbf{p}\|}{\|\mathbf{b}\|} \qquad\qquad e_{\mathbf{x}} = \frac{\|\bar{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|}$$

  - We have

$$\mathbf{A}(\bar{\mathbf{x}} - \mathbf{x}) = \mathbf{p} \quad \Longrightarrow \bar{\mathbf{x}} - \mathbf{x} = \mathbf{A}^{-1}\mathbf{p}$$

$$\therefore\ e_{\mathbf{x}} = \frac{\|\bar{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} = \frac{\|\mathbf{A}^{-1}\mathbf{p}\|}{\|\mathbf{x}\|} \leq \frac{\|\mathbf{A}^{-1}\| \cdot \|\mathbf{p}\|}{\|\mathbf{x}\|}.$$

Review of Linear Algebra

– Wen Shen pp. 130~133

- Condition number

$$\mathbf{Ax} = \mathbf{b} \Longrightarrow \|\mathbf{Ax}\| = \|\mathbf{b}\| \Longrightarrow \|\mathbf{A}\| \cdot \|\mathbf{x}\| \geq \|\mathbf{b}\| \Longrightarrow \frac{1}{\|\mathbf{x}\|} \leq \frac{\|\mathbf{A}\|}{\|\mathbf{b}\|}$$

  – We get

$$e_{\mathbf{x}} \leq \frac{\|\mathbf{A}^{-1}\| \cdot \|\mathbf{p}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{p}\| \cdot \frac{\|\mathbf{A}\|}{\|\mathbf{b}\|} = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| e_b = \kappa(\mathbf{A}) \cdot e_b$$

  – Condition number of $\mathbf{A}$: $\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$

    - In l_2-norm

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^{-1}\|_2 = \frac{\max_i |\lambda_i(\mathbf{A})|}{\min_i |\lambda_i(\mathbf{A})|}$$

✓ If $\kappa(A)$ is huge → error expansion → ill-conditioned system

Iterative Solvers for Linear Algebra

- What for?
  - Large, sparse and structured matrices
    - If $\mathbf{A}$ is large (ex.: $n = O(10^6)$), direct methods demand huge computing time.
    - Band matrices: often sparse and structured
    - Diagonally dominant matrices
- 2 kinds
  - **Fixed point iterative solvers**
    - Jacobi, Gauss-Seidel, SOR
  - Krylov solvers
    - Conjugate gradient, Arnoldi, Lanczos, GMRES, ……

Fixed point iterative solvers

– Wen Shen pp. 140~144

- Jacobi method

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots a_{nn}x_n &= b_n
\end{aligned}
$$

$$
\begin{cases}
x_1 = \frac{1}{a_{11}}\left(b_1 \qquad - a_{12}x_2 - \cdots - a_{1n}x_n\right), \\
x_2 = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1 \qquad - \cdots - a_{2n}x_n\right), \\
\quad \vdots \\
x_n = \frac{1}{a_{nn}}\left(b_2 - a_{n1}x_1 - a_{n2}x_2 - \cdots \qquad \right),
\end{cases}
$$

or

$$
x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\right)
$$

Fixed point iterative solvers

– Wen Shen pp. 140~144

- Jacobi method
  - Example

$$\begin{cases} 2x_1 - x_2 & = & 0 \\ -x_1 + 2x_2 - x_3 & = & 1 \\ -x_2 + 2x_3 & = & 2 \end{cases}$$

$$\Downarrow$$

$$\begin{cases} x_1^{k+1} & = & \frac{1}{2}x_2^k \\ x_2^{k+1} & = & \frac{1}{2}(1 + x_1^k + x_3^k) \\ x_3^{k+1} & = & \frac{1}{2}(2 + x_2^k) \end{cases}$$

Fixed point iterative solvers

– Wen Shen pp. 140~144

- Jacobi method

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^k \right)$$

  - Index *k*: number of iteration

– Algorithm

1. Choose a start vector $\mathbf{x}^0 = (x_1^0, x_2^0, \dots\dots, x_n^0)^t$
2. Apply the above formula in the iteration loop (*k*)
   - Two loops (*i:* outer & *j:* inner) per iteration
   - Computing x_i^{k+1} in loop of *i*: independent to each other → parallelizable

Fixed point iterative solvers

- Wen Shen pp. 140~144

- Jacobi method
  - Start vector $\mathbf{x}^0$
    - Anything except wrong fixed points (zero vector in case of non-zero solution problems)
    - Examples
      - $x_i^0 = 1$
      - $\mathbf{x}^0 = \mathbf{b}$
      - $x_i^0 = b_i / a_{ii}$
  - Memory consumption: You need two vectors $\mathbf{x}^k$ and $\mathbf{x}^{k+1}$.

Fixed point iterative solvers

– Wen Shen pp. 140~144

- Jacobi method
  - Stop criteria
    - $\left\|\mathbf{x}^{k+1}-\mathbf{x}^{k}\right\| \leq \varepsilon$
    - Residual $\mathbf{r}^{k}=\mathbf{A}\mathbf{x}^{k}-\mathbf{b}$: $\left\|\mathbf{r}^{k}\right\| \leq \varepsilon$
    - Maximum number of iteration: in case that it won’t converge

Fixed point iterative solvers

- Wen Shen pp. 140~144

- Gauss-Seidel method
  - Modified Jacobi

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^{k} \right)$$

  - Computing x_i^{k+1} in loop of *i* must be done after computing x_{i-1}^{k+1} → two times faster in serial computing but difficult to parallelize
  - Memory consumption → reduced. You need to save only one vector $\mathbf{x}$

Fixed point iterative solvers

– Wen Shen pp. 140~144

- Gauss-Seidel method
  - Example

$$\begin{cases} 2x_1 - x_2 & = & 0 \\ -x_1 + 2x_2 - x_3 & = & 1 \\ -x_2 + 2x_3 & = & 2 \end{cases}$$

$$\Downarrow$$

$$\begin{cases} x_1^{k+1} & = & \frac{1}{2}x_2^k \\ x_2^{k+1} & = & \frac{1}{2}(1 + x_1^{k+1} + x_3^k) \\ x_3^{k+1} & = & \frac{1}{2}(2 + x_2^{k+1}) \end{cases}$$

Fixed point iterative solvers

– Wen Shen pp. 140~144

- SOR (Successive Over Relaxation)
  - Modified Gauss-Seidel

$$x_i^{k+1} = (1-w)x_i^k + w \cdot \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right)$$

  - The parameter w(>0)
    - 1>w>0 : under relaxation
    - w=1 : Gauss-Seidel
    - 2>w>1 : over relaxation
    - w≥2 : divergence

Fixed point iterative solvers

– Wen Shen pp. 140~144

- SOR (Successive Over Relaxation)
  - Example

$$\begin{cases} 2x_1 - x_2 & = 0 \\ -x_1 + 2x_2 - x_3 & = 1 \\ -x_2 + 2x_3 & = 2 \end{cases}$$

$$\Big\Downarrow \; w=1.2$$

$$\begin{cases} x_1^{k+1} & = -0.2x_1^k + 0.6x_2^k \\ x_2^{k+1} & = -0.2x_2^k + 0.6 * (1 + x_1^{k+1} + x_3^k) \\ x_3^{k+1} & = -0.2x_3^k + 0.6 * (2 + x_2^{k+1}) \end{cases}$$

Do It Yourself

- Make your code of Jacobi iteration and apply it to Wen Shen 7.5: $\mathbf{Ax} = \mathbf{b}$

$$\mathbf{A} = \begin{pmatrix} 4 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & 0 & -1 & 0 & 0 \\ -1 & 0 & 4 & -1 & -1 & 0 \\ 0 & -1 & -1 & 4 & 0 & -1 \\ 0 & 0 & -1 & 0 & 4 & -1 \\ 0 & 0 & 0 & -1 & -1 & 4 \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} 1 & 5 & 0 & 3 & 1 & 5 \end{pmatrix}^{t}$$

✓ Initial guess: $\mathbf{x}^0 = \begin{pmatrix} 0.25 & 1.25 & 0 & 0.75 & 0.25 & 1.25 \end{pmatrix}^{t}$

Do It Yourself

- Edit your code to make one for Gauss-Seidel iteration and apply it to the same system

$$\mathbf{A} = \begin{pmatrix} 4 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & 0 & -1 & 0 & 0 \\ -1 & 0 & 4 & -1 & -1 & 0 \\ 0 & -1 & -1 & 4 & 0 & -1 \\ 0 & 0 & -1 & 0 & 4 & -1 \\ 0 & 0 & 0 & -1 & -1 & 4 \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} 1 & 5 & 0 & 3 & 1 & 5 \end{pmatrix}^{t}$$

✓ Initial guess: $\mathbf{x}^0 = \begin{pmatrix} 0.25 & 1.25 & 0 & 0.75 & 0.25 & 1.25 \end{pmatrix}^{t}$

Do It Yourself

- Edit your code to make one for SOR and apply it to the same system

$$\mathbf{A} = \begin{pmatrix} 4 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & 0 & -1 & 0 & 0 \\ -1 & 0 & 4 & -1 & -1 & 0 \\ 0 & -1 & -1 & 4 & 0 & -1 \\ 0 & 0 & -1 & 0 & 4 & -1 \\ 0 & 0 & 0 & -1 & -1 & 4 \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} 1 & 5 & 0 & 3 & 1 & 5 \end{pmatrix}^{t}$$

✓ Initial guess: $\mathbf{x}^0 = \begin{pmatrix} 0.25 & 1.25 & 0 & 0.75 & 0.25 & 1.25 \end{pmatrix}^{t}$

✓ Weight w=1.12

Do It Yourself

- [After this class]: Plot the errors for iterations solving the previous problems.

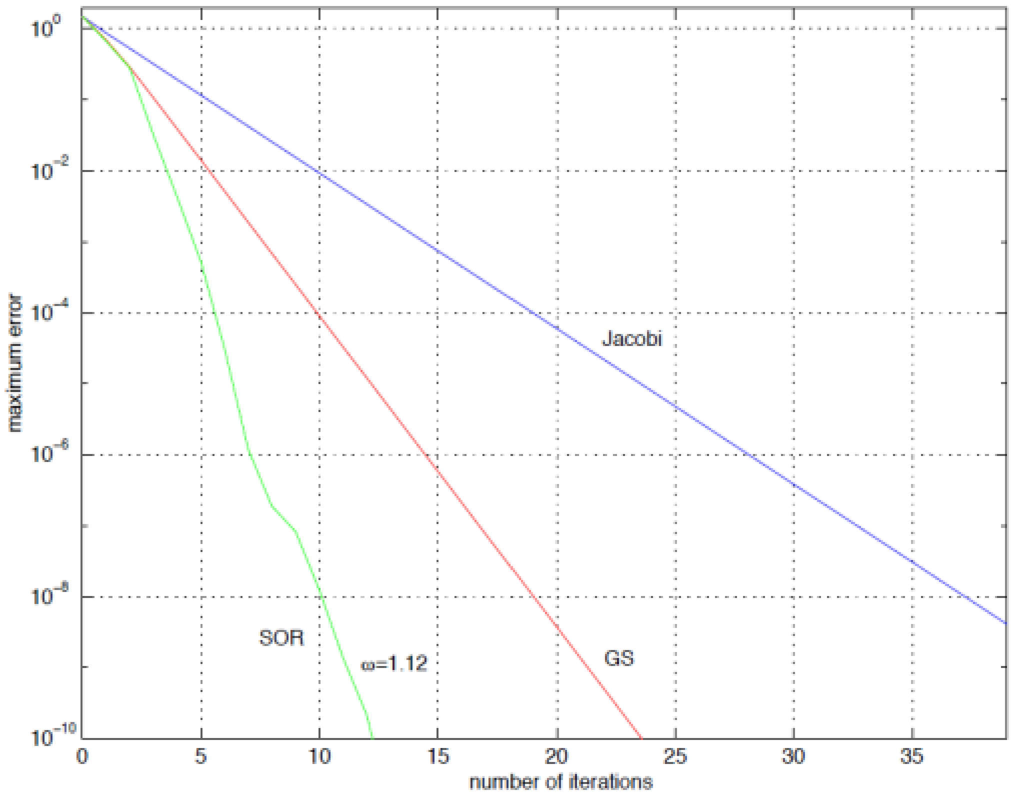


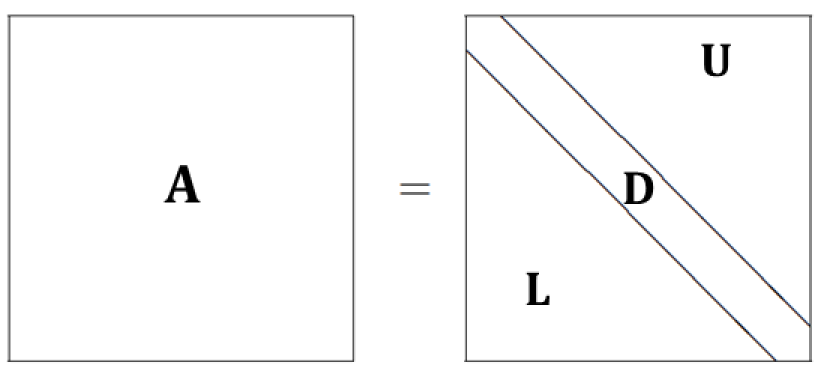
Figure from Wen Shen

Fixed point iterative solvers

  - Wen Shen pp. 147~150

- Standard form
  - In the matrix form
    - Problem: $\mathbf{Ax} = \mathbf{b} \rightarrow \mathbf{x} = \mathbf{Mx} + \mathbf{y}$
    - Fixed point iteration: $\mathbf{x}^{k+1} = \mathbf{Mx}^{k} + \mathbf{y}$
  - Splitting $\mathbf{A}$:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$$



Fixed point iterative solvers

– Wen Shen pp. 147~150

- Standard form
  - Jacobi

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

$$\Downarrow$$

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

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

Fixed point iterative solvers

– Wen Shen pp. 147~150

- Standard form
  - Gauss-Seidel

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

$$\Downarrow$$

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

$$\mathbf{x}^{k+1} = \mathbf{M}\mathbf{x}^{k} + \mathbf{y} \rightarrow \mathbf{y} = (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b} \;\&\; \mathbf{M} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}$$

Fixed point iterative solvers

– Wen Shen pp. 147~150

- Standard form
  - SOR

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

$$\Downarrow$$

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

$$\Downarrow$$

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

$$\mathbf{y} = (\mathbf{D} + w\mathbf{L})^{-1}\mathbf{b} \;\&\; \mathbf{M} = (\mathbf{D} + w\mathbf{L})^{-1}[(1-w)\mathbf{D} - w\mathbf{U}]$$

Fixed point iterative solvers

– Wen Shen pp. 147~150

- Error & convergence
  - Problem: $\mathbf{Ax} = \mathbf{b} \rightarrow \mathbf{x} = \mathbf{Mx} + \mathbf{y}$
  - Fixed point iteration: $\mathbf{x}^{k+1} = \mathbf{Mx}^k + \mathbf{y}$
  - Solution $\mathbf{s}$: $\mathbf{As} = \mathbf{b} \rightarrow \mathbf{s} = \mathbf{Ms} + \mathbf{y}$
  - Error vector $\mathbf{e}^k \coloneqq \mathbf{x}^k - \mathbf{s}$

    $\mathbf{e}^{k+1} = \mathbf{x}^{k+1} - \mathbf{s} = \mathbf{Mx}^k + \mathbf{y} - \mathbf{Ms} - \mathbf{y} = \mathbf{Me}^k$
  - Norms: $\|\mathbf{e}^{k+1}\| = \|\mathbf{Me}^k\| \leq \|\mathbf{M}\| \cdot \|\mathbf{e}^k\|$

    $\rightarrow \|\mathbf{e}^k\| \leq \|\mathbf{M}\|^k \cdot \|\mathbf{e}^0\|$
  - Convergence condition: $\|\mathbf{M}\| < 1$

Fixed point iterative solvers

– Wen Shen pp. 147~150

- Error & convergence
  - Smaller $\|\mathbf{M}\|$ → faster convergence
    - Jacobi: $\mathbf{M} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})$
    - Gauss-Seidel: $\mathbf{M} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}$
    - SOR: $\mathbf{M} = (\mathbf{D}+w\mathbf{L})^{-1}[(1-w)\mathbf{D}-w\mathbf{U}]$ → adjustable
  - Ex.)

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \Rightarrow$$

$\mathbf{M}$	l_1 norm	l_2 **norm**	l_∞ norm
Jacobi	1	**0.707**	1
G-S	0.875	**0.5**	0.75
SOR	0.856	**0.2**	0.88

$(w=1.2)$

Fixed point iterative solvers

– Wen Shen pp. 140~150

- **Convergence theorem**
  - If $\mathbf{A}$ is strictly diagonally dominant,

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \qquad \textit{for every} \quad i = 1, 2, \cdots, n.$$

  - then all three iteration methods converge to the exact solution, for every initial choice of $\mathbf{x}^0$

❖ The reverse is not always true.

參考: Conjugate Gradient

- CG was developed to solve $\mathbf{Ax} = \mathbf{b}$
  - $\mathbf{A}$: $n \times n$ symmetric positive-definite matrix
  - Minimizing $\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$ to find the solution $\mathbf{x}^*$
  - Nonzero vectors $\mathbf{u}_i$ are conjugate if $\mathbf{u}_i^T\mathbf{A}\mathbf{u}_i = 0$
  - Then $\mathbf{x}^* = \sum \alpha_i \mathbf{u}_i$. ($\alpha_i$:scalar)

$$\alpha_i = \frac{\mathbf{u}_i^T \mathbf{b}}{\mathbf{u}_i^T \mathbf{A} \mathbf{u}_i}$$

  - No need of matrix inversion if you find $\mathbf{u}_i$

參考: Conjugate Gradient

- CG was developed to solve $\mathbf{Ax} = \mathbf{b}$
  - From an arbitrary vector $\mathbf{x}_0$,
  - Iteration: $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{u}_i$.
  - Residual: $\mathbf{r}_i = \mathbf{b} - \mathbf{Ax}_i$
  - After n steps, $\mathbf{x}_n \cong \mathbf{x}^*$
  - We can set $\mathbf{u}_0 = \mathbf{r}_0$ and $\mathbf{u}_{i+1} = \mathbf{r}_{i+1} + \beta_i \mathbf{u}_i$.

$$\alpha_i = \frac{\mathbf{r}_i^{\mathbf{T}} \mathbf{u}_i}{\mathbf{u}_i^{\mathbf{T}} \mathbf{A} \mathbf{u}_i}$$

$$\beta_i = \frac{\mathbf{r}_{i+1}^{\mathbf{T}} \mathbf{r}_{i+1}}{\mathbf{r}_i^{\mathbf{T}} \mathbf{r}_i} = -\frac{\mathbf{r}_{i+1}^{\mathbf{T}} \mathbf{u}_i}{\mathbf{u}_i^{\mathbf{T}} \mathbf{A} \mathbf{u}_i}$$

參考: Conjugate Gradient

- Algorithm
  1. Initialization: $\mathbf{x}_0$ & $\mathbf{r}_0$
  2. Loop
     ① Calculation: α_i
     ② Iteration: $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{u}_i$
     ③ Iteration: $\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{A}\mathbf{u}_i$
     ④ Checking convergence: stop if converged
     ⑤ Calculation: β_i
     ⑥ Iteration: $\mathbf{u}_{i+1} = \mathbf{r}_{i+1} + \beta_i \mathbf{u}_i$
  3. output of $\mathbf{x}_{i+1}$

參考: Conjugate Gradient

- CG can be used to find a minimum of a convex function
  - Actually, $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i = -\nabla F(\mathbf{x}_i)$ where $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} - \mathbf{x}^{\mathbf{T}}\mathbf{b}$
  - Replace $\mathbf{r}_i$ with $-\nabla f(\mathbf{x}_i)$ for a convex function $f(\mathbf{x})$
  - Calculation of α_i
    - Fletcher-Reeves method
    - Polak-Ribière method
    - Hestenes-Stiefel method
    - Etc.

References

- Wen Shen,
  An Introduction to Numerical Computation
- Wikipedia
- A. Singh & P. Ravikumar, “Conjugate Gradient Descent”
- L. Vandenberghe, “Conjugate Gradient Method”

Further Study

- About **LU** factorization
- About **Arnoldi** (or **Lanczos**) iteration