Lecture 2 Direct methods for solving linear system

Weinan E[1,2] and Tiejun Li[2]

[1]Department of Mathematics,
Princeton University,
weinan@princeton.edu

[2]School of Mathematical Sciences,
Peking University,
tieli@pku.edu.cn
No.1 Science Building, 1575

Outline

Linear system

- Necessity
  1. Computers are good at performing high speed arithmetic operations
  2. Almost all of the scientific computing problems ends with a solution of a linear system
- Where does the linear system come from?
  1. Discretization of ODEs or PDEs
  2. Discretization of nonlinear systems
  3. Linear programming
  4. ······

Cramer's rule

- Cramer's rule

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{A} \in \mathbb{R}^{n\times n}, \ \boldsymbol{x}, \boldsymbol{b} \ \in \mathbb{R}^n$$

Case 1. $\det(\boldsymbol{A}) \neq 0$, $\boldsymbol{A}$ is nonsingular, then $x_i = \frac{\det(\boldsymbol{A}_i)}{\det(\boldsymbol{A})}$, where $\boldsymbol{A}_i$ is the matrix formed by replacing the i-th column of $\boldsymbol{A}$ with $\boldsymbol{b}$.

Case 2. $\det(\boldsymbol{A}) = 0$, $\boldsymbol{A}$ is singular, then

  - If $\text{rank}(\boldsymbol{A}, \boldsymbol{b}) = \text{rank}(\boldsymbol{A})$, infinite solutions;
  - If $\text{rank}(\boldsymbol{A}, \boldsymbol{b}) \neq \text{rank}(\boldsymbol{A})$, no solution.

- Cramer's rule is not suitable for computing (computational efficiency).

Upper triangular system

- Upper triangular system

$$\boldsymbol{U} \cdot \boldsymbol{x} = \boldsymbol{b}$$

where

$$\boldsymbol{U} = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ & u_{22} & \ddots & \vdots \\ & & \ddots & \ddots \\ & & & u_{nn} \end{pmatrix},$$

and $u_{ii} \neq 0, i = 1, \dots, n$.

- Direct solution by backward substitution

$$x_i = (b_i - \sum_{j=i+1}^{n} u_{ij} x_j)/u_{ii}, \quad i = n, n-1, \dots, 1$$

Here $\sum_{j=n+1}^{n}$ is defined as 0 when $i = n$.

- Similar strategy holds for other triangular matrices, such as lower, left, right triangular matrices etc.

A simple example

- Order $n = 30$, Upper triangular matrix

$$\boldsymbol{U} = \begin{pmatrix} 4 & 1 & 1 & & \\ & 4 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 4 & 1 \\ & & & & 4 \end{pmatrix},$$

- $\boldsymbol{b} = (1, 1, \dots, 1)^T$.
- What is the solution?

Computational efficiency for back substitution

- Total number of operations (addition, subtraction, multiplication, division)

$$\sum_{i=n}^{1} \Big[2\big(n - (i+1) + 1\big) - 1 + 1 + 1 \Big] = n^2$$

- Computational efficiency $O(n^2)$.

Outline

Gaussian elimination method

- How to obtain the solution of a general system?
- Idea: Transform the general matrix into upper triangular matrix.

  Step 1: Eliminate the first column.

$$a_{ij} \to a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}, \quad i = 2, \ldots, n; \; j = 1, \ldots, n$$

$$b_i \to b_i - \frac{a_{i1}}{a_{11}} b_1, \quad i = 2, \ldots, n$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{pmatrix} \to \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} & b_1^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{pmatrix}$$

Step 2: Eliminate the second column.

$$a_{ij}^{(2)} \to a_{ij}^{(2)} - \frac{a_{i2}^{(2)}}{a_{22}^{(2)}} a_{2j}^{(2)}, \quad i = 3, \dots, n; \; j = 2, \dots, n$$

$$b_i^{(2)} \to b_i^{(2)} - \frac{a_{i2}^{(2)}}{a_{22}^{(2)}} b_2^{(2)}, \quad i = 3, \dots, n$$

$$\begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} & b_1^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & a_{32}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{pmatrix} \to \begin{pmatrix} a_{11}^{(3)} & a_{12}^{(3)} & \cdots & a_{1n}^{(3)} & b_1^{(3)} \\ 0 & a_{22}^{(3)} & \cdots & a_{2n}^{(3)} & b_2^{(3)} \\ 0 & 0 & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(3)} & b_n^{(3)} \end{pmatrix}$$

Gaussian elimination method

Step 3: Repeat until the $n-1$-th column.

$$a_{ij}^{(n-1)} \to a_{ij}^{(n-1)} - \frac{a_{n-1,j}^{(n-1)}}{a_{n-1,n-1}^{(n-1)}} a_{i,n-1}^{(n-1)}, \quad i=n; \; j=n-1,n$$

$$\begin{pmatrix} a_{11}^{(n-1)} & a_{12}^{(n-1)} & \cdots & a_{1n}^{(n-1)} & b_1^{(n-1)} \\ 0 & a_{22}^{(n-1)} & \cdots & a_{2n}^{(n-1)} & b_1^{(n-1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{pmatrix} \to \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} & y_1 \\ 0 & u_{22} & \cdots & u_{2n} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & u_{nn} & y_n \end{pmatrix}$$

Step 4: Backward substitution to obtain $\boldsymbol{x}$.

$$\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}$$

A simple example

- Linear system

$$\begin{cases} x_1 + x_2 + x_3 & = & 6 \\ 2x_1 + 4x_2 + 2x_3 & = & 16 \\ -x_1 + 5x_2 - 4x_3 & = & -3. \end{cases}$$

- Augmented matrix

$$(\boldsymbol{A} \ \ \boldsymbol{b}) = \begin{pmatrix} 1 & 1 & 1 & 6 \\ 2 & 4 & 2 & 16 \\ -1 & 5 & -4 & -3 \end{pmatrix}$$

- Gaussian elimination.

Matrix form of Gaussian elimination method

- Gaussian elimination is equivalent to the following LU decomposition

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$$

and the solution of two triangular systems

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}, \quad \boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}$$

$$\boldsymbol{L} = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix}, \quad \boldsymbol{U} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix}$$

- The entries l_{ij} in matrix $\boldsymbol{L}$ are the same as those $\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$ in elimination steps, and the upper triangular matrix $\boldsymbol{U}$ is the same as that in Gaussian elimination.

Gaussian elimination algorithm

- Gaussian elimination algorithm:

$$
\begin{aligned}
&\text{for } k = 1, \dots, n-1 \\
&\quad \text{for } i = k+1, \dots, n \\
&\quad\quad l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \\
&\quad\quad \text{for } j = k, \dots, n \\
&\quad\quad\quad a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}
\end{aligned}
$$

Computational efficiency of Gaussian elimination

- Total number of triangulation

$$\sum_{k=1}^{n-1}\left[2\big((n+1)-k+1\big)+1\right](n-(k+1)+1)=\frac{2}{3}n^3+O(n^2)$$

- Computational efficiency

$$O\left(\frac{2}{3}n^3\right)$$

Symmetric positive definite (SPD) system

- The matrix form of Gaussian elimination for symmetric positive definite tridiagonal system has the following form

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$$

and we have $\boldsymbol{U} = \boldsymbol{D}\boldsymbol{L}^T$, where $\boldsymbol{D}$ is a diagonal matrix with $d_{ii} > 0$.

- Cholesky factorization for symmetric positive definite tridiagonal system

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$$

- $\boldsymbol{L}$ can be obtained by the following algorithm

$$l_{ij} = \frac{1}{l_{jj}}\Big(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}\Big), \quad j = 1, \dots, i-1,$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

Computational efficiency of Cholesky factorization

- Computational efficiency

$$\sum_{i=1}^{n}\left[\sum_{j=1}^{i-1}(2j-1)+(i-1)+(i-1)+1\right]=O\left(\frac{1}{3}n^3\right)$$

- The computational cost is a little less than direct Gaussian elimination by symmetry.

Tridiagonal system

- Tridiagonal system

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$$

where

$$\boldsymbol{A} = \begin{pmatrix} d_1 & c_1 & & \\ a_2 & d_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_n & d_n \end{pmatrix}.$$

- LU decomposition of tridiagonal system

$$\boldsymbol{L} = \begin{pmatrix} 1 & & & \\ \beta_2 & 1 & & \\ & \ddots & \ddots & \\ & & \beta_n & 1 \end{pmatrix}, \quad \boldsymbol{U} = \begin{pmatrix} \alpha_1 & c_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ & & & \alpha_n \end{pmatrix}$$

Tridiagonal system

- Thomas algorithm

$$\alpha_1 = d_1, \quad \beta_i = \frac{a_i}{\alpha_{i-1}}, \quad \alpha_i = d_i - \beta_i c_i, \quad i = 2, \ldots, n$$

- Computational efficiency

$$O(n)$$

Numerical solution of a linear system

- Numerical solution of a BVP

$$u''(x) = f(x), \quad x \in [0,1], \quad u(0) = u(1) = 0.$$

- Numerical discretization

  Define $h = \frac{1}{N}$, $\quad x_j = jh, f_j = f(x_j), j = 0, 1, \ldots, N$, and

$$u''(x_j) \approx \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2}$$

  then the ODE is reduced to a linear system $A \cdot X = b$, where

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

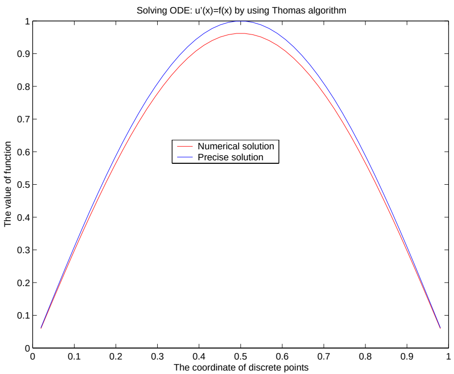
  $X = (u_1, u_2, \ldots, u_{N-1})^T$, $b = -h^2(f_1, f_2, \ldots, f_{N-1})^T$.

Numerical solution of a linear system

- Take $f(x) = -\pi^2 \sin \pi x$, we have the exact solution

$$u(x) = \sin \pi x.$$

- Take $N = 50$, we have the linear system and solve it with Thomas algorithm.
- Exact solution v.s. numerical solution



An example

- Linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}.$$

- Gaussian elimination

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & {\color{red}0} & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

- Even $\boldsymbol{A}$ is nonsingular, Gaussian elimination may NOT be proceeded directly.

Another example

- Linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1.0001 & 2 \\ 1 & 2 & 2 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}.$$

- Gaussian elimination

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1.0001 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0.0001 & 1 & 1 \\ 0 & 0 & 9999 & 10000 \end{pmatrix}$$

- If the precision $t = 3$, we will have $\bar{\boldsymbol{x}} = (0, 0, 1.000)$. But the roundoff exact solution is $\boldsymbol{x} = (1.000, -1.0001, 1.0001)$. It is totally different!
- Even direct Gaussian elimination could be applied, the result may be very bad!
- We need pivoting technique.

Outline

Pivoting

- What is pivoting?
- Complete pivoting is to let the largest element of the submatrix lie on the diagonal by interchanging rows or columns. A partial pivoting (or column pivoting) is to let the largest element in one column lie on the diagonal by interchanging two rows.
- The partial pivoting is more used.

Pivoting

- Example 1: Complete pivoting (move 3 — the largest one among the matrix — to a_{11})

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 2 & 3 \end{pmatrix} \to \begin{pmatrix} 2 & 2 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} \to \begin{pmatrix} 3 & 2 & 2 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- Example 2: partial pivoting (move 2 — the largest one among the first column — to a_{11})

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 2 & 3 \end{pmatrix} \to \begin{pmatrix} 2 & 2 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

Pivoting

- Pivoting

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

- Pivoting

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1.0001 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0.0001 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & & 0.9999 & 1 \end{pmatrix}$$

The numerical solution will be $\boldsymbol{x} = (1, -1, 1)$.

Matrix form for pivoting

- Matrix form for pivoting by row

$$\boldsymbol{PA} = \boldsymbol{LU}$$

  where $\boldsymbol{P}$ is a permutation matrix.
- Pivoting by row makes the computation more robust and stable.

Outline

Vector norms

Define $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$,

- Vector norms (definition of length)

$$
\begin{aligned}
&2-\text{norm} && \|\boldsymbol{x}\|_2 = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}} && \text{Euclidean norm}\\
&\infty-\text{norm} && \|\boldsymbol{x}\|_\infty = \max_i |x_i| &&\\
&1-\text{norm} && \|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i| &&\\
&p-\text{norm} && \|\boldsymbol{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}} &&
\end{aligned}
$$

- Properties of vector norms
  1. $\|\boldsymbol{x}\| \geq 0$ and $\|\boldsymbol{x}\| = 0$ iff $\boldsymbol{x} = 0$,
  2. $\|k\boldsymbol{x}\| = |k| \cdot \|\boldsymbol{x}\|$,
  3. $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ (Triangle inequality).

Matrix norms

Define $\boldsymbol{A} = (a_{ij})_{n \times n}$

- Matrix norm
  - Induced norm (subordinate norm)

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

From the definition of vector norms, we have

$$2 - \text{norm} \qquad \|\boldsymbol{A}\|_2 = \sqrt{\lambda_{\max}(\boldsymbol{A}^T\boldsymbol{A})} \ (= \sigma_{\max})$$

$$\infty - \text{norm} \qquad \|\boldsymbol{A}\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$$

$$1 - \text{norm} \qquad \|\boldsymbol{A}\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$$

  - Frobenius norm (why is it NOT an induced norm?)

$$\|\boldsymbol{A}\|_F = (\sum_{i,j=1}^{n} a_{ij}^2)^{\frac{1}{2}}$$

Properties of induced matrix norms

Properties of induced matrix norms:

1. $\|\boldsymbol{A}\| \geq 0$ and $\|\boldsymbol{A}\| = 0$ iff $\boldsymbol{A} = 0$,
2. $\|k\boldsymbol{A}\| = |k| \cdot \|\boldsymbol{A}\|$,
3. $\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$ (Triangle inequality),
4. $\|\boldsymbol{A}\boldsymbol{B}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{B}\|$,
5. $\|\boldsymbol{A}\boldsymbol{x}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{x}\|$.

Stability for the solution of linear system

- Example

$$A = \begin{pmatrix} 2.0002 & 1.9998 \\ 1.9998 & 2.0002 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 4 \end{pmatrix}.$$

The exact solution

$$x = (1, 1)^T$$

- Add perturbation $\delta b = (0.0002, -0.0002)^T$ to b, i.e. we have

$$\tilde{b} = (4.0002, 3.9998)^T$$

The perturbed solution

$$\tilde{x} = (1.5, 0.5)^T$$

- Relative error for solution and perturbation in ∞-norm

$$\frac{\|\tilde{x} - x\|_\infty}{\|x\|_\infty} = \frac{1}{2}, \qquad \frac{\|\tilde{b} - b\|_\infty}{\|b\|_\infty} = \frac{1}{20000}$$

The relative error is amplified 10000 times!!!

Condition number

- Condition number

$$Cond(A) = \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\|$$

For l^2-norm we have

$$Cond_2(\boldsymbol{A}) = \|\boldsymbol{A}\|_2 \cdot \|\boldsymbol{A}^{-1}\|_2$$

If $\boldsymbol{A}$ is symmetric, we have

$$Cond_2(\boldsymbol{A}) = \frac{\lambda_{\max}(\boldsymbol{A})}{\lambda_{\min}(\boldsymbol{A})}.$$

Remark 1: $Cond(\boldsymbol{A}) \geq 1.$

Remark 2: If $\det(\boldsymbol{A}) \approx 0$, $Cond(\boldsymbol{A}) \gg 1.$

High condition number matrix — Hilbert Matrix

- Hilbert matrix $H_n = (h_{ij})_{i,j=1}^{n}$ is defined as

$$h_{ij} = \frac{1}{i + j - 1}$$

- Hilbert matrix is a Symmetric Positive Definite (SPD) matrix
- Determinant of H_n

n	$\det(H_n)$
1	1
2	8.33333×10^{-2}
3	4.62963×10^{-4}
4	1.65344×10^{-7}
5	3.74930×10^{-12}
6	5.36730×10^{-18}

- $Cond_2(H_5) \sim O(10^5)$.

Explanation of condition number

- Original problem $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$;

  Perturbed problem $\boldsymbol{A}(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{b} + \delta\boldsymbol{b}$;

  Subtracting two equations we have

$$\delta\boldsymbol{x} = \boldsymbol{A}^{-1}\delta\boldsymbol{b}$$

  Take norm we have

$$\|\delta\boldsymbol{x}\| \leq \|\boldsymbol{A}^{-1}\|\|\delta\boldsymbol{b}\| = \|\boldsymbol{A}^{-1}\|\|\boldsymbol{A}\boldsymbol{x}\|\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}.$$

  With condition

$$\|\boldsymbol{A}\boldsymbol{x}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{x}\|$$

  we have

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq Cond(A)\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}.$$

Condition number

- Condition number characterize the stability
  If $Cond(\boldsymbol{A}) \gg 1$, stability is very bad;
  If $Cond(\boldsymbol{A}) \sim 1$, stability is good.
- Lesson we should learn:

  We should avoid handle the bad condition number problem in applications!

Stability

Loosely speaking, stability is to indicate how sensitive the solution of a problem may be to small relative changes in the input data. It is often quantized by condition number of a problem or an algorithm.

- Stability of the original problem ("Well-posedness")
  This means the well-posedness of the original problem. The linear system with high condition number is a typical ill-posed example, which is called the unstable problem.
- Stability of numerical algorithm
  This means the condition of the algorithm. The Gaussian elimination without pivoting for some linear system will be unstable.

Homework assignment 2

1. Using Thomas algorithm to solve the second order ODEs with one language (except matlab) (n=30, 50, 100). Compare the numerical solution with exact solution.

2. Compute 2-, 1- and ∞-condition number of n by n symmetric tridiagonal matrix

$$A_n = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

versus n with matlab and plot it as a figure.