Matrix Computations
Lecture 1: LU decomposition & Solving linear system

Zichong Ou
Supervisors: Jie Lu
ShanghaiTech University

Main content

- LU decomposition
- PLU decomposition
- LDL decomposition
- Iterative methods for solving linear systems

LU Decomposition

LU decomposition : Given $\mathbf{A} \in \mathbb{R}^{n\times n}$, find two matrices $\mathbf{L}, \mathbf{U} \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{A} = \mathbf{L}\mathbf{U},$$

where

- $\mathbf{L} \in \mathbb{R}^{n\times n}$ is lower triangular with unit diagonal elements (i.e., $\ell_{ii} = 1$ for all i) (unit lower triangular);
- $\mathbf{U} \in \mathbb{R}^{n\times n}$ is upper triangular.

Finding $\mathbf{U}$, $\mathbf{L}$ by Gauss Elimination

Let $\mathbf{A}^{(k)} = \mathbf{M}_k\mathbf{A}^{(k-1)}$, $\mathbf{A}^{(0)} = \mathbf{A}$. Note $\mathbf{A}^{(k)} = \mathbf{M}_k \cdots \mathbf{M}_2\mathbf{M}_1\mathbf{A}$.

Step k: Choose $\mathbf{M}_k$ such that

$\mathbf{M}_k\mathbf{a}_k^{(k-1)} = [\ a_{1k}^{(k-1)}, \ldots, a_{kk}^{(k-1)}, 0, \ldots, 0\ ]^T$.

- If $a_{kk}^{(k-1)} \neq 0$, let

$$\mathbf{M}_k = \mathbf{I} - \boldsymbol{\tau}^{(k)}\mathbf{e}_k^T, \qquad \boldsymbol{\tau}^{(k)} = \left[0, \ldots, 0, \frac{a_{k+1,k}^{(k-1)}}{a_{kk}^{(k-1)}}, \ldots, \frac{a_{n,k}^{(k-1)}}{a_{kk}^{(k-1)}}\right]^T,$$

$$\mathbf{A}^{(k)} = \mathbf{M}_k\mathbf{A}^{(k-1)} = \begin{bmatrix} a_{11}^{(k-1)} & \cdots & a_{1k}^{(k-1)} & \times & \cdots & \times \\ 0 & \ddots & \vdots & \vdots & & \vdots \\ \vdots & & a_{kk}^{(k-1)} & \vdots & & \vdots \\ \vdots & & 0 & \times & & \times \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \times & \cdots & \times \end{bmatrix}$$

$\mathbf{A}^{(k)} = \mathbf{M}_k\mathbf{A}^{(k-1)} = \mathbf{A}^{(k-1)} - \boldsymbol{\tau}^{(k)}\mathbf{A}^{(k-1)}(k,:)$

- $\mathbf{A}^{(n-1)} = \mathbf{U}$ is upper triangular
- $\mathbf{L} = \mathbf{M}_1^{-1} \ldots \mathbf{M}_{n-1}^{-1} = \mathbf{I} + \sum_{k=1}^{n-1} \boldsymbol{\tau}^{(k)}\mathbf{e}_k^T$

LU decomposition Code

```
function [L,U]= my_lu(A)
n= size(A,1);
L= eye(n); tau= zeros(n,1); U= A;
for k=1:1:n-1,
     rows= k+1:n;
     tau(rows)= U(rows,k)/U(k,k);
     U(rows,rows)= U(rows,rows)- tau(rows)*U(k,rows);
     U(rows,k)= 0;
     L(rows,k)= tau(rows);
end;
```

- complexity: $\mathcal{O}(2n^3/3)$
- works as long as $a_{kk}^{(k-1)}$—the so-called **pivots** —are all nonzero

Example & Exercise

- **Example 1**: Conduct the LU code to decompose

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 4 \end{bmatrix}$$

- **Exercise 1**: Conduct the LU code to decompose

$$\mathbf{A} = \begin{bmatrix} 2 & 2 & -2 \\ 4 & 7 & 7 \\ 6 & 18 & 22 \end{bmatrix}$$

PLU Decomposition

PLU decomposition : Given any matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, find $\mathbf{P}, \mathbf{L}, \mathbf{U} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{PA} = \mathbf{LU},$$

where

$\mathbf{P} \in \mathbb{R}^{n \times n}$ is a permutation matrix;

$\mathbf{L} \in \mathbb{R}^{n \times n}$ is unit lower triangular ;

$\mathbf{U} \in \mathbb{R}^{n \times n}$ is upper triangular.

Partial Pivoting

Given $\mathbf{A}^{(k-1)}, k = 1, \dots, n-1,\ \mathbf{A}^{(0)} = \mathbf{A}$.

1. Find $\mathrm{piv}(k) = \arg\max_{j\in[k,n]} \left|\mathbf{A}^{(k-1)}(j,k)\right|$
2. Let $\boldsymbol{\Pi}_k \in \mathbb{R}^{n\times n}$ be the interchange permutation that swaps row k and row $\mathrm{piv}(k)$ of $\mathbf{I}$
3. Determine the Gauss Transformation $\mathbf{M}_k = \mathbf{I}_n - \tau^{(k)}\mathbf{e}_k^T$, where

$$\boldsymbol{\tau}^{(k)} = \left[\begin{array}{c} \mathbf{0}_k \\ \left(\boldsymbol{\Pi}_k\mathbf{A}^{(k-1)}\right)(k+1:n,k) / \left(\boldsymbol{\Pi}_k\mathbf{A}^{(k-1)}\right)(k,k) \end{array} \right]$$

4. $\boldsymbol{A}^{(k)} = \mathbf{M}_k\left(\boldsymbol{\Pi}_k\mathbf{A}^{(k-1)}\right)$ (which satisfies $\boldsymbol{A}^{(k)}(k+1:n,k) = 0$)

Results

- $\mathbf{P} = \boldsymbol{\Pi}_{n-1}\cdots\boldsymbol{\Pi}_1$
- $\mathbf{U} = \mathbf{A}^{(n-1)}$
- $\mathbf{L} = \mathbf{I} + \sum_{k=1}^{n-1}\tilde{\tau}^{(k)}\mathbf{e}_k^T$ with $\tilde{\tau}^{(k)} = \boldsymbol{\Pi}_{n-1}\cdots\boldsymbol{\Pi}_{k+1}\tau^{(k)}$

PLU decomposition with Partial Pivoting Code

Find **L**, **U** s.t. **PA** = **LU** in MATLAB

```
for k=1:n-1
     [~ ,piv]=max(abs(A(k:n,k))); piv=piv-1+k;
     A([k piv],:)=A([piv k],:)  % swap row k and row piv
% If A(k,k)=0, then nothing to do
     if A(k,k)~ =0
         rows=k+1:n
         A(rows,k)=A(rows,k)/A(k,k) % compute l^(k)(k+1 : n)
         A(rows,rows)=A(rows,rows)-A(rows,k)*A(k,rows)
    end
end
```

- In the above code:
  $A(k, k : n) = \mathbf{U}(k, k : n),\ \forall k = 1, 2, \ldots, n$
  $A(k+1 : n, k) = \mathbf{L}(k+1 : n, k),\ \forall k = 1, 2, \ldots, n-1$
- $\mathcal{O}(n^2)$ comparisons for searching for the pivots
- $\mathcal{O}(2n^3/3)$ flops

Example & Exercise

- **Example 2**: Conduct the PLU code to decompose

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 4 \end{bmatrix}$$

- **Exercise 2**: Conduct the PLU code to decompose

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 4 & 17 \\ 3 & 6 & -12 & 3 \\ 2 & 3 & -3 & 2 \\ 0 & 2 & -2 & 6 \end{bmatrix}$$

LDM Decomposition

LDM decomposition: given $\mathbf{A} \in \mathbb{R}^{n \times n}$, find matrices $\mathbf{L}, \mathbf{D}, \mathbf{M} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{M}^T,$$

$\mathbf{L}$, $\mathbf{M}$ is **unit** lower/upper triangular, $\mathbf{D} = \mathrm{Diag}(d_1, \ldots, d_n)$

Idea : examine $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{M}^T$ column by column:

$$\mathbf{A}(:, j) = \mathbf{A}\mathbf{e}_j = \mathbf{L}\mathbf{v}$$

$$\mathbf{v} = \mathbf{D}\mathbf{M}^T\mathbf{e}_j$$

Observations:

1. $v_i = d_j m_{ji}$;
2. $v_i = 0, i = j + 1, \ldots, n$;
3. $\mathbf{A}(:, j) = \mathbf{L}\mathbf{v}$ can be partitioned as

$$\begin{aligned} \begin{bmatrix} \mathbf{A}(1:j, j) \\ \mathbf{A}(j+1:n, j) \end{bmatrix} &= \begin{bmatrix} \mathbf{L}(1:j, 1:j) & \mathbf{0} \\ \mathbf{L}(j+1:n, 1:j) & \mathbf{L}(j+1:n, j+1:n) \end{bmatrix} \begin{bmatrix} \mathbf{v}(1:j) \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{L}(1:j, 1:j)\mathbf{v}(1:j) \\ \mathbf{L}(j+1:n, 1:j)\mathbf{v}(1:j) \end{bmatrix} \end{aligned} \quad (1)$$

Solving LDM Decomposition

Recall from the last page that

$$\mathbf{A}(1:j,j) = \mathbf{L}(1:j,1:j)\mathbf{v}(1:j) \tag{2}$$

$$\mathbf{A}(j+1:n,j) = \mathbf{L}(j+1:n,1:j)\mathbf{v}(1:j) \tag{3}$$

Idea: Recursively find each column of **L**, each row of **M**, and each diagonal entry of **D**

For $j = 1 : n$

Step 1. Form $\mathbf{L}(1:j,1:j)$ using the columns $1,\dots,j-1$ of **L** and $\mathbf{L}(j,j) = 1$

Step 2. Solve the linear system $\mathbf{A}(1:j,j) = \mathbf{L}(1:j,1:j)\mathbf{v}(1:j)$ for $\mathbf{v}(1:j)$

Step 3. Compute $\mathbf{L}(j+1:n,j)$ according to (not needed for $j = n$)

$$\mathbf{L}(j+1:n,j) = (\mathbf{A}(j+1:n,j) - \mathbf{L}(j+1:n,1:j-1)\mathbf{v}(1:j-1))/\mathbf{v}(j)$$

Step 4. Set $d_j = v_j$, $m_{ji} = v_i/d_i$ for all $i = 1,\dots,j-1$

- Complexity: $\mathcal{O}(2n^3/3)$ (same as the previous LU code)

LDL Decomposition for Symmetric Matrices

If $\mathbf{A}$ is real symmetric, then the LDM decomposition may be reduced to

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T.$$

Solving LDL :

- recall that in the previous LDM decomposition, the key is to find the unknown
$$\mathbf{v} = \mathbf{D}\mathbf{M}^T\mathbf{e}_j$$
by solving $\mathbf{A}(1:j,j) = \mathbf{L}(1:j,1:j)\mathbf{v}(1:j)$ by forward substitution.
- Now, since $\mathbf{M} = \mathbf{L}$,
$$v_i = d_i\ell_{ji}.$$
Finding $\mathbf{v}$ is much easier and there is no need to run forward substitution.
  - With the knowledge of the columns $1,\ldots,j-1$ of $\mathbf{L}$, we can easily find $v_i = d_i\ell_{ji}$, $i = 1,\ldots,j-1$
  - Then, find v_j by $v_j = \mathbf{A}(j,j) - \mathbf{L}(j,1:j-1) * \mathbf{v}(1:j-1)$

An LDL Decomposition Code

```
function [L,D]= my_ldl(A)
n= size(A,1);
L= eye(n); d= zeros(n,1); M= eye(n);
v= zeros(n,1);
for j=1:n,
     v(1:j)= ForwardSubstitution(L(1:j,1:j),A(1:j,j));
     v(1:j-1)= L(j,1:j-1)'.*d(1:j-1);
     v(j)= A(j,j)- L(j,1:j-1)*v(1:j-1);
% replace for_subs.
     d(j)= v(j);
     for i=1:j-1,
         M(j,i)= v(i)'/d(i);
     end;
     L(j+1:n,j)= (A(j+1:n,j)-L(j+1:n,1:j-1)*v(1:j-1))/v(j);
end;
D= diag(d);
```

- complexity: $\mathcal{O}(n^3/3)$, half of LU or LDM

Example & Exercise

- **Example 3**: Conduct the LDL code to decompose

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 4 \end{bmatrix}$$

- **Exercise 3**: Conduct the LDL code to decompose

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 5 \\ 4 & 18 & 26 \\ 5 & 26 & 30 \end{bmatrix}$$

Cholesky Decomposition for Positive Definite Matrices

Cholesky decomposition : given a positive definite $\mathbf{A} \in \mathbb{S}^n$, decompose $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{G}\mathbf{G}^T,$$

where $\mathbf{G} \in \mathbb{R}^{n \times n}$ is lower triangular with positive diagonal elements.
Idea : if $\mathbf{A}$ is symmetric and PD, then its LDL decomposition

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$$

uniquely exist and has $d_i > 0$ for all $i = 1, \ldots, n$. Putting $\mathbf{G} = \mathbf{L}\mathbf{D}^{\frac{1}{2}}$ yields the Cholesky factorization.
Example 4 : Use the Algorithm 4.2.1 in the textbook to decompose the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 12 \\ 3 & 12 & 27 \end{bmatrix}$$

- complexity: $\mathcal{O}(n^3/3)$, similar to LDL

Extensions – Vandermonde system

A matrix $\mathbf{A} \in \mathbb{R}^{(n+1)\times(n+1)}$ is a Vandermonde matrix is

$$\mathbf{A} = \mathbf{A}(x_0, \ldots, x_n) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_n \\ \vdots & \vdots & & \vdots \\ x_0^n & x_1^n & \cdots & x_n^n \end{bmatrix} (\textit{Vandermonde matrix})$$

- If any $x_i \neq x_j$, $\mathbf{A}$ is nonsingular
- Using **LU** decomposition, solving $\mathbf{Az} = \mathbf{b}$ requires $O(n^3)$ flops
- A more efficient method with $O(n^2)$ complexity is in Chapter 4.6 of **[Golub-van-Loan'13]** ((Refer to it for details))

Extensions – Toeplitz system

A matrix $\mathbf{A} \in \mathbb{R}^{(n)\times(n)}$ is a Toeplitz matrix is

$$\mathbf{A} = \begin{bmatrix} h_0 & h_{-1} & \dots & \dots & h_{-n+1} \\ h_1 & h_0 & h_{-1} & & \vdots \\ \vdots & h_1 & h_0 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & h_{-1} \\ h_{n-1} & \dots & \dots & h_1 & h_0 \end{bmatrix} (\mathit{Toeplitz\ \ Matrix})$$

or $a_{ij} = h_{i-j}$ for all i, j

- See Chapter 4.7 in **[Golub-van-Loan'13]** for a more efficient method with $O(n^2)$ complexity to solve $\mathbf{Ax} = \mathbf{b}$

Iterative Methods for Linear Systems

- solving linear systems $\mathbf{Ax} = \mathbf{b}$ via **LU** requires $\mathcal{O}(n^3)$
- $\mathcal{O}(n^3)$ is too much for large-scale linear systems
- the motivation behind iterative methods is to seek less expensive ways to find an (approximate) linear system solution

The Key Insight of Iterative Methods

- assume $a_{ii} \neq 0$ for all i
- observe

$$\begin{aligned} \mathbf{b} = \mathbf{A}\mathbf{x} \quad & \Longleftrightarrow \quad b_i = a_{ii} x_i + \sum_{j \neq i} a_{ij} x_j, \quad i = 1, \ldots, n \\ & \Longleftrightarrow \quad x_i = \left(b_i - \sum_{j \neq i} a_{ij} x_j \right) / a_{ii}, \quad i = 1, \ldots, n \qquad (\dagger) \end{aligned}$$

- example

$$\begin{aligned} x_1 &= (b_1 - a_{12} x_2 - a_{13} x_3) / a_{11}, \\ x_2 &= (b_2 - a_{21} x_1 - a_{23} x_3) / a_{22}, \\ x_3 &= (b_3 - a_{31} x_1 - a_{32} x_2) / a_{33}. \end{aligned}$$

- idea: find an $\mathbf{x}$ that fulfils the equations in $(\dagger)$

Jacobi Iterations

input: a starting point $\mathbf{x}^{(0)}$
for $k = 0, 1, 2, \ldots$
$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii}$, for $i = 1, \ldots, n$
end

- complexity per iteration: $\mathcal{O}(n^2)$ for dense $\mathbf{A}$
- Let $\mathbf{D}_A = \mathrm{diag}(\mathrm{diag}(\mathbf{A}))$, $\mathbf{L}_A = \mathrm{tril}(\mathbf{A}, \text{-}1)$, and $\mathbf{U}_A = \mathrm{triu}(\mathbf{A}, 1)$ The Jacobi iterations has the form

$$\mathbf{M}_{\mathrm{J}} \mathbf{x}^{(k)} = \mathbf{N}_{\mathrm{J}} \mathbf{x}^{(k-1)} + \mathbf{b}$$

  where $\mathbf{M}_{\mathrm{J}} = \mathbf{D}_A$ and $\mathbf{N}_{\mathrm{J}} = -(\mathbf{L}_A + \mathbf{U}_A)$.
- convergence properties of Jacobi iterations
  - $\left\| \mathbf{x}^{(k)} - \mathbf{x}^{\star} \right\| \leq \left\| \mathbf{M}_{\mathrm{J}}^{-1} \mathbf{N}_{\mathrm{J}} \right\|^k \left\| \mathbf{x}^{(0)} - \mathbf{x}^{\star} \right\|$ with $\mathbf{A}\mathbf{x}^{\star} = \mathbf{b}$
  - it converges if the diagonal elements a_{ii}'s are "dominant" compared to the off-diagonal elements; see Theorem 11.2.2 in **[Golub-van-Loan'13]** for details

Gauss-Seidel Iterations

input: a starting point $\mathbf{x}^{(0)}$
for $k = 0, 1, 2, \ldots$
 for $i = 1, 2, \ldots, n$

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

 end
end

- use the most recently available $\mathbf{x}$ to perform update
- The Gauss-Seidel iteration is equivalent to

$$\mathbf{M}_{\mathrm{GS}} \mathbf{x}^{(k)} = \mathbf{N}_{\mathrm{GS}} \mathbf{x}^{(k-1)} + \mathbf{b}$$

 where $\mathbf{M}_{\mathrm{GS}} = \mathbf{D}_A + \mathbf{L}_A$ and $\mathbf{N}_{\mathrm{GS}} = -\mathbf{U}_A$.
- convergence properties of Gauss-Seidel iterations
  - $\left\| \mathbf{x}^{(k)} - \mathbf{x}^{\star} \right\| \leq \left\| \mathbf{M}_{\mathrm{GS}}^{-1} \mathbf{N}_{\mathrm{GS}} \right\|^{k} \left\| \mathbf{x}^{(0)} - \mathbf{x}^{\star} \right\|$
  - $\mathbf{A}$ is symmetric PD; see Theorem 11.2.3 in **[Golub-van-Loan'13]**

Successive Over-Relaxation (SOR):

The SOR iteration has the following term

$$\mathbf{M}_\omega \mathbf{x}^{(k)} = \mathbf{N}_{\mathrm{GS}} \mathbf{x}^{(k-1)} + \mathbf{b}$$

where $\mathbf{M}_\omega = \frac{1}{\omega}\mathbf{D}_A + \mathbf{L}_A$, $\mathbf{N}_\omega = (\frac{1}{\omega} - 1)\mathbf{D}_A + \mathbf{U}_A$, $0 < \omega < 2$.

- Idea: choosing proper ω to reduce $\left\|\mathbf{M}_\omega^{-1}\mathbf{N}_\omega\right\|$
- $\omega = 1$, Gauss-Seidel Iterations

input: a starting point $\mathbf{x}^{(0)}$
for $k = 0, 1, 2, \ldots$
  for $i = 1, 2, \ldots, n$
    $x_i^{(k+1)} = \omega \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) / a_{ii} + (1 - \omega) \cdot x_i^{(k)}$
  end
end

Exercise

- **Exercise 4**: Randomly generate a symmetric, positive definite matrix $\mathbf{A} \in \mathbb{R}^{100\times100}$ and a random vector $\mathbf{b} \in \mathbb{R}^{100\times1}$. Use Jacobi iterations, Gauss-Seidel iterations and SOR to solve $\mathbf{Ax} = \mathbf{b}$ and compare their convergence performance.

Extensions – Discretized Poisson Equation in Two Dimensions

The discretized Poisson equation in two dimensions: Suppose $F(x, y)$ is defined on R, we wish to find a function u that satisfies

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -F(x, y) \longrightarrow \mathbf{A}u = f$$

on $R = \left\{(x, y) : \alpha_x \leq x \leq \beta_x, \alpha_y \leq y \leq \beta_y\right\}$. **A** is a sparse symmetric positive definite matrix.

Refer to Chapter 4.8, Chapter 11.2 in **[Golub-van-Loan'13]**

- how to transform the differential equation into $\mathbf{A}u = f$?
- how to use the above iterative methods to solve $\mathbf{A}u = f$?

References

[Golub-van-Loan'13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4rd edition, JHU Press, 2013.