Matrix Computations

Chapter 2 Linear systems and LU decomposition

Section 2.1 Triangular Systems and LU Decomposition

Jie Lu
ShanghaiTech University

System of Linear Equations

Consider the system of linear equations (linear system)

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

- $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$ are given
- $\mathbf{x} \in \mathbb{R}^n$ is the solution to the system
- Extension to the complex case is simple

Solving the Linear System

Goal: Find the solution to $\mathbf{Ax} = \mathbf{b}$ in a numerically efficient way

- The problem is very easy if $\mathbf{A}$ is nonsingular and $\mathbf{A}^{-1}$ is known
    - How to compute $\mathbf{A}^{-1}$ efficiently?

- Solving the linear system may be easier in some special cases, e.g., triangular $\mathbf{A}$, orthogonal $\mathbf{A}$, circulant $\mathbf{A}$

Lower Triangular Systems

Example: Consider the 3×3 lower triangular system

$$\begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

If $\ell_{11}, \ell_{22}, \ell_{33} \neq 0$, then

- The first equation gives $x_1 = b_1/\ell_{11}$
- The second equation gives $x_2 = (b_2 - \ell_{21}x_1)/\ell_{22}$. Then, substituting x_1 yields x_2
- The third equation gives $x_3 = (b_3 - \ell_{31}x_1 - \ell_{32}x_2)/\ell_{33}$. Then, substituting x_1, x_2 yields x_3

Question: What happens if some of $\ell_{11}, \ell_{22}, \ell_{33}$ is zero?

Forward Substitution

For a general lower triangular system $\mathbf{Lx} = \mathbf{b}$ with $\mathbf{L} \in \mathbb{R}^{n \times n}$,

$$x_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij} x_j \right) \Big/ \ell_{ii}, \qquad \text{for } i = 1, 2, \ldots, n$$

The algorithm is called Forward Substitution for solving $\mathbf{Lx} = \mathbf{b}$

Forward substitution in MATLAB form:

```
function x= ForwardSubstitution(L,b)
n= length(b);
x= zeros(n,1);
x(1)= b(1)/L(1,1);
for i=2:1:n
     x(i)=(b(i)-L(i,1:i-1)*x(1:i-1))/L(i,i);
end
```

- Complexity: n^2 flops
- You may overwrite b with the solution to save memory

Upper Triangular Systems

Example: Consider the 3×3 upper triangular system

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

If $u_{11}, u_{22}, u_{33} \neq 0$, then

- The third equation gives $x_3 = b_3/u_{33}$
- The second equation gives $x_2 = (b_2 - u_{23}x_3)/u_{22}$. Then, substituting x_3 yields x_2
- The first equation gives $x_1 = (b_1 - u_{12}x_2 - u_{13}x_3)/u_{11}$. Then, substituting x_3, x_2 yields x_1

Question: What happens if some of u_{11}, u_{22}, u_{33} is zero?

Backward Substitution

For a general upper triangular system $\mathbf{Ux} = \mathbf{b}$ with $\mathbf{U} \in \mathbb{R}^{n \times n}$,

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \Big/ u_{ii}, \qquad \text{for } i = n, n-1, \ldots, 1$$

The algorithm is called Backward Substitution for solving $\mathbf{Ux} = \mathbf{b}$

Backward substitution in MATLAB form:

```
function x= BackwardSubstitution(U,b)
n= length(b);
x= zeros(n,1);
x(n)= b(n)/U(n,n);
for i= n-1:-1:1,
     x(i)= (b(i)- U(i,i+1:n)*x(i+1:n))/U(i,i);
end
```

- complexity: n^2 flops
- You may overwrite b with the solution to save memory

Column-Oriented Representation

Example: Consider the 3×3 lower triangular system

$$\begin{bmatrix} 2 & 0 & 0 \\ 1 & 5 & 0 \\ 7 & 9 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 5 \end{bmatrix}$$

From the first equation, we have $x_1 = 6/2 = 3$. Then the remaining two equations can be expressed as

$$\begin{bmatrix} 5 & 0 \\ 9 & 8 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} - x_1 \begin{bmatrix} 1 \\ 7 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} - 3 \begin{bmatrix} 1 \\ 7 \end{bmatrix} = \begin{bmatrix} -1 \\ -16 \end{bmatrix}$$

For a general $n \times n$ lower triangular system $\mathbf{Lx} = \mathbf{b}$, x_1 can be directly obtained. Then, form a $(n-1) \times (n-1)$ system according to

$$\mathbf{L}(2:n, 2:n)\mathbf{x}(2:n) = \mathbf{b}(2:n) - x_1 \cdot \mathbf{L}(2:n, 1)$$

Solving this new system for x_2 is simple
Repeated the process for the $(n-1) \times (n-1)$ system

Column-Oriented Representation (cont'd)

Column-Oriented Forward Substitution in MATLAB form:

```
for j=1:n-1
     b(j)=b(j)/L(j,j);
% Compute the first element of the solution to the
latest system
     b(j+1:n)=b(j+1:n)-b(j)*L(j+1:n,j);
% The right-hand side of the updated system
end
b(n)=b(n)/L(n,n);
% b has been overwritten by the solution
```

Complexity: n^2 flops

Exercise: Derive Column-Oriented Backward Substitution for solving upper triangular systems

- See Section 3.1 of the textbook

Multi-Right-Hand-side Problems

Compute the solution $\mathbf{X} \in R^{n \times q}$ to

$$\mathbf{LX} = \mathbf{B}$$

where $\mathbf{L} \in R^{n \times n}$ is lower triangular and $\mathbf{B} \in \mathbb{R}^{n \times q}$

It amounts to solving q triangular systems, but we can do Block Back Substitution. Partitioning $\mathbf{LX} = \mathbf{B}$ into

$$\begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{L}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{L}_{N1} & \mathbf{L}_{N2} & \cdots & \mathbf{L}_{NN} \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_N \end{bmatrix} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_N \end{bmatrix}$$

Multi-Right-Hand-side Problems (cont'd)

Solve the triangular system $\mathbf{L}_{11}\mathbf{X}_1 = \mathbf{B}_1$ for $\mathbf{X}_1$. Then, remove $\mathbf{X}_1$ from block equations 2 through N:

$$\begin{bmatrix} \mathbf{L}_{22} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{L}_{32} & \mathbf{L}_{33} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{L}_{N2} & \mathbf{L}_{N3} & \cdots & \mathbf{L}_{NN} \end{bmatrix} \begin{bmatrix} \mathbf{X}_2 \\ \mathbf{X}_3 \\ \vdots \\ \mathbf{X}_N \end{bmatrix} = \begin{bmatrix} \mathbf{B}_2 \\ \mathbf{B}_3 \\ \vdots \\ \mathbf{B}_N \end{bmatrix} - \begin{bmatrix} \mathbf{L}_{21} \\ \mathbf{L}_{31} \\ \vdots \\ \mathbf{L}_{N1} \end{bmatrix} \mathbf{X}_1$$

Repeat this process to the above system

```
pseudo code, not Matlab
for j=1:N
    Solve L_jj X_j = B_j;
    for i=j+1:N
        B_i = B_i - L_ij X_j;
    end
end
```

LU Decomposition

A "high-level' algebraic description of Gaussian Elimination

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

$$\mathbf{A} = \mathbf{L}\mathbf{U}, \quad \text{where}$$

$\mathbf{L} \in \mathbb{R}^{n \times n}$ is unit lower triangular ($\ell_{ii} = 1$ for all i)

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

Suppose $\mathbf{A}$ has an LU decomposition. Then, solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be recast as solving two triangular systems

1. solve $\mathbf{L}\mathbf{z} = \mathbf{b}$ for $\mathbf{z}$
2. solve $\mathbf{U}\mathbf{x} = \mathbf{z}$ for $\mathbf{x}$

Questions:

- Does LU decomposition always exist?
- How to find $\mathbf{L}$ and $\mathbf{U}$?

Gauss Transformations

A matrix description of the zeroing process in Gaussian elimination

Example: Suppose $x_1 \neq 0$ and $\tau = x_2/x_1$. Then,

$$\begin{bmatrix} 1 & 0 \\ -\tau & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ 0 \end{bmatrix}$$

Extension to $\mathbb{R}^n$: Let $\mathbf{x} \in \mathbb{R}^n$ s.t. $x_k \neq 0$ for some $1 \leq k \leq n$ and $\tau_i = \frac{x_i}{x_k}$ $\forall i = k+1, \ldots, n$. Then,

$$\underbrace{\begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -\tau_{k+1} & 1 & & \\ & & \vdots & & \ddots & \\ & & -\tau_n & & & 1 \end{bmatrix}}_{\mathbf{M}_k} \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Gauss Transformations (cont'd)

For $k = 1, \ldots, n$,

$$\mathbf{M}_k \mathbf{x} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -\tau_{k+1} & 1 & & \\ & & \vdots & & \ddots & \\ & & -\tau_n & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad \tau_i = \frac{x_i}{x_k}$$

$$\mathbf{M}_k = \mathbf{I} - \boldsymbol{\tau} \mathbf{e}_k^T, \qquad \boldsymbol{\tau} = [0, \ldots, 0, \tau_{k+1}, \ldots, \tau_n]^T$$

Multiplication by a Gauss Transformation

Let $\mathbf{M}_k = \mathbf{I} - \boldsymbol{\tau}\mathbf{e}_k^T$ be a Gauss transformation and $\mathbf{C} \in \mathbb{R}^{n\times r}$

$$\mathbf{M}_k\mathbf{C} = (\mathbf{I} - \boldsymbol{\tau}\mathbf{e}_k^T)\mathbf{C} = \mathbf{C} - \boldsymbol{\tau}(\mathbf{e}_k^T\mathbf{C}) = \mathbf{C} - \boldsymbol{\tau}\mathbf{C}(k,:) \quad \text{(outer product)}$$

Here, $\boldsymbol{\tau}$ does not necessarily depend on $\mathbf{C}$. Since $\boldsymbol{\tau}(1:k) = \mathbf{0}$, only $\mathbf{C}(k+1:n,:)$ is affected

```
for i= k+1:n
     C(i,:)= C(i,:)-tau(i)*C(k,:)
end
```

Complexity: $2(n-k)r$ flops

Exercise: Compute $(\mathbf{I} - \boldsymbol{\tau}\mathbf{e}_1^T)\mathbf{C}$ with

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \qquad \boldsymbol{\tau} = \begin{bmatrix} 0 \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

Upper Triangularizing

Problem : Find Gauss transformations $\mathbf{M}_1, \ldots, \mathbf{M}_{n-1} \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{M}_{n-1} \cdots \mathbf{M}_2 \mathbf{M}_1 \mathbf{A} = \mathbf{U}, \quad \mathbf{U} \text{ is upper triangular}$$

Step 1: Choose $\mathbf{M}_1$ s.t. $\mathbf{M}_1 \mathbf{a}_1 = [\ a_{11}, 0, \ldots, 0\ ]^T$

- If $a_{11} \neq 0$, let

$$\mathbf{M}_1 = \mathbf{I} - \boldsymbol{\tau}^{(1)} \mathbf{e}_1^T, \qquad \boldsymbol{\tau}^{(1)} = [\ 0, a_{21}/a_{11}, \ldots, a_{n1}/a_{11}\ ]^T.$$

$$\mathbf{M}_1 \mathbf{A} = \begin{bmatrix} a_{11} & \times & \ldots & \times \\ 0 & \times & \ldots & \times \\ \vdots & \vdots & & \vdots \\ 0 & \times & \ldots & \times \end{bmatrix}$$

Upper Triangularizing (cont'd)

Step 2: Set $\mathbf{A}^{(1)} = \mathbf{M}_1\mathbf{A}$

Choose $\mathbf{M}_2$ s.t. $\mathbf{M}_2\mathbf{a}_2^{(1)} = [\ a_{12}^{(1)}, a_{22}^{(1)}, 0, \ldots, 0\ ]^T$

- If $a_{22}^{(1)} \neq 0$, let

$$\mathbf{M}_2 = \mathbf{I} - \boldsymbol{\tau}^{(2)}\mathbf{e}_2^T, \qquad \boldsymbol{\tau}^{(2)} = [\ 0, 0, a_{32}^{(1)}/a_{22}^{(1)}, \ldots, a_{n,2}^{(1)}/a_{22}^{(1)}\ ]^T$$

$$\mathbf{M}_2\mathbf{A}^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \times & \ldots & \times \\ 0 & a_{22}^{(1)} & \times & \ldots & \times \\ \vdots & 0 & \times & & \times \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \times & \ldots & \times \end{bmatrix}$$

- Applying $\mathbf{M}_2$ to $\mathbf{A}^{(1)}$ does not change the first column of $\mathbf{A}^{(1)}$

$$\mathbf{M}_2\mathbf{a}_1^{(1)} = \mathbf{a}_1^{(1)} - \boldsymbol{\tau}^{(2)} \underbrace{a_{21}^{(1)}}_{=0} = \mathbf{a}_1^{(1)}$$

Upper Triangularizing (cont'd)

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

Step k: Choose $\mathbf{M}_k$ s.t. $\mathbf{M}_k \mathbf{a}_k^{(k-1)} = \left[a_{1k}^{(k-1)}, \ldots, a_{kk}^{(k-1)}, 0, \ldots, 0\right]^T$

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

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

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

- Applying $\mathbf{M}_k$ to $\mathbf{A}^{(k-1)}$ does not change the first $k-1$ columns of $\mathbf{A}^{(k-1)}$
- $\mathbf{A}^{(n-1)} = \mathbf{U}$ is upper triangular

Upper Triangularizing (cont'd)

Example: Upper triangularize $\mathbf{A} = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix}$

Computation of $\mathbf{L}$

When the pivots $a_{kk}^{(k-1)} \neq 0$ for all $k = 1, \ldots, n-1$,

$$\mathbf{U} = \mathbf{M}_{n-1} \cdots \mathbf{M}_2 \mathbf{M}_1 \mathbf{A} \text{ is upper triangular}$$

Find $\mathbf{L}$ based on $\mathbf{U}$

Facts: Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ be two lower (upper) triangular matrices. Then,

1. $\mathbf{AB}$ is lower (upper) triangular. In addition, if $\mathbf{A}$, $\mathbf{B}$ have unit diagonal entries, then $\mathbf{AB}$ is unit lower (upper) triangular.
2. $\det(\mathbf{A}) = \prod_{i=1}^{n} a_{ii}$.
3. If $\mathbf{A}$ is nonsingular, $\mathbf{A}^{-1}$ is lower (upper) triangular with $[\mathbf{A}^{-1}]_{ii} = 1/a_{ii}$.

Since every $\mathbf{M}_k$ is unit lower triangular (invertible),

$$\mathbf{L} = \mathbf{M}_1^{-1} \mathbf{M}_2^{-1} \cdots \mathbf{M}_{n-1}^{-1}$$

satisfies $\mathbf{A} = \mathbf{LU}$ and is unit lower triangular

Proof of Facts

Proof of Facts (cont'd)

Proof of Facts (cont'd)

A Naive Implementation of LU (Don't Use It)

```
function [L,U]= naive_LU(A)
n= size(A,1);
L= eye(n); tau= zeros(n,1); U= A;
for k=1:n-1,
     rows= k+1:n;
     tau(rows)= U(rows,k)/U(k,k);
     M= eye(n); M(rows,k)= -tau(rows);
     U= M*U;            % compute A^(k) = M_k A^(k-1)
     L= L*inv(M);
    % to eventually obtain L = M_1^-1 M_2^-1 ... M_{n-1}^-1
end
```

- The code treats each $\mathbf{A}^{(k)} = \mathbf{M}_k \mathbf{A}^{(k-1)}$ as a general matrix multiplication process, requiring $\mathcal{O}(n^3)$ flops. Can we utilize the structure of $\mathbf{M}_k$ to reduce complexity?
- The code calls for $n-1$ matrix inversion to compute $\mathbf{L}$. Why not directly compute the inverse of $\mathbf{A}$?

Computation of $\mathbf{L}$ (cont'd)

Fact: $\mathbf{M}_k^{-1} = \mathbf{I} + \boldsymbol{\tau}^{(k)}\mathbf{e}_k^T$ for each $k = 1, \ldots, n-1$

Verification: Since $[\boldsymbol{\tau}^{(k)}]_k = 0$,

$$\begin{aligned}(\mathbf{I} + \boldsymbol{\tau}^{(k)}\mathbf{e}_k^T)\mathbf{M}_k &= (\mathbf{I} + \boldsymbol{\tau}^{(k)}\mathbf{e}_k^T)(\mathbf{I} - \boldsymbol{\tau}^{(k)}\mathbf{e}_k^T) \\ &= \mathbf{I} + \boldsymbol{\tau}^{(k)}\mathbf{e}_k^T - \boldsymbol{\tau}^{(k)}\mathbf{e}_k^T + \boldsymbol{\tau}^{(k)} \underbrace{\mathbf{e}_k^T\boldsymbol{\tau}^{(k)}}_{=0} \mathbf{e}_k^T = \mathbf{I}\end{aligned}$$

Using the same spirit,

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

An Improved LU Code (Still Not Used by MATLAB)

```
function [L,U]= better_LU(A)
n= size(A,1);
L= eye(n); tau= zeros(n,1); U= A;
for k=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: $\mathcal{O}(2n^3/3)$
- Again, need nonzero pivots $a_{kk}^{(k-1)}$

Existence of LU Decomposition

Theorem

A matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ *has an LU decomposition if every (leading) principal submatrix* $\mathbf{A}(1:k, 1:k)$ *satisfies*

$$\det(\mathbf{A}(1:k, 1:k)) \neq 0$$

for $k = 1, 2, \ldots, n-1$*. If the LU decomposition of* $\mathbf{A}$ *exists and* $\mathbf{A}$ *is nonsingular, then the LU decomposition is unique and* $\det(\mathbf{A}) = u_{11} \cdots u_{nn}$*.*

Proof

Proof (cont'd)

Proof (cont'd)