Lecture 8

§1 LU Factorization by Gaussian elimination

1. Gaussian elimination （高斯消元法）

To reduce a general linear system $\boldsymbol{Ax} = \boldsymbol{b}$ to upper triangular form, we first choose $\boldsymbol{M}_1$ — with a_{11} as pivot – to set the first column of $\boldsymbol{A}$ below the first row to zero:

- The system becomes $\boldsymbol{M}_1\boldsymbol{Ax} = \boldsymbol{M}_1\boldsymbol{b}$; the solution is not changed.

Next, we build $\boldsymbol{M}_2$ — using a_{22} as pivot — to set the second column of $\boldsymbol{M}_1\boldsymbol{A}$ below the second row to zero:

- New system: $\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{Ax} = \boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b}$; the solution is still not changed.

This process continues for each successive column until all subdiagonal entries have been set to zero.

- The resulting upper triangular linear system is given by:

$$\boldsymbol{M}_{n-1} \cdot \ldots \cdot \boldsymbol{M}_1\boldsymbol{Ax} = \boldsymbol{M}_{n-1} \cdot \ldots \cdot \boldsymbol{M}_1\boldsymbol{b} \implies \underbrace{\boxed{\boldsymbol{MA}}}_{\text{upper triangular}}\boldsymbol{x} = \boldsymbol{Mb}.$$

  This system can be solved by back-substitution to obtain a solution to the original linear system $\boldsymbol{Ax} = \boldsymbol{b}$.

- This procedure is called Gaussian elimination.

通过 Gaussian elimination，我们

① 依次在等式两侧左乘相应的 elementary elimination matrix

② 直到左侧化为上三角矩阵

$$M_{n-1} \cdots M_1 A x = M_{n-1} \cdots M_1 b \quad （顺序为由 n-1 到 1）$$

$$\Rightarrow MAx = Mb$$

其中 $MA = U$ 为 upper triangular matrix

③ 用 back-substitution 得到最终结果

$$\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & & & \\ * & 1 & & \\ * & 0 & 1 & \\ * & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}$$

$$\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & * & 1 & \\ 0 & * & 0 & 1 \end{bmatrix} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix}$$

- Continuation of this process produces an upper triangular matrix.

2. LU Factorization （LU分解）

① 矩阵 $L_i = M_i^{-1}$ 为 unit lower triangular，即 L_i 为 lower triangular 且所有 diagonal entries 为 1

② 由于 $M = M_{n-1} \cdots M_1$，有

$$L = M^{-1} = M_1^{-1} \cdots M_{n-1}^{-1} = L_1 \cdots L_{n-1}$$

为一个 unit lower triangular matrix

注：根据 L_i 的性质，$L = L_1 \cdots L_{n-1}$ 可视作 $L_1, \cdots, L_{n-1}$ 的 union

但 $M = M_{n-1} \cdots M_1$ 不能直接写作 $M_1, \cdots, M_{n-1}$ 的 union

④ 由 Gaussian elimination，有 $MA = U$ 为 upper triangular matrix. 因此有

$A = LU$

其中 L 为一个 unit lower triangular matrix，U 为一个 upper triangular matrix，

因此，Gaussian elimination 通过对 A 的 LU factorization，将 A 分为了两个 factors

3、利用 LU factorization 解 linear system

Having obtained an LU factorization $\boldsymbol{A} = \boldsymbol{LU}$, the equation $\boldsymbol{Ax} = \boldsymbol{b}$ turns into

$$\boldsymbol{LUx} = \boldsymbol{b}$$

which can be solved by:

1. Solving the lower triangular system $\boldsymbol{Ly} = \boldsymbol{b}$ for $\boldsymbol{y}$ using forward-substitution.
2. Then solving the upper triangular system $\boldsymbol{Ux} = \boldsymbol{y}$ for $\boldsymbol{x}$ using back-substitution

- Note that $\boldsymbol{y} = \boldsymbol{Mb}$ coincides with the transformed right-hand side in Gaussian elimination.

⇝ Gaussian elimination and LU factorization are two ways of expressing the same solution procedure.

e.g. 用 Gaussian elimination 求解 linear system，并对 A 进行 LU factorization

We use Gaussian elimination to solve the linear system

$$\boldsymbol{Ax} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix} = \boldsymbol{b}.$$

We first set the subdiagonal entries of the first column of $\boldsymbol{A}$ to zero:

不需要计算

$$\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix},$$

需要计算

$$\boldsymbol{M}_1\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}.$$

Next, we eliminate the subdiagonal entries of the second column of $\boldsymbol{M}_1\boldsymbol{A}$:

$$\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} = \boldsymbol{U},$$

$$\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix} = \boldsymbol{Mb}.$$

We have reduced the original system to the equivalent upper triangular system

$$\boldsymbol{Ux} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix} = \boldsymbol{Mb},$$

which can be solved by back-substitution; we have $\boldsymbol{x} = \begin{bmatrix} -1 & 2 & 2 \end{bmatrix}^\top$.

To write out the LU factorization explicitly:

$$\boldsymbol{L}_1\boldsymbol{L}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} = \boldsymbol{L},$$

$= (M_2M_1)^{-1}$
$= M_1^{-1}M_2^{-1}$

so that

$$\boldsymbol{A} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} = \boldsymbol{LU}.$$

4. LU factorization algorithm

1° 逐步分析

① Step 1: update A via:

$$A' = M_1 A$$

$$= \begin{bmatrix} 1 & \vec{0}' \\ \hat{m}_1' & I \end{bmatrix} \begin{bmatrix} a_{11} & A(1,2:n) \\ * & A(2:n,2:n) \end{bmatrix}$$

(不需要关注 * 是什么)

$$= \begin{bmatrix} a_{11} & A(1,2:n) \\ \vec{0}' & \hat{m}_1' A(1,2:n) + A(2:n,2:n) \end{bmatrix}$$

其中 $\hat{m}_1' = -A(2:n,1)/a_{11}$

② Step 2: update A' via:

$$A^2 = M_2 A'$$

$$= \begin{bmatrix} 1 & 0 & \vec{0}' \\ 0 & 1 & \vec{0}' \\ \vec{0}' & \hat{m}_2' & I \end{bmatrix} \begin{bmatrix} a_{11}' & a_{12}' & A'(1,3:n) \\ 0 & a_{22}' & A'(2,3:n) \\ \vec{0}' & * & A'(3:n,3:n) \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}' & a_{12}' & A'(1,3:n) \\ 0 & a_{22}' & A'(2,3:n) \\ \vec{0}' & \vec{0}' & \hat{m}_2' A'(2,3:n) + A'(3:n,3:n) \end{bmatrix}$$

其中 $\hat{m}_2' = -A'(3:n,2)/a_{22}'$

- 在每个 step 中，我们 overwrite A，使 output 矩阵的 upper triangular part 对应 U
- 为了得到 L，我们需要储存 $\ell_k' = -\hat{m}_k'$ for $k = 1, \cdots, n-1$.
  由于 L 的主对角线全为 1，可以将剩下的 elements (ℓ_k') 储存在 output 矩阵的 lower triangular part

2° Algorithmic procedure

```
for k = 1 to n − 1                   /* loop over columns */
    if a_kk = 0 then stop            /* stop if pivot is zero */
    for i = k + 1 to n               /* compute multipliers
        ℓ_ik = a_ik / a_kk   = -m̂_k'     for current column */
    end
    for j = k + 1 to n
        for i = k + 1 to n           /* apply transformations
            a_ij = a_ij − ℓ_ik a_kj     to remaining submatrix */
        end
    end
end
```

3° Codes

```
function A = lu_plain(A)

n    = size(A,1);
for k = 1:n-1
    ind         = k+1:n;                        → 需要被 update 的行(列)的范围
    A(ind,k)    = A(ind,k)/A(k,k);              → 求出 l_k (直接储存在A中)
    A(ind,ind)  = A(ind,ind)-A(ind,k)*A(k,ind);
end                                             → update right lower block
```

- This code overwrites $\boldsymbol{A}$ with $\boldsymbol{L}$ and $\boldsymbol{U}$.
- We can obtain $\boldsymbol{L}$ and $\boldsymbol{U}$ via:

$$\boldsymbol{U} = \mathtt{triu}(\boldsymbol{A}) \quad \text{and} \quad \boldsymbol{L} = \mathtt{tril}(\boldsymbol{A}, \underbrace{-1}_{\text{不包含主对角线}}) + \mathtt{eye}(\mathtt{size}(\boldsymbol{A})).$$

4° Total flops: $\frac{2}{3}n^3 + O(n^2) = O(n^3)$

证明:

- 在每轮 iteration 中，需要的 flops 为

$$\begin{array}{ll} n-(k+1)+1 & \text{求解 } \boldsymbol{l}_k = -\boldsymbol{m}_k \\ +\ 2(n-k)^2 & \text{更新 lower right block: 1 multiplication + 1 subtraction} \\ \hline =\ (n-k)+2(n-k)^2 & \end{array}$$

- 总共需要 $\sum_{k=1}^{n-1}(n-k)+2(n-k)^2 = \sum_{i=1}^{n-1} 2i^2+i = 2\cdot\frac{n(n+1)(2n+1)}{6} + \frac{n(n-1)}{2} = \frac{2}{3}n^3 + O(n^2)$ 次 flops

注: Solving $A\boldsymbol{x}=\boldsymbol{b}$ (LU factorization of A + forward substitution + back-substitution) 的 total flops 为:
$\frac{2}{3}n^3 + O(n^2) = O(n^3)$

5、比较: inversion 与 factorization

① 计算 A^{-1} 的方法

若将 A^{-1} 写作 $A^{-1} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$，由于 $AA^{-1}=I$，即 $A[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n] = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_n]$，
有 求解 A^{-1} $\Longleftrightarrow$ 求解 $A\boldsymbol{x}_i = \boldsymbol{e}_i$ for all i (n个 linear system)

② 计算 A^{-1} 的 complexity:

- In a naive way (分别解 n 个 linear system): $n\cdot O(n^3) = O(n^4)$
- 利用 LU factorization (对 A LU分解 + n次 forward & backward substitution):
  $O(n^3) + nO(n^2) = O(n^3)$

③ 比较: inversion 与 factorization

- 计算 A^{-1} 的 cost 高于 LU factorization
- 即便要同时解多个方程组 ($A\boldsymbol{x} = \boldsymbol{b}_i, i=1,2,\cdots n$)，LU factorization 仍然更快.
  因为: (1) Initial cost: LU factorization of A < 计算 A^{-1}
  (2) Later cost: forward & backward substitution $\approx$ 计算 $A^{-1}\boldsymbol{b}_i = O(n^2)$

e.g. 计算 $A^{-1}B$ 应先对 A 做 LU分解，再利用 B 的每个 column 进行 forward & backward substitution