Linear System of Equations

Prof. Panruo Wu

NLA Chapter 20

Gaussian Elimination

Gaussian Eliminations (GE) is the method of choice to solve linear systems by hand, and on a computer. (this is actually remarkable).

To make it more flexible and stable, on computer we actually abstracted GE a bit into LU factorization, and added partial pivoting to stabilize it.

Gaussian Elimination -> LU

GE transforms a full linear system into an **upper-triangular** one by applying simple linear transformation on the left.

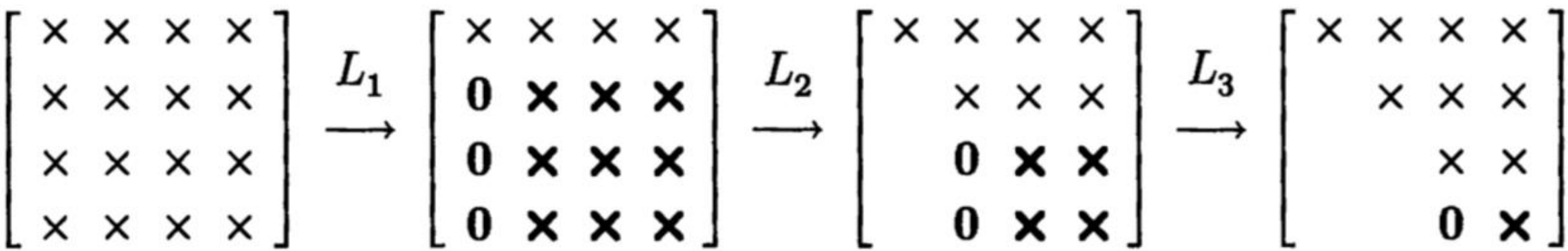
Looks familiar?

It's similar to Householder triangularization for QR; but the transformation is not orthogonal.

The idea is the same: we try to **introduce zeros below diagonal,** column by column. (Think about Gaussian Elimination)

This "elimination" is equivalent to multiplying A by a sequence of lower-triangular matrices L_k on the left:

$$\underbrace{L_{m-1} \cdots L_2 L_1}_{L^{-1}} A = U$$


L_1
L_2
L_3
A
L_1A
L_2L_1A
$L_3L_2L_1A$

An example.

We have a 4x4 matrix we wish to factorize:

$$A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix}.$$

The first step of GE looks like:

$$L_1 A = \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ -4 & & 1 & \\ -3 & & & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & 3 & 5 & 5 \\ & 4 & 6 & 8 \end{bmatrix}.$$

$$L_2L_1A = \begin{bmatrix} 1 & & \\ & 1 & \\ -3 & 1 & \\ -4 & & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & 3 & 5 & 5 \\ & 4 & 6 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & 2 & 4 \end{bmatrix}.$$

$$L_3L_2L_1A = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & 2 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & & 2 \end{bmatrix} = U.$$

Now, we need to find out the L factor as

$$L = L_1^{-1} L_2^{-1} L_3^{-1}$$

It turns out this is surprisingly easy:

$$\begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ -4 & & 1 & \\ -3 & & & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & & 1 & \\ 3 & & & 1 \end{bmatrix}.$$

In fact, the L has the same (negative) columns of the Li's:

$$\underset{A}{\begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix}} = \underset{L}{\begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & 3 & 1 & \\ 3 & 4 & 1 & 1 \end{bmatrix}} \underset{U}{\begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & & 2 \end{bmatrix}}.$$

General Formulas

Formally, for a $m \times m$ matrix, suppose x_k denotes the k-th column of the matrix **at the beginning of step k**.

Then the transformation L_k must be chosen such that:

$$x_k = \begin{bmatrix} x_{1k} \\ \vdots \\ x_{kk} \\ x_{k+1,k} \\ \vdots \\ x_{mk} \end{bmatrix} \quad \xrightarrow{L_k} \quad L_k x_k = \begin{bmatrix} x_{1k} \\ \vdots \\ x_{kk} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

To do this, we subtract l_{jk} times row k from row j, where l_{jk} is the **multiplier**

$$l_{jk} = \frac{x_{jk}}{x_{kk}}, (k < j \leq m)$$

The matrix L_k takes the form:

$$L_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -\ell_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -\ell_{mk} & & 1 \end{bmatrix}$$

As we noted before, L_k can be very easily inverted. And more luckily, the product $L = L_1^{-1} \cdots L_m^{-1}$ is also simple:

$$L = L_1^{-1} L_2^{-1} \cdots L_{m-1}^{-1} = \begin{bmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \ell_{m1} & \ell_{m2} & \cdots & \ell_{m,m-1} & 1 \end{bmatrix}.$$

The Gaussian Elimination Algorithm

Algorithm 20.1. Gaussian Elimination without Pivoting

$U = A,\ L = I$
for $k = 1$ **to** $m - 1$
    for $j = k + 1$ **to** m
        $\ell_{jk} = u_{jk}/u_{kk}$
        $u_{j,k:m} = u_{j,k:m} - \ell_{jk} u_{k,k:m}$

Operation count:

$$\frac{2}{3}m^3 \text{ flops}$$

Solution of $Ax = b$ by LU factorization

$$Ax = b \Leftrightarrow LUx = b$$
$$\text{Solve } y \text{ from } Ly = b$$
$$\text{Solve } x \text{ from } Ux = y$$

The last two forward/backward substitution takes $O(m^2)$.

Compared to QR factorization for solving linear system, LU factorization takes half the flops so is usually preferred in practice.

However, unlike QR, LU is **not** backward stable. And the above presented form of LU should not be used (unless you have a very nice matrix, such as diagonally dominant one).

Instability of GE without pivoting

Consider

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

This is full rank, and well conditioned ($\kappa(A) = 2.618$). However GE fails at the first step, because of the zero pivot.

Now let's look at a slightly perturbed version:

$$A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}$$

Okay, you could do GE on it. But what do you get?

$$L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{bmatrix}$$

Look at $U_{2,2} = 1 - 10^{20}$. In floating point arithmetic (even in double precision where $\epsilon_{\text{machine}} = 10^{-16}$), you will not get exactly the $1 - 10^{20}$, but rounded to -10^{20} (you lose the 1).

So you floating point factors are

$$\hat{L} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \hat{U} = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix}$$

The rounding seems innocent enough; but when you compute $\hat{L}\hat{U}$

$$\hat{L}\hat{U} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix}$$

Does this look like A?

If you use $\hat{L}\hat{U}$ to solve $Ax = b$ you will likely get nothing. E.g., with $b = [1,0]^T$, we get $\hat{x} = [0,1]^T$, whereas the correct solution is $x \approx [-1,1]^T$.

What happened?

(in)Stability of LU

In the previous example, LU seems stable, but not backward stable. In other words,

The $\hat{L}, \hat{U}$ are close enough to L, U, but $\hat{L}\hat{U}$ is nowhere close to $A = LU$.

But in general, LU is not even stable.

Additionally, the triangular matrices it generates may have condition number that is arbitrarily larger than condition of A. I.e., it's possible that:

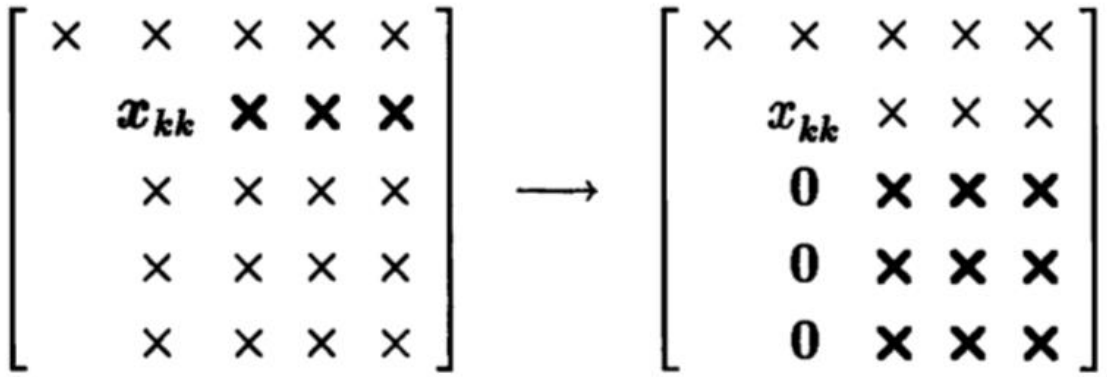
$$\kappa(L), \kappa(U) \gg \kappa(A)$$

So your forward/backward substitution may be ill-conditioned, even when A is not!

Pivoting

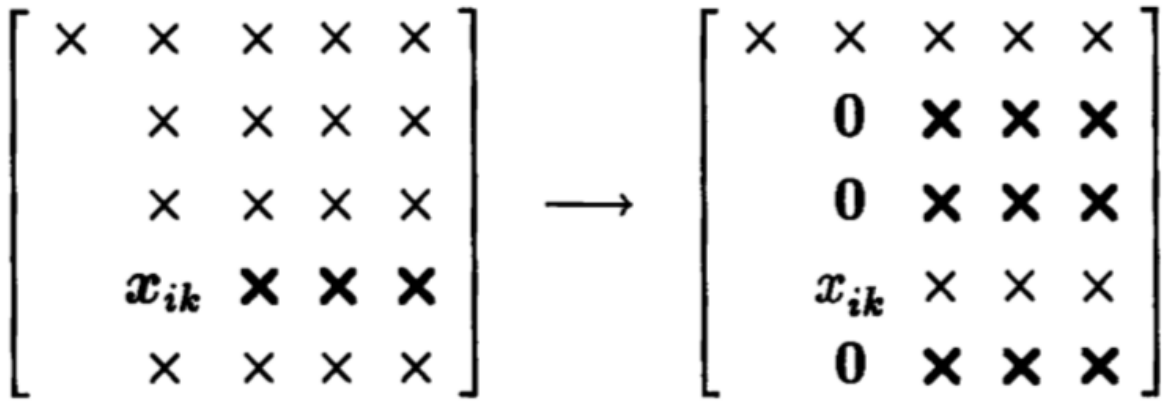
The stability of LU can be improved by **pivoting**.

The element x_{kk} on the diagonal is used to eliminate the following elements, and so is called the **pivot**.

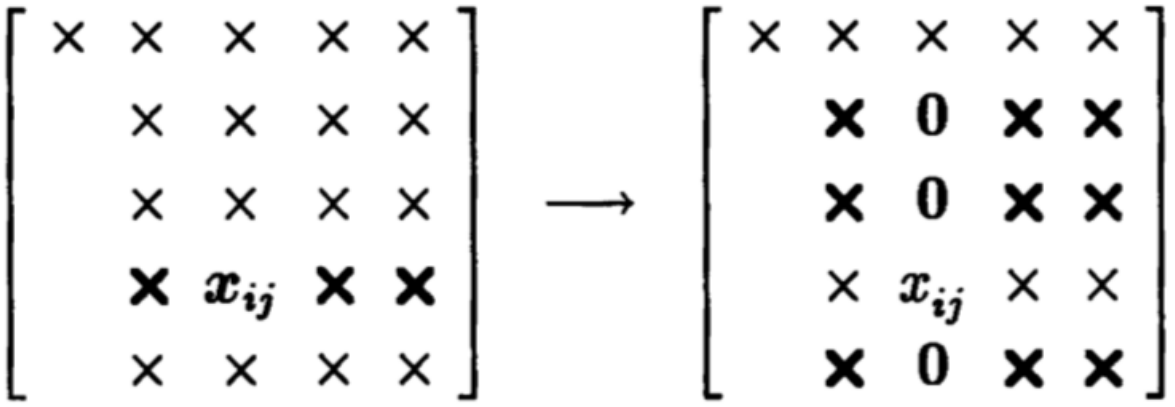


But there is no reason to choose this specific x_{kk} as pivot…

Instead, we could choose any row $i, (k \leq i \leq m)$. E.g.



In fact, we can even choose the column j $(k \leq j \leq m)$ instead of k.



Choosing the pivot

So we can pretty much pick any element in the submatrix $X_{k:m,k:m}$ as pivot, as long as it's nonzero. Typically we would pick the largest of them as pivot. (why?)

Okay, but once we start arbitrarily introducing zeros things will not become pretty… we want the same triangular structure as before. And there's a simple way of doing that:

Instead of leaving the pivot x_{ij} in-place, we move it to the diagonal, through row/column permutation.

This interchange of rows, and perhaps columns is called **pivoting**.

Partial Pivoted LU

The only modification to the unpivoted LU is that you select the largest element as your pivot, as swap the rows.

At the end, you actually factorized a row interchanged matrix:

$$PA = LU$$

where P is a permutation matrix (row permuted identity matrix).

But that's fine; it can also be used to solve linear equations (check this!)

More about the stability of LU

First, there's theorem that bounds the backward error of LU:

Theorem 22.1. *Let the factorization $A = LU$ of a nonsingular matrix $A \in \mathbb{C}^{m\times m}$ be computed by Gaussian elimination without pivoting (Algorithm 20.1) on a computer satisfying the axioms (13.5) and (13.7). If A has an LU factorization, then for all sufficiently small $\epsilon_{\text{machine}}$, the factorization completes successfully in floating point arithmetic (no zero pivots are encountered), and the computed matrices $\tilde{L}$ and $\tilde{U}$ satisfy*

$$\tilde{L}\tilde{U} = A + \delta A, \qquad \frac{\|\delta A\|}{\|L\|\,\|U\|} = O(\epsilon_{\text{machine}}) \tag{22.1}$$

for some $\delta A \in \mathbb{C}^{m\times n}$.

Does this mean LU is backward stable?

Only when $\|L\|\|U\| \approx \|A\|$.

The size of L, U

The stability of LU depends on the $\|L\|, \|U\|$ (the size of the factors).

With partial pivoting, $\|L\|$ should be small-ish, because every element of L is bounded by 1. (Why?)

But not so for $\|U\|$. The element of U can grow exponentially large than elements of A. (growth factor $\rho = \frac{\max_{i,j} |u_{ij}|}{\max_{i,j} |a_{ij}|}$)

$$A = \begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & 1 & & 4 \\ & & & 1 & 8 \\ & & & & 16 \end{bmatrix}$$

$$\rho = 2^{m-1}$$

So is LU not usable?

Now this is mysterious. Even though LU is not theorectically backward stable (we don't have a good bound on backward error), it is **utterly** backward stable in practice. (The last example don't really occur in real applications).

It's been used very extensively for more than 60 years, and nobody has ever reported instability of LU.

Why? Not completely known.

There's an (experimental) argument saying that if your matrix A is random, the L factor produced by partial pivoted LU is almost certainly well-conditioned; so $U = L^{-1}PA$ must also be well-conditioned.

Summary for LU

It’s the only algorithm we use in NLA that is not provably backward stable so it’s special.

It’s been very well optimized in software packages, such as Matlab \, LAPACK (DGETRF, DGESV), LINPACK etc.

It’s been used to benchmark the top500.org supercomputers in the world for 20+ years.

It’s a remarkable algorithm.

Symmetric Gaussian Elimination: Cholesky factorization.

If you have a **symmetric, positive definite** matrix A then you are in luck. It turns out there's a variant of GE, that takes only half the flops, and there's no need to do pivoting, and it's completely backward stable: Cholesky factorization.

A matrix is called symmetric positive definite (s.p.d.) iff it's symmetric and $x^T Ax > 0$ (for all $x \neq 0$). If we only have $x^T Ax \geq 0$ then it's call positive semi-definite.

Important property of s.p.d matrix: the eigenvalues are real, and they are all > 0.

Symmetric Gaussian Elimination

Now we have a s.p.d matrix A and we do a LU factorization on it. The first step looks like:

$$A = \begin{bmatrix} 1 & w^T \\ w & K \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ w & I \end{bmatrix} \begin{bmatrix} 1 & w^T \\ 0 & K - ww^T \end{bmatrix}$$

This is fine but after this step we lose the symmetry. To restore the symmetry, we immediately eliminate the first row:

$$\begin{bmatrix} 1 & w^T \\ 0 & K - ww^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & K - ww^T \end{bmatrix} \begin{bmatrix} 1 & w^T \\ 0 & I \end{bmatrix}$$

To recount, we did the following symmetric triangular elimination:

$$A = \begin{bmatrix} 1 & w^T \\ w & K \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ w & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & K - ww^T \end{bmatrix} \begin{bmatrix} 1 & w^T \\ 0 & I \end{bmatrix}$$

Cholesky factorization

For a general $a_{11} \neq 1$, we need $a_{11} > 0$, and get similar first step:

$$A = \begin{bmatrix} a_{11} & w^T \\ w & K \end{bmatrix} = \begin{bmatrix} \sqrt{a_{11}} & 0 \\ w/\sqrt{a_{11}} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & K - ww^T/a_{11} \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & w^T/\sqrt{a_{11}} \\ 0 & I \end{bmatrix}$$

This variant of LU is so important it has own name: Cholesky factorization.
Continuing the process we did last slide, we get:

$$A = R_1^T R_2^T \cdots R_m^T R_m \cdots R_2 R_1 = R^T R$$

But for this process to be successful, we need all the elements on diagonal to be positive during the process. **Positive definiteness** will give us this guarantee.

In fact, Cholesky factorization is the one of the cheapest way to test the if a matrix is p.d.

Cholesky factorization

- Why p.d. leads to positive a_{ii} during the process? We just need to prove that the steps produces successive p.d. matrices. I.e., $K - ww^T/a_{11}$ is p.d. Why?
  - Principal submatrix of a p.d. matrix is p.d., so $a_{11} > 0$
  - $A_2 = R_1^{-T} A R_1^{-1}$ is p.d. if A is p.d. (check the definition of p.d.)
- If A is not p.d., Cholesky factorization will break down and see some a_{ii} being negative or zero.
- Cholesky factorization is unique if it exists.
- Operation count: $\frac{1}{3}m^3$ flops. Half of LU.
- It's stable; because $\|R\| = \|R^T\| = \|A\|^{\frac{1}{2}}$ (use SVD), without any pivoting. The heaviest elements are on diagonal. (remember the bound of backward error for LU?)