Ch 6.1, 6.2, 6.5: LU Factorization, part 2 (details, variants)

Saturday, December 6, 2025 7:15 PM

Students saw this in APPM 3310

Cholesky...

① pivoting

② Block LU

③ LU for special types of matrices (symmetric, positive definite) "LDL^T" and "Cholesky" decompositions

① Pivoting

When doing Gaussian Elimination

$$\left[\begin{array}{ccc|c} \boxed{1} & 3 & -4 & 7 \\ 0 & \boxed{0} & 1 & 2 \\ 0 & 5 & 3 & 9 \end{array}\right]$$

1st "pivot"

2nd "pivot"

To continue, we need to swap rows 2 and 3.

Such swaps are called "pivoting".

- So if we have a 0 value for a pivot, we need to perform pivoting
- It turns out that if the pivot is almost 0, then we ought to perform pivoting (to minimize roundoff error)

and all good software for LU decompositions will do this

Ex. (ch 6.2 Burden and Faires)

$$0.003000 \cdot x_1 + 59.14\, x_2 = 59.17$$
$$5.291 \cdot x_1 - 6.130\, x_2 = 46.78 \quad \text{in 4-digit arithmetic}$$

(Exact soln is $x_1 = 10.00$, $x_2 = 1.000$)

$$\left[\begin{array}{cc|c} .003000 & 59.14 & 59.17 \\ 5.291 & -6.130 & 46.78 \end{array}\right] \qquad R_2 \leftarrow R_2 - \frac{5.291}{0.003000} \cdot R_1$$

$1763.6\overline{6}$ → in 4-digit precision → 1763

So absolute error is large! $(0.\overline{6})$

$$\left[\begin{array}{cc|c} 0.003000 & 59.14 & 59.17 \\ 0 & -104300 & -104400 \end{array}\right]$$

$= 46.78 - 1763 \times 59.17$ ought to be $-104309.37\overline{6}$

$= -6.130 - 1763 \times 59.14$ ought to be $-104309.37\overline{6}$

so $x_2 = \frac{-104400}{-104300} = 1.001 \neq 1.000$ (true answ), but not too bad considering 4-digit precision

but now $x_1 = \frac{59.17 - 59.14 \times 1.001}{.0003000} = -10.00 \neq 10.00$ (true answer)

small error; small denominator magnifies error

ie. we found $x_1 = -10$

but it should be $+10$. THAT'S BAD OBVIOUSLY

(ie our procedure was unstable: error is worse than expected w/ 4-digit precision)

Saturday, December 6, 2025 7:18 PM

Partial pivoting / maximal column pivoting

Strategy 1: at any given step, like (2^{nd} step)

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & \cdots & b_1 \\ 0 & a_{22} & a_{23} & \cdots & \vdots \\ 0 & a_{32} & a_{33} & \cdots & \\ 0 & a_{42} & a_{43} & & b_4 \end{array}\right] \begin{array}{l} \leftarrow R_1 \\ \leftarrow R_2 \\ \leftarrow R_3 \\ \leftarrow R_4 \end{array} \quad \text{Rows}$$

Check: swap R_2 with R_j where $|a_{j2}|$ is the largest of all $|a_{i2}|$ over $i \geq 2$

i.e.,

$$\left[\begin{array}{cccc|c} 3 & 2 & 1 & \cdots & \\ 0 & -1 & 2 & & \\ 0 & 5 & 4 & \cdots & \\ 0 & -19 & 7 & & \end{array}\right]$$

then swap R_2 and R_4

This would fix the bad example we just saw!

$$\left[\begin{array}{cc|c} .003000 & 59.14 & 59.17 \\ 5.291 & -6.130 & 46.78 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 5.291 & -6.130 & 46.78 \\ .003000 & 59.14 & 59.17 \end{array}\right]$$

and we'd get $x_1 = 10.00$, $x_2 = 1.000$. Correct! ✓

but... this pivoting strategy isn't perfect.

$$\left[\begin{array}{cc|c} 30.00 & 591400 & 591700 \\ 5.291 & -6.130 & 46.78 \end{array}\right]$$

Same system as above but multiplied by 10,000

Strategy 1 wouldn't suggest a pivot... but we should pivot!

w/o pivoting, we get $x_2 = 1.001$ (ok)

$x_1 = -10$ (No! Should be $x_1 = +10$)

Our strategy looked here ... but should also pay attention here too

Scaled Partial Pivoting strategy #2

Essentially, normalize each row so that it's largest entry in absolute value is 1

We really ought to recompute the normalization at every stage, but this is slow, so we usually just do it once at the beginning

the book argues it has $O(n^3)$ flop count, so non-negligible. Doing it just once is "cheap" by flop count. But... all types of pivoting are tricky because they make it much harder to efficiently parralelize the code

Complete/maximal (or "row and column") pivoting

Also permute columns too. This is ok if we keep track of it. This is quite slow and not done by default in standard software.

Saturday, December 6, 2025 7:18 PM

(16) Pivoting in the LU decomposition

Swapping rows means permuting rows. We can write down this effect by multiplying by a permutation matrix

def Take the identity matrix and swap/permute rows

Ex. Permutation $(1, 3, 2)$

is represented by the permutation matrix $P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix} = \begin{bmatrix} R_1 \\ R_3 \\ R_2 \end{bmatrix} \checkmark$$

PA permutes the rows of A

AP^T permutes the columns of A

More generally, if $k_1, k_2, \cdots, k_n$ is a permutation of $1, 2, \cdots, n$ then corresponding permutation matrix is $P = [p_{ij}]$, $p_{ij} = \begin{cases} 1 & j = k_i \\ 0 & \text{else} \end{cases}$

Fact: If P is a permutation matrix, then

1) $P^{-1} = P^T$, i.e., it's an orthogonal matrix
2) it has only 0 or 1 as entry
3) it's doubly stochastic meaning all rows sum to 1 and all columns sum to 1

Computationally, we prefer to work directly with the permutation if possible, but mathematically it can be nice to use the permutation matrix.

So, LU decomposition software that performs pivoting actually produces $PA = LU$ not $A = LU$

i.e. $A = P^{-1}LU$
$= P^T L U$

⚠ Matlab and Scipy have differing conventions. One returns P, the other P^T

So how do we use it to solve equations?

$A\vec{x} = \vec{b}$ i.e., $PA\vec{x} = P\vec{b} \equiv \tilde{b}$

$\tilde{b} := P\vec{b}$ is obtained by permuting $\vec{b}$. No need to explicitly multiply

so $LU\vec{x} = \tilde{b}$

and now do usual

($L\vec{y} = \tilde{b}$ via forward subs., then $U\vec{x} = \vec{y}$ via back subs.)

Saturday, December 6, 2025 7:18 PM

1c Some theory ch 6.4

Thm 6.17 For a $n \times n$ matrix A, the following are equivalent:

1. $A\vec{x} = \vec{0}$ has the unique solution $\vec{x} = 0$
2. $A\vec{x} = \vec{b}$, for any choice of $\vec{b}$, has a unique solution
3. A is nonsingular, i.e., A^{-1} exists
4. $\det(A) \neq 0$ — $\det(\,)$ is the determinant. Used in basic linear algebra courses. Not that common in numerics since its slow to compute (many tricks to approximate it)
5. Gaussian elimination w/ row interchanges can be performed to successfully solve $Ax = b$.

So, my corollary: if A is invertible, then it has a (possibly pivoted) LU decomposition.

Summary: pivoting is needed in some cases (ex.) and even if not needed, it's still a very good idea as it is a more stable way of computing the LU decomposition.

2 Block decompositions

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{matrix} \uparrow r \\ \downarrow \\ \uparrow n-r \\ \downarrow \end{matrix}$$

r $n-r$

n

1. Compute LU-decomp. of A_{11}, $A_{11} = L_{11}U_{11}$ **Recursion!**
2a. Solve for U_{12} in $L_{11}U_{12} = A_{12}$ (via forward subs.)
2b. Solve for L_{21} in $L_{21}U_{11} = A_{21}$ (via forward subs.)

} only $O(n^3)$ per RHS but $O(n)$ RHS so overall $O(n^3)$

thus

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & I_{n-r} \end{bmatrix} \begin{bmatrix} I_r & 0 \\ 0 & \tilde{A} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & I_{n-r} \end{bmatrix}$$

$\tilde{A} := A_{22} - L_{21}U_{12}$ is the Schur complement of A_{11}

3. Recurse! Let $\tilde{A} = L_{22}U_{22}$ be the LU factorization of $\tilde{A}$. Then we can combine:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}}_{U} \checkmark$$

Why? The "$O(n^3)$" steps: the little $r \times r$ LU factorization, $O(r^3)$
and $L_{21} \cdot U_{12}$ to compute $\tilde{A}$, $O((n-r)^3)$

However this naive decomposition isn't very stable, but it gives the idea of what's possible. See Demmel's book for stable block versions

No overall flop savings, but most computation is in matrix multiply (level-3 BLAS)

So, if level-3 BLAS is optimized, the LU code will be fast too!
(and also helps you parallelize the code, and efficient communication (memory movement))

Refs Ch 3.2.10 Golub and van Loan, 3rd ed '96

This block version was one of the main innovations of LAPACK (1991)

} well, kind of. See Demmel's book for a more accurate view

Saturday, December 6, 2025 7:18 PM

③ Special Factorizations (ch 6.6 Burden & Faires)

1. Diagonally dominant matrices. See book for details.

The big deal is that we can do LU w/o pivoting and this is stable!

2. Positive Definite (PD) matrices

Def A matrix is PD if ① it's symmetric, $A = A^T$ } sometimes you can drop this condition

② $x^T A x > 0 \quad \forall x \neq 0$

Equivalently, since symmetric matrices are diagonalizable and eigenvalues are real,

A is PD iff all eigenvalues $\lambda_i > 0$.

$A \in \mathbb{C}^{n \times n}$ is ok too: then want $A = A^*$ and $x^* A x > 0 \quad \forall x \neq 0$

Similarly, define a matrix A to be PSD (positive semidefinite) if

① $A = A^T$ (or $A = A^*$)

② $x^T A x \geq 0$

} or equivalently if all eigenvalues $\lambda_i \geq 0$

"A" is pos. def. is written $A > 0$ (squiggly), and "A" is PSD is written $A \geq 0$

Thm 6.23 $A > 0 \Rightarrow A^{-1}$ exists (and other facts, see book)

LU-factorizations of symmetric matrices

Instead of $A = LU$, absorb diagonal entries of U into the diagonal matrix D, and write $A = LDU$. Then by symmetry we can take

$U = L^T$ (transpose of lower triag. is upper triag.)

$A = LDL^T$ Known (not very creatively) as the LDL or LDL^T factorization.

If $A > 0$ (so A is symmetric), $D_{ii} > 0 \Rightarrow D^{1/2}$ exists

$$A = LD^{1/2} D^{1/2} L^T = (LD^{1/2})(LD^{1/2})^T$$

redefine $L \leftarrow LD^{1/2}$

$$= LL^T$$

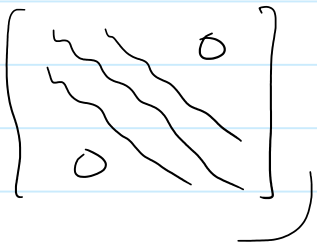
Recall: $(AB)^T = B^T A^T$, $D^T = D$ since diagonal

★ $A = LL^T$ is the Cholesky decomposition for pos. def. matrices.

Thm 6.26 $A > 0$ has a Cholesky decomposition, no need for pivoting, and the algorithm (via Gaussian Elim.) is stable.

Saturday, December 6, 2025 7:18 PM

3. Band matrices i.e. banded matrices



if only a few nonzero diagonals, we call it "banded"

ex. "tridiagonal"

You can 1) solve linear systems efficiently ($< n^3$) via the Thomas algorithm
and 2) compute LU decompositions efficiently ($< n^3$) via Crout factorization

In Ch.9 for finding eigenvalues, we'll also make use of upper Hessenberg matrices

Upper Hessenberg is almost upper triangular
except we allow nonzeros on the 1st subdiagonal