Direct methods for sparse matrices

Note Title 2024-12-06

$Ax = y$, $A \in \mathbb{R}^{m \times m}$

- Iterative methods (Krylov methods)
- Direct methods (factorizations)

Use a factorization of A to solve linear systems.

Ex: use the QR factorization $A = QR$ to solve $Ax = y$

$A = QR =$ 

Q: $m \times m$ orthogonal

R: $m \times m$ triangular

Idea: $A^{-1} = R^{-1} Q^{-1}$ lin. syst. with A = lin. syst. with Q, then with R

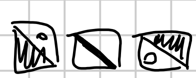
$$Ax = y \iff Q\underbrace{Rx}_{z} = y \iff \begin{cases} Qz = y & z = Q^T y \\ Rx = z & \text{backsubstitution} \end{cases}$$

To solve $Ax = y$, given A, y

1. First compute $A = QR$ $\quad O(n^3)$
2. $z = Q^T y$ $\quad O(n^2)$
3. solve $Rx = z$ by back-substitution $\quad O(n^2)$

Given many systems with the same A, we can do the expensive step 1 only once.

The same idea works with many other factorizations:

$A = USV^T$, $A = QDQ^T$, $A = LU$ , $A = LDL^T$ 

LU factorization: $A = LU$, L lower triangular,
U upper triangular.

Gaussian elimination

$$A = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}$$

One step of Gaussian elimination, introducing zeros in column 1, is equivalent to multiplication by a matrix in a special form:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ -L_{21} & 1 & 0 & 0 \\ -L_{31} & 0 & 1 & 0 \\ -L_{41} & 0 & 0 & 1 \end{bmatrix}}_{E_1} \cdot \begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

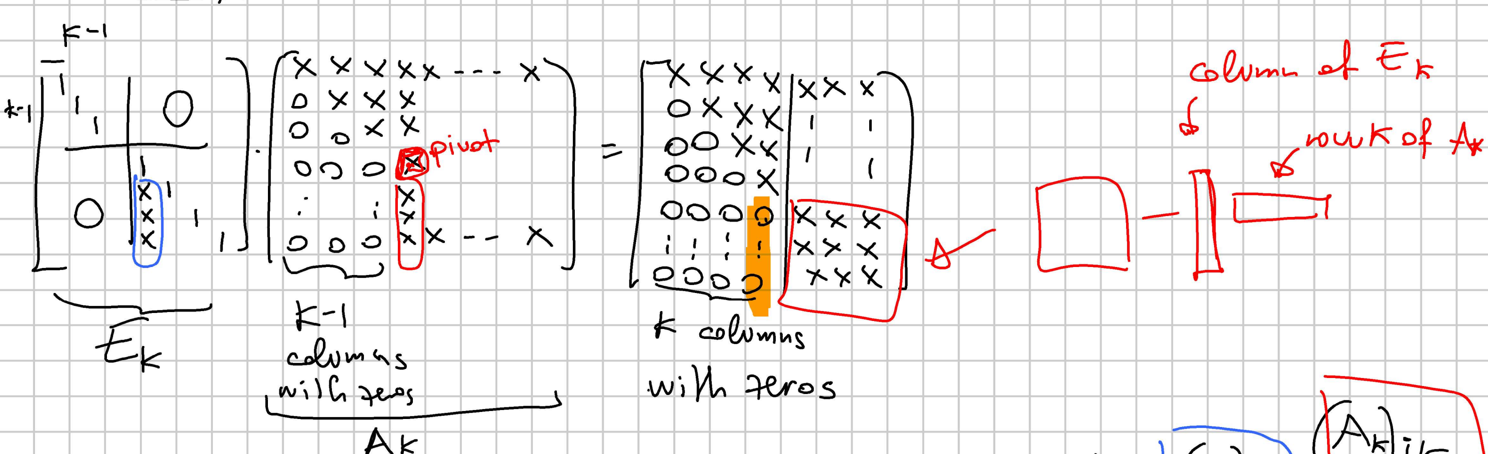
I + nonzeros in column 1

$\Leftrightarrow$ Gaussian elimination: $(\text{row } i) \leftarrow (\text{row } i) - L_{i1} \cdot (\text{row } 1)$

$$L_{i1} = \frac{A_{i1}}{A_{11}} \quad \text{produces zeros in column 1.}$$

E_1 is not orthogonal (as in QR), but it is lower triangular

Subsequent steps:



The multipliers you need at step k are given by $(E_k)_{i1} = \dfrac{(A_k)_{ik}}{(A_k)_{kk}}$.

pivot

$\boxed{A_1 = A}$

$$\boxed{\begin{aligned} A_2 &= E_1 A_1 \\ A_3 &= E_2 A_2 \\ &\vdots \\ A_m &= E_{m-1} A_{m-1} \end{aligned}}$$

$$\underbrace{\begin{bmatrix} x & x & \cdots & x \\ x & & & \\ \vdots & & & \\ x & & \cdots & x \end{bmatrix}}_{A_1} \to \underbrace{\begin{bmatrix} x & x & \cdots & x \\ 0 & x & & \\ 0 & \vdots & & \\ \vdots & & & \\ 0 & x & \cdots & x \end{bmatrix}}_{A_2} \to \underbrace{\begin{bmatrix} x & x & x & & \\ 0 & x & x & & \\ 0 & 0 & x & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & & x \end{bmatrix}}_{A_3} \to$$

$$\cdots \to \begin{bmatrix} x & x & \cdots & \cdots & x \\ 0 & x & & & \vdots \\ 0 & 0 & x & & \vdots \\ \vdots & 0 & \ddots & \ddots & \\ \vdots & \vdots & & 0 & x \\ 0 & 0 & \cdots & 0 \; 0 & x \end{bmatrix}$$

A_m upper triangular!

$$A_m = E_{m-1} A_{m-1} = E_{m-1} E_{m-2} A_{m-2} = \cdots = E_{m-1} E_{m-2} \cdots E_1 A$$

$$A = \underbrace{E_1^{-1} E_2^{-1} \cdots E_{m-1}^{-1}}_{L} \underbrace{A_m}_{U}$$

L: lower triangular (product of lower tr. matrices)

U: upper triangular

Remarkable:

$$E_1^{-1} E_2^{-1} \cdots E_{m-1}^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ L_{21} & 1 & 0 & & \vdots \\ L_{31} & L_{32} & 1 & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ L_{m1} & L_{m2} & \cdots & L_{m,m-1} & 1 \end{bmatrix}$$

Implementation:

step 0 $\quad L = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix} \qquad U = A$

step 1 $\quad \begin{bmatrix} 1 & & & 0 \\ x & \ddots & & \\ x & & \ddots & \\ \vdots & 0 & & \\ x & & & 1 \end{bmatrix} \qquad \underbrace{\begin{bmatrix} x & x & x & \cdots & x \\ 0 & x & & & \\ 0 & \vdots & & & \\ \vdots & & & & \\ 0 & x & \cdots & \cdots & x \end{bmatrix}}_{A_2} \qquad L_{i1} = \dfrac{A_{i1}}{A_{11}}$

step 2

$$\begin{bmatrix} 1 & & & & & \\ \times & 1 & & & 0 & \\ \times & \times & 1 & & & \\ \vdots & \vdots & 0 & 1 & & \\ & & \vdots & 0 & \ddots & \\ \times & \times & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \underbrace{\begin{bmatrix} \times & \times & \times & \cdots & \times \\ 0 & \times & \times & & \vdots \\ 0 & 0 & \times & & \vdots \\ \vdots & 0 & \times & & \\ & \vdots & \times & & \\ & & \vdots & & \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}}_{A_3}$$

$\vdots$

Computational cost: $\frac{2}{3}n^3$ ops + lower order terms.
$O(n^3)$

(vs. $\frac{4}{3}n^3$ for QR factorization)

Is this algorithm stable? <u>No</u>: very small pivots can cause issues. One can show that the computed $\tilde{L}, \tilde{U}$ are such that $\tilde{L} \cdot \tilde{U} = \hat{A}$

$$\|\hat{A} - A\| \leq \|\tilde{L}\| \cdot \|\tilde{U}\| \cdot O(\underset{\substack{\uparrow \\ \text{mach. precision}}}{u})$$

If $\|L\| \cdot \|U\| \gg \|A\|$, this will not be backward stable!

$$L_{ik} = \frac{(A_k)_{ik}}{(A_k)_{kk}} \qquad \text{if} \quad (A_k)_{kk} \ll (A_k)_{ik}$$

<u>Fix</u>: permute rows of the matrix A

Before step k of Gaussian elimination, swap the pivot row k with the row with largest $(A_k)_{ik}$ $(i = k, k+1, \dots m)$.

This ensures $|L_{ij}| \leq 1$ for all i, j

<u>Problem</u>: even with pivoting, there is a very small

number of matrices for which $\frac{\|U\|}{\|A\|} \sim 2^m$

But LU with pivoting is stable in most cases, and used in practice: $x = A \backslash y$ scipy.linalg.linsolve(A, y) ...

Matlab: $[L, U, P] = lu(A)$ returns L, U, P s.t. $LU = PA$

For sparse matrices: typically, $nnz(U), nnz(L) \gg nnz(A)$

high cost in both time and memory.

- This is matrix-dependent
- optimal row permutation to reduce fill-in is NP-complete!
- allocations, blocking tricky: better to use libraries.
- using LU for sparse system is efficient only if L, U are sparse.

Symmetric variants of LU:

$A = A^T \in \mathbb{R}^{m \times m}$, can we exploit symmetry to reduce the cost of LU factorization? Yes!

At each step, instead of multiplying $A_{k+1} = E_k A_k$, we multiply $A_{k+1} = E_k A_k \boxed{E_k^T}$

This produces A_{k+1} that is symmetric and has zeros in its first k columns and rows

$$A_{k+1} = \begin{array}{l} \left[\begin{array}{cccccc} \times & 0 & 0 & 0 & \cdots & 0 \\ 0 & \times & 0 & 0 & \cdots & 0 \\ 0 & 0 & \times & 0 & \cdots & 0 \\ 0 & 0 & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots & & \\ 0 & 0 & 0 & \times & \cdots & \times \end{array}\right] \end{array} \begin{array}{l} \}\, k \text{ rows} \\ \\ \\ \\ \\ \end{array}$$

symmetric!

$\overrightarrow{\text{columns}}$

At the end of elimination, $A_m = \begin{bmatrix} \times & & & \\ & \times & & \\ & & \ddots & \\ & & & \times \end{bmatrix}$ diagonal!

$$A = \underbrace{E_1^{-1} E_2^{-1} \cdots E_{m-1}^{-1}}_{\text{lower triangular}} \; \underbrace{A_m}_{\text{diagonal}} \; \underbrace{(E_{m-1}^{T})^{-1} \cdots (E_1^{T})^{-1}}_{\substack{\text{upper triangular} \\ \text{transpose of first factor}}}$$

$$A = L \cdot D \cdot L^T$$

This is called <u>LDL factorization</u>: a symmetric matrix (if all pivots are nonzero) can be written as $A = LDL^T$, with L lower tr., D diagonal.

All matrices A_k are symmetric $\rightarrow$ we can compute only <u>half</u> of its entries, and fill in the rest by symmetry!

<u>Cost</u>: $\frac{1}{3}n^3$ operations

<u>Pivoting</u> is more complicated. Matlab in $[L, D, P] = \mathrm{ldl}(A)$ returns matrices s.t. $LDL^T = \underbrace{PAP^T}_{\substack{A \text{ with row +} \\ \text{column exchanges}}}$

but, for stability reasons, we need to have <u>1×1</u> or <u>2×2</u> diagonal blocks in D. (Bunch-Kaufman, Bunch-Parlett pivoting)

Positive definite = symmetric + $x^T A x > 0$ for all vectors $x \neq 0$.

When A is positive definite, one can prove that d_{kk} (pivot at step k, which is also the (k,k) entry of D) is always positive.

This ensures that we can complete the factorization (even in a stable way) also with no pivoting!

$$A = LDL^T \qquad D = \begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{mm} \end{bmatrix} \qquad d_{kk} > 0 \text{ for all } k = 1, \dots m.$$

Slightly different form to write this factorization for positive definite A: we can write

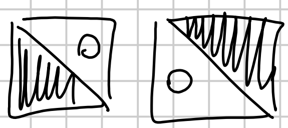
$$D = \begin{bmatrix} \sqrt{d_{11}} & & & \\ & \sqrt{d_{22}} & & \\ & & \ddots & \\ & & & \sqrt{d_{mm}} \end{bmatrix} \cdot \begin{bmatrix} \sqrt{d_{11}} & & & \\ & \sqrt{d_{22}} & & \\ & & \ddots & \\ & & & \sqrt{d_{mm}} \end{bmatrix}$$

$$= D^{\frac{1}{2}} \cdot D^{\frac{1}{2}}$$

$$\left(D^{\frac{1}{2}}\right)^T = D^{1/2}$$

So we can write

$$A = \underbrace{LD^{\frac{1}{2}}}_{R^T} \cdot \underbrace{D^{\frac{1}{2}} L^T}_{R} = R^T R =$$



For each symm. pos. def. matrix A, there exists R upper triangular such that $A = R^T R$.

This is known as Cholesky factorization.

16th 11:00 A1?