Cost of LU Factorization

Contents

Cost of PLU Factorization Algorithm

Notation: Big-O and Asymptotic

Let f, g be positive functions defined on $\mathbb{N}$.

- $f(n) = O\big(g(n)\big)$ ("f is *big-O of* g") as $n \to \infty$ if

complex simple

$$\frac{f(n)}{g(n)} \leqslant C, \quad \text{for all sufficiently large } n.$$

- $f(n) \sim g(n)$ ("f is *asymptotic to* g") as $n \to \infty$ if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1.$$

Examples $f(n) = 3n^3 + 2n^2 - 1$

- $f(n) = O(n^3)$ because $\dfrac{f(n)}{n^3} = \dfrac{3n^3+2n^2-1}{n^3} = 3 + \dfrac{2}{n} - \dfrac{1}{n^3} \lesssim 3+1+1=5$

for all large n.

Note $f(n) = O(5n^3)$, $f(n) = O(n^5)$, ...

- $f(n) \sim 3n^3$ because $\displaystyle\lim_{n\to\infty} \frac{3n^3+2n^2-1}{3n^3} = 1$.

Timing Vector/Matrix Operations – FLOPS

- One way to measure the "efficiency" of a numerical algorithm is to count the number of floating-point arithmetic operations (FLOPS) necessary for its execution. $(+, -, *, /, \text{sqrt})$
- The number is usually represented by $\sim cn^p$ where c and p are given explicitly.
- We are interested in this formula when n is large.

FLOPS for Major Operations

Vector/Matrix Operations

Let $x, y \in \mathbb{R}^n$ and $A, B \in \mathbb{R}^{n\times n}$. Then

- (vector-vector) $x^{\mathrm{T}}y$ requires $\sim 2n$ *flops*.
- (matrix-vector) Ax requires $\sim 2n^2$ *flops*.
- (matrix-matrix) AB requires $\sim 2n^3$ *flops*.

$$\vec{x}^T\vec{y} = [x_1 \; x_2 \cdots x_n]\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

(inner product)

$$= \underbrace{x_1y_1 + x_2y_2 + \cdots + x_ny_n}$$

1 ⊗ 1 ⊗ 1 ⊗ : total of n ⊗'s.

total of $(n-1)$ ⊕'s

$(2n-1)$ flops

$\sim 2n$ flops

$$= \sum_{i=1}^{n} x_i y_i$$

- Write $A = \begin{bmatrix} \vec{\alpha}_1^T \\ \hline \vec{\alpha}_2^T \\ \hline \vdots \\ \hline \vec{\alpha}_n^T \end{bmatrix}$ ← 1st row.

Then

$$A\vec{x} = \begin{bmatrix} \vec{\alpha}_1^T \\ \hline \vec{\alpha}_2^T \\ \hline \vdots \\ \hline \vec{\alpha}_n^T \end{bmatrix} \vec{x} = \begin{array}{c} \begin{bmatrix} \vec{\alpha}_1^T \vec{x} \\ \hline \vec{\alpha}_2^T \vec{x} \\ \hline \vdots \\ \hline \vec{\alpha}_n^T \vec{x} \end{bmatrix} \end{array} \begin{array}{l} \sim 2n \\ \sim 2n \\ \vdots \\ \sim 2n \end{array}$$

Total flops : $\sim 2n \times n = 2n^2$

GE w/ partial pivoting

- PLU factorization : $\sim \frac{2}{3}n^3$ $\quad (PA = LU)$
- Forward elimination : $\sim n^2$
- Backward substitution : $\sim n^2$

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

$$PA\vec{x} = P\vec{b}$$

$$L\underbrace{U\vec{x}}_{\vec{y}} = P\vec{b}$$

- $L\vec{y} = P\vec{b}$
- $U\vec{x} = \vec{y}$

Cost of PLU Factorization

- Pivot : $R_i \Leftrightarrow R_j$ (no flops)
- row replacement : $R_i \longrightarrow R_i + \left(-\frac{a_{ij}}{a_{jj}}\right) R_j$

Note that we only need to count the number of *flops* required to zero out elements below the diagonal of each column.

Suppose you are working out the j^{th} column.

- For each $i > j$, we replace R_i by $R_i + cR_j$ where $c = -a_{i,j}/a_{j,j}$. This requires approximately $2(n-j+1)$ *flops*:
  - 1 division to form c
  - $n-j+1$ multiplications to form cR_j
  - $n-j+1$ additions to form $R_i + cR_j$

$j^{th} \ldots n^{th}$: $n-j+1$

- Since $i \in \mathbb{N}[j+1, n]$, the total number of *flops* needed to zero out all elements below the diagonal in the jth column is approximately $2(n-j+1)(n-j)$.
- Summing up over $j \in \mathbb{N}[1, n-1]$, we need about $(2/3)n^3$ *flops*:

$n-(j+1)+1$

$= n-j-1+1 = n-j$

$$\sum_{j=1}^{n-1} 2\underbrace{(n-j+1)}_{\sim\, n-j}(n-j) \sim 2\sum_{j=1}^{n-1}(n-j)^2 = 2\sum_{j=1}^{n-1} j^2 \sim \frac{2}{3}n^3$$

$n-j$: $n-1, n-2, \cdots, 1$

Explanation of last step

Recall: $\sum_{j=1}^{n} k^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$

$$\sum_{j=1}^{n-1} j^2 = \frac{(n-1)\left[(n-1)+1\right]\left[2(n-1)+1\right]}{6}$$

$$= \frac{n(n-1)(2n-1)}{6} = \frac{2n^3 + (\text{lower-order terms})}{6}$$

$$\sim \frac{n^3}{3}.$$

Cost of Forward Elimination and Backward Substitution

Forward Elimination

- The calculation of $y_i = \beta_i - \sum_{j=1}^{i-1} \ell_{ij} y_j$ for $i > 1$ requires approximately $2i$ *flops*:
  - 1 subtraction
  - $i - 1$ multiplications
  - $i - 2$ additions
- Summing over all $i \in \mathbb{N}[2, n]$, we need about n^2 *flops*:

$$\sum_{i=2}^{n} 2i \sim 2\frac{n^2}{2} = n^2 .$$

Backward Substitution

- The cost of backward substitution is also approximately n^2 *flops*, which can be shown in the same manner.

Cost of G.E. with Partial Pivoting

Gaussian elimination with partial pivoting involves three steps:

- PLU factorization: $\sim (2/3)n^3$ *flops*
- Forward elimination: $\sim n^2$ *flops*
- Backward substitution: $\sim n^2$ *flops*

Summary

The total cost of Gaussian elimination with partial pivoting is approximately

$$\frac{2}{3}n^3 + n^2 + n^2 \sim \frac{2}{3}n^3$$

flops for large n.

Application: Solving Multiple Square Systems Simultaneously

To solve two systems $A\mathbf{x}_1 = \mathbf{b}_1$ and $A\mathbf{x}_2 = \mathbf{b}_2$. (Note both involve the same coeff. matrix.)

Method 1.

- Use G.E. for both.
- It takes $\sim (4/3)n^3$ *flops*.

```
%% method 1
x1 = A \ b1;      ~ 2/3 n^3
x2 = A \ b2;      ~ 2/3 n^3
```

Method 2.

- Do it in two steps:
  1. Do PLU factorization $PA = LU$.
  2. Then solve $LU\mathbf{x}_1 = P\mathbf{b}_1$ and $LU\mathbf{x}_2 = P\mathbf{b}_2$.
- It takes $\sim (2/3)n^3$ *flops*.

```
%% method 2
[L, U, P] = lu(A);      ~ 2/3 n^3
x1 = U \ (L \ (P*b1));  ~ 4n^2
x2 = U \ (L \ (P*b2));  ~ 4n^2
```

$\sim \frac{2}{3}n^3$

back. subs   for. elim

```
%% compact implementation
X = A \ [b1, b2];
x1 = X(:, 1);
x2 = X(:, 2);
```

$$A \underbrace{\begin{bmatrix} \vec{x}_1 & \vec{x}_2 \end{bmatrix}}_{= X} = \underbrace{\begin{bmatrix} \vec{b}_1 & \vec{b}_2 \end{bmatrix}}_{= B}$$

$$AX = B$$