Cost of LU Factorization

Contents

Cost of PLU Factorization Algorithm

Key result: In solving $A\vec{x} = \vec{b}$ where $A \in \mathbb{R}^{n \times n}$:

- PLU factorization requires about $\frac{2}{3}n^3$ flops
- Forward & Backward substitutions require only about n^2 flops

Notation: Big-O and Asymptotic

$f(1), f(2), f(3), \cdots$

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

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

"rough" idea of what $f(n)$ is like for large n.

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

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

more accurate description

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

- $f(n) = O(n^3)$ because $\frac{3n^3+2n-1}{n^3} = 3 + \frac{2}{n^2} - \frac{1}{n^3} \leqslant 3+1+1 = 5$ for all large n.

  ($f(n) = O(5n^3)$, $f(n) = O(n^4)$, $f(n) = O(n^5)$, $\cdots$)

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

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.

  $(+, -, *, \div, \text{sqrt})$

- The number is usually represented by $\sim cn^{\boxed{p}}$ where c and p are given explicitly.
- We are interested in this formula when n is large.

  that is, as $n \to \infty$.

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)

$$= \overbrace{x_1 y_1 + x_2 y_2 + \cdots + x_n y_n}^{(n-1)\ \oplus\text{'s}}$$

$$\underbrace{\otimes \quad \otimes \quad \cdots \quad \otimes}_{n\ \otimes\text{'s}}$$

Total flops : $n + (n-1)$

$= 2n-1 \sim \underline{\underline{2n}}$

$$A\vec{x} = \begin{bmatrix} \vec{\alpha}_1^T \\ \hline \vdots \\ \hline \vec{\alpha}_n^T \end{bmatrix} \vec{x} = \left.\begin{bmatrix} \vec{\alpha}_1^T\vec{x} \\ \hline \vdots \\ \hline \vec{\alpha}_n^T \vec{x} \end{bmatrix} \begin{array}{l} \to \sim 2n \\ \to \sim 2n \\ \to \sim 2n \end{array} \right] n \text{ times.}$$

$$A = \begin{bmatrix} \vec{\alpha}_1^T \\ \hline \vdots \\ \hline \vec{\alpha}_n^T \end{bmatrix}$$

$\vec{\alpha}_i^T$: i^{th} row of A.

<u>Total flops</u>: $(2n-1)\times n$

$= 2n^2 - n \sim 2n^2$

Cost of PLU Factorization

- pivot : $R_i \leftrightarrow R_j$ (No flops needed)
- row replacement : $R_i \to R_i + \left(-\frac{a_{ij}}{a_{jj}}\right) R_j$

(flop counting needed)

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

S'pose 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 $R_i + cR_j$

  $\Big\}\ 1 + 2(n-j+1) \sim 2(n-j+1)$

$$
\begin{array}{r}
\\ \\ i=4 \to \\ i=5 \to
\end{array}
\begin{bmatrix}
\times & \times & \times & \times & \times \\
0 & \times & \times & \times & \times \\
0 & 0 & \boxed{\times} & \boxed{\times} & \boxed{\times} \\
0 & 0 & \boxed{\times} & \boxed{\times} & \boxed{\times} \\
0 & 0 & \boxed{\times} & \boxed{\times} & \boxed{\times}
\end{bmatrix}
$$

$\uparrow$ $j=3$

$3 = 5 - 3 + 1$
$= n - j + 1$

- 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*:

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

$\uparrow$ reversing the term (change of indices)

Why $2\sum_{j=1}^{n-1} j^2 \sim \frac{2}{3}n^3$?

<u>One way</u> Recall $\sum_{j=1}^{n} j^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$.

So

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

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

$$\sim \frac{2n^3}{6} = \frac{n^3}{3}$$

__Another__ Using the following result.

$$\sum_{j=1}^{n-1} j^{p} \sim \frac{j^{p+1}}{p+1}$$

Think $\int x^{p}\,dx = \frac{x^{p+1}}{p+1} + C$

$$\sum_{j=1}^{n-1} j^{2} \sim \frac{j^{2+1}}{2+1} = \frac{j^{3}}{3}$$

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* ← heavy
- Forward elimination: $\sim n^2$ *flops*
- Backward substitution: $\sim n^2$ *flops*

← light

Summary

The total cost of Gaussian elimination with partial pivoting is approximately

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

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 matrix.)

Method 1. (inefficient)

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

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

Method 2. (efficient)

- 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 $\sim \underline{(2/3)n^3}$ *flops*.

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

$\frac{2}{3}n^3 + 4n^2 \sim \frac{2}{3}n^3$

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

$$A \underbrace{[\vec{x}_1 \ \vec{x}_2]}_{X} = \underbrace{[\vec{b}_1 \ \vec{b}_2]}_{B}$$