Lec 15: Square Linear Systems – Analysis

Efficiency

Notation: Big-O and Asymptotic

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

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

simpler

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

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

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

Example

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

- $f(n) = O(n^3)$ because $\dfrac{3n^3+2n^2+1}{n^3} = 3 + \dfrac{2}{n} + \dfrac{1}{n^3} < 4$ for n large.
- $f(n) \sim 3n^3$ because $\dfrac{3n^3+2n^2+1}{3n^3} \to 1$ as $n \to \infty$.

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.
  (+, −, ∗, /, 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*.

$$\vec{x}^{\,T}\vec{y} = \sum_{i=1}^{n} \underbrace{x_i y_i}$$

add n terms $\Downarrow$ $(n-1)$ $+$

1 multiplication $\Longrightarrow$ n $\times$

$n + (n-1)$

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

Cost of PLU Factorization

→ { • ~~pivoting (row swap)~~
• row replacement → linear combination

Note that we only need to count the number of *flops* required to zero out elements below the diagonal of each 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 $R_i + cR_j$
- 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*:

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

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

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

flops for large n.

Application: Solving Multiple Square Systems Simultaneously

To solve two systems $A\mathbf{x}_1 = \mathbf{b}_1$ and $A\mathbf{x}_2 = \mathbf{b}_2$.

Method 1.

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

```
%% method 1
 x1 = A \ b1;
 x2 = A \ b2;
```

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);
 x1 = U \ (L \ (P*b1));
 x2 = U \ (L \ (P*b2));
```

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