5. Complexity of matrix algorithms

- flop counts
- vector-vector operations
- matrix-vector product
- matrix-matrix product

Flop counts

floating-point operation (flop)

- one floating-point addition, subtraction, multiplication, or division
- other common definition: one multiplication followed by one addition

flop counts of matrix algorithm

- total number of flops is typically a polynomial of the problem dimensions
- usually simplified by ignoring lower-order terms

applications

- a simple, machine-independent measure of algorithm complexity
- not an accurate predictor of computation time on modern computers

Vector-vector operations

- inner product of two n-vectors

$$x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

n multiplications and $n-1$ additions $= 2n-1$ flops ($2n$ if $n \gg 1$)

- addition or subtraction of n-vectors: n flops

- scalar multiplication of n-vector : n flops

Matrix-vector product

matrix-vector product with $m \times n$-matrix A:

$$y = Ax$$

m elements in y; each element requires an inner product of length n:

$$(2n-1)m \text{ flops}$$

approximately $2mn$ for large n

special cases

- $m = n$, A diagonal: n flops
- $m = n$, A lower triangular: $n(n+1)$ flops
- A very sparse (lots of zero coefficients): #flops $\ll 2mn$

Matrix-matrix product

product of $m \times n$-matrix A and $n \times p$-matrix B:

$$C = AB$$

mp elements in C; each element requires an inner product of length n:

$$mp(2n-1) \text{ flops}$$

approximately $2mnp$ for large n

Exercises

1. evaluate $y = ABx$ two ways (A and B are $n \times n$, x is a vector)

- $y = (AB)x$ (MATLAB: `C = A*B; y = C*x;`)
- $y = A(Bx)$ (MATLAB: `z = B*x; y = A*z;`)

both methods give the same answer, but which method is faster?

2. evaluate $y = (I + uv^T)x$ where u, v, x are n-vectors

- $A = I + uv^T$ followed by $y = Ax$ (MATLAB: `y = (eye(n)+u*v')*x`)
- $w = (v^T x)u$ followed by $y = x + w$ (MATLAB: `y = x + (v'*x)*u`)

References

Lecture notes on

Complexity of matrix algorithms, EE103, L. Vandenberhge, UCLA