Matrix Computations
Chapter 1 Introduction
Section 1.3 Sensitivity Analysis and Complexities of Matrix Computations

Jie Lu
ShanghaiTech University

Linear system with perturbations

Consider the linear system

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

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is nonsingular and $\mathbf{b} \in \mathbb{R}^n$

Then consider the linear system with perturbations in $\mathbf{A}$ and $\mathbf{b}$

$$(\mathbf{A} + \epsilon\mathbf{F})\mathbf{x} = \mathbf{b} + \epsilon\mathbf{f}$$

where $\epsilon \in \mathbb{R}$, $\mathbf{F} \in \mathbb{R}^{n \times n}$, and $\mathbf{f} \in \mathbb{R}^n$

Let $\mathbf{x}(\epsilon)$ be the solution to the above perturbed system
$\mathbf{x}(0) = \mathbf{A}^{-1}\mathbf{b}$ is the solution to the unperturbed $\mathbf{A}\mathbf{x} = \mathbf{b}$

Linear system with perturbations (cont'd)

$$(\mathbf{A} + \epsilon\mathbf{F})\mathbf{x}(\epsilon) = \mathbf{b} + \epsilon\mathbf{f}, \quad \mathbf{x}(0) = \mathbf{A}^{-1}\mathbf{b}$$

Taking derivative w.r.t. ϵ on both sides of the above equation and then setting $\epsilon = 0$ lead to

$$\dot{\mathbf{x}}(0) = \mathbf{A}^{-1}(\mathbf{f} - \mathbf{F}\mathbf{x}(0))$$

where $\dot{\mathbf{x}}$ represents the derivative of $\mathbf{x}(\epsilon)$ w.r.t. ϵ

Applying Taylor series expansion to $\mathbf{x}(\epsilon)$ leads to

$$\mathbf{x}(\epsilon) = \mathbf{x}(0) + \epsilon\dot{\mathbf{x}}(0) + \mathcal{O}(\epsilon^2)$$

Combining the above two equations gives

$$\frac{\|\mathbf{x}(\epsilon) - \mathbf{x}(0)\|}{\|\mathbf{x}(0)\|} \leq |\epsilon| \cdot \|\mathbf{A}^{-1}\| \left(\frac{\|\mathbf{f}\|}{\|\mathbf{x}(0)\|} + \|\mathbf{F}\| \right) + \mathcal{O}(\epsilon^2)$$

where $\|\cdot\|$ represent any norm satisfying $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$ for all $\mathbf{A}, \mathbf{B}$ with proper sizes (e.g., p-norm, $p \in [1, \infty]$)

Condition number

For nonsingular $\mathbf{A} \in \mathbb{R}^{n \times n}$, define the condition number of $\mathbf{A}$ as

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$$

With the convention, $\kappa(\mathbf{A}) = \infty$ for singular $\mathbf{A}$

Then, using $\|\mathbf{b}\| \leq \|\mathbf{A}\|\|\mathbf{x}(0)\|$,

$$\underbrace{\frac{\|\mathbf{x}(\epsilon) - \mathbf{x}(0)\|}{\|\mathbf{x}(0)\|}}_{\text{relative error in } \mathbf{x}} \leq \kappa(\mathbf{A}) \left(\underbrace{|\epsilon| \frac{\|\mathbf{F}\|}{\|\mathbf{A}\|}}_{\text{relative error in } \mathbf{A}} + \underbrace{|\epsilon| \frac{\|\mathbf{f}\|}{\|\mathbf{b}\|}}_{\text{relative error in } \mathbf{b}} \right) + \mathcal{O}(\epsilon^2)$$

If $\kappa(\mathbf{A})$ is large, the solution is sensitive to the errors/perturbations in $\mathbf{A}$ and $\mathbf{b}$, so that $\mathbf{A}$ is said to be an ill-conditioned matrix; If $\kappa(\mathbf{A})$ is small, $\mathbf{A}$ is said to be well-conditioned

Condition number (cont'd)

We add a subscript to $\kappa(\cdot)$ based on the underlying norm. For example,

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2 \geq \|\mathbf{A}\mathbf{A}^{-1}\|_2 = \|\mathbf{I}\|_2 = 1$$

For any orthogonal matrix $\mathbf{Q}$, $\kappa_2(\mathbf{Q}) = \|\mathbf{Q}\|_2\|\mathbf{Q}^T\|_2 = 1 \cdot 1 = 1$

Condition numbers with different underlying norms are equivalent in the sense that for any $\kappa_\alpha(\cdot), \kappa_\beta(\cdot)$, there always exist $c_1, c_2 > 0$ such that

$$c_1\kappa_\alpha(\mathbf{A}) \leq \kappa_\beta(\mathbf{A}) \leq c_2\kappa_\alpha(\mathbf{A}) \quad \forall \mathbf{A} \in \mathbb{R}^{n\times n}$$

Therefore, if a matrix is ill-conditioned in terms of $\kappa_\alpha(\cdot)$, it is also ill-conditioned in terms of any other $\kappa_\beta(\cdot)$

Floating point arithmetic

- Real numbers are stored in computers using floating point format, which represents a real number using a block of 64 bits (0s and 1s), or 8 bytes (groups of 8 bits)
    - Each of the 2^{64} possible sequences of bits corresponds to a specific real number
- Real numbers can be approximated as floating point numbers to an accuracy of around 10 digits (good enough for almost all practical applications)
- When computers carry out arithmetic operations on numbers represented in floating point format, the result is rounded to the nearest floating point number, and the very small error in the computed result is called (floating point) round-off error
    - Not included in this course; belong to the field of numerical analysis

Complexities of Matrix Computations

- Every vector/matrix operation such as $\mathbf{x}+\mathbf{y}$, $\mathbf{y}^T\mathbf{x}$, $\mathbf{A}\mathbf{x}$, ... incurs computational costs, and they cost more as the vector and matrix sizes increase
- A very rough estimate of computational costs can be found by counting the total number of floating point arithmetic operations
- flops: one flop means one floating point arithmetic operation
- We typically look at floating point arithmetic operations such as
  - addition
  - subtraction
  - multiplication
  - division
- In this course, complexity means the number of flops required by a specific method

Complexities of Vector Operations

Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$,

- Scalar-vector multiplication $\alpha\mathbf{x}$: n multiplications $\Longrightarrow$ n flops
- Vector addition $\mathbf{x} + \mathbf{y}$: n additions $\Longrightarrow$ n flops
- Inner product $\mathbf{y}^T\mathbf{x} = \sum_{i=1}^{n} x_i y_i$: n multiplications and $n-1$ additions $\Longrightarrow$ $2n-1$ flops (often simplified to $2n$ flops)

Complexities of Matrix Computations

- Scalar-matrix multiplication $\alpha\mathbf{A}$ for $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\alpha \in \mathbb{R}$: mn flops
- Matrix addition $\mathbf{A}+\mathbf{B}$ for $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$: mn flops
- Matrix transposition $\mathbf{A}^T$: 0 flops (Copying entries of $\mathbf{A}$ to those of $\mathbf{A}^T$ take some time, but not reflected in the flop count)
- Matrix-vector multiplication $\mathbf{Ax}$ for $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$: m inner products, so $m(2n-1)$ flops (simplified to $2mn$ flops)
- Matrix multiplication $\mathbf{AB}$ for $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$: p matrix-vector multiplications, so $pm(2n-1)$ flops

Big O complexities

- Mostly we are only interested in the *order* of complexity
- You don't need the exact flop count (which can be hard to count for complicated algorithms)
- big O notation: Given two functions $f(n), g(n)$, the notation

$$f(n) = \mathcal{O}(g(n))$$

means that there exist $C > 0$ and n_0 such that $|f(n)| \leq C|g(n)|$ for all $n \geq n_0$

- Big O complexities of standard vector/matrix operations:
  - $\mathbf{x} + \mathbf{y}$: $\mathcal{O}(n)$ flops
  - $\mathbf{y}^T\mathbf{x}$: $\mathcal{O}(n)$ flops
  - $\mathbf{A}\mathbf{x}$: $\mathcal{O}(mn)$ flops
  - $\mathbf{A}\mathbf{B}$: $\mathcal{O}(mnp)$ flops

Big O complexities (cont'd)

- **Example**: Suppose the exact flop count of an algorithm is

$$f(n) = 3n^3 + 8n^2 + 2n + 20240307$$

  - $O(n^3)$ flops
  - For sufficiently large n, n^3 dominates
  - For small n, 20240307 costs more

- **Example**: Suppose the exact flop counts of two algorithms are

$$f_1(n) = n^3, \quad f_2(n) = \frac{1}{2}n^3.$$

  - Their big O complexities are both $O(n^3)$
  - But the second algorithm may be twice faster

Complexities of complex vector/matrix operations

For complex vector and matrix operations, define one flop as one real flop

- 1 complex addition = 2 real additions = 2 flops
- 1 complex multiplication = 4 real multiplications + 2 real additions = 6 flops

$\vdots$

The scaling factors do not affect big O complexities

Tips for saving computations

- Apply matrix operations wisely
- example: try this on MATLAB

```
>> A=randn(5000,2); B=randn(2,10000);
C=randn(10000,10000);
>>
>> tic; D= A*B*C; toc
Elapsed time is 12.238567 seconds.
>> tic; D= (A*B)*C; toc      % ask MATLAB to do AB
first
Elapsed time is 12.640961 seconds.
>> tic; D= A*(B*C); toc      % ask MATLAB to do BC
first
Elapsed time is 0.222270 seconds.
```

Tips for saving computations (cont'd)

Example: Compute $\mathbf{D} = \mathbf{ABC}$, $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{n \times p}, \mathbf{C} \in \mathbb{R}^{p \times p}$, $n \ll \min\{m, p\}$

-
- Compute $\mathbf{AB}$ first and then $\mathbf{D} = (\mathbf{AB})\mathbf{C}$. The flop count is

$$\mathcal{O}(mnp) + \mathcal{O}(mp^2) = \mathcal{O}(m(n+p)p) \approx \mathcal{O}(mp^2)$$

- Compute $\mathbf{BC}$ first and then $\mathbf{D} = \mathbf{A}(\mathbf{BC})$. The flop count is

$$\mathcal{O}(np^2) + \mathcal{O}(mnp) = \mathcal{O}((m+p)np)$$

- The second option is preferable when n is much smaller than m, p

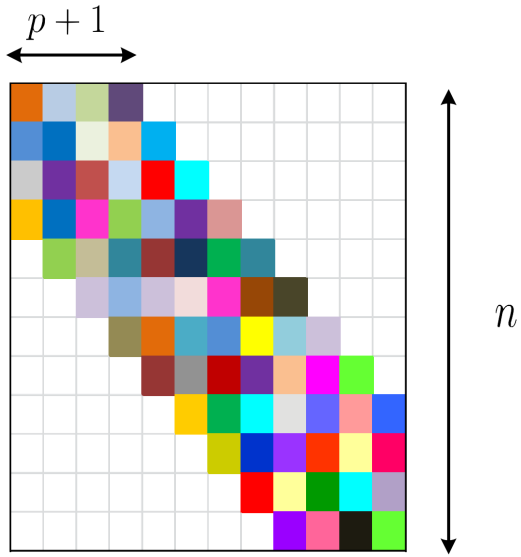
Tips for saving computations (cont'd)

- Utilize favorable matrix structures
- **Example**: Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ and suppose

  $a_{ij} = 0$ for all i, j such that $|i - j| > p$

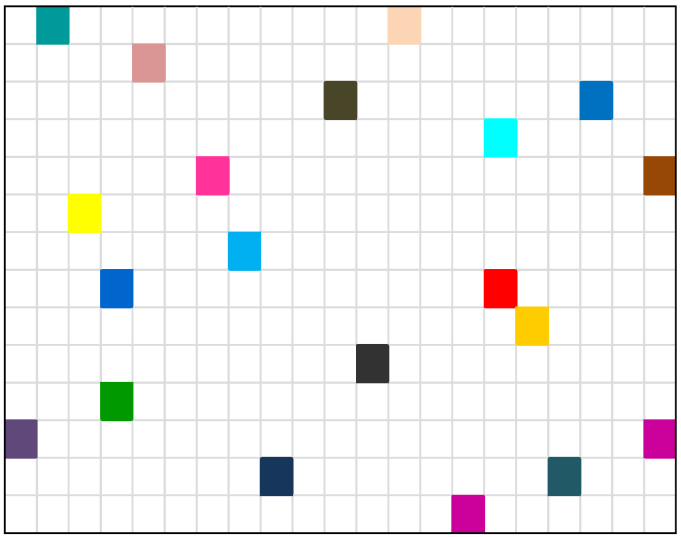
  for some integer $p > 0$

  - Such a structured $\mathbf{A}$ is called band diagonal
  - If we don't use structures, computing $\mathbf{Ax}$ requires $O(n^2)$
  - Zero flops for $+0$ or $\times 0$
  - If we use the band diagonal structures, we can compute $\mathbf{Ax}$ with $O(pn)$



Tips for saving computations (cont'd)

- Utilize sparsity
- A vector or matrix is said to be sparse if it contains many zero elements
    - Assume unstructured sparsity



Tips for saving computations (cont'd)

Let $nnz(\cdot)$ denote the number of nonzero elements of a vector/matrix

Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

- $\alpha\mathbf{x}$: $nnz(\mathbf{x})$ for $\alpha \neq 0$

- $\mathbf{x} + \mathbf{y}$: between 0 and $\min\{nnz(\mathbf{x}), nnz(\mathbf{y})\}$ flops
  $\Longrightarrow O(\min\{nnz(\mathbf{x}), nnz(\mathbf{y})\})$
  - 0 flops if the sparsity patterns of $\mathbf{x}, \mathbf{y}$ do not overlap
  - $nnz(\mathbf{x})$ flops if the index set of $\mathbf{x}$'s nonzero entries is contained in that of $\mathbf{y}$'s nonzero entries

- $\mathbf{y}^T\mathbf{x}$: between 0 and $2\min\{nnz(\mathbf{x}), nnz(\mathbf{y})\}$ flops
  $\Longrightarrow O(\min\{nnz(\mathbf{x}), nnz(\mathbf{y})\})$
  - Question: When are the minimum and maximum number of flops attained?

Tips for saving computations (cont'd)

Given $\mathbf{A}, \tilde{\mathbf{A}} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times p}$, $\mathbf{x} \in \mathbb{R}^n$,

- $\alpha\mathbf{A}$: $nnz(\mathbf{A})$
- $\mathbf{A} + \tilde{\mathbf{A}}$: between 0 and $\min\{nnz(\mathbf{A}), nnz(\tilde{\mathbf{A}})\}$
  $\Longrightarrow \mathcal{O}(\min\{nnz(\mathbf{A}), nnz(\tilde{\mathbf{A}})\})$
- $\mathbf{Ax}$ with dense $\mathbf{x}$: $nnz(\mathbf{A})$ multiplications and a number of additions that is no more than $nnz(\mathbf{A})$, so between $nnz(\mathbf{A})$ and $2nnz(\mathbf{A})$ flops
  $\Longrightarrow \mathcal{O}(nnz(\mathbf{A}))$
  - For diagonal $\mathbf{A}$, only $nnz(\mathbf{A})$ multiplications are needed, no additions, so $nnz(\mathbf{A})$ flops
- $\mathbf{AB}$: At most $2\min\{nnz(\mathbf{A})p, nnz(\mathbf{B})m\}$ flops
  $\Longrightarrow \mathcal{O}(\min\{nnz(\mathbf{A})p, nnz(\mathbf{B})m\})$

Reference: S. Boyd and L. Vandenberghe, *Introduction to Applied Linear Algebra – Vectors, Matrices, and Least Squares*, 2018. Available online at `https://web.stanford.edu/~boyd/vmls/vmls.pdf`