Medical Image Reconstruction Term II – 2010

Topic 1: Mathematical Basis Lecture 1

Professor Yasser Mostafa Kadah

Topic Today

- Matrix Computations
  - How to solve linear system of equation $Ax=b$ on a computer !

Matrix Vector Multiplication

- Consider an nxm matrix A and nx1 vector x:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

- Matrix vector multiplication b=Ax is given as,

$$b_i = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{im}x_m = \sum_{j=1}^{m} a_{ij}x_j$$

Matrix Vector Multiplication

- If b = Ax, then b is a linear combination of the columns of A.

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{n2} \end{bmatrix} x_2 + \cdots + \begin{bmatrix} a_{1m} \\ a_{2m} \\ \vdots \\ a_{nm} \end{bmatrix} x_m$$

- Computer pseudo-code:

$b \leftarrow 0$
for $j = 1, \ldots, m$
  for $i = 1, \ldots, n$
    $b_i \leftarrow b_i + a_{ij} x_j$

Computational Complexity: Flop Count

- Real numbers are normally stored in computers in a floating-point format.
- Arithmetic operations that a computer performs on these numbers are called floating-point operations (flops)
- Example: Update $b_i \leftarrow b_i + a_{ij}x_j$
  - 1 Multiplication + 1 Addition = 2 flops
  - Matrix-vector multiplication : 2 nm flops or O(nm)
  - For nxn matrix x nx1 vector: $O(n^2)$ operation
    - Doubling problem size quadruples effort to solve

Matrix-Matrix Multiplication

- If A is an nxm matrix, and X is mxp, we can form the product B = AX, which is nxp such that,

$$b_{ij} = \sum_{k=1}^{m} a_{ik} x_{kj}$$

- Pseudo-code:

$$
\begin{aligned}
&B \leftarrow 0 \\
&\text{for } i = 1, \ldots, n \\
&\quad \text{for } j = 1, \ldots, p \\
&\quad\quad \text{for } k = 1, \ldots, m \\
&\quad\quad\quad b_{ij} \leftarrow b_{ij} + a_{ik} x_{kj}
\end{aligned}
$$

  - 2mnp flops
  - Square case: $O(n^3)$

Systems of Linear Equations

- Consider a system of n linear equations in n unknowns

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

- Can be expressed as Ax=b such that

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Systems of Linear Equations

- Theorem: Let A be a square matrix. The following six conditions are equivalent

(a) A^{-1} *exists.*

(b) *There is no nonzero* y *such that* $Ay = 0$.

(c) *The columns of* A *are linearly independent.*

(d) *The rows of* A *are linearly independent.*

(e) $\det(A) \neq 0$.

(f) *Given any vector* b, *there is exactly one vector* x *such that* $Ax = b$.

Methods to Solve Linear Equations

- Theoretical: compute A^{-1} then premultiply by it:

$$A^{-1} A x = A^{-1} b \quad \Rightarrow \quad x = A^{-1} b$$

- Practical: A^{-1} is never computed!
  - Unstable
  - Computationally very expensive
  - Numerical accuracy
- Gaussian elimination ??
  - Computational complexity?
  - Numerical accuracy?
- Explore ways to make this solution simpler

Elementary Operations

- A linear system of equation Ax=b remains the same if we:
  - Add a multiple of one equation to another equation.
  - Interchange two equations.
  - Multiply an equation by a nonzero constant.
- Explore ways of solving the linear system using these elementary operations
- Gaussian elimination is an example of such method

$$\left[\begin{array}{c|ccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ \hline 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right] \quad \left[\begin{array}{c|c|ccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ \hline 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \hline 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right]$$

Triangular systems of equations

- Lower triangular systems

$$G = \begin{bmatrix} g_{11} & 0 & 0 & \cdots & 0 \\ g_{21} & g_{22} & 0 & \cdots & 0 \\ g_{31} & g_{32} & g_{33} & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ g_{n1} & g_{n2} & g_{n3} & \cdots & g_{nn} \end{bmatrix}$$

- Consider linear system Gy=b: Forward Substitution

$$y_1 = b_1 / g_{11}$$

$$y_2 = (b_2 - g_{21} y_1) / g_{22}$$

- Upper triangular system: Backward Substitution
- Efficient computation for such special matrices

Cholesky Decomposition

- Cholesky Decomposition Theorem: Let A be positive definite. Then A can be decomposed in exactly one way into a product $A = R^TR$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} r_{11} & 0 & 0 & \cdots & 0 \\ r_{12} & r_{22} & 0 & \cdots & 0 \\ r_{13} & r_{23} & r_{33} & & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ r_{1n} & r_{2n} & r_{3n} & \cdots & r_{nn} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & r_{2n} \\ 0 & 0 & r_{33} & \cdots & r_{3n} \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & & r_{nn} \end{bmatrix}$$

Solution Using Cholesky Decomposition

- Consider problem Ax=b
- Then, use Choleskly decomposition to put $A= R^TR$
- Then, $A\,x = b \Rightarrow R^TRx= b$
- Let $Rx= y$ then solve $R^Ty= b$
  - triangular system of equations that is easy to solve
- Then, solve $Rx=y$
  - Another triangular system of equations that is easy to solve

LU Decomposition

- LU Decomposition Theorem: Let A be an nxn matrix whose leading principal submatrices are all nonsingular. Then A can be decomposed in exactly one way into a product A= L U as:

$$
\begin{bmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\
a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\
\vdots & \vdots & \vdots & & \vdots \\
a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn}
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
l_{21} & 1 & 0 & \cdots & 0 \\
l_{31} & l_{32} & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & & \vdots \\
l_{n1} & l_{n2} & l_{n3} & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\
0 & u_{22} & u_{23} & \cdots & u_{2n} \\
0 & 0 & u_{33} & \cdots & u_{3n} \\
\vdots & \vdots & \vdots & & \vdots \\
0 & 0 & 0 & \cdots & u_{nn}
\end{bmatrix}
$$

Vector Norm

- Measure of distance
- Definition:

A *norm* (or *vector norm*) on $\mathbb{R}^n$ is a function that assigns to each $x \in \mathbb{R}^n$ a non-negative real number $\|x\|$, called the norm of x, such that the following three properties are satisfied for all $x, y \in \mathbb{R}^n$ and all $\alpha \in \mathbb{R}$:

$$\|x\| > 0 \text{ if } x \neq 0, \quad \text{and} \quad \|0\| = 0 \qquad \text{(positive definite property)} \quad (2.1.1)$$

$$\|\alpha x\| = |\alpha| \, \|x\| \qquad \text{(absolute homogeneity)} \quad (2.1.2)$$

$$\|x + y\| \leq \|x\| + \|y\| \qquad \text{(triangle inequality)} \quad (2.1.3)$$

- Example: Euclidean norm

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}$$

Vector Norm

- General definition of p-norm:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

- Examples:
  - 2-norm: Euclidean distance
  - 1-norm: (taxicab norm or Manhattan norm)

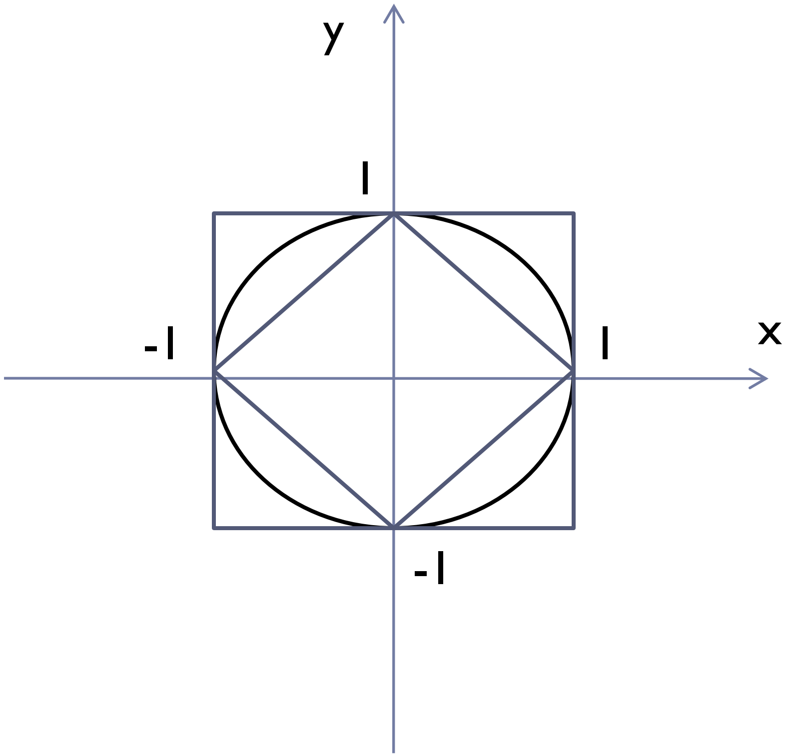
$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

  - ∞-norm:

$$\|x\|_\infty = \max_{1 \le i \le n} |x_i|$$

Vector Norm

- Example:
  - draw the circles defined by the following equations: $||x||_1 = 1$, $||x||_2 = 1$, $||x||_\infty = 1$



Matrix Norm

- A matrix norm is a function that assigns to each $A \in \Re^{n \times n}$ a real number ||A|| such that:

$$\begin{aligned} \|A\| &> 0 \text{ if } A \neq 0 && (2.1.18) \\ \|\alpha A\| &= |\alpha| \, \|A\| && (2.1.19) \\ \|A + B\| &\leq \|A\| + \|B\| && (2.1.20) \\ \|AB\| &\leq \|A\| \, \|B\| \quad \text{(submultiplicativity)} && (2.1.21) \end{aligned}$$

- Example: Frobenius norm (commonly used)

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

Matrix Norm

- Induced (operator) norm $$\| A \| = \max_{x \neq 0} \frac{\| Ax \|}{\| x \|}$$
- Special case: induced p-norm or Matrix p-norm
$$\| A \|_p = \max_{x \neq 0} \frac{\| Ax \|_p}{\| x \|_p}$$
  - Theoretically important
  - Expensive to compute
  - Frobenius norm is NOT the matrix 2-norm
- Theorem: $\| Ax \| \leq \| A \| \, \| x \|$
- Examples:

$$(a)\ \| A \|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{n} | a_{ij} |. \qquad (b)\ \| A \|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} | a_{ij} |$$

Condition Number

- Consider a linear system equation and its perturbation:

$$Ax = b \text{ and } A(x + \delta x) = b + \delta b$$

- Then, $A\delta x = \delta b$ or $\delta x = A^{-1}\delta b$
- Hence,

$$\|\delta x\| \leq \|A^{-1}\| \, \|\delta b\|$$

- Also,

$$\|b\| \leq \|A\| \, \|x\|$$

- Combining equations:

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \, \|A^{-1}\| \frac{\|\delta b\|}{\|b\|}$$

- Define the condition number as:

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

Condition Number

- Using induced matrix norm, $\kappa(A) \geq 1$
- Matrices with $\kappa(A) \geq 1000$ are considered ill-Conditioned
  - Numerical errors in solving Ax=b are amplified in the solution by the condition number
- Estimation of condition number: from eigenvalues: divide maximum eigenvalue by the minimum eigenvalue or
  - $\kappa(A) = \lambda_{max}/\lambda_{min}$
- For singular matrices, $\kappa(A) = \infty$
- Condition number improvement by scaling equations possible
  - Example:

$$\begin{bmatrix} 1 & 0 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \epsilon \end{bmatrix}$$

Roundoff Errors

- Floating point number presentation $.123456 \times 10^7$
  - Mantissa $.123456$
  - Exponent **7**
- Problems occur when adding numbers of very different scales
- If a computation results in a number that is too big to be represented, an overflow is said to have occurred.
- If a number that is nonzero but too small to be represented is computed, an underflow results.
- Machine epsilon: smallest positive floating point number s such that $fl(1+s)>1$ (Homework to compute)

Sensitivity Analysis

- Using perturbation analysis, show how stable the solution is for a particular matrix A and machine precision s.
  - Condition number describes the matrix only
  - Be careful with choice of single vs. double precision since time gain may end up causing major errors in result !

Least-Squares Problem

- To find an optimal solution to linear system of equations Ax=b that does not have to be square and it is desired to minimize the 2-norm of the residual

$$\begin{bmatrix} \phi_1(t_1) & \phi_2(t_1) & \cdots & \phi_m(t_1) \\ \phi_1(t_2) & \phi_2(t_2) & \cdots & \phi_m(t_2) \\ \phi_1(t_3) & \phi_2(t_3) & \cdots & \phi_m(t_3) \\ \vdots & \vdots & & \vdots \\ \phi_1(t_n) & \phi_2(t_n) & \cdots & \phi_m(t_n) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

  - n>m : overdetermined system (least-squares solution)
  - n<m: underdetermined system (minimum-norm solution)

Orthogonal Matrices

- An orthogonal matrix has its inverse the same as its transpose

$$QQ^T = I \qquad Q^TQ = I \qquad Q^T = Q^{-1}$$

  - Determinant = 1
  - Condition number = 1 (ideal)
  - Orthogonal transformations preserve length
  - Orthogonal transformations preserve angle
  - Example: rotators and reflectors

QR Decomposition

- Any nxn matrix A can be decomposed as a product QR where Q is an orthogonal matrix and R is an upper triangular matrix
- Solution of Ax=b is again straightforward:
  - QRx=b
  - Let Rx= y and solve Qy=b (solution is simply y= Q^Tb)
  - Then solve triangular system Rx=y as before
- Advantage of QR solution: excellent numerical stability
- Overdetermined case (A is nxm with n>m): QR decomposition is still possible with :

$$R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}$$

Singular Value Decomposition (SVD)

- Let A be an nxm nonzero matrix with rank r. Then A can be expressed as a product:

$$A = U \Sigma V^T$$

- Where:
  - U is an nxn orthogonal matrix
  - V is an mxm orthogonal matrix
  - Σ is an nxm diagonal matrix of singular values in the form:

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \sigma_r & & \\ & & & & 0 & \\ & & & & & \ddots \end{bmatrix}$$

Solution of Least Squares Using SVD

- Condition number can be shown to be equal to:

$$\kappa_2(A) = \frac{\sigma_1}{\sigma_n}$$

- In order to improve condition number, we can solve the equation after replacing the smallest singular values by zero until the condition number is low enough
  - Regularization of the ill-conditioned provlem
  - "Pseudo-inverse" or "Moore-Penrose generalized inverse"

$$A^\dagger = V\Sigma^\dagger U^T$$

- Highest numerical stability of all methods but $O(n^3)$

Computational Complexity

- Cholesky's algorithm applied to an nxn matrix performs about $n^3/3$ flops.
- LU based decomposition applied to an nxn matrix performs about $2n^3/3$ flops.
- Gaussian elimination applied to an nxn matrix performs about $2n^3/3$ flops.
- QR decomposition: $2nm^2-2m^3/3$ flops
- SVD has $O(n^3)$ flops

- All are still too high for some problems
  - Need to find other methods with lower complexity

Iterative Solution Methods

- Much less computations of $O(n^2)$
- Steepest descent based methods
- Conjugate gradient based methods

Steepest Descent Methods

- Looks for the error b-Ax and tries to remove this error in its direction

$$
\begin{aligned}
& r \leftarrow r - Ax \\
& p \leftarrow \quad ? \\
& k \leftarrow 0 \\
& \text{do until satisfied or } k = l \\
& \quad \left[\begin{aligned}
& q \leftarrow Ap \\
& \alpha \leftarrow p^T r / p^T q \\
& x \leftarrow x + \alpha p \\
& r \leftarrow r - \alpha q \\
& p \leftarrow \quad ? \\
& k \leftarrow k + 1
\end{aligned} \right. \\
& \text{if not satisfied, set flag}
\end{aligned}
$$

Set $p \leftarrow r$ to get steepest descent.

Conjugate Gradient (CG) Methods

- Removes the error in "mutually-orthogonal" directions
  - Maximum n iterations needed to reach exact solution
  - Better performance compared to steepest descent

$$
\begin{aligned}
& r \leftarrow r - Ax \\
& p \leftarrow r \\
& \nu \leftarrow r^T r \\
& k \leftarrow 0 \\
& \text{do until converged or } k = l \\
& \quad \left[\begin{array}{l}
q \leftarrow Ap \\
\mu \leftarrow p^T q \\
\alpha \leftarrow \nu / \mu \\
x \leftarrow x + \alpha p \\
r \leftarrow r - \alpha q \\
\nu_+ \leftarrow r^T r \\
\beta \leftarrow \nu_+ / \nu \\
p \leftarrow r + \beta p \\
\nu \leftarrow \nu_+ \\
k \leftarrow k + 1
\end{array} \right. \\
& \text{if not converged, set flag}
\end{aligned}
$$

Exercise

- Write a program to compute Machine Epsilon
- Look for Matlab functions that implement the topics discussed in this lecture
  - Read help
- Implement code for steepest descent and conjugate gradient methods and compare results to SVD based solution (pseudo-inverse) using only a few iterations