①

Tuesday 9 - 10:15

③ Gaussian elimination/direct methods in solving $Ax=b$

"exact"

① • Solve $Ax=b$ directly, expensive $O(n^3)$ $O(10^{-16})$
• later you will see indirect or iterative methods $O(10^{-8})$ or set tolerance cheaper
• advantage of direct: once factor $A$ can solve for any RHS $\vec{b}$.

Three cases arise when solving $Ax=b$

1) No solution: $b \notin R(A)$
2) infinitely many solutions $N(A) \neq \emptyset$ (non-trivial)
✓ 3) exactly one solution (A invertible)

① Elimination - forward and backward sub

- based on how to solve triangular systems.
- we use gaussian elimination to put $A$ into upper triangular form $U$ and then use back-substitution on $U$
• Recall a matrix is upper triangular
$\Leftrightarrow$ $\boxed{u_{ij} = 0 \quad i > j}$ 0 below diagonal

$$\begin{pmatrix} a_{11} & - & a_{1n} \\ | & & | \\ a_{m1} & - & a_{mn} \end{pmatrix} \sim \begin{pmatrix} u_{11} & u_{12} & - & u_{1n} \\ & \ddots & & | \\ & & \cancel{u_{n-1}}\,u_{n-1\,n-1} & u_{n-1\,n} \\ & & & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ | \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ | \\ b_n \end{pmatrix}$$

Start at bottom: $u_{nn} x_n = b_n \Leftrightarrow \boxed{x_n = \frac{b_n}{u_{nn}}}$

$u_{n-1n-1} x_{n-1} + u_{n-1n} x_n = b_{n-1}$

$$\boxed{x_{n-1} = \frac{b_{n-1} - u_{n-1n} x_n}{u_{n-1n-1}}}$$

known computed prior iterate

$u_{n-2n-2} x_{n-2} + u_{n-2n-1} x_{n-1} + u_{n-2n} x_n = b_{n-2}$

$$\boxed{x_{n-2} = \frac{b_{n-2} - u_{n-2,n} x_n - u_{n-2,n-1} x_{n-1}}{u_{n-2n-2}}}$$

- computationally cheap! $O(n^2)$

```
for i = n, n-1, ..., 1
  x_i = b_i
  for j = i+1, i+2, ..., n
      x_i = x_i - u_ij x_j
  end
  x_i = x_i / u_ii
end
```

- Similiarly, forward substitution for a lower triangular matrix $\boxed{l_{ij} = 0 \quad j > i}$

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & 0 \\ l_{31} & l_{32} & l_{33} & \\ | & | & & \ddots \\ l_{n1} & l_{n2} & l_{n3} & l_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ | \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ | \\ b_n \end{pmatrix}$$

Forward sub - start at the top!

$x_1 = \frac{b_1}{l_{11}}$

$x_2 = (b_2 - l_{21} x_1) / l_{22}$

$x_3 = (b_3 - l_{31} x_1 - l_{32} x_2) / l_{33} \quad \dots$

algorithm) for $i = 1, \ldots, n$
$x_i = b_i$
for $j = 1, \ldots, i-1$
$x_i = x_i - e_{ij} x_j$
end
$x_i = x_i / e_{ii}$
end

, Why do we care about triangular systems? - use gaussian elimination (as we have been doing) to transform $Ax = b$ in upper tri system $Ux = y$ and use backsubstition to solve

- The elementary row operations are applied to augmented system $\boxed{[A|b]}$ to introduce zeros below diagonal without changing solution

Elementary row ops

✓ 1) Multiply row by scalar
2) Add scalar multiple of 1 row to another
3) permute rows

$$A = \begin{pmatrix} a_{11} & a_{12} & \text{—} & a_{1n} \\ | & & & | \\ a_{m-11} & & & a_{m-1n} \\ a_{m1} & \text{—} & & a_{mn} \end{pmatrix}$$

1) 0 all elements in first column below the diagonal

2) make sure pivot $\neq 0$ if it is swap in row
Three types of pivoting: (don't want pivot too small)
① partial: swap w/ largest in row (most comm
② rook: swap w/ largest in row or column
③ full: swap w/ largest in full submatrix (expensive)

$$\boxed{row(i') = row(i) - \frac{a_{i1}}{pivot1} \, row(i)} \quad \forall i \ 2 \le i \le n$$

$$A' = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \text{---} & a_{1n} \\ 0 & & & & \\ | & & A'[2:m, 2:n] & & \\ 0 & & & & \end{bmatrix}$$

- continue inductively on $(m-1) \times (n-1)$ submatrix

## ② LU decomposition

- So far, we have always written GE as equation on row operations. we can write it in terms of matrix L. (premult by L) $LA = U$

- look at first ~~row~~ transformations $L_1$
- $1^{st}$ row of A should not be affected by $L_1$: $\vec{l_1} = [1, 0, 0 \text{ --- } 0]$

$$L_1 = \begin{pmatrix} l_1 \\ | \\ l_n \end{pmatrix}$$

$$\boxed{a'_{2j} = a_{2j} - \left(\frac{a_{21}}{a_{11}}\right) a_{1j}}$$

$$l_2 = \begin{bmatrix} -\frac{a_{21}}{a_{11}} & 1 & 0 & \text{---} & 0 \end{bmatrix}$$

↑ Column1 of $l_2$ multiplied by row 1 of A

↑ column 2 of L / row 2 of A

⋮

$$\boxed{L_1 = \begin{bmatrix} 1 & & & 0 \\ -a_{21}/a_{11} & 1 & & \\ -a_{31}/a_{11} & & \ddots & \\ | & 0 & & \\ -a_{n1}/a_{nn} & & & 1 \end{bmatrix}}$$

$A' = L_1 A$

2) $A'' = L_2 A' = L_2 L_1 A$

$$L_2 = \begin{bmatrix} 1 & & & & \\ 0 & 1 & & & 0 \\ | & -\frac{a'_{32}}{a'_{22}} & \ddots & & \\ | & | & 0 & \ddots & \\ 0 & -\frac{a'_{n2}}{a'_{22}} & & & 1 \end{bmatrix}$$

- After $\boxed{n-1}$ elimination steps:

$$\boxed{U = L_{n-1} L_{n-2} \cdots L_2 L_1 A}$$

$$\Rightarrow A = (L_{n-1} L_{n-2} \cdots L_2 L_1)^{-1} U$$
$$= \boxed{\underbrace{L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1}}_{L} U} = LU$$

- since inverse of lower triangular matrix is lower tri and product of lower tri $L$ is lower tri

## Special properties of $L_i$

- easy to compute inverses
- each takes a row & subtracts constant times another row
- since the inverse should undo whatever $L_1$ - it should add that constant

$\Rightarrow$ only a sign difference $L_1^{-1} = \begin{bmatrix} 1 & & \\ \frac{a_{21}}{a_{11}} & 1 & 0 \\ | & \ddots & \\ \frac{a_{n1}}{a_{11}} & 0 & 1 \end{bmatrix}$

- Also simple to multiply - multiplication factors preserved at original location

$$L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} = \begin{bmatrix} 1 & & & \\ \frac{a_{21}}{a_{11}} & 1 & & \\ \vdots & \frac{a'_{32}}{a'_{22}} & \ddots & \\ \frac{a_{n1}}{a_{11}} & \frac{a'_{n2}}{a'_{22}} & \cdots & 1 \end{bmatrix} = \boxed{L}$$

- fills in each column

⇒ L always has ones on diagonals (unit lower triangular)

- U has pivots on diagonal

$\det(A) = \det(LU) = \det(L)\det(U)$
$= \det(U)$

② LU with pivoting: $PA = LU$

P, permutation matrix takes care of row swaps.

$\det(P) = \pm 1$ so

$\det(PA) = \det(P)\det(A) = \det(U)$

$\Leftrightarrow \det(A) = \pm\det(U)$

Why do we need P?

Gaussian elimination from top to bottom can fail (ignore 0 & small pivots) numerically don't want to introduce large numbers

Ex. $A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ • swap rows 1 & 2 to get upper triangular.

- How to use to solve $Ax = b$
- Once have $A = LU$ can use to solve for any RHS $b$

$Ax = L(Ux) = b$. Let $Ux = y$

① Solve $Ly = b$ for $y$ (forward sub)
② Solve $Ux = y$ for $x$ (back sub)

Further properties of matrix form.
Let us consider a vector $x \in \mathbb{R}^n$ and zeroing entries $i > j$

$$L_j \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ x_{j+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

column of A

We can write it succinctly in outer product form:

Recall $(\vec{e_i})_j = \begin{cases} 0 & j \neq i \\ 1 & j > i \end{cases}$ $\quad (0 - 1 \; 0 - 0)$ $\quad i$

$= I - \left(\frac{x_{j+1}}{x_j}\right) e_{j+1} e_j^T$ (To zero out $x_{j+1}$)

↑ look at $(j+1)^{th}$ element

$\left[I - \left(\frac{x_{j+1}}{x_j}\right) e_{j+1} e_j^T\right] x$

$$(0 - 1 \; 0 - 0) \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{pmatrix} = x_j$$
$\quad j$

$= x - \frac{x_{j+1}}{x_j} e_{j+1} \underbrace{(e_j^T x)}_{x_j}$

$$= x - x_{j+1} e_{j+1} = \begin{pmatrix} x_1 \\ \vdots \\ x_{j+1} \\ \vdots \\ x_n \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ x_{j+1} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ 0 \\ x_{j+2} \\ \vdots \\ x_n \end{pmatrix}$$

3x3 case $L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 \\ x_2/x_1 \\ x_3/x_1 \end{bmatrix} [1\ 0\ 0] =$

Example $= \begin{bmatrix} 1 & 0 & 0 \\ -x_2/x_1 & 1 & 0 \\ -x_3/x_1 & 0 & 1 \end{bmatrix}$ ✓

To zero all elements below $j$ put in $j^{th}$ column

we define $\boxed{L_j = I - \tau e_j^T}$

$$\boxed{\tau_k = \begin{cases} 0 & k \le j \\ \frac{x_k}{x_j} & k > j \end{cases}}$$

Gaussian transformation

- We can use closed form to prove inverse property:

$$\boxed{L_j^{-1} = I + \tau e_j^T}$$

Proof: $(I - \tau e_j^T)(I + \tau e_j^T)$

$= I - \cancel{\tau e_j^T + \tau e_j^T} - \tau(e_j^T \tau)e_j^T$

But $e_j^T \tau = 0$, since $\tau_j = 0$ by definition.

(3) We have special factorizations to exploit special properties in matrices:

1) symmetric $A = A^T$ (less storage)

2) positive definite: $\forall \vec{x} \neq \vec{0} \in \mathbb{R}^n$ $\boxed{x^T A x > 0}$

$A > 0$

- all eigenvalues are positive
- positive diagonals $a_{ii} > 0$
- invertible

Ex: $A^T A$ is positive definite

$x^T A^T A x = (Ax)^T Ax = \|Ax\|_2^2 \geq 0$

$= 0 \iff x = 0$

- negative definite: $z^T M z < 0 \quad z \neq 0$
- positive semi-definite: $x^T A x \geq 0$ (could have 0 eigenvalues) non-invertible
- indefinite: $x^T A x > 0$ and $y^T A y < 0 \quad \exists x, y$
  positive & negative eigenvalues

- $A_r$ submatrix positive definite

Special Factorizations

A matrix is symmetric positive definite (spd) $\Leftrightarrow \exists$ lower triangular matrix $L$ w/ real positive diagonal entries s.t.

$$\boxed{A = LL^T = R^T R} \quad L \text{ unique}$$

Cholesky Factorization

- no need for pivoting - positive diagonal so can take sqrt

Symmetric Case
- In GE, we use row operations to introduce 0's below diagonal in each column
- To maintain symmetry we can apply the same operations to columns of matrix to introduce 0's in first row

- $\boxed{LDL^T}$ Since $A$ is symmetric, it is clear that strictly upper triangular portion of $U$ must be symmetric to strictly lower triangular part of $L$
- we can't write $U = L^T$ since diagonals of $L$ & $U$ aren't the same

- L is unit lower w/ all 1s on diagonal
  & U has pivots on diagonal
- Store pivots on diagonal $D$
  Let $\boxed{U = DL^T}$ $\Rightarrow$ $A = LU = \boxed{LDL^T}$

Special case) Cholesky - positive definite

all diagonal elements $a_{ii} > 0$, so we can take the sqrt

$A = L D^{\frac{1}{2}} D^{\frac{1}{2}} L^T$

$= \boxed{(LD^{\frac{1}{2}})(LD^{\frac{1}{2}})^T}$ , since $D^T = D$

$D^{-1} = (\frac{1}{d_{11}}, \dots, \frac{1}{d_{nn}})$

$= \boxed{L_1 L_1^T}$

since nonzero no pivoting necessary for Cholesky

- Exercise: symmetric and triangular $\Rightarrow$ diagonal

(4) Conditioning and perturbation studies for $Ax = b$

- ill-conditioning

(i) will small perturbations in the data & round-off error lead to small or large errors in solution vector $x$?

First, we must define matrix and vector norms

- norm on a vector space $V$ is a function

$\|\cdot\|: V \to \mathbb{R}^+ \cup \{0\}$ that satisfies:

1) absolute homogeneity: $\|\alpha v\| = |\alpha|\,\|v\|$ $\forall \alpha \in \mathbb{R}$, $v \in V$

2) subadditivity (triangle inequality)
$\|u+v\| \le \|u\| + \|v\|$

3) nondegeneracy $\|v\| = 0 \Leftrightarrow v = 0$

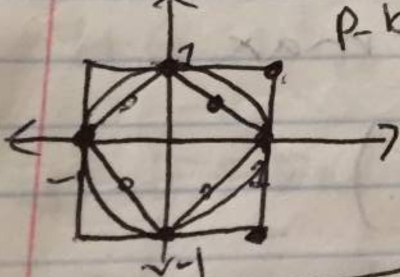
It gives the length of a vector & is a convex function

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

1) Standard Euclidean 2 norm
$\|\vec{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2} = \sqrt{x^T x}$ ← dot product

2) Infinity (maximum) norm
$\|x\|_\infty = \max_{1 \le i \le n} |x_i|$

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




p-balls

## Matrix norms

(1) Frobenius norm is extension of Euclidean norm: $\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2\right)^{\frac{1}{2}}$

sqrt of sum of all elements squared

$= \boxed{\sqrt{tr(A^T A)}} = \sqrt{tr(AA^T)}$

$tr(A^TA) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^T a_{ji} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ji}^2$ ✓

- Submultiplicative) $\|AB\|_F \leq \|A\|_F \|B\|_F$

2-norm) $\|A\|_2 = \max_{x \neq 0} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2}$

$= \sup_{\|x\|_2 = 1} \|Ax\|_2$

- max amplitude of length of vector x when subjected to A
- difficult to calculate - will see later that it is the maximum singular value SVD

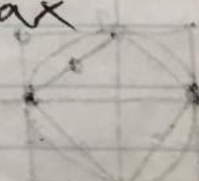
Other p-norms which are simpler to calculate:

- key idea is to express in terms of vector norms

- $\|A\|_\infty = \sup_{\|x\|_\infty = 1} \|Ax\|_\infty$ = max row sum

$= \max_{1 \leq i \leq m} \sum_{j=1}^{n} |a_{ij}|$ | Ex: $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ $\boxed{\|A\|_\infty = 7}$

- $\|A\|_1 = \sup_{\|x\|_1 = 1} \|Ax\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{m} |a_{ij}|$ = max col sum (Eg: $\|A\|_1 = 6$)



Proof: ~~Take~~ We want $\max \|Ax\|_1$

s.t. $\|x\|_1 = \sum_{j=1}^{n} |x_j| = 1$

$\|Ax\|_1 = \| \sum_{j=1}^{n} x_j \vec{a}_j \|_1 \leq \sum_{j=1}^{n} |x_j| \, \|a_j\|_1$

triangle inequality & scalar

$$\le \underbrace{\sum_{j=1}^{n} |x_j|}_{1} \left( \max_{1\le j\le n} \|a_j\|_1 \right) \to \max_{1\le j\le n} \|a_j\|_1$$

$$= \boxed{\max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|}$$

This is achieved when $x \to e_j$

<u>Important Properties</u>

1) All induced matrix norms are <u>submultiplicative</u>: $\|AB\| \le \|A\| \|B\|$

$\|Ax\| \le \|A\| \|x\|$

2) 2 norm and Frobenius are invariant under orthogonal transformations

$\|QA\|_2 = \|A\|_2$ } for orthogonal matrix

$\|QA\|_F = \|A\|_F$ } $Q^T Q = I = QQ^T$

Proof: $\|QAx\|_2^2 = x^T A^T \underbrace{Q^T Q}_{I} A x$

$= (Ax)^T Ax = \|Ax\|_2^2$

$\|QA\|_F^2 = tr(A^T \underbrace{Q^T Q}_{I} A) = tr(Q^T Q)$ ✓

① Effect on $\vec{x}$ from perturbation in $\vec{b}$

$A\vec{x} = b$

$A\vec{y} = \delta\vec{b}$ ← perturbation in $\vec{b}$

$\vec{y} = \vec{x} + \delta\vec{x}$

$\to \boxed{A(\vec{x} + \delta\vec{x}) = \vec{b} + \delta\vec{b}}$

* How big is relative change in $\vec{x}$ as a result of relative change in $\vec{b}$?

$\frac{\|\delta \vec{x}\|}{\|\vec{x}\|}$ related $\frac{\|\delta \vec{b}\|}{\|\vec{b}\|}$ ?

Subtract 2 equations: $A(x+\delta x) = b + \delta b$
$Ax = b$

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

use submultiplicativity:
$\boxed{\|\delta x\| = \|A^{-1}\delta b\| \leq \|A^{-1}\| \|\delta b\|}$

$Ax = b \Rightarrow \|Ax\| \leq \|A\| \|x\| = \|b\|$

$\Rightarrow \boxed{\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}}$

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

At worst the relative change is
$\boxed{\|A\| \|A^{-1}\| = \kappa(A)}$ — condition number

ill-conditioned matrices: $\kappa(A) = \infty$

Ex: $A = \begin{bmatrix} 1 & 1 \\ 1 & 1.001 \end{bmatrix}$ $\boxed{A^{-1} = \begin{bmatrix} 1001 & -10000 \\ -10000 & 10000 \end{bmatrix}}$

close to being singular

$\Rightarrow \kappa(A) \approx 40{,}000$ in Frobenius norm

- ill-conditioning is a property of the system, not algorithm. Check

$K(A)$ ~~before~~ when solving matrix vector eqn

② Round-off error accumulation (LU)

Ex: from small pivot

Solve $A\vec{x} = \vec{b}$ $\quad A = \begin{bmatrix} 0.01 & 1 \\ 1 & -1 \end{bmatrix}$ $\quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$K(A) \approx 3$ well-conditioned

$\sim \begin{bmatrix} 0.01 & 1 \\ 0 & -101 \end{bmatrix}$ $\quad c = \begin{bmatrix} 1 \\ -100 \end{bmatrix}$ Solve $Ux = c$

$$\boxed{x = \begin{bmatrix} 0.99099099 \\ 0.99099099 \end{bmatrix}}$$

Now suppose we can only work w/ 2 pt floating pt arithmetric

- $-101$ truncated to $-100$

$\Rightarrow \boxed{x_2 = 1}$ and $\frac{1}{100} x_1 = 1 - x_2 = 1 - 1 = 0$

$\boxed{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}$ $\Rightarrow$ huge change in answr!

How to write in floating point (2 digits)

$\boxed{0.01 = 10 \times 10^{-3}}$ $\qquad -101 = -10 \times 10^{-3}$

can't store the additional 1

$\boxed{1 = 10 \times 10^{-1}}$

Imagine this on a real example of $\varepsilon = 10^{-16}$ machine precision.

* problem is caused by small pivot 0.01. The large multiplication (100), Info in

$a_{22} = -1$ is overwhelmed + truncated by large $-100$ subtraction
$\Rightarrow$ catastrophic cancellation. (CC)

To avoid CC: swap rows for larger pivot:

$$-\tfrac{1}{100}\begin{bmatrix} 1 & -1 \\ 0.01 & 1 \end{bmatrix} x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\sim \left[\begin{array}{cc|c} 1 & -1 & 0 \\ 0 & \underset{\substack{\uparrow \\ \text{truncated} \\ \text{rounding error}}}{1.01} & 1 \end{array}\right]$$

$$\sim \left[\begin{array}{cc|c} 1 & -1 & 0 \\ 0 & 1 & 1 \end{array}\right] \text{ w/ rounding error}$$

$x_2 = 1 \quad x_1 = x_2 = 1 \quad \boxed{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}$

$\Rightarrow$ very small perturbation of exact

* need to avoid small pivots
- partial pivoting is most common strategy

o even well-conditioned matrices $K(A) \sim 1$ can suffer from round-off error (algorithmic design)

$\boxed{PA = LU}$ for more stable LU