Recap regarding solving matrix eqs.

- **Recap**: We've been discussing how to solve matrix eq. of form $A\bar{v} = \bar{g}$

  Here: More common notation: $A\bar{x} = \bar{b}$

  - Gaussian elimination

    - Can be used to solve $A\bar{x}=\bar{b}$ for general (dense) A, $\mathcal{O}(N^3)$ FLOPs, or more accurately, $\mathcal{O}(\frac{2}{3}N^3)$

    - We've only looked at special cases, for tridiagonal A. (More efficient)

  - Today: LU decomposition

  - Later: Iterative methods for solving $A\bar{x} = \bar{b}$

- Classification:

  Direct methods

  - Gives in theory the exact answer in a finite number of steps.
  - In practice, can suffer from num. instabilities
  - Typically works with the entire matrix at once
    ↳ keeps the full matrix in memory

  - Gauss elim
  - LU decomp.

  Iterative methods

  - Iterate closer and closer to exact answer, but will never get there exactly.
  - Can often work without full matrix in memory, and are less susceptible to round-off errors.

  - Jacobi's it. meth.
  - Gauss-Seidel
  - Relaxation methods

Lower-upper (LU) decomposition

- We'll introduce it as an approach for solving $A\bar{x} = \bar{b}$
- Actually a starting point for many different matrix tasks
- Plan:
  1) What is it?
  2) What is it good for? + What's the difficulty?
  3) An algorithm for doing it

1) What is LU decomp.?

- <u>Theorem:</u> If A is non-singular ($\leftrightarrow$ invertible $\leftrightarrow$ non-zero eigenvals) then A can be written as

$$A = LU$$

where L is <u>lower triangular</u> and U is <u>upper triang.</u>

- <u>Ex:</u> A is 4×4

$$\underbrace{\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{\substack{[a_{ij}] \\ \text{16 elements}}} = \underbrace{\underbrace{\begin{bmatrix} \cdot & & & \\ \cdot & \cdot & & 0 \\ \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{[l_{ij}]} \times \underbrace{\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot \\ & 0 & \cdot & \cdot \\ & & & \cdot \end{bmatrix}}_{[u_{ij}]}}_{\text{20 elements}}$$

- 16 eqs. for 20 unknowns
- under constrained
- can choose 4 elements freely to get unique solution
- common to set $l_{ii} = 1$

$$\Rightarrow \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix},$$

- Comp. complexity: To determine L and U for a given $N \times N$ A $(N \times N)$ is $\mathcal{O}(N^3)$, or more precisely $\mathcal{O}(\frac{2}{3}N^3)$

  Note: Complexity of the decomposition $A = LU$ is the same as for solving $A\bar{x} = \bar{b}$ with Gaussian elim.

2) What is LU decomp. good for?

Assume we have performed LU decomp. We can now

- solve $A\bar{x} = \bar{b}$, $\mathcal{O}(N^2)$
- very easily find $\det(A)$, $\mathcal{O}(N)$
- find A^{-1}, $\mathcal{O}(N^3)$. (Would have cost $\mathcal{O}(N^4)$ by doing it as N matrix eqs solved with Gaussian elim.)

- Solving $A\bar{x} = \bar{b}$ with LU decomp:
  - Have $A = LU$
  - Thus $A\bar{x} = \underbrace{LU\bar{x} = \bar{b}}$

    Solve for $\bar{x}$ in two steps:

    Define $\bar{\omega} \equiv U\bar{x}$ (don't know what $\bar{x}$ is, so don't know $\bar{\omega}$)

    1. Solve $L\bar{\omega} = \bar{b}$ for $\bar{\omega}$
    2. Solve $U\bar{x} = \bar{\omega}$ for $\bar{x}$ Done!

① Solve $L\bar{w} = \bar{b}$ for $\bar{w}$

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

solve by forward subst:

- $l_{11} w_1 = b_1 \implies \boxed{w_1 = \frac{1}{l_{11}} b_1}$

- $l_{21} w_1 + l_{22} w_2 = b_2 \implies \boxed{w_2 = \frac{1}{l_{22}} \left[b_2 - l_{21} w_1 \right]}$

- Similarly:

$$w_3 = \frac{1}{l_{33}} \left[b_3 - l_{31} w_1 - l_{32} w_2 \right]$$

$$w_4 = \frac{1}{l_{44}} \left[b_4 - l_{41} w_1 - l_{42} w_2 - l_{43} w_3 \right]$$

General: $$\boxed{w_i = \frac{1}{l_{ii}} \left[b_i - \sum_{j=1}^{i-1} l_{ij} w_j \right]}$$

$$\left(\text{Count FLOPs: } \sum_{i=1}^{N} (2i-1) = N^2 \right)$$

which is less than $O(N^3)$ for the decomp.

- Now we have $\bar{w}$ and can move to step 2

② Solve $U\bar{x} = \bar{w}$ for $\bar{x}$

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

Solve for $\bar{x}$ by <u>back subst.</u>

- $u_{44} x_4 = w_4 \Rightarrow \boxed{x_4 = \frac{1}{u_{44}} w_4}$

- $u_{33} x_3 + u_{34} x_4 = w_3 \Rightarrow \boxed{x_3 = \frac{1}{u_{33}} \left[w_3 - u_{34} x_4 \right]}$

- Similarly:

$$x_2 = \frac{1}{u_{22}} \left[w_2 - u_{23} x_3 - u_{24} x_4 \right]$$

$$x_1 = \frac{1}{u_{11}} \left[w_1 - u_{12} x_2 - u_{13} x_3 - u_{14} x_4 \right]$$

<u>General:</u>

$$\boxed{\begin{aligned} x_N &= \frac{1}{u_{NN}} w_N \\ x_i &= \frac{1}{u_{ii}} \left[w_i - \sum_{j=i+1}^{N} u_{ij} x_j \right], \quad i = N-1, N-2, \dots, 1 \end{aligned}}$$

⊗ Ended Thursday lecture here

Again, $O(N^2)$ FLOPs, so forward subst + back subst is $O(N^2)$.

<u>Conclusion:</u>

If we have $A = LU$,

we can solve $A\bar{x} = \bar{b}$ by two-step procedure

① $L\bar{w} = \bar{b}$ → find $\bar{w}$ by forward subst

② $U\bar{x} = \bar{w}$ → find $\bar{x}$ by back subst

$O(N^2)$

- A difficulty

  - We need to store the full matrix A $(N \times N)$ in memory for the LU decomp.

    ↳ $N \times N$ floating-point numbers

    ↳ $N^2 \times 8$ bytes
    (64 bits)

Ex: $N = 10^4$

Need $10^4 \times 10^4 \times 8$ bytes

$= 8 \times 10^8$ bytes

$\approx 10^9$ bytes

$= 1$ GB

of memory

  - Also, the decomp. will be slow $O(N^3)$ when N is large...

- Easy to find $\det(A)$

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

$$= \det(L) \cdot \det(U)$$

$$\det(A) = 1 \times (u_{11}\, u_{22} \cdots u_{NN})$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \bullet & 1 & 0 \\ \bullet & \bullet & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & 0 & \bullet \end{bmatrix}$$

$$\det(A) = \prod_{i=1}^{N} u_{ii}$$

or

$$\log(\det(A)) = \sum_{j=1}^{N} \log(u_{ii})$$

- Can find A^{-1} at $\mathcal{O}(N^3)$ compl

- We know:
  - $$A^{-1}A = I = \begin{bmatrix} 1 & & 0 \\ & 1 & \\ 0 & & 1 \end{bmatrix} = AA^{-1}$$
  - $A = LU$
- Will find A^{-1}
- Write as column vectors

$$A^{-1} = \begin{bmatrix} \underbrace{\begin{bmatrix} \circ \\ \circ \\ \circ \\ \circ \end{bmatrix}}_{\bar{a}_1^{-1}} \cdots \underbrace{\begin{bmatrix} \circ \\ \circ \\ \circ \\ \circ \end{bmatrix}}_{\bar{a}_4^{-1}} \end{bmatrix} = \begin{bmatrix} \bar{a}_1^{-1} & \bar{a}_2^{-1} & \bar{a}_3^{-1} & \bar{a}_4^{-1} \end{bmatrix}$$

where e.g. $\bar{a}_1^{-1} = \begin{bmatrix} a_{11}^{-1} \\ a_{21}^{-1} \\ a_{31}^{-1} \\ a_{41}^{-1} \end{bmatrix}$

- Know that

$$AA^{-1} = \underbrace{LU}_{A} \times \underbrace{\begin{bmatrix} \bar{a}_1^{-1} & \bar{a}_2^{-1} & \bar{a}_3^{-1} & \bar{a}_4^{-1} \end{bmatrix}}_{A^{-1}} = \underbrace{\begin{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix}}_{I}$$

- This is four matrix eqs.

1) $(LU)\,\bar{a}_1^{-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$\vdots$

4) $(LU)\,\bar{a}_4^{-1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

so four eqs of the form
$$(LU)\,\bar{x} = \bar{b}$$

- In general, to find A^{-1} when we have $A=LU$ requires solving N eqs. of the form $(LU)\bar{x}=\bar{b}$

- Each eq. takes $\mathcal{O}(N^2)$ FLOPs

$\Rightarrow$ Can find A^{-1} in $\mathcal{O}(N^3)$ FLOPs, which is the same as complexity for doing the decomp. $A=LU$

$\Rightarrow$ In total, LU decomp + finding A^{-1} is $\underline{\mathcal{O}(N^3)}$

(would have been $\mathcal{O}(N^4)$ if we had solved the N eqs. using Gaussian elim.)

<u>Note</u>:
Basis for many ML methods, like Gaussian Proc.

3) An algorithm for LU decomp.

- $A = LU$
- How to determine the elements in L and U

$$\begin{bmatrix} \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{bmatrix} & \begin{matrix} a_{12} & a_{13} & a_{14} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{matrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}$$

- First column: a_{i1}

- $a_{11} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{11} \\ 0 \\ 0 \\ 0 \end{bmatrix} = u_{11}$ $\qquad \boxed{u_{11} = a_{11}}$

- $a_{21} = \begin{bmatrix} l_{21} & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{11} \\ 0 \\ 0 \\ 0 \end{bmatrix} = l_{21} u_{11}$ $\qquad \boxed{l_{21} = \dfrac{a_{21}}{u_{11}}}$

Note: what if $u_{11} \approx 0$?

- $a_{31} = l_{31} u_{11}$ $\qquad \boxed{l_{31} = \dfrac{a_{31}}{u_{11}}}$

- $a_{41} = l_{41} u_{11}$ $\qquad \boxed{l_{41} = \dfrac{a_{41}}{u_{11}}}$

- Second column : a_{i2}

  - $$a_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{12} \\ u_{22} \\ 0 \\ 0 \end{bmatrix} = u_{12}$$

    $$\Rightarrow \boxed{u_{12} = a_{12}}$$

  - $$a_{22} = \begin{bmatrix} l_{21} & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{12} \\ u_{22} \\ 0 \\ 0 \end{bmatrix} = u_{12} l_{21} + u_{22}$$

    $$\Rightarrow \boxed{u_{22} = a_{22} - u_{12} l_{21}}$$

  - $$a_{32} = \begin{bmatrix} l_{31} & l_{32} & 1 & 0 \end{bmatrix} \begin{bmatrix} u_{12} \\ u_{22} \\ 0 \\ 0 \end{bmatrix} = u_{12} l_{31} + u_{22} l_{32}$$

    $$\Rightarrow \boxed{l_{32} = \frac{a_{32} - l_{31} u_{12}}{u_{22}}}$$

  - $a_{42} = \ldots$ $$\Rightarrow \boxed{l_{42} = \frac{a_{42} - l_{41} u_{12}}{u_{22}}}$$

- Cont. for third and fourth columns ...

- General algorithm:

$$\boxed{l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}}{u_{jj}} \quad , \quad i > j}$$

$$\boxed{u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \quad , \quad i \leq j}$$

- Pivoting Need to avoid $u_{ii} \approx 0$ for numerical stability.

  - Use permutation matrix to interchange rows &

    Instead of $A = LU$

    will have $\underline{A = PLU}$ or equiv. $P^T A = LU$

    P : pivot matrix

    $P^T = P^{-1} \Leftrightarrow PP^T = I$