Next: Numerical Linear Algebra

Basically, the only math your computer can do is linear algebra. There are very few instances when non-linear problems are solved without some element of linear algebra.

What are the standard tools needed in linear algebra?

- Products:
  - vector-vector (inner product)
  - matrix-vector
  - matrix-matrix
- Solutions:
  - solve $A\vec{x} = \vec{b}$
  - minimize $\| A\vec{x} - \vec{b} \|$ (least squares)
- Factorizations
  - $A = LU$ - $A = USV^T$
  - $A = QR$
- Eigen computations
  - Find all $\lambda_i, \vec{v}_i$ s.t. $A\vec{v}_i = \lambda_i \vec{v}_i$.

Efficient methods for doing all these calculations are the building blocks of almost all scientific computing.

Tell Levi Strauss anecdote.

Ignore things like AB, $A\vec{x}$, $\vec{u}^T\vec{v}$ for now, these are computations that merely need to be optimized, a CS endeavor.

Note: Computational cost: $A \sim n \times n$, $\vec{u}, \vec{v} \sim n \times 1$

$$\vec{u}^T\vec{v} \sim O(n) \text{ flops}$$
$$A\vec{x} \sim O(n^2) \text{ flops}$$
$$A^TA \sim O(n^3) \text{ flops}$$

Consequence for larger computers:

A computer with twice the speed/storage can only compute A^TA with $A \sim \sqrt[3]{2}n$ in the same time as original machine.

First problem to tackle: solve $A\vec{x} = \vec{b}$ using Gaussian elimination

Recall: Let $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$

$A\vec{x} = \vec{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$ can be solved using row reduction:

$$\left(\begin{array}{cc|c} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \end{array}\right) \sim \left(\begin{array}{cc|c} a_{11} & a_{12} & b_1 \\ 0 & a_{22} - \frac{a_{12}a_{21}}{a_{11}} & b_2 - \frac{a_{21}b_1}{a_{11}} \end{array}\right)$$

x_2 can be computed as $x_2 = \dfrac{b_2 - \frac{a_{21}b_1}{a_{11}}}{a_{22} - \frac{a_{12}a_{21}}{a_{11}}}$

Then $x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2)$ (this is called <u>back-substitution</u>)

This algorithm is <u>very</u> easy to break.

Ex: $a_{11} = 0$ or $a_{22} - \frac{a_{12}a_{21}}{a_{11}} = 0$.

In order to systematically solve $A\vec{x} = \vec{b}$ using Gaussian elimination, one must use <u>pivoting</u>.

<u>Ex:</u> $A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ must be (row) pivoted to $\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$.

On every step, it must be ensured that the pivot element is non-zero.

How expensive is Gaussian elimination?
Lets count operations: (treat $+, -, \div, \times$ as having the same cost)

To put A in echelon form: $\begin{pmatrix} a_{11} & \cdot & \cdots & \vdots \\ & a_{22} & & \vdots \\ 0 & & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix}$

Loop over columns $j = 1, .., n-1$

Loop over rows $i = j+1, \dots, n$

① compute $\frac{a_{ij}}{a_{jj}}$ (1 flop)

② compute row $i - \frac{a_{ij}}{a_{jj}}$ row j ($2(n-j)$ flops)

③ compute $b_i - \frac{a_{ij}}{a_{jj}} b_j$ (2 flop)

Total (for each j)
$2(n-j) + 3$ flops

Now compute the total:

$$\sum_{j=1}^{n-1} \sum_{i=j+1}^{n} \left(2(n-j)+3\right) = \sum_{j=1}^{n-1} (n-j)\left(2(n-j)+3\right)$$

$$\approx 2\sum_{j=1}^{n-1} (n-j)^2 \approx \mathcal{O}(n^3) \text{ flops.}$$

(of course there are more careful derivations)

Another Way to think of

Gaussian Elimination : LU factorization:

- Each row operation corresponds to multiplication by a lower triangular matrix.

<u>Ex:</u> $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -7 & 0 & 1 \end{pmatrix}}_{L_1} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -21 \end{pmatrix}$$

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}}_{L_2} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -21 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & -9 \end{pmatrix}}_{U.}$$

So $L_2 L_1 A = U$

$A = (L_2 L_1)^{-1} U$

$= \underbrace{L_1^{-1} L_2^{-1}}\, U$

These "undo" the row operation in L_1, L_2

$L_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 7 & 0 & 1 \end{pmatrix}$

$L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}$

$L_1^{-1} L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 7 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 7 & 2 & 1 \end{pmatrix}$

Gaussian Elimination & LU factorization are equivalent.

Easy to alter GE code to save LU (exercise):

Then to solve $A\vec{x}=\vec{b}$, $(LU\vec{x}=\vec{b})$

(1) solve $L\vec{y}=\vec{b}$ ⟵ Forward substitution

(2) solve $U\vec{x}=\vec{y}$ ⟵ Backward substitution

Operation Count? (For the L solve)

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & & \ddots & \\ & & & l_{nn} \end{pmatrix} \vec{y} = \vec{b}$$

$y_1 = b_1 / l_{11}$

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

are all 1's in our case

$\vdots$

```
do i = 1, n
   y_i = b_i          <- not included in op. count.
   do j = 1, i-1
      y_i = y_i - l_ij * y_j
   end do
end do
```

$$\text{Opcnt}: \sum_{i=1}^{n}\sum_{j=1}^{i-1} 2 = 2\sum_{i=1}^{n}(i-1)$$

$$= 2\sum_{1}^{n-1} k = 2\cdot\frac{n(n-1)}{2} = n^2 + O(n)$$

This is fast