Gaussian Elimination & LU ~~factorization~~.

Is how solve $Ax=b$ unless know something nice or horrible about A.

For today's lecture, A is an n-by-n invertible matrix.

The LU factorization of A produces
a lower triangular (henceforth Δar) matrix L, and
an upper Δar matrix U
such that $LU = A$.

1. Whence L?
2. Pivoting to avoid 0.
3. Why L?

It is easy to solve equations in Δ matrices, so once we compute L and U, it is easy to solve equations in A.

"Easy": count ops

where $f(n) \approx \Theta(g(n))$ if $\exists\, c_1, c_2, n_0$ s.t.
$$c_1 g(n) \le f(n) \le c_2 g(n) \quad \text{for all } n \ge n_0$$

To compute Ax requires $2n^2 - n$ flops - floating point ops
Think of $\Theta(n^2)$ ops to include memory refs, branches, etc

Forward Substitution: solving $Lx = b$

L looks like
$$\begin{pmatrix} L(1,1) & 0 & 0 \cdots 0 \\ L(2,1) & L(2,2) & 0 \cdots 0 \\ \vdots & & \vdots \\ L(n,1) & \cdots & L(n,n) \end{pmatrix} \begin{pmatrix} x(1) \\ \vdots \\ x(n) \end{pmatrix} = \begin{pmatrix} b(1) \\ \vdots \\ b(n) \end{pmatrix}$$

This says $L(1,1) \cdot x(1) = b(1)$, so set $x(1) = b(1)/L(1,1)$

what if $L(1,1)=0$? is singular

$L(2,1)\, x(1) + L(2,2)\, x(2) = b(2)$,

so, once we know $x(1)$, we can set

$$x(2) = \frac{1}{L(2,2)}\left(b(2) - L(2,1)x(1)\right)$$

$$x(3) = \frac{1}{L(3,3)}\left(b(3) - L(3,1)x(1) - L(3,2)x(2)\right)$$

flops to compute $x(1)$: 1 (\ )

$x(2)$: 3 (\ - *)

$x(3)$: 5 (\ - - * *)

$x(n)$: $2n-1$ one \ $(2n-1)$ − $(2n-1)$ *

So, # flops to compute $x = 1 + 3 + 5 + \cdots + 2n-1 = n^2$

(from 1st lecture) or $\Theta(n^2)$ ops

One way to write this:

```
for i in 1 to n
  | x(i) = b(i)
  | for j in 1 to (i-1)
  |   | x(i) = x(i) - L(i,j) x(j)
  | x(i) = x(i) / L(i,i)
```

Another way to write this:

```
x = b
for i in 1 to n
|  x(i) = x(i) / L(i,i)
|  for j in i+1 to n
|  |  x(j) = x(j) - L(j,i) x(i)
```

Keep on board

This code is an operator that multiplies by L^{-1}

Backwards solve: solve $Ux = b$

$$\begin{pmatrix} U(1,1) & U(1,2) & \cdots & U(1,n) \\ 0 & U(2,2) & \cdots & U(2,n) \\ \vdots & & & \vdots \\ 0 & \cdots & 0 & U(n,n) \end{pmatrix} x = b$$

Same thing, but start with $x(n)$, so also n^2 flops

So, to solve $Ax=b$, solve $LUx=b$ by

1. Find y s.t. $Ly=b$
2. Find x s.t. $Ux=y$

$\Rightarrow \quad LUx = Ly = b$

Total # flops $= 2n^2$

Why do I call this easy?

A has n^2 entries, so is time $\sim$ #entries in A.
the ideal.

Multiplying Ax uses $2n^2 - n$ flops, so is comparable to multiplication.

How do we get L and U?

And, why not compute A^{-1}, or L^{-1} or U^{-1}?

Answers are mostly numerics and run time (it takes longer)

To start, let's ask "why even compute L or U"

Main goal: produce a sequence of operators

$T_1, \ldots, T_k$ so that $x = T_1(T_2(\cdots(T_k(b)\cdots)$

which write as $x = T_1 \circ T_2 \circ \cdots \circ T_k(b)$

e.g. $T_1 = L^{-1}$ $T_2 = U^{-1}$

Computing L: apply operations to A until it is upper Δar.

Elimination: making entries of A zero.

Idea: gradually zero out entries of A until is upper Δar.

in order, zero out
$$\begin{array}{llll} A(2,1) & A(3,1), \dots, & A(n,1) \\ & A(3,2) & A(4,2) \cdots & A(n,2) \\ & & \ddots & \\ & & & A(n,n-1) \end{array}$$

Example: $A(2,:) = A(2,:) - A(1,:) \cdot \dfrac{A(2,1)}{A(1,1)}$

subtract $\dfrac{A(2,1)}{A(1,1)}$ times first row from second.

To avoid changing A, let $M = A$, and modify M

Let O_{ij}^{c} be operation on a vector x that subtracts $c \cdot x[i]$ from $x[j]$

$$O_{ij}^{c}(x)[k] = \begin{cases} x[k] & \text{if } k \neq j \\ x[j] - c\,x[i] & k = j \end{cases}$$

$$O_{ij}^{-c} \circ O_{ij}^{c} \circ x = x$$

Write O_{ij}^{c} as a matrix $= I - c\,E_{j,i}$,

where $E_{j,i}$ is zero except $E(j,i) = 1$

$$E(2,1) = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \qquad E_{j,i}\,x = x[i]\,e_j$$

Can apply to a matrix M by applying to each of its columns.

We begin with $M = O_{1,2}^{M(2,1)/M(1,1)} \circ M$

Eliminate all entries in first column, except $M(1,1)$, by

for i in 2 to n

$c = M(i,1)/M(1,1)$

$M = O_{1,i}^{c} \circ M$

In 2nd column, eliminate all entries in rows $3 \dots n$

for i in 3 to n

$c = M(i,2)/M(2,2)$

$M = O_{2,i}^{c} \circ M$

As $M(i,1) = 0$ for $i > 1$ at this point, none of those entries change.

Full algorithm:

$M = A$

For i in 1 to $n-1$

For j in $i+1$ to n (so $j > i$)

$c_{ij} = M(j,i)/M(i,i)$

$M = O_{ij}^{c_{ij}} \circ M$

Example:
$$\begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} \to \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} \to \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & 5 & 5 \\ 0 & 4 & 6 & 8 \end{bmatrix}$$

$$\to \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 4 & 6 & 8 \end{bmatrix} \to \cdots \to \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix} = U$$

If we store the $c_{i,j}$, then can use them to solve linear equations:

$$O_{n-1,n}^{c_{n-1,n}} \cdots\cdots O_{1,2}^{c_{1,2}} \circ A = U$$

So solve $Ax = b$ by applying these operators to b, giving $Ux = O_{n-1,n}^{c_{n-1,n}} \circ \cdots \circ O_{1,2}^{c_{1,2}} \circ b$

Then solve for x.

As an algorithm this is

for i in 1 to $n-1$

for j in $i+1$ to n

$b = O_{i,j}^{c_{i,j}} b \iff b[j] = b[j] - c_{i,j}\, b[i]$

Is same as forward solve in L with

$L(i,i) = 1$ and $L(j,i) = c_{i,j}$!

This is how we build L.

How many ops?

for i in 1 to $n-1$
  for j in $i+1$ to n
    apply O^c_{ij} to M - but only in cols $i+1$ to n

$\sim (n-i)^2$

$$\sum_{i=1}^{n-1} (n-i)^2 \approx \frac{1}{3} n^3, \text{ so } \Theta(n^3) \text{ ops}$$

Why not build L^{-1}? Seems to take $\Theta(n^3)$ ops.
Is wasteful and we do not need it.

What to do when $L(i,i) = 0$?

Pivot. Find k s.t. $L(k,i) \neq 0$, and
  eliminate entries in column i using k^{th} row.

Or, swap rows k and i (virtually)

Instead of L, construct PL for a permutation matrix L.

A permutation matrix is a matrix P
  that is all zeros except for one 1 in every
    row and column.

For each i, let $\pi(i)$ be s.t. $P[i, \pi(i)] = 1$

$Px = x(\pi(1)), x(\pi(2)), \ldots, x(\pi(n))$,

a permutation of x.

It is easy to mult a vector by P –
takes $\Theta(n)$ ops by using formula.

P is orthogonal, so $P^T P = I \Rightarrow P^{-1} = P^T$
$\Rightarrow$ is easy to apply P^{-1}!

We compute P during the algorithm,
and go back and adjust $c_{i,j}$ at the end –
is more work if do it along the way.

Example $A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ compute LU for $P^T A$

$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ $P^T A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$

But, this is not the only problem!

What if $A = \begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix}$ $L = \begin{bmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{bmatrix}$ $U = \begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - 1/\varepsilon \end{bmatrix}$

If $\varepsilon < 10^{-16}$, rounding error gives $\tilde{U} = \begin{bmatrix} \varepsilon & 1 \\ 0 & -1/\varepsilon \end{bmatrix}$

and $L\tilde{U} = \begin{bmatrix} \varepsilon & 1 \\ 1 & 0 \end{bmatrix}$ a very different matrix.

So, pivot not just when 0, but when small.

Try to keep L and U small.

A theorem of Wilkinson says if compute with precision u,

$$\|A - \tilde{L}\tilde{U}\| \leq u \cdot \theta(\|\tilde{L}\| \cdot \|\tilde{U}\|)$$

Is a good reason to think about L and U.