UCLA Math151A
Fall 2021
Lecture 22
2021/11/15

Misc remarks on Numerical Quadrature

Start on Numerical Linear Algebra

$$\sum_{i=1}^{n} w_i f(x_i)$$

T.R.: n=2, DOE=1

S.R.: n=3, (bad) DOE=2, (good) DOE=3

C.T.R.: O(h^2)

C.S.R.: O(h^4)

G.Q: n=arbitrary, DOE = 2n-1

Remark 22.1.

If you know f is smooth,
G.Q. with $n = 10, 20, 40, \ldots, O(100)$ often sufficient.

Remark 22.2.

In practice, best to use an existing implementation of G.Q. rather than writing one's own.

Remark 22.3.

Traperzoidal/Simpson's rule is still quite effective and easy to implement.

Remark 22.4.

Another simple/effective technique: interpolate f(x) with cubic spline s(x) and integrate.

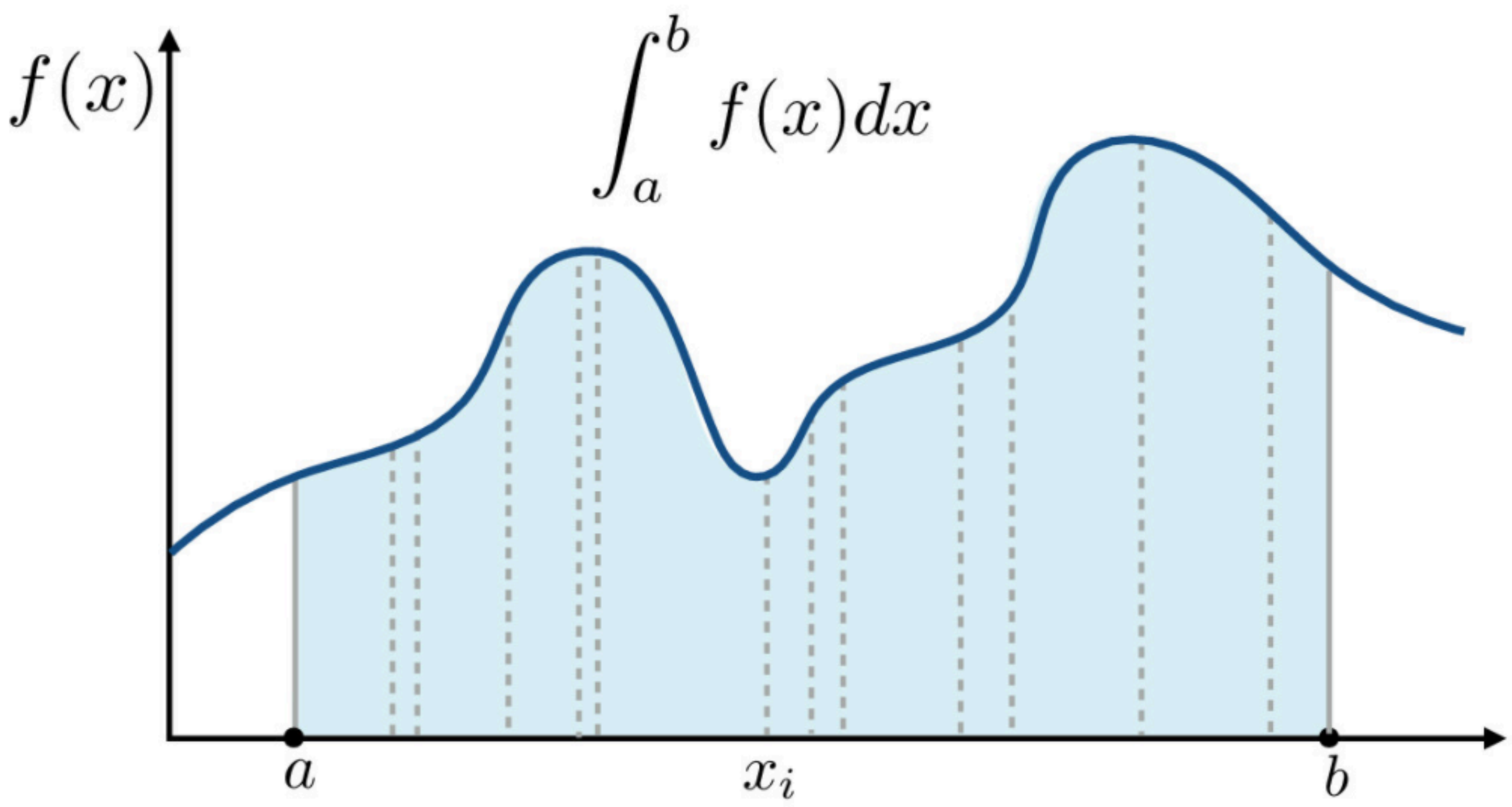
Remark 22.5.
If $f(x)$ is extremely noisy (e.g., stock price),
best to use Monte Carlo integration based on probability theory

Monte-Carlo

[not required by 151A]

expectation converges to the true integral

Simple idea: estimate the integral of a function by averaging random samples of the function's value.



- Downside: error decreases as $\sim \frac{1}{\sqrt{n}}$,

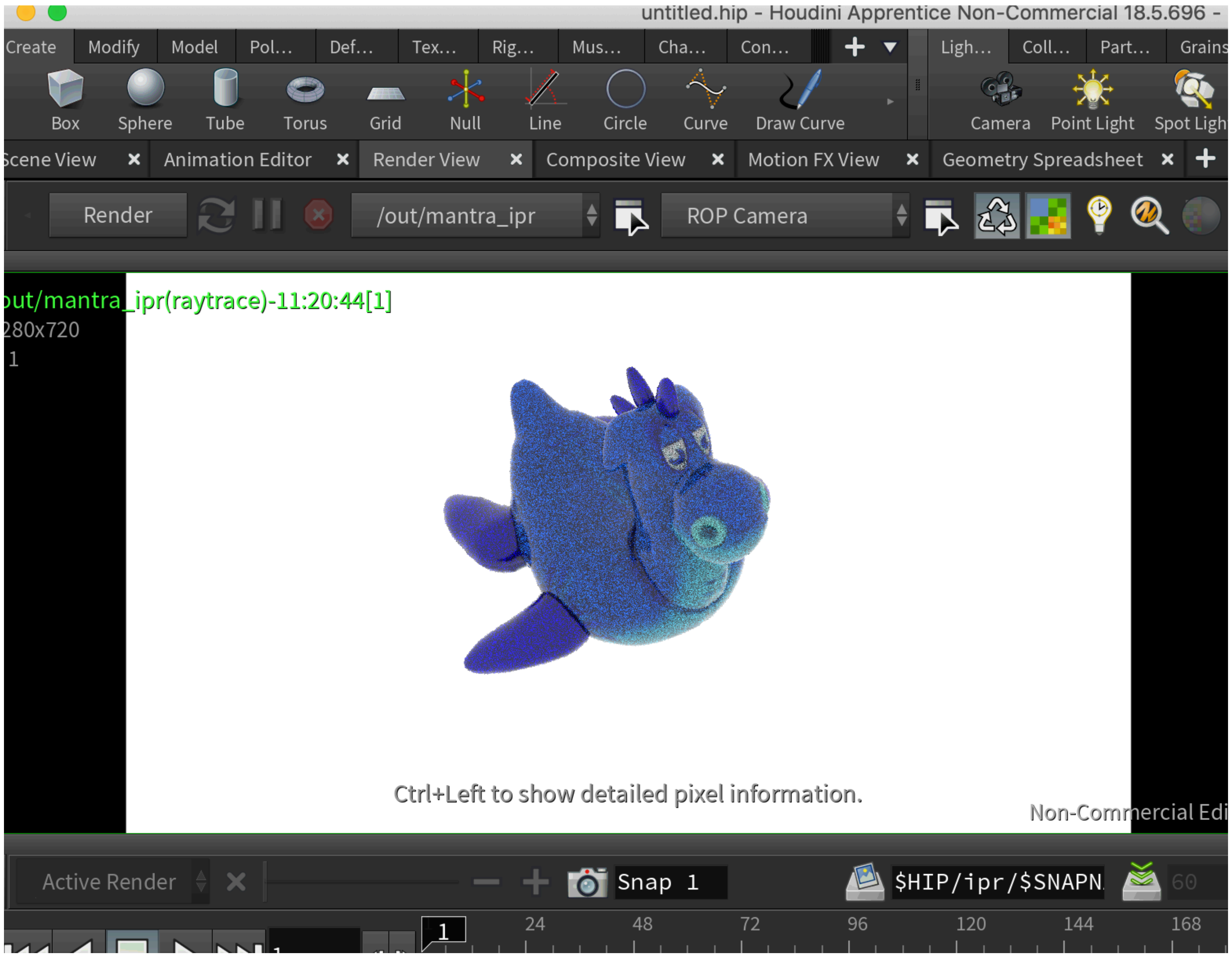
  compare to $\sim \frac{1}{n^2}$ for composite traperzoidal rule,

  $\sim \frac{1}{n^4}$ for composite simpson's rule.

- Positive: accuracy independent of $f(x)$

  (compare to traperzouidal rule needing $f \in C^2$, e.g.).

Live Demo of Using Monte-Carlo for rendering CGI



Remark 22.6.

Not covered: **Romberg integration.**

Basic idea is use Richardson extrapolation repeatedly.

Remark 22.7.

Not covered: adaptive quadrature.

Remark 22.8.

Misc topic: integratls over unbuonded domains.

$$\text{e.g., } \int_0^\infty e^{-x^2}dx.$$

We cannot approximate infinity in a computer.

The idea is to transform variables to make integral bounds finite,

$$\text{e.g., } z = \frac{x}{1+x},\ z(0) = 0, z(\infty) = 1.$$

$$\int_0^\infty e^{-x^2}dx = \int_0^1 \frac{1}{(1-z)^2}e^{-(\frac{z}{1-z})^2}dz.$$

NEW TOPIC of the COURSE:

Direct Methods for Solving Linear Systems of Equations

$\boxed{\text{Large linear systems:}}$

Matrix equation: $Ax = b, A \in \mathbb{R}^{n\times n}, x, b \in \mathbb{R}^n$.
Goal is to find x, the solution of the system.

the rest of the course we will assume that $\det(A) \neq 0$.

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$Ax = b$ is equivalent to a linear system:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ &\vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n &= b_n \, , \end{aligned}$$

n equations n unknowns

$$\sum_{j=1}^{n} a_{ij}x_j = b_j, \quad 1 \leq i \leq n$$

Why do we care about solving linear system?

Because they show up nearly everywhere in applied math.

For example, solve for coefficients of cubic spline interpolant (We've seen this before).

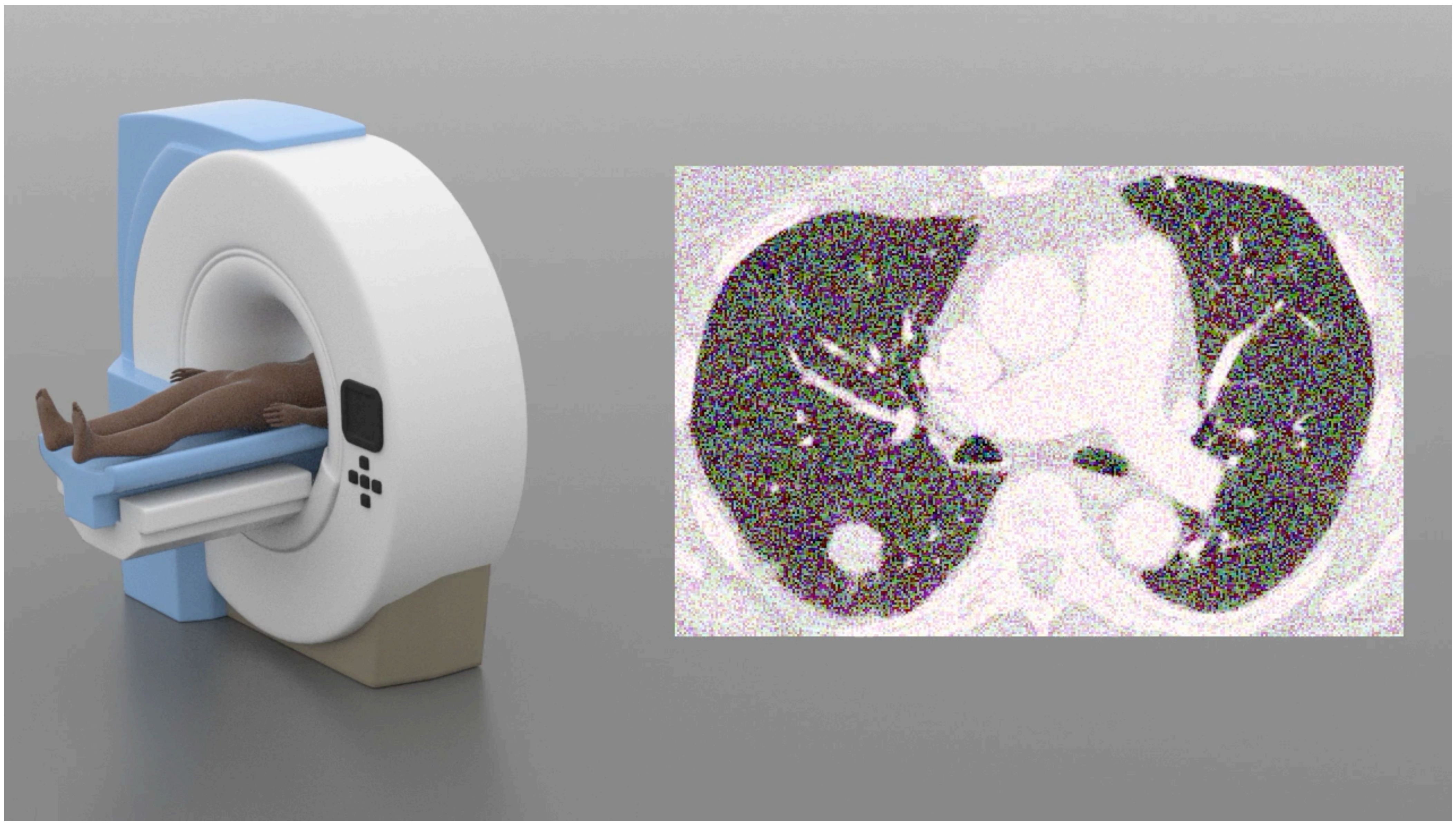
Another example, if you want to use Newton's method in higher dimensions than d = 1, recall

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (d = 1)$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}_F(\mathbf{x}_n)^{-1}\mathbf{f}(\mathbf{x}_n) \qquad (d > 1)$$

have to invert the Jacobian matrix!

Third example, to find a numerical solution to partial differential equations, e.g., heat equation, $$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$



can require solving $Au = b$ at each time step.

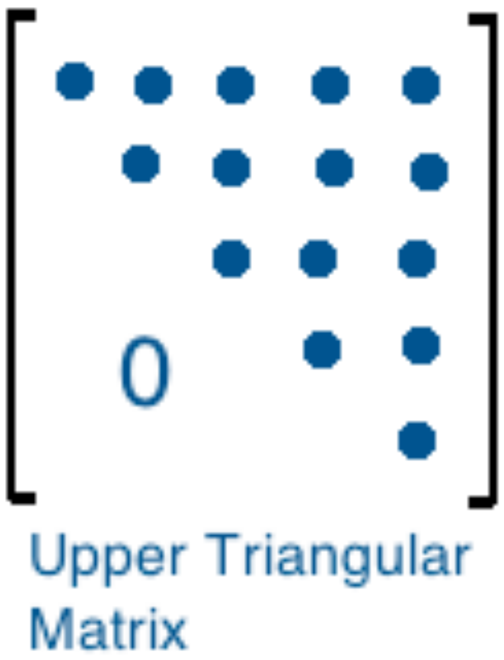
$\boxed{\text{Methods for solving } Ax = b\text{:}}$

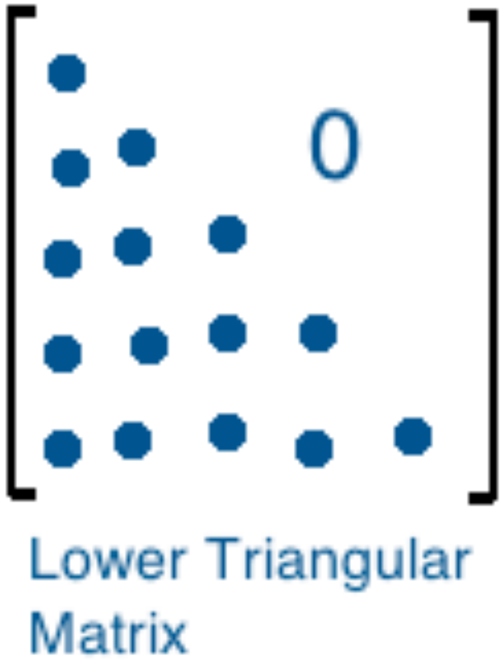
Direct Methods: find the exact solution (assuming exact arithmetic) (151A)

Iterative Methods: find approximate solutions (151B)

Gaussian Elimination

Goal is to transform $Ax = b$ into system $Ux = y$





with elementary row operations, where U is upper triangular.

Why? It makes the system simple to solve.

Elementary row operations:

- Replace a row E_i with a non-zero scalar's multiple of the row λE_i
- Replace E_i with $E_i + \lambda E_j$
- Swap E_i and E_j

e.g. $\begin{pmatrix} 1 & -3 \\ 2 & 1 \end{pmatrix}$ $\quad E_1 \leftarrow 3E_1 \quad$ $\begin{pmatrix} 3 & -9 \\ 2 & 1 \end{pmatrix}$

Fact:

every elementary row operation can be represented by applying an invertible matrix P.

e.g. $$\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -3 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 3 & -9 \\ 2 & 1 \end{pmatrix}$$

A sequence of elementary row operations can be used to transform $Ax = b$ to $Ux = y$.

Back Substitution

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Upper triangular matrices are easy to invert.

1. Starts with the last equation because it has only one unknown.

2. Solve the second from last equation (n-1)th using xn solved for previously. This solves for xn-1.

3. Keep going up.