Elimination and $PA = LU$

Goal: solve $Ax = b$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$A \qquad x = b$$

In 18.06, we learn Gaussian elimination:

"Convert A to upper triangular matrix using elementary row operations"

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Triangular system is easy to solve with backward substitution.

LU factorization

Gaussian elimination factors A into

$$A = L\,U$$

(L: lower triangular; U: upper triangular)

Computationally, useful for $Ax_j = b_j \quad j = 1, \ldots, m.$

solve $L\tilde{b}_j = b_j$ (forward sub) and $Ux_j = \tilde{b}_j$ (backward sub)

$$\Rightarrow \quad Ax_j = LUx_j = L\tilde{b}_j = b_j \quad \checkmark$$

Theoretically, useful to analyze G.E.

Example

$$\underbrace{\begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ -4 & & 1 & \\ -3 & & & 1 \end{bmatrix}}_{E_1} \underbrace{\begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & 3 & 5 & 5 \\ & 4 & 6 & 8 \end{bmatrix}}_{E_1 A}$$

E_2 $\qquad\qquad$ E_1A $\qquad\qquad$ E_2E_1A

$$\begin{bmatrix} 1 & & \\ & 1 & \\ -3 & & 1 \\ -4 & & & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & 3 & 5 & 5 \\ & 4 & 6 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & 2 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & & 2 \end{bmatrix}$$

E_3 $\qquad\qquad$ E_2E_1A $\qquad\qquad$ $U = E_3E_2E_1A$

$$E_3E_2E_1A = U$$

$$A = \underbrace{E_1^{-1}E_2^{-1}E_3^{-1}}U$$

L = lower triangular

$$E_1^{-1} = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & & 1 & \\ 3 & & & 1 \end{bmatrix} = \text{"undo row exchanges by adding rows back!"}$$

$$E_1^{-1}E_2^{-1}E_3^{-1} = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & 3 & 1 & \\ 3 & 4 & 1 & 1 \end{bmatrix}$$

"multipliers" from elimination go in place in L w/sign flip.

Note: $A = LU$ can be computed "in-place,"
L and U overwrite lower/upper triangles of A.

Algorithm ($A = LU$)

$U = A$, $L = I$

for $k = 1$ to $n-1$ (each column)

  for $j = k+1$ to n (each row below pivot u_{kk})

    $l_{jk} = u_{jk} / u_{kk}$

    $u_{j,k:n} = u_{j,k:n} - l_{jk} u_{k,k:n}$

  end

end

Cost of $A = LU$

traditional measure: count arithmetic

Floating Point OPerations = FLOPS

zero in col. k costs $\sim 2(n-k)$ flops

$n-k-1$ zeros $\sim 2(n-k)^2$ flops

contributions from columns 1 through $n-1$

$$\#\text{flops} = \sum_{k=1}^{n-1} 2(n-k)^2 \sim \frac{2}{3}n^3 \quad (n \to \infty)$$

Solving $L\tilde{b} = b$ and $Ux = \tilde{b}$, in contrast,

$$\#\text{flops} \sim n^2$$

In practice, there is more to an algorithms performance than just flop count!
We will revisit other factors later in the course.

When to store L in practice?

We can solve $Ax = b$ and even $AX = B$ (many RHS) w/out explicitly storing L. Do Gaussian elimination on

$$[A \mid B]$$ augmented matrix.

Implicitly apply L^{-1} through row ops:

Elimination $AX = B \Rightarrow UX = L^{-1}B$

Back Substitution $X = U^{-1}(L^{-1}B)$

In many situations, we don't know $b_1, b_2, \ldots, b_m$ at same time. E.g., implicit time-stepping for ODEs or iterative solvers for nonlinear eq.'s. Store L and save!

1) $A = LU$ once

2) $L\tilde{b}_j = b_j$ and $Ux_j = \tilde{b}_j$ $j = 1, \ldots, m$
$\Rightarrow$ $2m$ triangular solves

Stability of GE

W/out row interchanges, small pivots make GE hopelessly unstable.

Here is a classic 2×2 example:

$$\hat{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \xrightarrow{\text{Perturb}} A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}$$

GE fails even though A is invertible (divide by zero)

$$L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1-10^{20} \end{bmatrix}$$

rounded L and U

$$\tilde{L} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \qquad \tilde{U} = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix}$$

← rounded

$$\tilde{L}\tilde{U} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix}$$

$$A - \tilde{L}\tilde{U} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

not even close!

$Ax = b$ $\qquad$ $\tilde{L}\tilde{U} = b$

$b = (1,0)^T$ $\qquad$ $\tilde{x} = (0,1)^T$

$x \approx (-1,1)^T$

Solution of $Ax = b$ is not close to solution of $\tilde{L}\tilde{U}x = b$, even though $Ax = b$ is well-conditioned in this instance.

In general, GE behaves like a back. stable algorithm unless $\|L\|$ and $\|U\|$ are much bigger than $\|A\|$.

Thm If LU factorization of A exists & is computed by Gaussian elimination (without pivoting) in floating point arithmetic, then computed factors $\tilde{L}$ and $\tilde{U}$ satisfy

$$\tilde{L}\tilde{U} = A + \delta A \quad \text{where} \quad \frac{\|\delta A\|}{\|L\|\|U\|} = O(\varepsilon_{\text{mach}}).$$

Partial Pivoting (a partial solution)

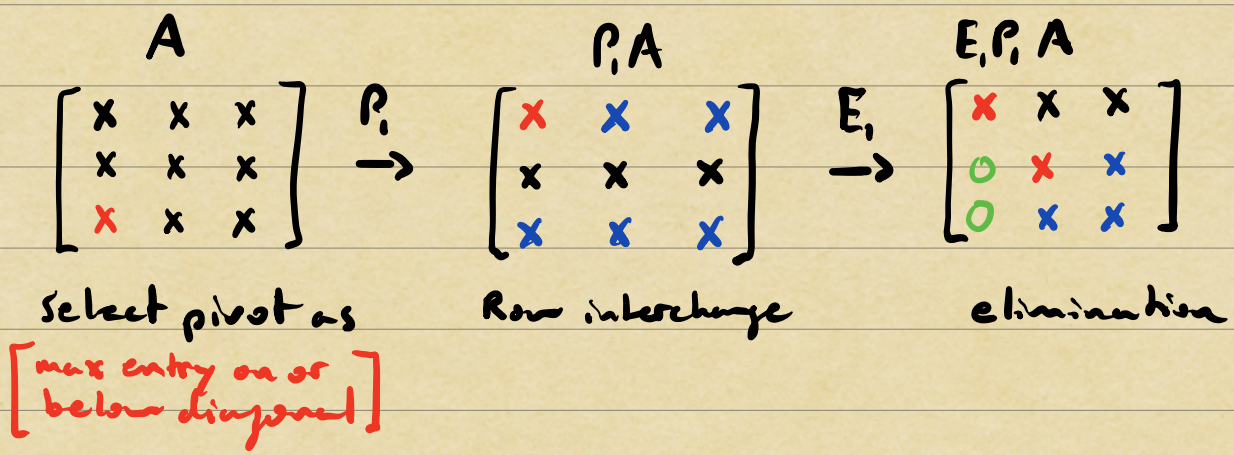
Can we control $\|L\|$ and $\|U\|$?

Recall $L = \begin{bmatrix} 1 & & & \\ \ell_{21} & 1 & & \\ \ell_{31} & \ell_{32} & \ddots & \\ \ell_{m1} & \ell_{m2} & \cdots & 1 \end{bmatrix}$ where $\ell_{jk} = \dfrac{x_{jk}}{x_{kk}}$ (multipliers)

$(k \le j \le m)$

$(x_{ij} = \overset{(i,j)^{\text{th}}}{\text{entry}}$ of j^{th} column of $E_k \cdots E_1 A)$

Idea: use row permutations to avoid small pivots and control growth in $\|L\|$.



With partial pivoting, $x_{kk} \geq x_{jk}$ for $k \leq j \leq m$

so $\ell_{jk} \leq 1$ - no growth in L!

Note: can also incorporate column interchanges

⇒ Column pivoting, complete pivoting, etc. (See LNT Lec. 21)

Unfortunately, entries of U can still be exponentially large relative to A. See the counterexample in Julia notebook

is-gaussian-elimination-unstable

Fortunately, this almost never happens in practice. In daily use, GE and LU behave like back. stable algorithms! See the experiments w/random A in the Julia notebook.

This is the mystery of Gaussian Elimination

Theoretically unstable - practically stable