And Now You Will Too!

- Programming Quiz: Monday 1PM
- A-Rod ruined my day yesterday
- Therefore, I am going to crush you, just like A-Rod crushes a Joe Borowski hanging slider.

Just Kidding

LU-Decomposition

$$\text{LU-DECOMPOSITION}(A)$$

1. $$n \leftarrow \text{rows}[L]$$
2. for $$k \leftarrow 1$$ to $$n$$
3. do
4. $$u_{kk} \leftarrow a_{kk}$$
5. for $$i \leftarrow 1$$ to $$n$$
6. do
7. $$\ell_{ik} \leftarrow a_{ik}/u_{kk}$$
8. $$u_{ki} \leftarrow a_{ki}$$
9. for $$i \leftarrow k + 1$$ to $$n$$
10. do
11. for $$j \leftarrow k + 1$$ to $$n$$
12. do
13. $$a_{ij} \leftarrow a_{ij} - \ell_{ik} u_{kj}$$
LU ≈ Gaussian Elimination

- We either have $A = LU$ or we have $MA = U$, and $L = M^{-1}$
- Because of the special structure of $M$, we have a (fairly) remarkable relationship
  $$M^{-1} = (M_{n-1} \cdots M_2 M_1)^{-1} = M_1^{-1} M_2^{-1} \cdots = L$$

\[ L = \begin{pmatrix}
1 & 1 \\
m_{21} & m_{31} & m_{32} & 1 \\
& \ddots & \ddots & \ddots \\
m_{n1} & m_{n2} & m_{n3} & \cdots & 1
\end{pmatrix} \]

where the $m_{ik}$ are the multipliers from Gaussian elimination!

- So $L$ and $U$ can be derived directly from the elimination process:
  $$\ell_{ik} = m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \quad u_{kj} = a_{kj}^{(k)}$$

Recall!

- A square matrix $P$ is a permutation matrix if there is a single 1 in each row and column.
- A square matrix whose columns all have length (norm) 1, and that are (pairwise) orthogonal is called orthogonal.
- If $Q \in \mathbb{R}^{n \times n}$ is orthogonal then (by definition) $Q^T Q = I$, so then $Q^T = Q^{-1}$.
- What effect does (right)-multiplying by a permutation matrix have? (Shuffles Columns)
- What effect does (left)-multiplying by a permutation matrix have? (Shuffles Rows)
- To make Gaussian Elimination work, we sometimes need to swap two rows.
- The resulting “transformation” matrix is a symmetric permutation matrix $P$

Zero Pivots in $MA = U$

- We may need to swap rows before every iteration of the elimination ($MA = U$)
- In fact, we may want to perform row exchanges
- In Gaussian Elimination, (with row swaps), really what we end up with is
  $$M_{n-1} P_{n-1} \cdots M_2 P_2 M_1 P_1 A = U$$
- Let’s show how we can get all of the permutations “pushed” to the outside, and thus show that we can think of it as just reordering the rows of $A$ one time. Leaving us with our desired factorization: $PA = LU$

Does $P$ Mess up our Triangular Solves?

- Note that the system $PAx = Pb$ is equivalent to the original system, which is then equivalent to $LUx = Pb$.
- We can solve the system in two steps:
  - First solve the system $Ly = Pb$ (forward substitution).
  - Then solve the system $Ux = y$ (backward substitution).
- $Pb$ is really nothing more than a “permuted” version of $b$.
- Typically permutation matrices $P$ are (compactly) represented by an array $\pi[1, \ldots, n]$.
  $$\pi[i] = 1 \Rightarrow P_i \pi[i] = 1, P_{ij} = 0 \forall j \neq \pi[i]$$
- Recall: left multiply just takes linear combinations of the rows.
- $PA$ has $(i, j)$ entry of $a_{\pi[i], j}$ and $Pb$ has $b_{\pi[i]}$ in the $i^{th}$ position.
Simple Case

- Suppose for simplicity that \( A \in \mathbb{R}^{3 \times 3} \), so Gaussian Elimination produces:
  \[
  M_2 P_2 M_1 P_1 A = U
  \]
- \( P_2 \) is orthogonal, so \( P_2^T P_2 = I \), thus
  \[
  M_2 P_2 M_1 P_1 A = M_2 P_2 M_1 P_2^T P_2 P_1 A = M_2 \hat{M}_1 P_2 P_1 A = U.
  \]
where \( \hat{M}_1 = P_2 M_1 P_2^T \)
- That is \( \hat{M}_1 \) has the rows and columns of \( M_1 \) permuted by \( P_2 \).

Specifically

- For example,
  \[
  P_2 = \begin{pmatrix}
  1 & 0 & 0 \\
  0 & 0 & 1 \\
  0 & 1 & 0 
  \end{pmatrix}
  \quad
  M_1 = \begin{pmatrix}
  1 & 0 & 0 \\
  -m_{21} & 1 & 0 \\
  -m_{31} & 0 & 1 
  \end{pmatrix}
  \quad
  \hat{M}_1 = P_2 M_1 P_2^T = \begin{pmatrix}
  1 & 0 & 0 \\
  -m_{31} & 1 & 0 \\
  -m_{21} & 0 & 1 
  \end{pmatrix}
  \]

In The End

- In General (by inserting enough reordering permutation matrices), we get
  \[
  \hat{M} P A = U
  \]
  or
  \[
  P A = L U
  \]
with \( L = \hat{M}^{-1} \)
- We’ll do an example here...

Why Exchange Rows?

- We may want to exchange rows, even if the pivot element is just very small in magnitude.
- Pivoting on small numbers can lead to a significant loss of accuracy.
  Suppose we are computing rounding to four significant digits
- Consider the simple system:
  \[
  Ax = \begin{pmatrix}
  .0001 & .5 \\
  .4 & .3 
  \end{pmatrix}
  \begin{pmatrix}
  x_1 \\
  x_2 
  \end{pmatrix} = \begin{pmatrix}
  .5 \\
  .1 
  \end{pmatrix}
  \]
whose exact solution is \( x = (.9999, .9998)^T \)
Partial Pivoting

- This horrible loss of accuracy in the solution can come from having a small pivot. So let's be smarter:
- Pivot on element with the largest (magnitude) element remaining in the column:
  \[ \ell^* = \arg \max_{i \geq k} |a_{ik}|. \]
- Swap rows \( k \) and \( \ell^* \) at step \( k \) of the algorithm
- While this doesn’t always eliminate numerical instability, it usually works well.
- We could (though using both row and column permutations), implement a complete pivoting strategy in which the largest remaining matrix element is used as the pivot element.

The LUP Decomposition – The “Book Way”

- The element \( a_{11} \) is called the pivot element.
- Note that the above decomposition method fails whenever the pivot element is zero.
- In this case, we can permute the rows of \( A \) to obtain a new pivot element.
- In fact, for numerical stability, it is desirable to have the pivot element be as large as possible in absolute value.
- If no nonzero pivot is available, \( A \) is singular.
- This leads to the following modified factorization.

  \[
  QA = \begin{bmatrix} a_{k1} & w^T \\ v/a_{k1} & A' \end{bmatrix}
  \begin{bmatrix} a_{k1} & w^T \\ 0 & A' - vw^T/a_{k1} \end{bmatrix}
  \]

  \( (1) \)

Finding the LUP Decomposition (cont.)

- As before, we obtain \( L', U' \), and \( P' \) and we get

  \[
  PA = \begin{bmatrix} 1 & 0 \\ 0 & P' \end{bmatrix} QA
  \]

  \( (3) \)

  \[
  = \begin{bmatrix} 1 & 0 \\ 0 & P' \end{bmatrix} \begin{bmatrix} 1 & 0 \\ v/a_{k1} & I \end{bmatrix} \begin{bmatrix} a_{k1} & w^T \\ 0 & A' - vw^T/a_{k1} \end{bmatrix}
  \]

  \( (4) \)

  \[
  = \begin{bmatrix} 1 & 0 \\ P'v/a_{k1} & I \end{bmatrix} \begin{bmatrix} a_{k1} & w^T \\ 0 & P'(A' - vw^T/a_{k1}) \end{bmatrix}
  \]

  \( (5) \)

  \[
  = \begin{bmatrix} 1 & 0 \\ P'v/a_{k1} & I \end{bmatrix} \begin{bmatrix} a_{k1} & w^T \\ 0 & L'U' \end{bmatrix}
  \]

  \( (6) \)

  \[
  = \begin{bmatrix} 1 & 0 \\ P'v/a_{k1} & L' \end{bmatrix} \begin{bmatrix} a_{k1} & w^T \\ 0 & U' \end{bmatrix}
  \]

  \( (7) \)
Next Time!

- **Quiz:** in lab April 23!
- Two or Three *simple* programming questions.
- You *will* be able to use any of your *own* code from the previous labs.
- You *will not* be allowed to access the Internet, not even to check if the Red Sox are beating the Yankees.