Applications of LU

(1) Solve linear equations. How?

\[ PA = LU \quad \sim \quad PAx = Pb \quad \Rightarrow \quad UX = Pb \]

(i) Solve \[ Ly = Pb \]

(ii) Solve \[ Ux = y \]

Isn't this complicated or expensive?
(No: The factorization itself is cheap--and reusable.)

(2) Solve a matrix equation. How?

Given: \[ A\mathbf{x} = \mathbf{b} \]

Simplifying assumption:

\[ A, X, B \text{ are square and have same size.} \]

We can solve this column-by-column:

\[
\begin{pmatrix}
A
\end{pmatrix}
\begin{pmatrix}
\mathbf{x}
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{b}
\end{pmatrix}
\]

No different than solving lots of linear systems with the same \( A \) and lots of different right-hand side vectors \( \mathbf{b} \). Can reuse \( L \) and \( U \).

Example: \[ A\mathbf{x} = \mathbf{I} \quad (\mathbf{I} \text{ is the identity matrix.}) \]

\[ \sim \text{ Solved by } \mathbf{x} = A^{-1} \]

\[ \sim \text{ Can find inverse } A^{-1} \text{ using LU} \]

LU \quad BW+FW subst

Computational Cost: \[ O(n^3) + n \cdot O(n^3) = O(n^3) \]
The factor $U$ in pivoted LU looks like it is in upper echelon form, and most of the time it is... but this is not guaranteed.

For example, $U$ can contain linearly dependent rows.

Demo: LU and upper echelon form, Part I

If you hit a column of all zeros, then to achieve echelon form, you would need to "move right" (and just keep eliminating in the same row).

Then our pivot/elimination split trick no longer works, and $L$ is no longer lower triangular!

(Pivoting is the problem here!)

But: We can still use the same process as pivoted LU to compute an invertible matrix $M$ so that

\[ MA = U \]

so that $U$ is in upper echelon form. But $M$ cannot easily be factored into elimination and permutation matrices--and thus not easily inverted!

Nonetheless, we can obtain the "echelon factorization":

\[ A = M^{-1}U \]

Demo: LU and upper echelon form, Part II
(4) Find the basis of a span. How?

Given: \( \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_k \) \quad \text{linearly dependent}

Want: \( \mathbf{y}_1, \ldots, \mathbf{y}_k \in \text{span} \{ \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_k \} \quad \text{span} \mathbf{A} = \mathbf{V} \) \quad \text{linearly independent}

Define \( \mathbf{A} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} \)

Obtain \( \mathbf{MA} = \mathbf{U} \) \quad \text{in echelon form}

Non-zero rows of \( \mathbf{U} \) form a basis of \( \mathbf{V} \).

(5) Find the determinant of a matrix. How?

\[ \det \mathbf{A} = \mathbf{LU} \]

\[ \Rightarrow \det (\mathbf{P}) \det (\mathbf{A}) = \det (\mathbf{L}) \det (\mathbf{U}) \]

\[ \pm 1 \quad 1 \quad \text{product of diagonal entries} \]
We'd like to find the rank* of a matrix. Is that possible using a computer?

Two randomly vectors almost surely do not point in the same direction.

Two random vectors are almost surely not linearly dependent.

Computers do not represent numbers exactly. (in floating point)

Every floating point number: 

$$3.14159277777777$$

Model that as:  

True value + (small?) "random" error

In a vector:

Inexact vector

Suppose we would like to test two inexact vectors for linear dependence.

True:  \( \mathbf{u} = \alpha \mathbf{v} \) (linearly dependent)

Computed:  \( \hat{\mathbf{u}} \neq \alpha \hat{\mathbf{v}} \) (not linearly dependent)

Lesson:  We cannot hope for exact equality on a computer. Instead, we must define some sort of tolerance.

*rank: Number of linearly independent rows/columns
Suppose we take that into account. How would we compute the rank?

Just compute echelon form. In exact arithmetic, "missing" row rank would appear as rows of zeros.

On a computer, we cannot hope for exact zeros.

Demo: Computing the Rank

Lesson: To find the rank computationally, we must specify a threshold on (for example) the minimum norm of an echelon form row.
Echelon factorization of $A$ is not much help:

$$A = M^r U = N^t$$

nullspace not obvious

Idea: Start with echelon factorization of $A^r$:

$$A^r = M^r U \Rightarrow A = U^t M^r$$

Echelon form

$$N(U^t) = \{ \begin{bmatrix} 0, 0, 0, \ldots, 0, 1, 0, \ldots, 0 \end{bmatrix}, \ldots \}$$

because these vectors "hit" the zero columns in $U^t$.

So we know a few vectors $\tilde{x}$ so that $U^t \tilde{x} = \mathbf{0}$.

But $A = U^t M^r$

We're looking for $\tilde{y}$ so that $M^t \tilde{y} = \tilde{x}$ (for each of our $\tilde{x}$).

Easy: $\tilde{y} = M^t \tilde{x}$.

$\Rightarrow N(A) = M^t N(U^t)$