Can the SVD $A = U \Sigma V^T$ be used to solve square linear systems? At what cost (once the SVD is known)?

$A \mathbf{x} = \mathbf{b}$

$U \Sigma V^T \mathbf{x} = \mathbf{b}$

$\mathbf{x} = V \Sigma^{-1} U^T \mathbf{b}$
\[
\begin{pmatrix}
\sigma_1 \\
\vdots \\
\sigma_n
\end{pmatrix}
\begin{pmatrix}
x_1 \\
\vdots \\
x_n
\end{pmatrix}
= \begin{pmatrix}
\sigma_1 x_1 \\
\vdots \\
\sigma_n x_n
\end{pmatrix}
\]
Consider a ‘tall and skinny’ linear system, i.e. one that has more equations than unknowns:

\[ A_{\text{top}} x = b_{\text{top}} \]

In the figure: \( m > n \). How could we solve that?
Instead of \( A x = b \), ask for
"Which \( x \) makes \( \| A x - b \|_2^2 \) as small as possible?"

\[
\left( \begin{array}{c}
\tilde{x} \\
-\tilde{b}
\end{array} \right) \quad \text{"residual"}
\]

\[
\| A \tilde{x} - b \|_2^2 = \| r \|_1^2 = r_1^2 + \ldots + r_m^2
\]

\( \Rightarrow \) "Least squares"
Solving Least-Squares

How can I actually solve a least-squares problem $Ax \approx b$?

Suppose $Q$ is orthogonal: $\|Qy\|_2 = \|y\|_2$

$$\min_x \|Ax - b\|_2^2 = \|y^T(EV^T)^{-1} - b\|_2^2$$

$$= \min_y \|Ey - \bar{u}b\|_2^2$$
\[ = \sqrt{\left( \begin{pmatrix} \sigma_1 \\
0 \\
\vdots \\
0_n \end{pmatrix} y_1 - z_1 \right)^2 + \ldots + \left( \sigma_n y_n - z_n \right)^2} \]

\[ |\sigma_1| \geq |\sigma_2| \ldots \geq |\sigma_n| \geq 0 \]
\[ \sigma_1 \cdots \sigma_k > 0 \quad \delta_{k+1} \cdots \delta_l = 0 \]

\[ \sum = (\sigma_1 y_1 - z_1)^2 + \cdots + (\sigma_k y_k - z_k)^2 \]

\[ + \left( \sum_{k+1}^{\ell} (\sigma_{k+1} y_{k+1} - z_{k+1}) \right)^2 \]

\[ + \left( \sigma_{\ell+1} y_{\ell+1} - z_{\ell+1} \right)^2 \]

\[ + \left( \sigma_{\ell+1} y_{\ell+1} - z_{\ell+1} \right)^2 \]

\[ + 2^\ell \]

\[ y_i = \frac{z_i}{\sigma_i} \Rightarrow (\sigma_i y_i - z_i)^2 \]

\[ = \left( \sigma_i \frac{z_i}{\sigma_i} - z_i \right)^2 = 0 \]
\[
\min \| y - U^\dagger b \|_1
\]

\[
y_i = \begin{cases} 
U^\dagger y_i & \sigma_i = 0 \\
0 & \sigma_i = 0
\end{cases}
\]

doesn't matter pick
\[ \Sigma^{-1} = \begin{pmatrix} 1/\sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/\sigma_n \end{pmatrix} \]

\[ \Sigma^+ = \begin{pmatrix} 1/\sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/\sigma_n \end{pmatrix}^{-1} \]

"Pseudo inverse of a diagonal"
with that:

\[ \tilde{y} = \Sigma^+ U^T g \]

Still wanted: \( \| Ax - b \|^2 \)

\[ \tilde{y}^2 = V^T x \quad (=) \quad V\tilde{y} = x \]

fill & skinny \[ \bar{x} = V \Sigma^+ U^T g \]

square & inv. \[ \bar{x} = V \Sigma^{-1} U^T g \]
\[ \nu \Sigma^+ \nu^T = A^+ \]

\[ \nu \Sigma^+ \nu^T \downarrow \]

\[ \text{pseudoinv. of } A \]

\[ x = A^+ \bar{b} \]

\[ (x = A^{-1} \bar{b}) \]

\[ \text{solves } \min ||Ax - \bar{b}||_2^2 \]
\[ A^T A x = A^T b \]

(normal equations)

\[ \text{cond} (A^T A) \approx \text{cond} (A)^2 \text{cond}(A) \]

\[ \approx \text{cond} (A)^2 \left(10^9\right)^2 \]
In-class activity: SVD and Least Squares
How could the solution process for $Ax \cong b$ be with an SVD $A = U\Sigma V^T$ be ‘packaged up’?
The Normal Equations

You may have learned the ‘normal equations’ $A^T A x = A^T b$ to solve $A x \approx b$. Why not use those?
\[ p(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_n x^n \]

\[
\begin{bmatrix}
\alpha_0 \\
\vdots \\
\alpha_n
\end{bmatrix}
\begin{bmatrix}
y_0 \\
y_1 \\
\vdots \\
y_n
\end{bmatrix}
\]

\[ p(x_0) = y_0 \]
\[ p(x_n) = y_n \]
Outline

Python, Numpy, and Matplotlib
Making Models with Polynomials
Making Models with Monte Carlo
Error, Accuracy and Convergence
Floating Point
Modeling the World with Arrays
  The World in a Vector
  What can Matrices Do?
  Graphs
  Sparsity
Norms and Errors
The ‘Undo’ Button for Linear Operations: LU
Repeating Linear Operations:
Eigenvalues and Steady States
Eigenvalues: Applications
Approximate Undo: SVD and Least Squares
SVD: Applications
  Solving Funny-Shaped Linear Systems
  Data Fitting
  Norms and Condition Numbers
  Low-Rank Approximation
Interpolation
Iteration and Convergence
Solving One Equation
Solving Many Equations
Finding the Best: Optimization in 1D
Optimization in $n$ Dimensions
Fitting a Model to Data

How can I fit a model to measurements? E.g.:

- Number of days in polar vortex
- Mood
- Time $t$

**Diagram:**

- Vertical axis: Mood
- Horizontal axis: Number of days in polar vortex
- Points representing data points
- Trend line suggesting a decrease in mood as the number of days in the polar vortex increases.
Demo: Data Fitting using Least Squares
Outline

- Python, Numpy, and Matplotlib
- Making Models with Polynomials
- Making Models with Monte Carlo
- Error, Accuracy and Convergence
- Floating Point
- Modeling the World with Arrays
  - The World in a Vector
  - What can Matrices Do?
  - Graphs
  - Sparsity
- Norms and Errors
- The ‘Undo’ Button for Linear Operations: LU
- Repeating Linear Operations: Eigenvalues and Steady States
- Eigenvalues: Applications

Approximate Undo: SVD and Least Squares

**SVD: Applications**
- Solving Funny-Shaped Linear Systems
- Data Fitting
- Norms and Condition Numbers
- Low-Rank Approximation

- Interpolation
- Iteration and Convergence
- Solving One Equation
- Solving Many Equations
- Finding the Best: Optimization in 1D
- Optimization in \( n \) Dimensions