Pr. 1.
For $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$:

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{A} \\ \boldsymbol{A}^T & \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \\ \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T & \boldsymbol{0} \end{bmatrix} \quad (\boldsymbol{\Sigma}^T = \boldsymbol{\Sigma} \text{ since } \boldsymbol{A} \text{ is square}).$$

Hence

$$\boldsymbol{B} \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \\ \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T v_i \\ \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T u_i \end{bmatrix} = \begin{bmatrix} \sigma_i u_i \\ \sigma_i v_i \end{bmatrix} = \sigma_i \begin{bmatrix} u_i \\ v_i \end{bmatrix}.$$

Thus $\boldsymbol{x} \triangleq \begin{bmatrix} u_i \\ v_i \end{bmatrix}$ is an eigenvector of $\boldsymbol{B}$ corresponding to the eigenvalue σ_i. Similarly:

$$\boldsymbol{B} \begin{bmatrix} u_i \\ -v_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \\ \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} u_i \\ -v_i \end{bmatrix} = \begin{bmatrix} -\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T v_i \\ \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T u_i \end{bmatrix} = \begin{bmatrix} -\sigma_i u_i \\ \sigma_i v_i \end{bmatrix} = -\sigma_i \begin{bmatrix} u_i \\ -v_i \end{bmatrix},$$

so that $\boldsymbol{z} \triangleq \begin{bmatrix} u_i \\ -v_i \end{bmatrix}$ is an eigenvector of $\boldsymbol{B}$ corresponding to the eigenvalue $-\sigma_i$. Note that $\boldsymbol{z}'\boldsymbol{x} = \begin{bmatrix} u_i^T & -v_i^T \end{bmatrix} \begin{bmatrix} u_i \\ v_i \end{bmatrix} = 0$ so $\boldsymbol{x}$ and $\boldsymbol{z}$ are orthogonal. Thus the eigenvalues of $\boldsymbol{B}$ are exactly $\{\pm\sigma_i\}$.
We must normalize the eigenvectors to have unit norm. The unit-norm eigenvector of $\boldsymbol{B}$ corresponding to the eigenvalue σ_i is $\begin{bmatrix} u_i \\ v_i \end{bmatrix} / \sqrt{2}$ (check that this is unit norm!) and the unit-norm eigenvector corresponding to the eigenvalues $-\sigma_i$ is $\begin{bmatrix} u_i \\ -v_i \end{bmatrix} / \sqrt{2}$, where u_i and v_i are the left and right singular vectors of $\boldsymbol{A}$ associated with the singular value σ_i.

Pr. 2.

(a) Here $\boldsymbol{A} = \boldsymbol{x}\boldsymbol{y}^T$ is written in outer product form. We can express $\boldsymbol{A}$ as

$$\boldsymbol{A} = \boldsymbol{x} \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix} = \begin{bmatrix} y_1 \boldsymbol{x} & \dots & y_n \boldsymbol{x} \end{bmatrix},$$

which has (at most) one linearly independent column because for every $i \neq j$, if $\boldsymbol{A}_{:,i}$ and $\boldsymbol{A}_{:,j}$ denote the ith and jth column of $\boldsymbol{A}$, then

$$y_i \boldsymbol{A}_{:,j} - y_j \boldsymbol{A}_{:,i} = y_i y_j \boldsymbol{x} - y_j y_i \boldsymbol{x} = 0.$$

Because $\boldsymbol{y} \neq \boldsymbol{0}$ is given, we can choose i and j such that at least one of y_i or y_j is nonzero; thus the columns of $\boldsymbol{A}$ are linearly dependent. Because $\boldsymbol{x}, \boldsymbol{y} \neq \boldsymbol{0}$ is given, $\boldsymbol{A}$ has rank one.
(If either of $\boldsymbol{x}$ or $\boldsymbol{y}$ were zero, then $\boldsymbol{A}$ would have rank zero.)
We can see this another way via the SVD. Write $\boldsymbol{A} = \sigma \boldsymbol{u}\boldsymbol{v}^T$ where $\sigma \triangleq \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle \cdot \langle \boldsymbol{y}, \boldsymbol{y} \rangle}$, $\boldsymbol{u} \triangleq \boldsymbol{x} / \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ and $\boldsymbol{v} \triangleq \boldsymbol{y} / \sqrt{\langle \boldsymbol{y}, \boldsymbol{y} \rangle}$. Comparing this with the SVD form $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ we have that the first column of $\boldsymbol{U}$ and $\boldsymbol{V}$ are equal to the $\boldsymbol{u}$ and $\boldsymbol{v}$ derived. The remaining columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ can be any set of orthogonal vectors that is in the ortho-complement of $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively Further, $\boldsymbol{\Sigma}$ has one nonzero element, given by σ. Because there is at most one nonzero singular value, the rank of $\boldsymbol{A}$ is 1, implying $\boldsymbol{A}$ has one linearly independent column.

(b) Typing the given `Julia` code yields Figure 1, where there $n = 100$ non-zero singular values. The discrepany arises because of numerical round-off error due to representing the matrix $\boldsymbol{A}$ in finite-precision arithmetic. Typing `rank(A)` gives the theoretically correct answer despite the $n - 1$ singular values not being identically equal to zero. This is because `Julia`'s `rank` function first computes the singular values and counts how many are greater than $\max(m, n)\epsilon$ where ϵ is machine precision (equal to 2.204×10^{-16} on my machine).

(c) Typing `@which rank(A)`

```
rank(A::AbstractArray{T<:Any,2}) at linalg\generic.jl:308
```

Clicking on the link to `linalg/generic.jl` reveals the rank function

```
function rank(A::AbstractMatrix)
    m,n = size(A)
    (m == 0 || n == 0) && return 0
    sv = svdvals(A)
    return sum(sv .> maximum(size(A))*eps(sv[1]))
end
```

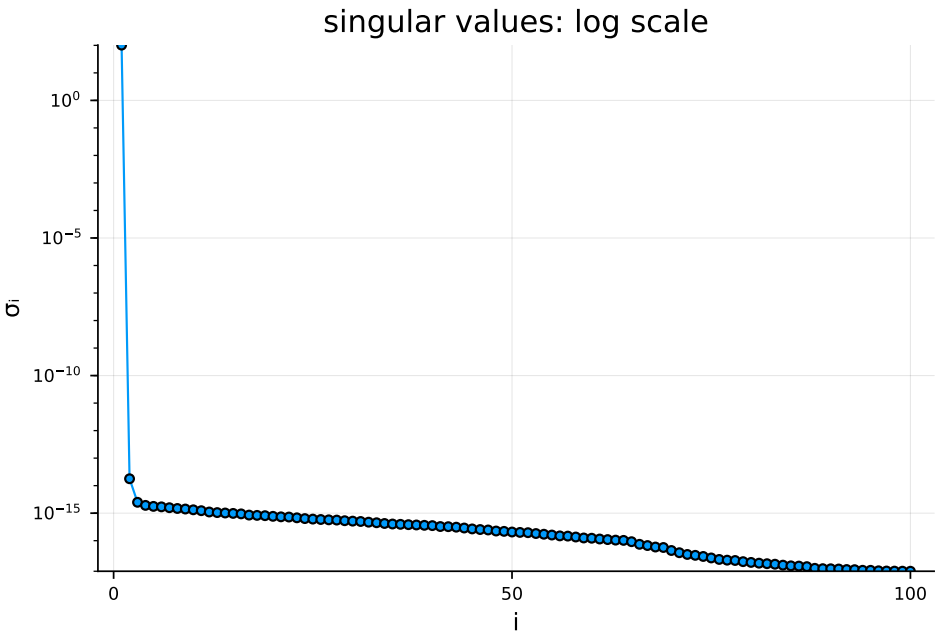


FIGURE 1. The singular values of $\boldsymbol{A} = \boldsymbol{x}\boldsymbol{y}^T$.

This reveals that the 'numerical rank' of the matrix $\boldsymbol{A}$ is determined by the formula

$$r = \{\text{Number of } \sigma_i > \epsilon \max(m, n)\},$$

where ϵ is machine precision. Note the 'continuous' looking spectrum of the noise-only singular values in Figure 1. This is the signature of noise-only singular values.
For a 1×2 array in double precision (Float64), the tolerance is $2\epsilon = 4.408 \times 10^{-16}$ on a typical 64-bit computer.

Pr. 3.
Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be a square matrix. The SVD of $\boldsymbol{A}$ always exists and is given by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. When the rank of $\boldsymbol{A}$ is n, $\boldsymbol{\Sigma}$ is an $n \times n$ diagonal matrix with strictly positive entries; otherwise it is an $n \times n$ diagonal matrix with $r < n$ strictly positive entries and is hence positive semi-definite. Since $\boldsymbol{V}$ is an orthogonal matrix, $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$ and hence we can express $\boldsymbol{A}$ as:

$$\begin{aligned}
\boldsymbol{A} &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \\
&= \boldsymbol{U}\boldsymbol{I}\boldsymbol{\Sigma}\boldsymbol{V}^T \quad \text{Because } \boldsymbol{A} \text{ is square, } \boldsymbol{U}, \boldsymbol{V} \text{ and } \boldsymbol{I} \text{ have the same dimensions.} \\
&= \underbrace{\boldsymbol{U}\boldsymbol{V}^T}_{\triangleq \boldsymbol{Q}} \underbrace{\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T}_{\triangleq \boldsymbol{S}}.
\end{aligned}$$

Recall from HW1 that $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{V}^T$ is an orthogonal matrix because $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{I}\boldsymbol{V}^T = \boldsymbol{I}$. The matrix $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is symmetric because $\boldsymbol{S}^T = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{S}$. It is positive definite when its eigenvalues, which are exactly equal to the n strictly positive singular values of $\boldsymbol{A}$, are strictly positive, and it is positive semi-definite when the rank of $\boldsymbol{A}$ is less than n. Note $\boldsymbol{S} = \boldsymbol{S}^T$ and all the eigenvalues of $\boldsymbol{S}$ being non-negative (or positive) is a sufficient and necessary test for positive semi-definiteness (resp. positive definiteness) of a matrix.

Pr. 4.
(a) We have that

$$\begin{aligned}
\boldsymbol{A} = \boldsymbol{x}\boldsymbol{y}^T &= \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2} ||\boldsymbol{x}||_2 ||\boldsymbol{y}||_2 \frac{\boldsymbol{y}^T}{||\boldsymbol{y}||_2} \\
&= \left[\frac{\boldsymbol{x}}{||\boldsymbol{x}||_2} \quad \boldsymbol{U}^\perp\right] \begin{bmatrix} ||\boldsymbol{x}||_2||\boldsymbol{y}||_2 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix} \left[\frac{\boldsymbol{y}}{||\boldsymbol{y}||_2}^T \quad \boldsymbol{V}^\perp\right]. \qquad (1)
\end{aligned}$$

We know that $\boldsymbol{x}\boldsymbol{y}^T$ is a rank–1 matrix, and thus has only one nonzero singular value. Furthermore, $\frac{\boldsymbol{x}}{||\boldsymbol{x}||_2}$ and $\frac{\boldsymbol{y}}{||\boldsymbol{y}||_2}$ are unit norm vectors, and $||\boldsymbol{x}||_2||\boldsymbol{y}||_2$ is a positive real number. Therefore, $\sigma_1 = ||\boldsymbol{x}||_2||\boldsymbol{y}||_2$ is the only nonzero singular value of $\boldsymbol{A}$, and $\boldsymbol{u}_1 = \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2}$, $\boldsymbol{v}_1 = \frac{\boldsymbol{y}}{||\boldsymbol{y}||_2}$ are the corresponding left and right singular vectors. The full SVD of $\boldsymbol{A}$ is given by (1), where $\boldsymbol{U}^\perp$ is any set of orthonormal vectors orthogonal to $\boldsymbol{u}_1$, and $\boldsymbol{V}^\perp$ is any set of orthonormal vectors orthogonal to $\boldsymbol{v}_1$.
Using the definition of pseudoinverse,

$$\boldsymbol{A}^\dagger = \frac{\boldsymbol{y}}{||\boldsymbol{y}||_2}\frac{1}{||\boldsymbol{x}||_2||\boldsymbol{y}||_2}\frac{\boldsymbol{x}^T}{||\boldsymbol{x}||_2} = \frac{\boldsymbol{y}\boldsymbol{x}^T}{||\boldsymbol{x}||_2^2||\boldsymbol{y}||_2^2}.$$

(b) This is a special case of (a) in which $\boldsymbol{y} = \boldsymbol{x}$. Thus:

$$\boldsymbol{A}^\dagger = \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2}\frac{1}{||\boldsymbol{x}||_2||\boldsymbol{x}||_2}\frac{\boldsymbol{x}^T}{||\boldsymbol{x}||_2} = \frac{\boldsymbol{x}\boldsymbol{x}^T}{||\boldsymbol{x}||_2^4}.$$

Pr. 5.
Approach 1:
If $\boldsymbol{y} \in \mathcal{N}(\boldsymbol{A}^T)$, then $\boldsymbol{A}^T\boldsymbol{y} = \boldsymbol{0}$, so $\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{y} = 0$ for any vector $\boldsymbol{x}$, so $(\boldsymbol{A}\boldsymbol{x})^T\boldsymbol{y} = 0$ and $\boldsymbol{A}\boldsymbol{x} \perp \boldsymbol{y}$. Thus $\boldsymbol{y} \in \mathcal{R}(\boldsymbol{A})^\perp$ because $\mathcal{R}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x}\}$. This shows that $\mathcal{N}(\boldsymbol{A}^T) \subseteq \mathcal{R}(\boldsymbol{A})^\perp$.
Conversely, if $\boldsymbol{y} \in \mathcal{R}(\boldsymbol{A})^\perp$ then $\boldsymbol{y} \perp \boldsymbol{A}\boldsymbol{x}$ for any $\boldsymbol{x}$ so $0 = (\boldsymbol{A}\boldsymbol{x})'\boldsymbol{y} = \boldsymbol{x}(\boldsymbol{A}'\boldsymbol{y})$ so choosing $\boldsymbol{x} = \boldsymbol{A}'\boldsymbol{y}$ we have $\|\boldsymbol{A}'\boldsymbol{y}\| = 0$ so $\boldsymbol{y} \in \mathcal{N}(\boldsymbol{A}^T)$. Thus $\mathcal{N}(\boldsymbol{A}^T) = \mathcal{R}(\boldsymbol{A})^\perp$.
Approach 2:
Let $\boldsymbol{A} \in \mathbb{R}_r^{m\times n}$. Then $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and $\boldsymbol{A}^T = \boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T$. Using our discussion on the anatomy of the SVD:

$$\boldsymbol{A} = \left[\underbrace{\boldsymbol{u}_1 \ \ldots \ \boldsymbol{u}_r}_{\text{basis for } \mathcal{R}(\boldsymbol{A})} \ \underbrace{\boldsymbol{u}_{r+1} \ \ldots \ \boldsymbol{u}_m}_{\text{basis for } \mathcal{R}(\boldsymbol{A})^\perp}\right] \boldsymbol{\Sigma} \left[\underbrace{\boldsymbol{v}_1 \ \ldots \ \boldsymbol{v}_r}_{\text{basis for } \mathcal{N}(\boldsymbol{A})^\perp} \ \underbrace{\boldsymbol{v}_{r+1} \ \ldots \ \boldsymbol{v}_n}_{\text{basis for } \mathcal{N}(\boldsymbol{A})}\right]^T$$

$$\boldsymbol{A}^T = \left[\underbrace{\boldsymbol{v}_1 \ \ldots \ \boldsymbol{v}_r}_{\text{basis for } \mathcal{R}(\boldsymbol{A}^T)} \ \underbrace{\boldsymbol{v}_{r+1} \ \ldots \ \boldsymbol{v}_n}_{\text{basis for } \mathcal{R}(\boldsymbol{A}^T)^\perp}\right] \boldsymbol{\Sigma}^T \left[\underbrace{\boldsymbol{u}_1 \ \ldots \ \boldsymbol{u}_r}_{\text{basis for } \mathcal{N}(\boldsymbol{A}^T)^\perp} \ \underbrace{\boldsymbol{u}_{r+1} \ \ldots \ \boldsymbol{u}_m}_{\text{basis for } \mathcal{N}(\boldsymbol{A}^T)}\right]^T$$

We see that $\mathcal{R}(\boldsymbol{A})^\perp = \text{span}(\{\boldsymbol{u}_{r+1}, \ldots, \boldsymbol{u}_m\}) = \mathcal{N}(\boldsymbol{A}^T)$.

Pr. 6.
(a) The problem of finding the line $y = \alpha x + b$ that best fits the points $(1, 2)$, $(2, 1)$ and $(3, 3)$ is equivalent to the problem: Find $\boldsymbol{\beta}$ that minimizes $||\boldsymbol{z} - \boldsymbol{A}\boldsymbol{\beta}||_2$, with $\boldsymbol{z} \triangleq \begin{bmatrix}2\\1\\3\end{bmatrix}$, $\boldsymbol{A} \triangleq \begin{bmatrix}1 & 1\\2 & 1\\3 & 1\end{bmatrix}$, $\boldsymbol{\beta} = \begin{bmatrix}\alpha\\b\end{bmatrix}$.
Thus the optimal LLS solution is $\widehat{\boldsymbol{\beta}} = \boldsymbol{A}^\dagger\boldsymbol{z} = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^T\boldsymbol{z} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{z}$.
Here $(\boldsymbol{A}^T\boldsymbol{A})^{-1} = \begin{bmatrix}14 & 6\\6 & 3\end{bmatrix}^{-1} = \frac{1}{14\cdot 3 - 6\cdot 6}\begin{bmatrix}3 & -6\\-6 & 14\end{bmatrix} = \frac{1}{6}\begin{bmatrix}3 & -6\\-6 & 14\end{bmatrix}$, and $\boldsymbol{A}^T\boldsymbol{z} = \begin{bmatrix}13\\6\end{bmatrix}$,
so $\widehat{\boldsymbol{\beta}} = \frac{1}{6}\begin{bmatrix}3 & -6\\-6 & 14\end{bmatrix}\begin{bmatrix}13\\6\end{bmatrix} = \frac{1}{6}\begin{bmatrix}3\\6\end{bmatrix} = \begin{bmatrix}0.5\\1\end{bmatrix}$.
We repeat this calculation using the SVD. Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ denote a SVD of $\boldsymbol{A}$. Numerically computing an SVD yields:

$$\boldsymbol{U} = \begin{bmatrix}-0.3231 & 0.8538 & 0.4082\\-0.5475 & 0.1832 & -0.8165\\-0.7719 & -0.4873 & 0.4082\end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix}4.0791 & 0\\0 & 0.6005\\0 & 0\end{bmatrix}, \quad \boldsymbol{V} = \begin{bmatrix}-0.9153 & -0.4027\\-0.4027 & 0.9153\end{bmatrix},$$

so $\widehat{\boldsymbol{\beta}} = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T = \boldsymbol{V}\begin{bmatrix}0.2451 & 0 & 0\\0 & 1.6653 & 0\end{bmatrix}\boldsymbol{U}^T\boldsymbol{z} = \begin{bmatrix}0.5\\1\end{bmatrix}$.
Unsurprisingly we get the same answer since $\boldsymbol{A}^+ = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}$ when $\boldsymbol{A}$ has full column rank.
(b) Here we are asked to find the line $y = \alpha x + b$ that best fits the points $(2, 1)$, $(1, 2)$ and $(3, 3)$. We could repeat the above computation, or we can recognize that all that has changed is that the first two points are swapped between (a) and (b). Thus the solution $\boldsymbol{\beta}$ of the two problems will be identical because the least-squares criterion is the same for any ordering of the (x_i, y_i) points. A more formal way of seeing is to first note that the new $\boldsymbol{A}$ matrix for (b) is exactly same as the $\boldsymbol{A}$ matrix in (a) except for the first two rows swapping places. Denote the

"$\boldsymbol{A}$" matrix in part (a) by $\boldsymbol{A}_a$ and the "$\boldsymbol{A}$" matrix in part (b) by $\boldsymbol{A}_b$. Then $\boldsymbol{A}_b = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\triangleq \boldsymbol{P}} \boldsymbol{A}_a$.

Note that $\boldsymbol{P}$ is a orthogonal matrix because $\boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}$ Such a $\boldsymbol{P}$ is called a permutation matrix. We also have that the new $\boldsymbol{z}_b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \boldsymbol{P}\boldsymbol{z}_a$. Thus the new problem is Find $\boldsymbol{\beta}_b$ that minimizes $||\boldsymbol{z}_b - \boldsymbol{A}_b\beta||_2 = ||\boldsymbol{P}\boldsymbol{z}_a - \boldsymbol{P}\boldsymbol{A}_a\beta||_2 = ||\boldsymbol{z}_a - \boldsymbol{A}_a\beta||_2$, because $\boldsymbol{P}$ is an orthogonal matrix. Thus we get the same solution $\boldsymbol{\beta}$.

Pr. 7.
Define for integer values d:

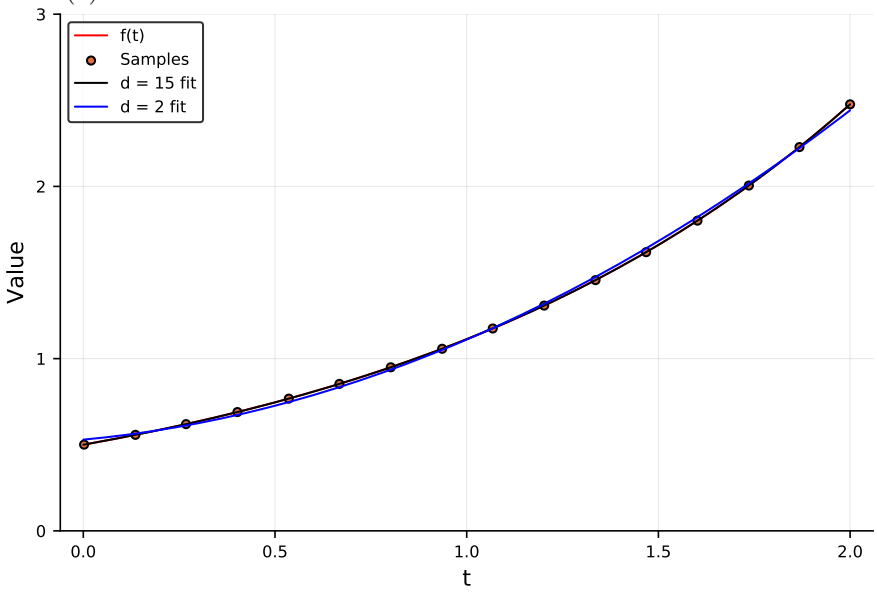
$$\boldsymbol{y} \triangleq \begin{bmatrix} f(t_1) \\ \vdots \\ f(t_{16}) \end{bmatrix} \in \mathbb{R}^{16}, \quad \boldsymbol{A} \triangleq \begin{bmatrix} 1 & t_1 & t_1^2 & \dots & t_1^d \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & t_{16} & t_{16}^2 & \dots & t_{16}^d \end{bmatrix} \in \mathbb{R}^{16\times(d+1)}, \quad \boldsymbol{x} \triangleq \begin{bmatrix} x_1 \\ \vdots \\ x_{d+1} \end{bmatrix} \in \mathbb{R}^{d+1}.$$

The solution $\widehat{\boldsymbol{x}} = \boldsymbol{A}^\dagger \boldsymbol{y}$ yields the the desired optimal least-squares estimate of the coefficients of the degree-d polynomial $p_d(t) = \sum_{i=1}^{d+1} x_i t^{i-1}$ that minimizes the error $||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}||_2$. The $\boldsymbol{A}$ of the form above is called a Vandermonde matrix.
Here is `Julia` code:

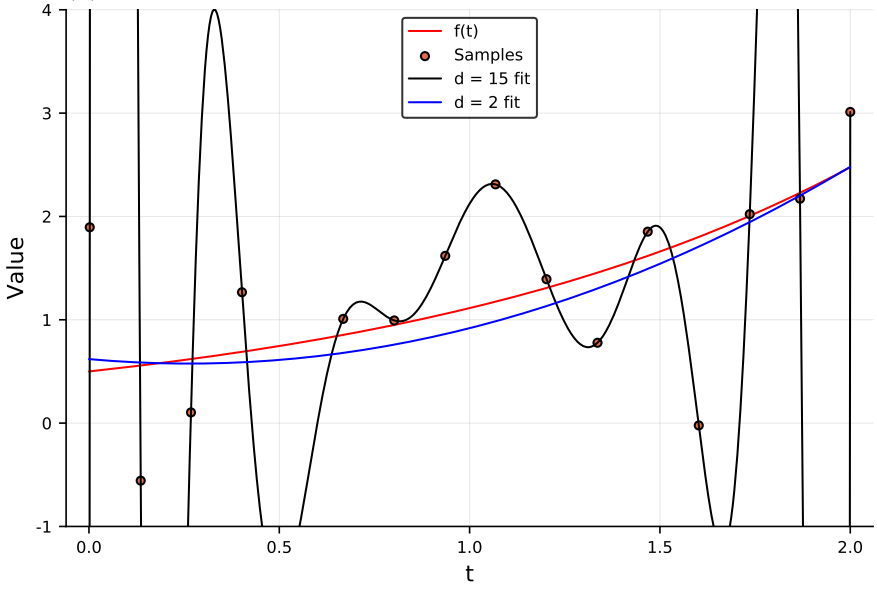
```
T = [2.1e-3, 0.136, 0.268, 0.402, 0.536,
     0.668, 0.802, 0.936, 1.068, 1.202,
     1.336, 1.468, 1.602, 1.736, 1.868, 2.000]

#f = 0.5 * exp.(0.8*T)                      # part a
f = 0.5 * exp.(0.8*T) + randn(length(T))  # part b
d = length(T)-1

A15 = [TT^j for TT in T, j=0:d]
A2  = A15[:,1:3]
x16 = pinv(A15)*f
x2  = pinv(A2)*f
t = linspace(0,2,500)

using Plots; pyplot()
plot(t, 0.5*exp.(0.8*t), color =:red, label = "f(t)", ylim=(-1,4))
scatter!(T,f, marker=:circle, label = "Samples")

a16 = [tt^j for tt in t, j=0:d]
a2 = a16[:,1:3]

plot!(t,(a16*x16), color =:black, label = "d = 15 fit")
plot!(t,a2*x2, color =:blue, label = "d = 2 fit", xaxis = "t", yaxis = "Value")
#savefig("tmp2.pdf")
```

This yielded the upper figure below. Note that the error for $d = 15$ is much smaller than for $d = 2$. In fact the $d = 15$ error is exactly zero (to within numerical precision) because $\text{rank}(\boldsymbol{A}) = 16 = dim(\boldsymbol{y})$ so the solution is exact! Comparing the least-squares coefficients obtained for $d = 2$ and $d = 15$ reveals that the first three terms are about the same. When noise is added we get the lower figure below. The polynomial for $d = 15$ passes *exactly* through the noisy samples. This property follows from the previous argument because $rank(\boldsymbol{A}) = 16$ so that error will be zero, relative to the samples, but large relative to the function. This example illustrates why choosing large model orders ($= d$) can make things worse by "over fitting" noise.

Part (a) - when there is no noise:



Part (b) - when there is noise.



Use the `cond(A)` command in `Julia` to obtain the condition number: the ratio of the largest and smallest singular value of a matrix. The condition number is important because the measurement error manifests as a $\delta\boldsymbol{y}$ so that the error in the least squares solution (relative to the noise-less case) is given by exactly by $\delta\widehat{\boldsymbol{x}} = \boldsymbol{A}^\dagger(\boldsymbol{y}+\delta\boldsymbol{y}) - \boldsymbol{A}^\dagger\boldsymbol{y} = \boldsymbol{A}^\dagger\delta\boldsymbol{y}$. A large condition number means that small $\delta\boldsymbol{y}$ can get amplified, producing large $\delta\boldsymbol{x}$ and hence instabilities as manifested in the overfitting.
An estimate of the degree to which $\delta\boldsymbol{x}$ can change as a function of the condition number κ is given by the relation:

$$\frac{||\delta\boldsymbol{x}||}{||\boldsymbol{x}||} \leq \kappa \frac{||\delta\boldsymbol{y}||}{||\boldsymbol{y}||}.$$

Here $||\delta\boldsymbol{y}||$ is the norm of the noise. The condition number $\kappa = \sigma_1(\boldsymbol{A})/\sigma_{\min(m,n)}(\boldsymbol{A})$ is large when $d = 15$ (check it) compared to when $d = 2$.

Pr. 8.
Here $\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $\boldsymbol{A}^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and the optimal least-squares estimate is: $\boldsymbol{x}^* = \boldsymbol{A}^\dagger\boldsymbol{b} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

(a) Consider $\boldsymbol{A}_1 = \begin{bmatrix} 1 & \delta \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & \delta \end{bmatrix}$ so that by inspection an SVD of $\boldsymbol{A}_1$ is $\boldsymbol{A}_1 = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T$ where $\sigma_1 = \sqrt{1+\delta^2}$, $\boldsymbol{u}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\boldsymbol{v}_1 = \begin{bmatrix} 1/\sqrt{1+\delta^2} \\ \delta/\sqrt{1+\delta^2} \end{bmatrix}$. We do not need to compute the other singular vectors because the optimal (minimum-norm) least-squares solution is simply: $\boldsymbol{z}^* = \boldsymbol{A}_1^\dagger \boldsymbol{b} = \frac{1}{\sigma_1} \boldsymbol{v}_1 \boldsymbol{u}_1^T \boldsymbol{b} = \frac{1}{1+\delta^2} \begin{bmatrix} 1 \\ \delta \end{bmatrix}$, for which $\boldsymbol{z}^* \to \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ as $\delta \to 0$, so $||\boldsymbol{z}^* - \boldsymbol{x}^*||_2 \to 0$.

(b) Consider now the case where $\boldsymbol{A}_2 = \begin{bmatrix} 1 & 0 \\ 0 & \delta \end{bmatrix}$, which has rank$(\boldsymbol{A}) = 2$ for $\delta > 0$. Here the optimal least-squares solution is: $\boldsymbol{z}^* = \boldsymbol{A}_2^\dagger \boldsymbol{b} = \begin{bmatrix} 1 \\ 1/\delta \end{bmatrix}$ so that $||\boldsymbol{x}^* - \boldsymbol{z}^*||_2 \to \infty$ as $\delta \to 0$.

This exercise illustrates that the manner in which error is introduced can impact a solution negligibly or dramatically!

Pr. 9.

Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^H$. If rank$(\boldsymbol{A}) = n$, then $\boldsymbol{A}$ has n linearly independent columns, $n \leq m$, and $A^H A$ is invertible. Then:

$$(\boldsymbol{A}^H\boldsymbol{A})^{-1}\boldsymbol{A}^H = (\boldsymbol{V}\boldsymbol{\Sigma}^H\boldsymbol{U}^H\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^H)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^H\boldsymbol{U}^H = (\boldsymbol{V}\boldsymbol{\Sigma}^H\boldsymbol{\Sigma}\boldsymbol{V}^H)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^H\boldsymbol{U}^H$$

$$= \boldsymbol{V}(\boldsymbol{\Sigma}^H\boldsymbol{\Sigma})^{-1}\boldsymbol{\Sigma}^H\boldsymbol{U}^H = \boldsymbol{V} \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2^2} & \dots & 0 \\ & \ddots & \ddots & \\ 0 & 0 & \dots & \frac{1}{\sigma_n^2} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & \dots & 0 \\ & \ddots & & & \ddots & \\ 0 & 0 & \dots & \sigma_n & \dots & 0 \end{bmatrix} \boldsymbol{U}^H$$

$$= \boldsymbol{V} \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \dots & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2} & \dots & 0 & \dots & 0 \\ & \ddots & & & \ddots & \\ 0 & 0 & \dots & \frac{1}{\sigma_n} & \dots & 0 \end{bmatrix} \boldsymbol{U}^H = \boldsymbol{A}^\dagger.$$

Pr. 10.

A possible `Julia` implementation is

```
function compute_normals(I, L)
#
# Syntax:       N = compute_normals(I, L)
#
# Inputs:       I is an m x n x d matrix whose d slices contain m x n images
#               of a common scene under different lighting conditions
#
#               L is a 3 x d matrix whose columns are the lighting direction
#               vectors for the images in I, with d >= 3
#
# Outputs:      N is an m x n x 3 matrix containing the unit-norm
#               surface normal vectors for each pixel in the scene
#

    m, n, d = size(I) # Parse inputs
    L = mapslices(normalize, L, 1) # Normalize lighting direction vectors

    # Solve least squares problem
    # Here, using pinv() is efficient because L is a small matrix
    # and we apply pinv(L) to many (mn) pixels.
    I = reshape(I, m * n, d)
    N = I * pinv(L)
    N = reshape(N, m, n, 3)

    # alternative one-line "Julia way" that avoids reshape:
    # N = mapslices((v) -> pinv(L')*v, I, 3) # apply pinv to each pixel
    # However, it ran slower than the above, per autograder output.

    N = mapslices(normalize, N, 3) # Normalize normal vectors
    return N
end
```