Pr. 1.

(a) Let $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, …, $\boldsymbol{A}_K$ and $\boldsymbol{B}$ denote $M \times N$ matrices. Determine the solution of the "matrix fitting" least-squares optimization problem:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}=(x_1,\ldots,x_K)\in\mathbb{F}^K}{\arg\min} \left\| \sum_{k=1}^{K} x_k \boldsymbol{A}_k - \boldsymbol{B} \right\|_{\mathrm{F}}.$$

(b) Write (by hand) a simple `Julia` expression for computing $\hat{\boldsymbol{x}}$ given $\boldsymbol{B}$ and the $\boldsymbol{A}_k$ matrices when $K = 3$.

(c) Optional challenge. How would you handle the general K case in `Julia`?

Pr. 2.
Let $\boldsymbol{A}$ be an $M \times N$ matrix, and let $\boldsymbol{B}$ be an $M \times K$ matrix.

(a) Determine the solution to

$$\hat{\boldsymbol{X}} = \underset{\boldsymbol{X}\in\mathbb{F}^{N\times K}}{\arg\min} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|_{\mathrm{F}}.$$

(b) Write (by hand) a short `Julia` expression for computing $\hat{\boldsymbol{X}}$ efficiently given $\boldsymbol{A}$ and $\boldsymbol{B}$ in the case that $\boldsymbol{B}$ is very wide, *i.e.*, $M \approx N \ll K$.

Pr. 3.

(a) Consider the **linear LS** problem $\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$. We have seen that when $\boldsymbol{A}$ has **full column rank**, the solution is $\hat{\boldsymbol{x}} = (\boldsymbol{A}'\boldsymbol{A})^{-1}\boldsymbol{A}'\boldsymbol{y}$, which involves inverting $\boldsymbol{A}'\boldsymbol{A}$.

Express the **condition number** of $\boldsymbol{A}'\boldsymbol{A}$ in terms of the **singular values** of $\boldsymbol{A}$.

(b) The **Tikhonov regularized solution** is $\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|^2 = (\boldsymbol{A}'\boldsymbol{A} + \beta\boldsymbol{I})^{-1}\boldsymbol{A}'\boldsymbol{y}$. Here we invert a different matrix. Express the condition number of that matrix in terms of the singular values of $\boldsymbol{A}$.

Verify that the regularized solution has a "better" condition number.

Pr. 4.
A matrix $\boldsymbol{T}$ has the property $\boldsymbol{T}^3 = \boldsymbol{I}$. What are the possible **eigenvalues** of $\boldsymbol{T}$?

Pr. 5.
The **power method** uses the iteration

$$\boldsymbol{x}_{k+1} = \frac{\boldsymbol{A}\boldsymbol{x}_k}{\|\boldsymbol{A}\boldsymbol{x}_k\|_2}$$

and **converges** to $\boldsymbol{v}_1$, the leading **eigenvector** of $\boldsymbol{A}$ associated with its largest (in magnitude) **eigenvalue**, provided $\boldsymbol{x}_0'\boldsymbol{v}_1 \neq 0$ and the largest (in magnitude) eigenvalue is unique.

(a) Suppose $\boldsymbol{A}$ is a Hermitian symmetric matrix with distinct (in magnitude) eigenvalues. Let λ_1 denote the largest eigenvalue (meaning largest, not largest in magnitude!) and assume that it is known. Describe how you can use one run of the power method (with a modified input and some simple processing of the output) to compute the eigenvector associated with the *smallest* eigenvalue of $\boldsymbol{A}$, assuming it is unique. Here we mean smallest value, *not* smallest magnitude. (You are not allowed to invert $\boldsymbol{A}$ because that is too expensive in large applications.)

Hint: What simple transformation of $\boldsymbol{A}$ maps its smallest eigenvalue to its largest (in magnitude)?

(b) How would you modify the scheme if λ_1 is not known? (You may use more than one power iteration in this case, but you still may not invert $\boldsymbol{A}$.)

Pr. 6.
Consider two **vector spaces** $\mathcal{V}_\alpha$ and $\mathcal{V}_\beta$ with corresponding **norms** $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$, respectively. A function $f : \mathcal{V}_\alpha \mapsto \mathcal{V}_\beta$ is called an L-**Lipschitz function** with respect to those norms if for every $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, there exists a real constant $L > 0$ such that

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|_\beta \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_\alpha.$$

The constant L is called the **Lipschitz constant** of f and we say that f is **Lipschitz continuous** with Lipschitz constant L. In general, the norm on the left hand side of the inequality can differ from the norm on the right hand side. When the norm $\|\cdot\|$ is not specified, it is assumed that f is a Lipschitz function with respect to the Euclidean norm.

(a) Show that $f(\boldsymbol{x}) = \nabla g(\boldsymbol{x}) = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$ is a Lipschitz function when $\boldsymbol{A}$ is a $M \times N$ matrix.
Express its Lipschitz constant concisely.

Note that $f(\boldsymbol{x})$ is the gradient of the **least squares** cost function $g(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$.

(b) Show that the largest **singular value** of a matrix is a 1-Lipschitz function (of its MN entries), with respect to both the **spectral** norm and the **Frobenius norm**.
Hint: do the spectral norm first, using the triangle inequality, then use a previous HW problem.

Functions with Lipschitz continuous gradients are important because if the function has a Lipschitz constant L, then for any $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^N$,

$$f(\boldsymbol{x}) \leq f(\boldsymbol{z}) + \nabla f(\boldsymbol{z})^T(\boldsymbol{x} - \boldsymbol{z}) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2.$$

This inequality is very helpful in analyzing iterative algorithms, such as **gradient descent**, because substituting $\boldsymbol{x} = \boldsymbol{x}_{k+1}$ and $\boldsymbol{z} = \boldsymbol{x}_k$ yields a bound on the value of the cost function after k iterations that can be used to select the step size to ensure that the value of the cost function will decrease after every iteration (unless we are already at a minimizer where the gradient is zero).

Pr. 7.
(Projection onto orthogonal complement of nullspace)

(a) Determine a simple **orthonormal basis** for the **nullspace** of the matrix $\boldsymbol{X} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

(b) Determine a simple orthonormal basis for the **orthogonal complement** of the nullspace of that matrix $\boldsymbol{X}$.

(c) Determine the **projection** of the vector $\boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$ onto $\mathcal{N}^{\perp}(\boldsymbol{X})$.

(d) Repeat the previous three parts for the matrix $\boldsymbol{Y} = \mathbf{1}_3 \mathbf{1}_3'$ and the vector $\boldsymbol{w} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$.

Here a numerical solution is fine.

(e) Write a function called `orthcompnull` that projects an input vector $\boldsymbol{x}$ onto the orthogonal complement of the nullspace of any input matrix $\boldsymbol{A}$.

For full credit, your final version of the code should be computationally efficient in the case where $\boldsymbol{x}$ is a matrix with n rows and many columns, each of which we want to project onto the orthogonal complement of the nullspace of input matrix $\boldsymbol{A}$.

In `Julia`, your file should be named `orthcompnull.jl` and should contain the following function:

```julia
"""
`z = orthcompnull(A, x)`

In:
* `A` `m x n` matrix
* `x` vector of length `n`, or matrix with `n` rows and many columns

Out:
* `z` : vector or matrix of size ??? (you determine this)

Projects `x` onto the orthogonal complement of the null space
of the input matrix `A`

For full credit, your solution should be computationally efficient!
"""
function orthcompnull(A, x)
```

Submit your solution to the autograder by emailing it as an attachment to `eecs551@autograder.eecs.umich.edu`.

Test your code yourself using the examples above (and others as needed) *before* submitting to the autograder.

(f) Submit your code (a screen capture is fine) to gradescope so that the grader can verify that your code is computationally efficient. (The autograder checks only correctness, not efficiency.)

Optional problem(s) below
(not graded, but solutions will be provided for self check; do not submit to gradescope)

Pr. 8.
Let $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{x}\boldsymbol{x}'$ denote a **rank-one update** of a matrix $\boldsymbol{A}$, where

- $\boldsymbol{A}$ is diagonal with real entries and $a_{11} > a_{22} > \ldots > a_{nn}$
- $x_i > 0$ so that (as shown in previous HW) the eigenvalues of $\boldsymbol{B}$ differ from those of $\boldsymbol{A}$.

(a) Determine the **eigenvectors** of $\boldsymbol{B}$ in terms of $\boldsymbol{x}$, $\boldsymbol{A}$ and the **eigenvalues** of $\boldsymbol{B}$ (found in a previous HW).

(b) Prove that the eigenvectors are orthogonal. Hint: a simple **partial fraction expansion** may be helpful.

Pr. 9.
Determine the **eigenvalues** and **eigenvectors** of $\boldsymbol{A} = \boldsymbol{x}\boldsymbol{x}' + \boldsymbol{y}\boldsymbol{y}'$, where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}^N$.
You need only find the eigenvector(s) that correspond to nonzero eigenvalue(s).
Assume that $\boldsymbol{x}'\boldsymbol{y} = \rho \neq 0$ and focus on the case where $\boldsymbol{A}$ has its maximum possible **rank**.
Hint: The desired eigenvector(s) of $\boldsymbol{A}$ must be in the **range** of $\boldsymbol{A}$. Hence, any eigenvector must be linearly related to $\boldsymbol{x}$ and $\boldsymbol{y}$. Also, write $\boldsymbol{A}$ as $\boldsymbol{Z}\boldsymbol{Z}'$ for some simple matrix $\boldsymbol{Z}$. Check your answers numerically.

Pr. 10.
Let matrix $\boldsymbol{A}$ have **compact SVD** $\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r'$ where $\boldsymbol{U} = [\boldsymbol{U}_r \ \boldsymbol{U}_0]$ and $\boldsymbol{V} = [\boldsymbol{V}_r \ \boldsymbol{V}_0]$. Express the following subspaces (**ranges** and **nullspaces**) in terms of $\boldsymbol{U}_r$, $\boldsymbol{U}_0$, $\boldsymbol{V}_r$, and/or $\boldsymbol{V}_0$.

(a) $\mathcal{N}^\perp(\boldsymbol{A}')$

(b) $\mathcal{R}(\boldsymbol{A}')$

(c) $\mathcal{R}^\perp(\boldsymbol{A}\boldsymbol{A}^+)$

(d) $\mathcal{N}(\boldsymbol{A}^+)$

(e) $\mathcal{R}(\boldsymbol{A}^+\boldsymbol{A})$
