Pr. 1.

(a) Prove the "**vec trick**," *i.e.*, that for arbitrary (possibly complex-valued) matrices of compatible sizes:

$$\operatorname{vec}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T) = (\boldsymbol{B} \otimes \boldsymbol{A})\operatorname{vec}(\boldsymbol{X}), \tag{1}$$

where **vec(.)** is the operator that stacks the columns of the input matrix into a vector, and $\otimes$ denotes the **Kronecker product**. Note that here we really mean transpose $\boldsymbol{B}^T$ even if $\boldsymbol{B}$ is complex valued!

(b) One application of (1) is computing a 2D **discrete Fourier transform** (DFT). The (1D) DFT of a vector $\boldsymbol{x} \in \mathbb{C}^N$ is the vector $\boldsymbol{f}_x \in \mathbb{C}^N$ with entries

$$[\boldsymbol{f}_x]_k = \sum_{n=1}^{N} x_n \exp\left(\frac{-2\pi i(k-1)(n-1)}{N}\right), \quad k = 1, \ldots, N.$$

(Here we are using the linear algebra (and `Julia`) convention where the first array is 1, whereas DSP books usually use $0, \ldots, N-1$.) We can represent the above computation in matrix-vector form as $\boldsymbol{f}_x = \boldsymbol{F}_N \boldsymbol{x}$, where $\boldsymbol{F}_N$ is the $N \times N$ DFT matrix with entries

$$(\boldsymbol{F}_N)_{kn} = \exp\left(\frac{-2\pi i(k-1)(n-1)}{N}\right).$$

We can generate $\boldsymbol{F}_N$ in `Julia` as

```
# N x N DFT matrix
using FFTW
using LinearAlgebra
F = fft(Matrix(I, N, N), 1)
```

One can verify that $\boldsymbol{F}_N^H \boldsymbol{F}_N = N\boldsymbol{I}_N$, so the (1D) **inverse DFT** of $\boldsymbol{f}_x$ can be computed as $\boldsymbol{x} = (1/N)\boldsymbol{F}_N^H \boldsymbol{f}_x$.

In two dimensions, we compute the **2D DFT**, call it $\boldsymbol{F}_X$, of the $M \times N$ matrix $\boldsymbol{X}$, by computing the 1D DFT of each column of $\boldsymbol{X}$ followed by the 1D DFT of each row of the result (or vice versa).

Explain why

$$\boldsymbol{F}_X = \boldsymbol{F}_M \boldsymbol{X} \boldsymbol{F}_N^T$$

computes the 2D DFT of $\boldsymbol{X}$; then use (1) to write $\operatorname{vec}(\boldsymbol{F}_X)$ as the product of a matrix and $\operatorname{vec}(\boldsymbol{X})$.

(c) Show that

$$\boldsymbol{X} = \frac{1}{MN} \boldsymbol{F}_M^H \boldsymbol{F}_X \overline{\boldsymbol{F}_N}$$

computes the **2D inverse DFT** of $\boldsymbol{F}_X$, where $\overline{\boldsymbol{Y}}$ denotes (element-wise) complex conjugate; then use (1) to write $\operatorname{vec}(\boldsymbol{X})$ as the product of a matrix and $\operatorname{vec}(\boldsymbol{F}_X)$.

Hint: Use the fact that $\boldsymbol{F}_N^H \boldsymbol{F}_N = \boldsymbol{F}_N \boldsymbol{F}_N^H = N\boldsymbol{I}_N$.

Pr. 2.

In the ordinary **least-squares** problem we found the (minimum norm) $\hat{\boldsymbol{x}}$ that minimized $\|\boldsymbol{r}\|_2$ where $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ is the residual. We saw that the optimal $\boldsymbol{x}$ can be expressed entirely in terms of $\boldsymbol{b}$ and an SVD of $\boldsymbol{A}$. Let $\boldsymbol{A}$ be an $m \times n$ matrix so that $\boldsymbol{x}$ is an $n \times 1$ vector and $\boldsymbol{b}$ is an $m \times 1$ vector. Thus $\boldsymbol{r}$ is an $m \times 1$ vector.

In applications with **heteroscedastic measurement errors**, we prefer to minimize $\|\boldsymbol{W}\boldsymbol{r}\|_2$, *i.e.*, a weighted squared error, where $\boldsymbol{W}$ is a diagonal matrix having diagonal entries $w_1, \ldots w_m \geq 0$. Determine the optimal $\boldsymbol{x}$ for this **weighted least-squares** problem. (Again, you should express the answer in terms of $\boldsymbol{b}$, $\boldsymbol{W}$, and an SVD of an appropriate matrix.)

Pr. 3.

Let $\boldsymbol{A}$ be a diagonal matrix with real entries that are distinct and non-zero. Let $\boldsymbol{x}$ be a vector with all non-zero entries. Determine how the **eigenvalues** of the **rank-one update** $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{x}\boldsymbol{x}'$ are related to the eigenvalues of $\boldsymbol{A}$ and the vector $\boldsymbol{x}$.

Hint: They will be implicitly related (not in a closed form expression) as the solution of an equation involving the eigenvalues of $\boldsymbol{A}$ and the elements of $\boldsymbol{x}$. Hint: $\boldsymbol{G} + \boldsymbol{H} = \boldsymbol{G}(\boldsymbol{I} + \boldsymbol{G}^{-1}\boldsymbol{H})$ if $\boldsymbol{G}$ is invertible.

Pr. 4.

(a) For $\delta > 0$, determine the solution of the **regularized least-squares** problem

$$\hat{\boldsymbol{x}}(\delta) = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \delta^2 \|\boldsymbol{x}\|_2^2.$$

Express the answer in terms of the SVD of an appropriate matrix.
Use the fact that $\boldsymbol{B}^+ = (\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{B}'$ when $\boldsymbol{B}'\boldsymbol{B}$ is invertible to simplify the expression.
Hint: Rewrite the cost function so it looks like a usual least-squares problem.

(b) What does $\hat{\boldsymbol{x}}(\delta)$ tend to as $\delta \to \infty$? Does this answer make sense?

(c) Write (by hand, not code) an iteration based on the **gradient descent** (GD) method such that the iterates converge to the minimizer $\hat{\boldsymbol{x}}(\delta)$.

(d) Determine a condition on the step size μ that guarantees convergence of your GD method. Express the condition in terms of the original problem quantities: $\boldsymbol{A}$, $\boldsymbol{b}$, and δ.

Pr. 5.
Let $\boldsymbol{A}$ be a Hermitian symmetric (square) $n \times n$ matrix. Suppose $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}'$ with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq 0$. and $0 \geq \lambda_{k+1} \geq \ldots \geq \lambda_n < 0$. Determine an **SVD** of $\boldsymbol{A}$.

Pr. 6.
Make a clearly stated problem that could be used on an EECS 551 exam based the topics covered so far, and provide your own solution to the problem. Your problem could include something about `Julia`, but in that case should be about concepts, not mere syntax, and should not require submitting code to an autograder.
If you make a good problem (not too trivial, not too hard) then it might be used on an actual exam.
To earn full credit:

- Your problem and solution must fit on a single page with a horizontal line that clearly separates the question (above) from the answer (below).
- Your solution to the problem must be correct.
- Your problem should not be excessively trivial (nor beyond the scope of the course).
- Your problem must not be identical to any quiz or homework or clicker problem.
- Your problem should not be of the form "prove that ..." but it is fine to pose a True/False problem with an answer that needs some proving to explain.
- You must upload your problem/solution to gradescope as usual for grading.
- You must submit your problem/solution in pdf format to Canvas for distribution by the deadline for this HW. (We can export from Canvas easily but not from gradescope.)
- Do not submit any other parts of your HW assignment to Canvas; only this 1 page pdf file.
- Do not put your name on the problem/solution that you upload to Canvas because we plan to export all problems/solutions to distribute as practice problems.

Pr. 7.

(Nesterov-accelerated gradient descent for least squares problems)

This problem describes an accelerated gradient descent method due to Nesterov for the least squares problem

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

that converges provably faster (in a worst-case sense) than the standard gradient descent method. The method is initialized with $t_0 = 1$ and $\boldsymbol{z}_0 = \boldsymbol{x}_0$. and consists of the following iteration for $k = 0, 1, \ldots$:

$$\begin{aligned} t_{k+1} &= \frac{1 + \sqrt{1 + 4t_k^2}}{2} \\ \boldsymbol{x}_{k+1} &= \boldsymbol{z}_k - \mu \boldsymbol{A}' \left(\boldsymbol{A}\boldsymbol{z}_k - \boldsymbol{b}\right) \qquad \text{(gradient step)} \\ \boldsymbol{z}_{k+1} &= \boldsymbol{x}_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right) (\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \qquad \text{(momentum step)}\ . \end{aligned}$$

The sequence $\{\boldsymbol{x}_k\}$ generated by the Nesterov-accelerated algorithm converges to $\hat{\boldsymbol{x}}$ when $0 < \mu \leq 1/\sigma_1^2(\boldsymbol{A})$.

(a) Write a function called `lsngd` that implements the above Nesterov-accelerated least squares gradient descent algorithm.

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

```
"""
`x = lsngd(A, b ; x0 = zeros(size(A,2)), nIters = 200, mu = 0)

Perform Nesterov—accelerated gradient descent to solve the LS problem
``\\argmin_x 0.5 \\| A x — b \\|_2``

In:
— `A` `m x n` matrix
— `b` vector of length `m`

Option:
— `x0` initial starting vector (of length n) to use; default 0 vector.
— `nIters` number of iterations to perform; default 200.
— `mu` step size, must satisfy ``0 < \\mu <= 1 / \\sigma_1(A)^2`
to guarantee convergence, where ``\\sigma_1(A)`` is the first (largest) singular value.
Ch.5 will explain a default value for `mu`.

Out:
`x` vector of length `n` containing the approximate solution

"""
function lsngd(A::AbstractMatrix{<:Number}, b::AbstractVector{<:Number}
        ; x0::AbstractVector{<:Number} = zeros(eltype(b), size(A,2)),
        nIters::Int = 200, mu::Real = 0)
```

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

The template above includes type declarations for each of the input variables. Such annotations are not essential in general, but often can be helpful.

(b) After your code passes, use it to generate a plot of $\log_{10}(\|\boldsymbol{x}_k - \hat{\boldsymbol{x}}\|/\|\hat{\boldsymbol{x}}\|)$ as a function of $k = 0, 1, \ldots, 200$ using $\mu = 1/\sigma_1^2(\boldsymbol{A})$ for $\boldsymbol{x}_0 = \boldsymbol{0}$ and $\boldsymbol{A}$ and $\boldsymbol{b}$ generated as

```
using Random: seed!
m = 100; n = 50; sigma = 0.1;
seed!(0); A = randn(m, n); xtrue = rand(n);
b = A * xtrue + sigma * randn(m);
```

Submit your plot to gradescope.

(c) Compare the convergence characteristics of the accelerated gradient descent method to the standard gradient descent method. For a given step size μ, which converges faster to the true solution? Does this hold for different values of (allowable) μ values? Turn your plots for at least two values of μ that illustrate and compare the rates of convergence of standard gradient descent and Nesterov's accelerated method. You must plot both algorithms on the same graph to compare them. The accelerated gradient descent method does not necessarily decrease $\|\boldsymbol{x}_k - \boldsymbol{x}_*\|$ monotonically, so some wiggles in its curves are expected.

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

Pr. 8.

(a) Suppose $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ has SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}' = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i'$
Spherical manifold optimization problem:

$$\boldsymbol{x}_{\text{opt}} = \arg\max_{\|\boldsymbol{x}\|_2=1} \|\boldsymbol{A}\boldsymbol{x}\|_2, \qquad \widetilde{\boldsymbol{x}}_{\text{opt}} = \arg\max_{\|\boldsymbol{x}\|_2=1} \|\boldsymbol{x}'\boldsymbol{A}\|_2 \tag{2}$$

- $\boldsymbol{x}_{\text{opt}} =?$
- $\widetilde{\boldsymbol{x}}_{\text{opt}} =?$
- When are $\boldsymbol{x}_{\text{opt}}$ and $\widetilde{\boldsymbol{x}}_{\text{opt}}$ unique (to within a sign ambiguity)?
- Do answers change if replace $\boldsymbol{A}$ with $-\boldsymbol{A}$ in objective function? Explain why or why not?
- What constraints could you add to above manifold optimization problem (with the same objective function) so you get $\boldsymbol{u}_r$ and $\boldsymbol{v}_r$? Hint: Assume you know bases vectors for $\mathcal{N}(\boldsymbol{A})$ and $\mathcal{N}(\boldsymbol{A}')$.

(b) Suppose $\boldsymbol{A} \in \mathbb{F}^{m \times m}$ has eigenvalue decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}' = \sum_{i=1}^{m} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i'$, with $\lambda_1 \geq \ldots \geq \lambda_n$ and $\lambda_i \in \mathbb{R}$
Spherical manifold optimization problem:

$$\boldsymbol{x}_{\text{opt}} = \arg\max_{\|\boldsymbol{x}\|_2=1} \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} \tag{3}$$

- Prove that $\boldsymbol{x}_{\text{opt}} = z\boldsymbol{u}_1$ where $|z| = 1$
- What is $\boldsymbol{x}_{\text{opt}}'\boldsymbol{A}\boldsymbol{x}_{\text{opt}}$?
- When is $\boldsymbol{x}_{\text{opt}}$ unique (aside from the sign ambiguity)?

(c) Suppose $\boldsymbol{A} \in \mathbb{F}^{m \times m}$ has eigenvalue decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}' = \sum_{i=1}^{m} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i'$ where $\lambda_i \in \mathbb{R}$
Spherical manifold optimization problem:

$$\boldsymbol{x}_{\text{opt}} = \arg\max_{\|\boldsymbol{x}\|_2=1} \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} \text{ subject to } \boldsymbol{x} \perp \boldsymbol{u}_1 \perp \boldsymbol{v}_2, \ldots, \boldsymbol{u}_{k-1} \tag{4}$$

Claim:

$$\boldsymbol{x}_{\text{opt}} = \boldsymbol{u}_k \tag{5}$$

- Prove that $\boldsymbol{x}_{\text{opt}} = z\boldsymbol{u}_k$ where $|z| = 1$, via an equivalent formulation involving projections
- What is $\boldsymbol{x}_{\text{opt}}'\boldsymbol{A}\boldsymbol{x}_{\text{opt}}$?
- When is $\boldsymbol{x}_{\text{opt}}$ unique (aside from the sign ambiguity)?

Pr. 9.
Suppose $\boldsymbol{b} = \boldsymbol{\Phi}\boldsymbol{z}$ for some vector $\boldsymbol{z}$, where $\boldsymbol{\Phi}$ is a **Parseval tight frame**.

(a) For $\delta \geq 0$, Express the solution of the **regularized least-squares** problem

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \delta^2 \|\boldsymbol{x}\|_2^2$$

in terms of $\boldsymbol{z}$ as simply as possibly.

(b) What value of δ would be the most appropriate in this special case where $\boldsymbol{\Phi}$ is a tight frame?

(c) What is the **condition number** of $\boldsymbol{\Phi}$ here?

Pr. 10.
Show that $\boldsymbol{\Phi}$ is a tight frame with frame bound α, then $\boldsymbol{\Phi}'\boldsymbol{\Phi} \preceq \boldsymbol{I}$.
