Pr. 1.
Given an $N \times N$ matrix $\boldsymbol{A}$ and initial vector $\boldsymbol{x}_0 \in \mathbb{F}^N$, the **power iteration** uses the iterative recursion $\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k / \|\boldsymbol{A}\boldsymbol{x}_k\|_2$. If $\boldsymbol{A}$ is an $N \times N$ Hermitian matrix for which the eigenvalue with the largest magnitude exceed all others, *i.e.*, $|\lambda_n| < |\lambda_1|$, $n = 2, \dots, N$, then the sequence $\{\boldsymbol{x}_k\}$ generated by the power iteration **converges** to $\mathrm{e}^{\imath\phi}\boldsymbol{v}_1$ as $k \to \infty$, where $\boldsymbol{v}_1$ is the leading unit-norm **eigenvector** associated with the largest (in magnitude) **eigenvalue** λ_1, and $\mathrm{e}^{\imath\phi}$ is an arbitrary phase factor. if $\boldsymbol{v}_1'\boldsymbol{x}_0 = [\boldsymbol{V}'\boldsymbol{x}_0]_1 \neq 0$. (Ch. 8 considers more generality.)

(a) Let $\boldsymbol{B}$ denote an $N \times N$ Hermitian symmetric matrix with distinct (in magnitude) eigenvalues, ordered such that $\lambda_N < \lambda_{N-1} < \ldots < \lambda_2 < \lambda_1$. Suppose that λ_1 is known. Describe how to use *one* run (where $k \to \infty$) of the power iteration (possibly with a modified input matrix) to compute the eigenvector $\boldsymbol{v}_N$ associated with the *smallest* eigenvalue of $\boldsymbol{B}$, namely λ_N, assuming it is unique. Here we mean smallest value, *not* smallest magnitude. (Do *not* assume $\boldsymbol{B}$ is invertible.)

Hint. What simple transformation of $\boldsymbol{B}$ maps its smallest eigenvalue to its largest (in magnitude)?
Remember that eigenvalues can be negative so it is possible that $|\lambda_1| < |\lambda_N|$.

(b) Describe a simple way to compute λ_N from the $\boldsymbol{v}_N$ result of part (a). The eigenvector $\boldsymbol{v}_N$ is not unique because it could be scaled by $\mathrm{e}^{\imath\phi}$. Does that "sign ambiguity" affect your calculation of λ_N?

(c) Describe how to modify the scheme to find $\boldsymbol{v}_N$ if λ_1 is unknown. (You may apply the power iteration more than once in this case, but do not invert $\boldsymbol{B}$. Use as few applications of the power iteration as possible for full credit.)

Pr. 2.
This problem examines important properties of **positive semidefinite** and **positive definite** matrices.
Recall from Ch. 3 that $\boldsymbol{B}'\boldsymbol{B} \succeq \boldsymbol{0}$ for any matrix $\boldsymbol{B}$.
For each part, you may use results from previous parts if helpful.

(a) (Optional) Show that $\boldsymbol{B}'\boldsymbol{B} \succ \boldsymbol{0}$ if $\boldsymbol{B}$ has full column rank.

(b) (Optional) Show that $\boldsymbol{A} \succ \boldsymbol{0}$ implies $\boldsymbol{A}$ is invertible.

(c) (Optional) Show that $\boldsymbol{A} \succeq \boldsymbol{0}$ and $\boldsymbol{B} \succeq \boldsymbol{0} \Rightarrow \boldsymbol{A} + \boldsymbol{B} \succeq \boldsymbol{0}$.

(d) Show that $\boldsymbol{A} \succ \boldsymbol{0}$ and $\boldsymbol{B} \succeq \boldsymbol{0} \Rightarrow \boldsymbol{A} + \boldsymbol{B} \succ \boldsymbol{0}$.

(e) Show that $\boldsymbol{A}'\boldsymbol{A} + \boldsymbol{B}'\boldsymbol{B}$ is invertible if $\boldsymbol{B}$ has full column rank.
(This property is relevant to **regularized LS** problems.)

(f) Show $\boldsymbol{A}'\boldsymbol{A} + \boldsymbol{B}'\boldsymbol{B}$ is invertible if $\mathcal{N}(\boldsymbol{A}) \cap \mathcal{N}(\boldsymbol{B}) = \{\boldsymbol{0}\}$, i.e., if $\boldsymbol{A}$ and $\boldsymbol{B}$ have disjoint null spaces other than $\boldsymbol{0}$.

(g) (Optional) Show that $\boldsymbol{A} \succ \boldsymbol{0}$, $\boldsymbol{B} \succ \boldsymbol{0} \Rightarrow \boldsymbol{B}\boldsymbol{A}\boldsymbol{B} \succ \boldsymbol{0}$.

Pr. 3.
(Projection onto orthogonal complement of null space)

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

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

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

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

Here a numerical solution using `Julia` is fine.

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

For full credit, your final version of the code should be computationally efficient in the case where the input $\boldsymbol{X}$ is a matrix with n rows and many columns, where the code must project each column of $\boldsymbol{X}$ onto $\mathcal{N}^\perp(\boldsymbol{A})$.

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

```
"""
   R = orthcompnull(A, X)

Project each column of `X` onto the orthogonal complement of the null space
of the input matrix `A`.

In:
* `A` `M × N` matrix
* `X` vector of length `N`, or matrix with `N` rows and many columns

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

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

Email your solution 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.)

Pr. 4.

In many **artificial neural network** models used in machine learning, the final layer is often "dense" or "fully connected" and often is affine or linear. This problem focuses on the linear case.

In a **supervised learning** setting, we are given training data $(\boldsymbol{x}_n, y_n), n = 1, \dots, N$ consisting of pairs of features $\boldsymbol{x}_n \in \mathbb{R}^M$ and responses $y_n \in \mathbb{R}$. A linear artificial neuron makes a prediction simply by computing the inner product of an input feature $\boldsymbol{x} \in \mathbb{R}^M$ with a (learned) weight vector $\boldsymbol{w} \in \mathbb{R}^M$, *i.e.*, $\hat{y}_n = \boldsymbol{w}'\boldsymbol{x}_n$. We want to train the weight vector $\boldsymbol{w}$ to minimize the average **loss** over the training data by solving the following optimization problem:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w}), \quad L(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n), \quad \hat{y}_n = \boldsymbol{w}'\boldsymbol{x}_n.$$

Determine analytically the optimal weight vector $\hat{\boldsymbol{w}}$ in the case where the loss function is the squared error

$$\ell(y_n, \hat{y}_n) = \frac{1}{2}(y_n - \hat{y}_n)^2.$$

You may assume that the data matrix $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_N \end{bmatrix}$ has full row rank. State any conditions needed on N for this assumption (and hence your answer) to be valid.

Hint. You should be able to set this up as a **linear least-squares** problem and express your answer in terms of the training data feature correlation matrix $\boldsymbol{K}_x = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n'$ and the cross-correlation between the training data features and responses $\boldsymbol{K}_{yx} = \frac{1}{N} \sum_{n=1}^{N} y_n \boldsymbol{x}_n'$.

Pr. 5.

(Nesterov's fast gradient descent for least squares problems)

To solve the least squares problem

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}), \qquad f(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2,$$

this problem describes a fast gradident method (FGM), also known as accelerated gradient descent, due to Nesterov that yields a sequence $\{\boldsymbol{x}_k\}$ that converges provably faster (in a worst-case sense) to the minimizer $\hat{\boldsymbol{x}}$ than the standard gradient descent method does. The method is initialized with $t_0 = 1$ and $\boldsymbol{z}_0 = \boldsymbol{x}_0$, where $\boldsymbol{x}_0$ is an initial guess for $\hat{\boldsymbol{x}}$, and consists of the following iteration for $k = 0, 1, \ldots$:

$$\boldsymbol{z}_{k+1} = \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k) \qquad \text{(usual GD update: "primary" sequence)}$$

$$t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right) \qquad \text{(magic momentum factors)}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{z}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{z}_{k+1} - \boldsymbol{z}_k) \qquad \text{(update with momentum: "secondary" sequence) .}$$

The "secondary" sequence $\{\boldsymbol{x}_k\}$ generated by FGM converges to the minimizer $\hat{\boldsymbol{x}} = \boldsymbol{A}^+\boldsymbol{y}$, when $0 < \mu \le 1/\sigma_1^2(\boldsymbol{A})$. Iterative methods like this are useful when $\boldsymbol{A}$ is too large to compute $\boldsymbol{A}^+\boldsymbol{y}$ directly, and also when there is no known solution for $\hat{\boldsymbol{x}}$.

(a) Write a function called `lsngd` that implements the above FGM for the least-squares problem. It should return the final $\boldsymbol{x}_K$, where $K \ge 1$ is the number of iterations.

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

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

Perform Nesterov's fast gradient method (accelerated gradient descent)
to solve the LS problem
``\\hat{x} = \\argmin_x ½ ‖ A x - y ‖₂²``

# In:
- `A` `M × N` matrix
- `y` 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 < μ ≤ 1 / σ₁(A)²`` to ensure convergence,
   where ``σ₁(A)`` is the spectral norm `A`.
   Ch.6 explains a practical default value for `mu`.

# Out:
`x` vector of length `N` containing the final approximate solution
"""
function lsngd(A::AbstractMatrix{<:Number}, y::AbstractVector{<:Number} ;
    x0::AbstractVector{<:Number} = zeros(eltype(y), size(A,2)),
    nIters::Int = 200, mu::Real = 0)
```

Email your solution 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 to examine the convergence rate of the sequence $\{\boldsymbol{x}_k\}$ to the ideal minimizer $\hat{\boldsymbol{x}} = \boldsymbol{A}^+\boldsymbol{y}$, by generating 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 $\boldsymbol{y}$ generated as follows. Here the problem size is small enough that you can compute $\hat{\boldsymbol{x}}$ using `A \ y` for plotting purposes.

```
using Random: seed!
M = 100; N = 50; sigma = 0.1
seed!(0); A = randn(M, N); xtrue = rand(N)
y = A * xtrue + sigma * randn(M);
```

Submit your plot to gradescope.

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

Pr. 6.

(a) Consider the **LLS** problem $\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$. 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 $\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_2^2 = (\boldsymbol{A}'\boldsymbol{A} + \beta \boldsymbol{I})^{-1}\boldsymbol{A}'\boldsymbol{y}$. Here we invert a different matrix. Express the condition number of that matrix in terms of the singular values of $\boldsymbol{A}$ and $\beta > 0$.

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

Pr. 7.

Given constants a, b, c, consider $\{(x, y, z) \in \mathbb{R}^3 : ax + by + cz = 0\}$, a plane that intersects the origin.

(a) Describe how you would use an **SVD** to find **basis vectors** for the plane.
How many bases vectors are required to express a point on the plane?

(b) Find the point on the plane that is closest to an arbitrary point $(\alpha, \beta, \gamma) \in \mathbb{R}^3$. Your expression should be general and reasonably simple and should not require any SVD computations.

Anytime a problem says "find" something, it is also implied that you must also show how *you found it.*

(c) Using your preceding answer (and probably `Julia` or a calculator), find the point on the plane $x + 2y + 3z = 0$ that is closest to the point $(4, 5, 6)$.

Pr. 8.

Hand-written digit classification using feature-based linear regression (discussion task)

This task illustrates how to do linear regression with image features for classifying handwritten digits. We focus on just the two digits "0" and "1," although the principles generalize to all digits.
Download the `task-3-classify-ls.ipynb` jupyter notebook file from Canvas under the discussion folder and follow all instructions to complete the task. You may work individually, but we recommend that you work in pairs or groups of three.
When you are finished, upload your solutions to gradescope. Note that the submission for the task is separate from the rest of the homework because the task allows you to submit as a group. Only upload one submission per group! Whoever uploads the group submission must add all group members in gradescope, using the "View or edit group" option on the right-hand sidebar after uploading a PDF and matching pages. Make sure to add all group members, because this is how they will receive credit.

Pr. 9.

Exam problem for 20 points: This problem's score will *not* be dropped!

Work with *one* partner to make a clearly stated problem that could be used on the upcoming ECE 551 midterm based on the topics covered so far, and provide your own solution to the problem. Submit your problem to **Canvas** to earn credit. Your problem must *not* be solvable simply by plugging numbers into `Julia`.
You have two choices of problem formats: CanvasQuiz (e.g., multiple-choice) or `Julia` autograder. In both cases you must also submit a correct solution to your problem to earn full credit.
For a CanvasQuiz problem, your submission to Canvas must be in the markdown format described here: `https://github.com/gpoore/text2qti`. We will use that tool to convert your submission into practice problems for the whole class to use on Canvas, so for full credit you must submit a `.txt` file that satisfies the formatting specifications on that page. You can choose to make: a multiple choice question (including true-false or multiple correct answers), or problem with a single numerical answer, as documented in the above link. (Short answer or essay questions do not earn credit here.) Multiple choice questions about `Julia` are also fine.
Alternatively, your problem can be a `Julia` problem that is suitable for use with an autograder. For such a `Julia` problem, submit a `.pdf` file to Canvas that includes a *typed* problem description, including the function specification,

and shows your `Julia` code answer to the problem. This type of problem is a bit more work to create, but may be more likely to be selected for an exam than a CanvasQuiz question, and is excellent review.
A good problem (not too trivial, not too hard) might be used on an actual exam.

To reduce formatting issues, use a code editor (like VSCode) not a text editor, because markdown is like code. Then check the syntax of your file using this web site:
`http://ec2-34-207-154-191.compute-1.amazonaws.com/`

To further improve the formatting, you may submit a **draft** of your problem to Canvas by **2PM on Tue. Oct. 07**. We will process your drafts and post them on Canvas so that you can check formatting. If needed, you can submit an updated version to Canvas before the official deadline. Proper file formatting greatly affects the score earned!

To earn full credit:

- Determine what is the *last* section from the course notes that is needed to solve your problem.
  If the last required section is 4.2, then start your problem with a corresponding code: `S4.2` .
  Use the *section* number, not the page number, and no space between the `S` and the section number.
  This coding will help students find problems from sections of interest during review.

  So for this example, the first line of your problem statement should look something like this:
  `0. S4.2 The pseudoinverse of a matrix...` .
- Your solution to the problem must be correct and must have the proper markdown format.
  Do *not* include a problem or quiz `title` or `points` . Just the question/answer(s).
  There are **examples** in the file `quizexample1.txt` in the Files/homework/quiz-examples folder on Canvas.
  You can see how those examples look in a practice Canvas quiz called "quizexample1" under "Quizzes" on Canvas.
  `https://umich.instructure.com/courses/788795/quizzes/444520`
- A CanvasQuiz question must include some brief explanation of the answers, as shown in the examples.
- Name your file like `uniqname1-uniqname2.txt` where "uniqname" is your UM uniqname. For example, Jeff and Amaya would submit the file `fessler-amurguia.txt` or `fessler-amurguia.pdf` if they were partners. Just make one submission per pair!
- Your problem should not be excessively trivial (nor beyond the scope of the course).
- Your problem must not be essentially identical to any quiz or homework or clicker or sample exam 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 would need some proving to explain.
- Have a couple classmates review your problem *before* you submit it to make sure it makes sense and has clear answers.
- Do not submit any other parts of your HW assignment to Canvas.
- Do not put your name(s) inside the problem/solution that you upload to Canvas.

Pr. 10.
Please complete the mid-semester course evaluation for ECE 551 before the deadline given by UM (most likely Oct. 19 in F25). That deadline will likely be between the due date of this HW and the due data of the next HW, so it will be a separate entry on gradescope. When you complete the evaluation, the system will give you a confirmation page. Submit a screenshot of that confirmation to gradescope to earn credit for this problem. Your evaluation is very important to us for improving the course. Your evaluation is entirely anonymous.

Non-graded problem(s) below
(Solutions will be provided for self check; do not submit to gradescope.)

Pr. 11.
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 **null spaces**) in terms of $\boldsymbol{U}_r$, $\boldsymbol{U}_0$, $\boldsymbol{V}_r$, and/or $\boldsymbol{V}_0$.

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

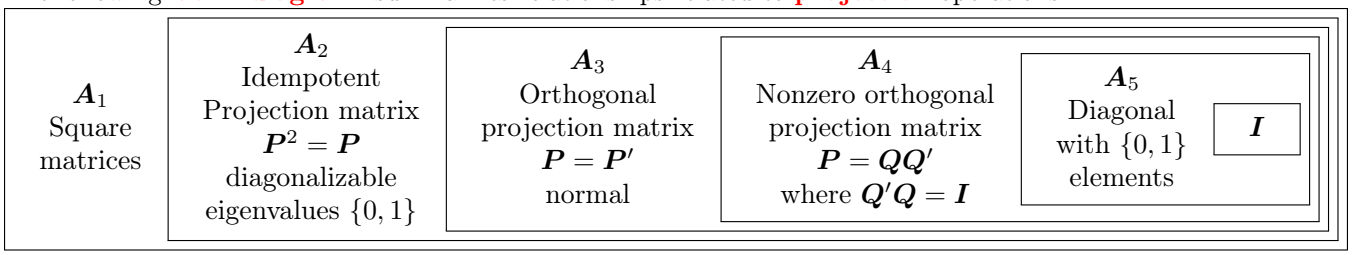
(b) $\mathcal{R}(\boldsymbol{A}')$

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

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

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

Pr. 12.
The following **Venn diagram** summarizes relationships related to **projection** operations.



Each category is a *strict superset* of the categories nested with in it.
Provide example matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_5$ that belong to the each of the categories above but *not* the next category nested within it. Try to provide the simplest possible example in each case.

Pr. 13.
Throughout the problems below you may reuse properties you derived in earlier parts as long as you refer back to the properties you use.

(a) Let $\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i'$ denote a **compact SVD** of $\boldsymbol{A}$ where r denotes the **rank** of $\boldsymbol{A}$.
Then in terms of the singular vectors $\{\boldsymbol{u}_i\}$ and $\{\boldsymbol{v}_i\}$:

(1) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})} =$
(2) $\boldsymbol{P}_{\mathcal{R}^\perp(\boldsymbol{A})} =$
(3) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A}')} =$
(4) $\boldsymbol{P}_{\mathcal{R}^\perp(\boldsymbol{A}')} =$
(5) $\boldsymbol{P}_{\mathcal{N}(\boldsymbol{A})} =$
(6) $\boldsymbol{P}_{\mathcal{N}^\perp(\boldsymbol{A})} =$
(7) $\boldsymbol{P}_{\mathcal{N}(\boldsymbol{A}')} =$
(8) $\boldsymbol{P}_{\mathcal{N}^\perp(\boldsymbol{A}')} =$

(b) Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$ denote a (full) SVD of $\boldsymbol{A}$ where r is the rank of $\boldsymbol{A}$.
Then in terms of the singular vectors $\{\boldsymbol{u}_i\}$ and $\{\boldsymbol{v}_i\}$:

(1) $\boldsymbol{P}^2_{\mathcal{R}(\boldsymbol{A})} = ?$
(2) $\boldsymbol{P}^3_{\mathcal{R}(\boldsymbol{A})} = ?$
(3) $\boldsymbol{P}^k_{\mathcal{R}(\boldsymbol{A})} =?$ for a positive integer k
(4) $(\boldsymbol{I} - \boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})})\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})} =?$
(5) $(\boldsymbol{I} - \boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})})^k =?$ for a positive integer k
(6) $(\boldsymbol{I} - \boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})})^k \boldsymbol{P}^j_{\mathcal{R}(\boldsymbol{A})} = ?$ for positive integers j and k

(7) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})} - \boldsymbol{P}'_{\mathcal{R}(\boldsymbol{A})} = ?$
(8) $\text{rank}\big(\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})}\big) = ?$
(9) $\text{rank}\big(\boldsymbol{P}_{\mathcal{R}^\perp(\boldsymbol{A})}\big) = ?$
(10) $\text{rank}\big(\boldsymbol{I} - \boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})}\big) = ?$

(c) Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$ be an SVD of $\boldsymbol{A}$ and r be the rank of $\boldsymbol{A}$.
Then in terms of the singular vectors $\{\boldsymbol{u}_i\}$ and $\{\boldsymbol{v}_i\}$:

(1) $\left\|\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})}\right\|_{\text{F}} = ?$
(2) $\left\|\boldsymbol{P}^k_{\mathcal{R}(\boldsymbol{A})}\right\|_{\text{F}} = ?$ for a positive integer k
(3) $\left\|\boldsymbol{I} - \boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})}\right\|_{\text{F}} = ?$
(4) $\left\|(\boldsymbol{I} - \boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})})^k\right\|_{\text{F}} = ?$ for a positive integer k
(5) $\left\|\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})}(\boldsymbol{I} - \boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})})\right\|_{\text{F}} = ?$
(6) $\prod_{i=1}^k (\boldsymbol{I} - \boldsymbol{P}_{\mathcal{R}(\boldsymbol{u}_i)}) - \boldsymbol{P}_{\mathcal{R}^\perp(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_k)} = ?$ for any positive integer k

(d) Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$ be an SVD of $\boldsymbol{A}$ and r be the rank of $\boldsymbol{A}$. Find expressions for both a **compact SVD** and a **full SVD** for each of the following matrices.

(1) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})} = ?$
(2) $\boldsymbol{P}^\perp_{\mathcal{R}(\boldsymbol{A})} = ?$
(3) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{u}_1)}\boldsymbol{A} = ?$
(4) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_k)}\boldsymbol{A} = ?$ for $k = 1, \ldots, r$
(5) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_k)}\boldsymbol{A} = ?$ for $r+1 \le k \le \min(m,n)$
(6) $\boldsymbol{A}\boldsymbol{P}_{\mathcal{R}(\boldsymbol{v}_1)} = ?$
(7) $\boldsymbol{A}\boldsymbol{P}_{\mathcal{R}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_k)} = ?$ for $k = 1, \ldots, r$
(8) $\boldsymbol{A}\boldsymbol{P}_{\mathcal{R}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_k)} = ?$ for $1 \le k \le \min(m,n)$ (Hint: think carefully about r.)
(9) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_k)}\boldsymbol{A}\boldsymbol{P}_{\mathcal{R}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_k)} = ?$ for $k = 1, \ldots, r$
(10) $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_j)}\boldsymbol{A}\boldsymbol{P}_{\mathcal{R}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_k)} = ?$ for $1 \le j,k \le \min(m,n)$

Pr. 14.
Let $\boldsymbol{A}$ be an $M \times N$ matrix with SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$. Let $\boldsymbol{u}_k$ denote the kth column of $\boldsymbol{U}$, and $\boldsymbol{v}_k$ denote the kth column of $\boldsymbol{V}$. Let σ_k be the kth diagonal element of $\boldsymbol{\Sigma}$, and let $\boldsymbol{A}$ have rank r. Let $\boldsymbol{I}_d$ denote the $d \times d$ identity matrix. Simplify the following **pseudoinverse** expressions as much as possible.

(a) (1) $(\boldsymbol{u}_i)^+ =$
(2) $(\boldsymbol{u}_i\boldsymbol{u}_i')^+ =$
(3) $(\boldsymbol{u}_i\boldsymbol{v}_i')^+ =$
(4) $(\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i')^+ =$

(b) (1) $\boldsymbol{U}^+ =$
(2) $\boldsymbol{V}^+ =$
(3) $\boldsymbol{U}\boldsymbol{U}^+ =$
(4) $\boldsymbol{V}\boldsymbol{V}^+ =$
(5) $\boldsymbol{\Sigma}^+ =$
(6) $(\boldsymbol{U}\boldsymbol{\Sigma})^+ =$
(7) $(\boldsymbol{\Sigma}\boldsymbol{V}')^+ =$
(8) $\boldsymbol{A}^+ = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}')^+ =$
(9) $(\boldsymbol{U}\boldsymbol{V}')^+ =$

(c) (1) For $1 \le k \le M$, $\boldsymbol{U}[:, 1:k]^+ =$
(2) For $1 \le k \le N$, $\boldsymbol{V}[:, 1:k]^+ =$
(3) For $1 \le k \le \min(M, N)$. $\boldsymbol{\Sigma}[1:k, 1:k]^+ =$
(4) For $1 \le k \le M$, $\boldsymbol{U}[:, 1:k]\boldsymbol{U}[:, 1:k]^+ =$
(5) For $1 \le k \le N$, $\boldsymbol{V}[:, 1:k]\boldsymbol{V}[:, 1:k]^+ =$
(6) For $1 \le k \le M$, $\boldsymbol{U}[:, 1:k]^+\boldsymbol{U}[:, 1:k] =$
(7) For $1 \le k \le N$, $\boldsymbol{V}[:, 1:k]^+\boldsymbol{V}[:, 1:k] =$

(d) (1) For $1 \le k \le M$, $\boldsymbol{I}_M - \boldsymbol{U}[:, 1:k]\boldsymbol{U}[:, 1:k]^+ =$
(2) For $1 \le k \le N$, $\boldsymbol{I}_N - \boldsymbol{V}[:, 1:k]\boldsymbol{V}[:, 1:k]^+ =$
(3) For $1 \le k \le M$, $\boldsymbol{I}_k - \boldsymbol{U}[:, 1:k]^+\boldsymbol{U}[:, 1:k] =$
(4) For $1 \le k \le N$, $\boldsymbol{I}_k - \boldsymbol{V}[:, 1:k]^+\boldsymbol{V}[:, 1:k] =$

(e) (1) For $1 \le k \le r$, $(\boldsymbol{U}[:, 1:k]\boldsymbol{\Sigma}[1:k, 1:k]\boldsymbol{V}[:, 1:k]')^+ =$
(2) $(\boldsymbol{A}')^+ =$
(3) $(\boldsymbol{A}^+)' =$
(4) For $\alpha \neq 0$, $(\alpha\boldsymbol{A})^+ =$
(5) $(\boldsymbol{A}\boldsymbol{A}^+)' =$
(6) $(\boldsymbol{A}^+\boldsymbol{A})' =$

(f) (1) $\boldsymbol{A}\boldsymbol{A}^+ =$
(2) $\boldsymbol{A}^+\boldsymbol{A} =$
(3) $\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} =$
(4) $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ =$
(5) $(\boldsymbol{P}_{\mathcal{R}(\boldsymbol{A})})^+ =$

(g) (1) $(\boldsymbol{A}'\boldsymbol{A})^+\boldsymbol{A}' =$
(2) $\boldsymbol{A}'(\boldsymbol{A}\boldsymbol{A}')^+ =$
(3) If $r = N$, then $\boldsymbol{A}^+\boldsymbol{A} =$
(4) If $r = M$, then $\boldsymbol{A}\boldsymbol{A}^+ =$
(5) If $r = N$, then $(\boldsymbol{A}'\boldsymbol{A})^{-1}\boldsymbol{A}' =$
(6) If $r = M$, then $\boldsymbol{A}'(\boldsymbol{A}\boldsymbol{A}')^{-1} =$