Pr. 1.

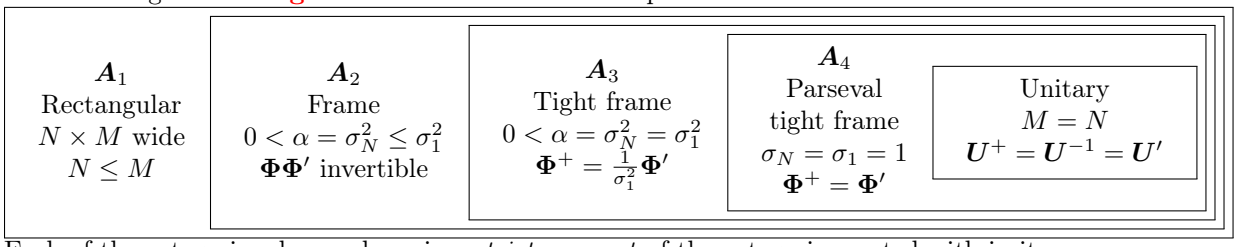
(a) Consider the **linear LS** problem $\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$. We have seen that when $\boldsymbol{A}$ has **full column rank**, the solution is $\hat{\boldsymbol{x}} = (\boldsymbol{A}'\boldsymbol{A})^{-1}\boldsymbol{A}'\boldsymbol{y}$, which 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 $\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_2^2 = (\boldsymbol{A}'\boldsymbol{A} + \beta \boldsymbol{I})^{-1}\boldsymbol{A}'\boldsymbol{y}$. 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. 2.
The following **Venn diagram** describes the relationship between **frames** and other matrices.



Each of the categories shown above is a *strict superset* of the categories nested with in it.

(a) Provide example matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_4$ that belong to the each of the categories above but *not* the next category nested within it. In each case, choose the simplest possible example from these options:
$\boldsymbol{B}_0 = [0]$, $\boldsymbol{B}_1 = [1]$, $\boldsymbol{B}_2 = [2]$, $\boldsymbol{B}_3 = [i]$, $\boldsymbol{B}_4 = [1\ 1]$, $\boldsymbol{B}_5 = [1\ 1]/\sqrt{2}$,
$\boldsymbol{B}_6 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\boldsymbol{B}_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}/\sqrt{2}$, $\boldsymbol{B}_8 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$, $\boldsymbol{B}_9 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $\boldsymbol{B}_{10} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\boldsymbol{B}_{11} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$, $\boldsymbol{B}_{12} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$.

(b) A subset of unitary matrices is certain **diagonal** matrices. Describe the necessary and sufficient conditions for a diagonal matrix to be unitary.

Pr. 3.
Given a $N \times N$ matrix $\boldsymbol{A}$ and initial vector $\boldsymbol{x}_0 \in \mathbb{F}^N$, the **power method** uses the iterative recursion

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

For this problem, assume $\boldsymbol{A}$ is a $N \times N$ Hermitian symmetric matrix with distinct (in magnitude) eigenvalues, ordered such that $\lambda_N < \lambda_{N-1} < \ldots < \lambda_2 < \lambda_1$. (Ch. 7 considers more general cases.) Under these assumptions, the sequence $\{\boldsymbol{x}_k\}$ **converges** (ignoring some sign or phase factors; see Ch. 7) to $\boldsymbol{v}_1$, the leading **eigenvector** of $\boldsymbol{A}$ associated with its largest (in magnitude) **eigenvalue**, if $\boldsymbol{v}_1'\boldsymbol{x}_0 = [\boldsymbol{V}'\boldsymbol{x}_0]_1 \neq 0$.

(a) Suppose that λ_1 is known. Describe how you can use one run of the power method (possibly with a modified input matrix) to compute the eigenvector $\boldsymbol{v}_N$ associated with the *smallest* eigenvalue of $\boldsymbol{A}$, namely λ_N, 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)?
Keep in mind 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 $e^{\imath\phi}$. Does that "sign ambiguity" affect your calculation of λ_N?

(c) Describe how you would modify the scheme to find $\boldsymbol{v}_N$ if λ_1 is unknown. (You may apply the power method more than once in this case, but you still may not invert $\boldsymbol{A}$. For full credit, your procedure should use as few applications of the power method as possible.)

Pr. 4.
In many **artificial neural networks** 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** 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.
This problem examines important properties of **positive semidefinite** and **positive definite** matrices.
Recall that the lecture notes show 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 least-squares** 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. 6.

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

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

(b) Specify conditions on the given matrices that ensure $\hat{\boldsymbol{x}}$ is unique.

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

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

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

(a) Determine a simple **orthonormal basis** for the **nullspace** 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 nullspace 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 $\boldsymbol{Y} = \mathbf{1}_3\mathbf{1}_3'$ 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 nullspace 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. 8.
(Nesterov-accelerated gradient descent for least squares problems)
To solve the least squares problem

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

this problem describes an accelerated gradient descent method 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 $k = 0, 1, \ldots$:

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

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

(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 × 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 \\leq 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)
```

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 $\hat{\boldsymbol{x}} = \boldsymbol{A}^+\boldsymbol{b}$, 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 $\boldsymbol{b}$ generated as follows. Here the problem size is small enough that you can compute $\hat{\boldsymbol{x}}$ using `A \ b` for plotting purposes.

```
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 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 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 $\|\boldsymbol{x}_k - \hat{\boldsymbol{x}}\|$ monotonically, so some wiggles in its curves are expected.

Pr. 9.
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. 10.
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 EECS 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 format for your problem: 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 any of the many CanvasQuiz questions submitted, and is excellent review.
A good problem (not too trivial, not too hard) might be used on an actual exam.

To reduce formatting issues, 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. 12**. 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 formatting will affect 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 pseudo-inverse 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 folder on Canvas.
  You can see how those examples look in a practice Canvas quiz called "quizexample1" here:
  `https://umich.instructure.com/courses/461718/quizzes/250602`
- 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, Eric and Zongyu would submit the file `echeek-zonyul.txt` or `echeek-zonyul.pdf` if they were partners.
- 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.
  You will do peer reviews of these questions (Canvas announcement about that later).

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

Pr. 11.
A matrix $\boldsymbol{T}$ has the property $\boldsymbol{T}^3 = \boldsymbol{I}$. What are the possible **eigenvalues** of $\boldsymbol{T}$?

Pr. 12.
Let $\boldsymbol{A} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $\boldsymbol{B} = \begin{bmatrix} 2 & 0 \\ 2 & 3 \\ 0 & 0 \end{bmatrix}$. Determine the cosine of the **angle between the subspaces** $\mathcal{R}(\boldsymbol{A})$ and $\mathcal{R}(\boldsymbol{B})$.

Pr. 13.
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. 14.
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. 15.
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) $\mathcal{R}(\boldsymbol{A}^+\boldsymbol{A})$
