Last (family) name:_____

First (given) name:_____

I have neither given nor received aid on this examination, nor concealed any violation of the Honor Code.
The only online resources I have used during this exam are those listed below as well as the following sites (list URL and exam problem #):

Signature: _____

ID Number: _____

EECS 551 Midterm 2, 2020-10-27 EDT (24 hour online)

- There are 6 problems for a total of 100 points.
- This part of the exam has 13 pages. Make sure your copy is complete.
- This is a 24-hour "online" exam. Many of the exam questions will be on Canvas; this pdf has additional questions that you should answer by submitting to gradescope unless instructed otherwise in each problem.
- This is an "open book" exam. During the exam, you may use all of the course materials on the Canvas EECS 551 site including the course notes on google drive, as well as wikipedia, the built-in JULIA help, and the online JULIA manual. If you use any other resources for solving the CanvasQuiz questions, then you must cite the source along with your honor code statement above. *If you use any other resources for solving the gradescope questions, then cite the source as part of your solution submitted to gradescope.* Be sure to *sign* the honor code above and scan (e.g., photograph) the top part of this page and submit to gradescope.
- You may use without rederiving any of the results presented in the course notes.
- You must complete the exam entirely on your own.
- If you need an exam question to be clarified, post a private question to the instructors on piazza.
- Clearly box your final answers. For full credit, show your complete work clearly and legibly. Answers must be submitted properly to gradescope to earn credit.
- For multiple-choice questions, select *all* correct answers.
- To "disprove" any statement, provide a concrete counter-example. For maximum credit, make that counter-example as small and simple as possible, *e.g.*, having the smallest possible matrix dimensions and using the simplest numbers like "0" and "1" as much as possible. For example, to disprove the statement "any square matrix $\boldsymbol{A}$ is invertible," the smallest and simplest counter-example is the 1×1 matrix $\boldsymbol{A} = [0]$.

autograder instructions

For any problem involving writing a JULIA function, carefully follow these instructions.

- Submit a mathematical explanation of your solution to gradescope.
- Submit a readable screenshot of your code to gradescope for grading *efficiency*.
- Test your code thoroughly *before* submitting, to maximize credit earned.
  (Passing on the 1st or 2nd submission will earn full credit; partial credit will decrease rapidly after that.)
- Submit your tested code by email attachment to `eecs551@autograder.eecs.umich.edu` as usual.
  Incorporate any necessary `using` statements.
  Name your function correctly.
  Name the file attachment like `thefunction.jl` where `thefunction` is the function name.

For all autograder problems on Midterm2, scores vs # of attempts were (8,8,5,2,1,0).

(Solutions)

Regrade policy

- For any student who submits a regrade request, we may regrade the entire exam, and your final score may increase **or decrease**.
- Submit any regrade request on gradescope, with a clear description of why you think the problem should be regraded. Saying just "please look at it again" is insufficient and will not be considered.
- Regrade requests must be within 3 days (72 hours) of when the exam scores were released on gradescope.
- After scores are returned, discussing your solutions with a professor or GSI nullifies the opportunity to submit a regrade request. Your request needs to be based on your answer at the time of the exam. (Wait to discuss until *after* requesting a regrade, if you plan to make such a request.) (Submitting reports of errors in the solution to Canvas is fine.)
- Some questions were graded by student graders who may not have been very discriminating about the precision of your justifications for your answers. If you submit a regrade request, that means you think you were unfairly given too few points on some part of the exam, so it is logical for us to see if you were unfairly given too many points on some other part of the exam! Of course we want fairness overall. Minor mistakes are inevitable when grading numerous exams, and those mistakes go in both directions. One point on any midterm is only 0.2% of your overall final score, and unlikely to affect your final grade. You should look over the solutions and your answers to all problems carefully *before* submitting a regrade request to make sure that you really want to have your whole exam reevaluated.
- For elaboration on these solutions, please come to office hours.

1. (5 points)

```
0. Determine the spectral norm of the pseudo-inverse of the matrix
   $\A = \x \y'$ where $\x = \1_M$ is the vector of $M$ ones
   and $\y$ is a unit-norm vector in $\FN$.
a) $1$
b) $1/M$
*c) $1/\sqrt{M}$
d) $\sqrt{M}$
e) $M$
f) $1/N$
g) $1/\sqrt{N}$
h) $\sqrt{N}$
i) $N$
j) $MN$
k) $1/(MN)$
l) None of these
```

.....

() [87% correct]

2. (5 points)

```
0. When $\y=\0_M$ (the vector of $M$ zeros)
   and $\A \in \FMN$ has rank $0 < r < \min(M,N)$,
   the set of minimizers over $\x$ of the LS cost function
   $\| \A \x - \y \|_2$
%... $\A^+\y + \nullspace(\A) = \0 + \range(\Vz)$ is a $N-r$ dim subspace of $\FN$
[*] is a convex set
[ ] is the single point $\0$
[ ] is a subspace of dimension $N$
[ ] is a subspace of dimension $M$
[ ] is a subspace of $\FM$
[*] is a subspace of $\FN$
[*] is a subspace of dimension $N - r$
[ ] is a subspace of dimension $M - r$
[ ] None of these
```

.....

(HW 5.2) [72% correct]

3. (5 points)

```
0. Of the lines of Julia code shown as possible answers below,
   which one most efficiently projects a vector $\y$ onto $\range(\A \A^+)$
   for a nonzero matrix $\A$?
   Assume that prior to those lines, the code is
   ```
   using LinearAlgebra
```
```

```
   U,s,V = svd(A)
   r = rank(Diagonal(s))
   Ur = U[:,1:r]
   Vr = V[:,1:r]
   ```
%... Use compact SVD: $\A = \Ur \Sr \Vr' \implies \A \A^+ = \Ur \Ur'$
%   The correct solution works even if $r=0$!
a) `U' * (U * y)`
b) `V' * (V * y)`
c) `U * (U' * y)`
d) `V * (V' * y)`
e) `V * (U' * y)`
f) `Ur' * (Ur * y)`
g) `Vr' * (Vr * y)`
*h) `Ur * (Ur' * y)`
i) `Vr * (Vr' * y)`
j) `Vr * (Ur' * y)`
k) None of these
```

.....

*(HW 6.7) [97% correct]*

---

**4**. (5 points)

```
0. Let $\U = \bmat \u_1 & \ldots & \u_9 \emat \in \fF^{11 \times 9}$
   have orthonormal columns.
   Define
   $\A = \bmat \u_1 & \u_2 & \u_3 \emat$,
   $\B = \bmat \u_4 & \u_5 & \u_6 \emat$,
   $\C = \bmat \u_7 & \u_8 & \u_9 \emat$.
   Which of the following statements are correct?
%  and let $\P_{\cS}$ denote the orthogonal projection matrix
%  for a subspace $\cS$.
%... A has linearly independent columns so its null space is $\0_9$.
%    B and C have orthogonal range spaces so
%    $\P_{\range(\A)} \P_{\range(\B)} = \A\A'\B\B' = \0_{11 \times \11}$.
%    $\range(\C) = \span(\{\u_7, \ldots, \u_9\})
%    \subseteq \null(\A') \cap \null(\B')
%    = \span(\{\u_4, \ldots, \u_{11}) \cap
%    \span(\{\u_1, \u_2, \u_3, \u_7, \ldots, \u_{11}) =
%    \span(\{\u_7, \ldots, \u_{11})$
[ ] $\P_{\range(\A)} + \P_{\range(\B)} + \P_{\range(\C)} = \I_{11}$
[*] $\P_{\range(\A)} + \P_{\range(\B)} + \P_{\range(\C)} =
    \P_{\range(\bmat \A & \B & \C \emat)}$
[*] $\P_{\null(\A)} = \0$
[*] $\P_{\range(\A)} \P_{\range(\B)} = \0$
[ ] $\null(\A') = \null(\B') \oplus \null(\C')$
[ ] $\null(\A') \cap \null(\B') \subseteq \null(\C')$
[ ] $\null(\A') \cap \null(\B') = \null(\C')$
[*] $\range(\C) \subseteq \null(\A') \cap \null(\B')$
[ ] $\range(\C) = \null(\A') \cap \null(\B')$
```

.....
```

(HW 3.9) [50% correct] correct answers selected by 80-90%

---

5. (5 points)

```
0. The nuclear norm of $\B = \bmat
   x & 0 & 0 & x \\
   0 & y & 2y & 0 \\
   0 & y & 2y & 0 \\
   3x & 0 & 0 & 3x \emat$
   for $x \geq 0$ and $y \leq 0$ is:

%...$\|\B\|_* = |x| \sqrt{20} + |y| \sqrt{10} = x \sqrt{20} - y \sqrt{10}$
a) $x \sqrt{20} + y \sqrt{10}$
b) $|x y|$
c) $14 |x y|$
d) $x + y$
e) $8x + 6y$
*f) None of these
```

.....

(HW 7.2) [84% correct] 14% chose the incorred answer that neglected that $y < 0$

---

6. (5 points)

```
0. For any $c \geq 0$,
   define $\cS_c$ to be the set of $N \times N$ orthogonal projection matrices
   having spectral radius at most $c$.
   The set $\cS_c$ is convex when
%... If $\A,\B \in \cS_c$ then by HW j5s we do have
%    $\rho(\alpha \A + (1 - \alpha) \B) \leq c$.
%    However, that convex combination is not an orthogonal projection matrix
%    in general.  Consider A = [1 0; 0 0] and B = [0 0; 0 1]
%    But for $c=0$, the set is simply the 0 matrix which is a convex set.
%    For $c=1/2$, the set is also simply the 0 matrix because eigs are all 0 or 1.
[*] $c = 0$
[ ] $c > 0$
[*] $c = 1/2$
[ ] $c = 1$
[ ] Never
```

.....

(HW 4.6, 7.5) [42% correct; 23% chose just $c = 0$] (5 points for c=0 and c=1/2; 3 points for c=0 only; 0 else)

ls/fit0

1. (9 points)
Given a data vector $\boldsymbol{y} \in \mathbb{R}^M$, we can perform polynomial regression, *i.e.*, $y_m \approx f(t_m)$, $m = 1, \ldots, M$, where f is a polynomial, by solving optimization problems of the form $\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathbb{R}^N}{\arg\min} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2$ for a suitably chosen $M \times N$ matrix $\boldsymbol{A}$.
Find a simple expression for the residual norm squared $\|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{y}\|_2^2$ for the case of fitting a 0-degree polynomial.
A simple expression is one that involves only vectors like $\boldsymbol{y}$ (no matrices).

. . . . . .

For fitting a 0-degree polynomial, $f(t) = x_1 t^0 = x_1 1$ so $\boldsymbol{A} = \boldsymbol{1}_M$ so $\hat{\boldsymbol{x}} = \boldsymbol{A}^+\boldsymbol{y} = \boldsymbol{1}^+\boldsymbol{y} = \frac{1}{M}\boldsymbol{1}'\boldsymbol{y}$ and $\left\|\frac{1}{M}\boldsymbol{1}\boldsymbol{1}'\boldsymbol{y} - \boldsymbol{y}\right\|_2^2 = \left\|\left(\frac{1}{M}\boldsymbol{1}\boldsymbol{1}' - \boldsymbol{I}\right)\boldsymbol{y}\right\|_2^2 = \left\|\boldsymbol{P}_{\boldsymbol{1}}^{\perp}\boldsymbol{y}\right\|_2^2 = \boldsymbol{y}'\boldsymbol{P}_{\boldsymbol{1}}^{\perp}\boldsymbol{y} = \|\boldsymbol{y}\|_2^2 - \frac{1}{M}(\boldsymbol{1}'\boldsymbol{y})^2$.
The solution $\left\|\frac{\boldsymbol{1}'\boldsymbol{y}}{M}\boldsymbol{1} - \boldsymbol{y}\right\|_2^2$ is acceptable because it has no matrices in it.
The expressions $\left\|\left(\frac{1}{M}\boldsymbol{1}\boldsymbol{1}' - \boldsymbol{I}\right)\boldsymbol{y}\right\|_2^2$, $\left\|\frac{1}{M}\boldsymbol{1}\boldsymbol{1}'\boldsymbol{y} - \boldsymbol{y}\right\|_2^2$ contain the outer product matrix $\boldsymbol{1}\boldsymbol{1}'$, so do not earn full credit.
(HW 4.9) [84% correct; some found $\hat{\boldsymbol{x}}$ but not $\|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{y}\|_2^2$]

code/subspace/inrange

2. (14 points)
Write a JULIA function that determines whether a given vector $\boldsymbol{z} \in \mathbb{F}^M$ is in the range of a given matrix $\boldsymbol{B} \in \mathbb{F}^{M \times N}$.
The function should return `true` or `false`, and return value `true` indicates that $\boldsymbol{z}$ is in the range of $\boldsymbol{B}$ to within numerical precision.
You may not use `pinv` or `eigen`. Hint: use `isapprox(x,y)` to see if vectors `x` and `y` are equal to within numerical precision. (Do not try to write your own numerical precision test!) If you use any other JULIA function that has optional tolerance parameter(s), just use JULIA's default setting.
A full credit solution must be as efficient as possible.
Template:

```
"""
    tf = inrange(B, z)
Return `true` or `false` depending on whether `z` is in the range of `B`
to within numerical precision.  Must be as compute efficient as possible.

In:
* `B` a `M × N` matrix
* `z` vector of length `M`
"""
function inrange(B::AbstractMatrix, z::AbstractVector)
```

See submission instructions near front of exam.

. . . . . .

Mathematically we want to see if $\boldsymbol{P}_{\mathcal{R}(\boldsymbol{B})}\boldsymbol{z} = \boldsymbol{z}$ or equivalently $\boldsymbol{U}_r(\boldsymbol{U}_r'\boldsymbol{z}) = \boldsymbol{z}$ where $\boldsymbol{U}_r$ comes from the compact SVD of $\boldsymbol{B}$. Exact equality is unlikely due to numerical precision, so we use check to see if $\boldsymbol{U}_r(\boldsymbol{U}_r'\boldsymbol{z}) \approx \boldsymbol{z}$. As seen in the documentation, JULIA's `isapprox(x,y)` checks (by default) to see if $\|\boldsymbol{x} - \boldsymbol{y}\| \leq \varepsilon \max(\|\boldsymbol{x}\|, \|\boldsymbol{y}\|)$ for a small relative tolerance ε. Here we have $\|\boldsymbol{U}_r(\boldsymbol{U}_r'\boldsymbol{z})\|_2 \leq \|\boldsymbol{z}\|_2$. (Do you know why?)
A mathematically equivalent approach is to see if $\boldsymbol{U}_0'\boldsymbol{z} = \boldsymbol{0}$. The `isapprox` documentation explains that checking to see if $\boldsymbol{x} \approx \boldsymbol{0}$ requires an absolute tolerance, so this approach requires more care to get to work.
Checking if $\mathsf{rank}(\boldsymbol{B}) = \mathsf{rank}\left(\begin{bmatrix}\boldsymbol{B} & \boldsymbol{z}\end{bmatrix}\right)$ is slower because it requires two SVD calls.
We cannot use backslash here, because $\boldsymbol{B}$ could be zero.

Some students will likely try to use the **Gram Schmidt process** they learned as undergraduates, even though we did not cover that method in 551. The usual Gram-Schmidt process is designed to start with **linearly independent** vectors and we have no guarantee of that here. Perhaps it could be possible to develop some modified version that works generally enough even to handle the case where $\boldsymbol{B}$ is zero? It is not obvious to me how to do it and it will be interesting to see if any student does. And if so, the next step will be to compare the compute effort of that procedure versus an SVD. The closely related QR decomposition for a square matrix is $O(n^3)$ so it seems unlikely to be faster than SVD. Using a (compact) SVD is the simplest and most straightforward way to solve this problem.
Solution:

```
using LinearAlgebra: svd, rank, Diagonal
"""
    tf = inrange(B, z)
Return `true` or `false` depending on whether `z` is in the range of `B`
to within numerical precision.  Must be as compute efficient as possible.

In:
* `B` a `M × N` matrix
* `z` vector of length `M`
"""
function inrange(B::AbstractMatrix, z::AbstractVector)
    (U,s,_) = svd(B)
    r = rank(Diagonal(s)) # efficient way to find rank
    Ur = U[:,1:r] # rank() considers precision
    Pz = Ur * (Ur'*z) # parens important for efficiency
    return isapprox(z, Pz) # precision again
end
```

Test:

```
include(ENV["hw551test"] * "inrange.jl")
M,N = 9,5
A = rand(M,N)
x = rand(N)
b = A * x
@assert inrange(A, b)
@assert !inrange(A, b .+ 1e-4)
inrange(ones(3,5), 7*ones(3)) # wide
```

(HW 3.2, 3.4, 3.9) [85% correct math (mean 2.7/3): 7% used `B\z` *which fails for* `B=zeros(N,N)`*;*
97 passed autograder with tries (82,10,3,2);
52% efficient (mean 2.1/3): 11% missing parens; 10% `rank(B)` *not* `rank(Diagonal(s))`*.*
A few students had nice "super efficient" versions that handled special cases like "if $r = M$ *then return* `true`*."*
A few students tried to pursue extra efficiency using conditions like "if $2r > M$ *then use* $\boldsymbol{U}_0$ *instead of* $\boldsymbol{U}_r$*."*
A drawback of that approach is that it requires the ***full SVD*** *of* $\boldsymbol{B}$*, so it is unclear if it will actually save time.*
If anyone wants to investigate that, I would be interested to know the outcome.
Calling JULIA*'s* `svd` *eventually calls* `LAPACK.gesvd!` *here that then calls compiled code in* `liblapack`*.]*

ls/pinv-y-b

3. (10 points)
For a given nonzero matrix $\boldsymbol{A} \in \mathbb{F}^{M\times N}$ and measurement $\boldsymbol{y} \in \mathbb{F}^M$, the solution to the least-squares problem $\arg\min_{\boldsymbol{x}\in\mathbb{F}^N} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2$ is not unique in general, but the *minimum norm* least-squares solution *is* unique and is $\hat{\boldsymbol{x}} = \boldsymbol{A}^+\boldsymbol{y}$. This problem asks you to think about what other vectors $\boldsymbol{b} \in \mathbb{F}^M$ would lead to the same $\hat{\boldsymbol{x}}$?

- Specifically, describe concisely the set $\mathcal{S} = \left\{\boldsymbol{b} \in \mathbb{F}^M \;:\; \boldsymbol{A}^+\boldsymbol{b} = \boldsymbol{A}^+\boldsymbol{y}\right\}$ in terms of one or more of the four fundamental spaces associated with $\boldsymbol{A}$ and possibly in terms of other problem quantities.
- Discuss whether $\mathcal{S}$ is convex. (Always? Never? Under certain necessary and sufficient conditions?)
- Discuss whether $\mathcal{S}$ is a subspace. (Always? Never? Under certain necessary and sufficient conditions?)

. . . . . .

- For SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$ with $\boldsymbol{U} = \begin{bmatrix}\boldsymbol{U}_r & \boldsymbol{U}_0\end{bmatrix}$, we have $\mathcal{S} = \boldsymbol{y} + \mathcal{N}(\boldsymbol{A}^+) = \boldsymbol{y} + \mathcal{R}(\boldsymbol{U}_0)$.
- This set is *always* convex. (It is a "flat" - a plane that does not intersect the origin in general.)
- This set is a subspace iff $\boldsymbol{y} \in \mathcal{R}(\boldsymbol{U}_0)$.
  If $\boldsymbol{y} \notin \mathcal{R}(\boldsymbol{U}_0)$, then the set does not include $\boldsymbol{0}$, so not a subspace.
  If $\boldsymbol{y} \in \mathcal{R}(\boldsymbol{U}_0)$, then $\boldsymbol{y} + \mathcal{R}(\boldsymbol{U}_0) = \mathcal{R}(\boldsymbol{U}_0)$ which is a subspace.

(F19 student, HW 5.2) [75% correct set; 82% correct about convexity (65% correct about set and convexity + 17% also correct convexity based on their incorrect set); 53% correct about subspace.]

code/ls/ls-diag1

4. (14 points)
Let $\mathcal{D}$ denote the set of $N \times N$ diagonal matrices. Complete the following JULIA function so that it returns $\hat{\boldsymbol{D}} = \arg\min_{\boldsymbol{D}\in\mathcal{D}} \|\boldsymbol{X}\boldsymbol{D}-\boldsymbol{Y}\|_{\mathrm{F}}$, when given matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ in $\mathbb{F}^{M\times N}$, where each column of $\boldsymbol{X}$ is nonzero.
Your code must return a variable of type `Diagonal` in the `LinearAlgebra` package.
The `::Diagonal` declaration for the function is a reminder of that.
To earn full credit, the code must be as efficient as possible, and should work for both real and complex inputs.
Template:

```
"""
    D = bestdiag(X::Matrix, Y::Matrix)
Solve `\\arg\\min_{D diagonal} \\| X D - Y \\|_F`
In:
* `X`, `Y` `M × N` matrices
Out:
* `D` `N × N Diagonal`
"""
function bestdiag(X::Matrix, Y::Matrix)::Diagonal
```

See submission instructions near front of exam.

. . . . . .

$\|\boldsymbol{X}\boldsymbol{D}-\boldsymbol{Y}\|_{\mathrm{F}}^2 = \left\|\begin{bmatrix}\boldsymbol{x}_1 & \dots & \boldsymbol{x}_N\end{bmatrix}\mathrm{Diag}\{d_n\} - \begin{bmatrix}\boldsymbol{y}_1 & \dots \boldsymbol{y}_N\end{bmatrix}\right\|_{\mathrm{F}}^2 = \left\|\begin{bmatrix}d_1\boldsymbol{x}_1 & \dots & d_N\boldsymbol{x}_N\end{bmatrix} - \begin{bmatrix}\boldsymbol{y}_1 & \dots \boldsymbol{y}_N\end{bmatrix}\right\|_{\mathrm{F}}^2$
$= \sum_{n=1}^N \|\boldsymbol{x}_n d_n - \boldsymbol{y}_n\|_2^2$. So the minimizing d_n is simply $\hat{d}_n = \boldsymbol{x}_n^+\boldsymbol{y}_n = (\boldsymbol{x}_n'\boldsymbol{y}_n)/(\boldsymbol{x}_n'\boldsymbol{x}_n) = (\boldsymbol{x}_n'\boldsymbol{y}_n)/\|\boldsymbol{x}_n\|_2^2$.
Solution (and several alternatives):

```
using LinearAlgebra: Diagonal, norm
"""
    D = bestdiag(X::Matrix, Y::Matrix)
Solve `\\arg\\min_{D diagonal} \\| X D - Y \\|_F`
In:
```

```
* `X`, `Y` `M × N` matrices
Out:
* `D` `N × N Diagonal`
"""
function bestdiag(X::Matrix, Y::Matrix)::Diagonal
    N = size(X,2)
    d = [(X[:,n]'*Y[:,n]) / norm(X[:,n])^2 for n=1:N] # comprehension
    return Diagonal(d)
end

# alternative solution using sum with vec:
#    d = vec(sum(conj(X).*Y, dims=1) ./ sum(abs.(X.^2), dims=1))

# alternative solution using a loop is OK,
# but it is less concise and it is trickier to initialize type:

#    T = promote_type(eltype(X), eltype(Y)) # professional version!
#    d = zeros(T, N) # ok to simply use T = eltype(X)
#    for n=1:N
#        x = X[:,n]
#        d[n] = (x'*Y[:,n]) / (x'x)
#    end

# alternative concise but inefficient (due to X'Y matrix multiply) approach:
#    d = diag(X'Y) ./ sum(abs.(X).^2, dims = 1)
```

Test:

```
using LinearAlgebra: diag
include(ENV["hw551test"] * "bestdiag.jl")
#include("ls-diag1sol.jl")
M,N = 9,5
X = randn(M,N)
Y = randn(M,N)
D = bestdiag(X, Y)
D2 = Diagonal(diag(X'Y) ./ sum(abs2.(X), dims = 1))
D2 ≈ D
```

(HW 6.6, 7.10) [50% correct and complete math; 89 passed autograder with tries (35,41,7,5,1); 48% efficient (mean 2.1/3) (16% using `pinv` *; 18% using backslash instead of analytical solution).]*

`code/ls/pseudosvd`

5. (14 points)
Write a JULIA function that, given a nonzero matrix $\boldsymbol{A} \in \mathbb{C}^{M\times N}$, returns a tuple of the three matrices in a compact SVD of $\boldsymbol{A}^+$ *without* calling the `eigen` or `pinv` functions. See code template below for specifics.
To earn full credit, your solution must be as efficient as possible.
Template:

```
"""
    (Ur,Sr,Vr) = pseudosvd(A)
Return 3 matrices in a compact SVD of the Moore-Penrose pseudo-inverse of `A`
without calling (or duplicating) Julia's built-in pseudo-inverse function.
The returned triplet of matrices should satisfy `Ur*Sr*Vr' = A^+`
to within appropriate numerical precision.
"""
function pseudosvd(A::AbstractMatrix)
```

See submission instructions near front of exam.

.....

$\boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r' \Longrightarrow \boldsymbol{A}^+ = \boldsymbol{V}_r\boldsymbol{\Sigma}_r^{-1}\boldsymbol{U}_r'$, however this is *not* a compact SVD of $\boldsymbol{A}^+$, as noted in the course notes.
To form a valid compact SVD we must permute the singular values into descending order:

$$\boldsymbol{A}^+ = \underbrace{(\boldsymbol{V}_r\boldsymbol{P}')}_{\tilde{\boldsymbol{U}}_r}\underbrace{(\boldsymbol{P}\boldsymbol{\Sigma}_r^{-1}\boldsymbol{P}')}_{\tilde{\boldsymbol{\Sigma}}_r}\underbrace{(\boldsymbol{P}\boldsymbol{U}_r')}_{\tilde{\boldsymbol{V}}_r'},$$

where $\boldsymbol{P} = \boldsymbol{P}'$ is the $r \times r$ permutation matrix that reverses the ordering.
Of course the compact SVD of $\boldsymbol{A}^+$ is not unique because one can scale matching columns of $\tilde{\boldsymbol{U}}_r$ and $\tilde{\boldsymbol{V}}_r$ by the same phase factor. Such scaling will fail the autograder and any student whose solution is correct and just differs by such sign factors should submit a regrade request through gradescope. It is unlikely there are many (if any) such cases since student code and the autograder code are calling exactly the same `svd` function.
Solution:

```
using LinearAlgebra: svd, rank, Diagonal
"""
    (Ur,Sr,Vr) = pseudosvd(A)
Return 3 matrices in a compact SVD of the Moore-Penrose pseudo-inverse of `A`
without calling (or duplicating) Julia's built-in pseudo-inverse function.
The returned triplet of matrices should satisfy `Ur*Sr*Vr' = A^+`
to within appropriate numerical precision.
"""
function pseudosvd(A::AbstractMatrix)
    (U,s,V) = svd(A)
    r = rank(Diagonal(s)) # efficiency (and numerical precision)
    Sr = Diagonal(1 ./ s[r:-1:1]) # reverse ordering!
    return (V[:,r:-1:1], Sr, U[:,r:-1:1])
end
```

Test:

```
using LinearAlgebra: pinv, I, Diagonal, diag
include(ENV["hw551test"] * "pseudosvd.jl")

function mytest(A)
    U,S,V = pseudosvd(A)
    # the following 4 tests are sufficient to confirm a valid compact SVD
    @assert U*S*V' ≈ pinv(A)
    @assert U'U ≈ I
    @assert V'V ≈ I
    @assert sort(diag(S); rev=true) == diag(S)

    U1,s1,V1 = svd(pinv(A)) # compare to SVD of pinv(A) directly
    r = rank(A)
    U1 = U1[:,1:r]; S1 = Diagonal(s1[1:r]); V1 = V1[:,1:r] # compact SVD

    for k=1:r # resolve possible "sign flips" before comparing
        phase = U1[:,k]' * U[:,k] # exp(ı ϕ) factor
        U1[:,k] *= phase
        V1[:,k] *= phase
    end
    @assert isapprox(U1, U) && isapprox(V1, V) && isapprox(S1, S)
end

for T in [Float32, ComplexF64] # test both real and complex
    # test tall and wide and zero
    # (zero not required in specification but it works too!)
    for A in [zeros(T,3,5), rand(T,5,7), rand(T,8,6)]
        mytest(A)
    end
end
```

Notice that the test code is longer than the solution code!
The subtle part here is comparing the output of the solution code to the output of `svd(pinv)`. That comparison is not essential because we know that the solution code is returning component matrices that have appropriate properties, but I included it for completeness because many students made similar tests.
Here I only checked for sign flips because the (nonzero) test matrices have unique singular values. It is possible (with more effort) to also develop a test for checking consistency in the case of repeated singular values.
(HW 3.8, 4.5, 5.15, 6.14, 7.3) [87% correct math; 98 passed autograder; tries (50,31,12,2,2,1);
48% efficient (mean 2.4/3) (20% used `rank(A)` *not* `rank(Diagonal(s))` *]*

6. (9 points)
Express $\|\boldsymbol{x}\boldsymbol{y}'\|_\infty$ in terms of appropriate norms of vectors $\boldsymbol{x}$ and of $\boldsymbol{y}$. Explain.

. . . . . .

$\|\boldsymbol{x}\boldsymbol{y}'\|_\infty = \|\boldsymbol{x}\|_\infty \|\boldsymbol{y}\|_1$ because $\|\cdot\|_\infty$ is the maximum absolute row sum of a matrix and the ith row of the rank-1 outer product is $x_i\boldsymbol{y}'$ which has absolute sum $|x_i| \|\boldsymbol{y}\|_1$.
(f19 student, HW 7.2) [92%]

Exam scores with *approximate* grades.

Gradescope part (excluding code) 104 students: min 15, max 46/46, std dev 6.1, mean 39.0, median 40
autograder part: range 0-24/24, mean 20.5, std dev 5.4
Canvas part (excluding one 0): min 6.7, max 30/30, std dev 5.3, mean 24.4, median 25,

Overall, 104 students: median=83.7, mean=86.9, std=14.2, range 35.9-100