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 Final, 2020-12-15 EST (24 hour online)

- There are 6 problems for a total of 100 points.
  This part of the exam has 16 pages. Make sure your copy is complete.
- This is a 24-hour "online" exam. Some 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.
- You can start the Canvas part any time during the 24 hour period, and you must finish it within **2 hours** of when you started. You can answer each Canvas question only once. When you first start the quiz in Canvas, you will get an alert like: "Once you have submitted an answer, you will not be able to change it later. You will not be able to view the previous question." After you select one or more options, you click the "Next" button and that will finalize your selections for that question.

  If you have a temporary internet connection loss, you should be able to reconnect and continue where you left off, during the 2 hour time window. We set this time window long enough to allow for this possibility. It should take less than an hour to finish the Canvas part, so there is a 1-hour buffer to handle any connection issues.

  To be fair to students taking the Canvas part at different times, we will *not* reply to *any* questions about the Canvas part on Piazza during the exam period, so do not waste your time posting there.

  We have vetted the Canvas questions carefully and there is nothing intentionally ambiguous there. If you think there is some ambiguity in a Canvas question, then answer based on your best interpretation about what was meant. If you think there is an ambiguity on a Canvas question/answer, then upload an explanation along with your Honor Code statement as the last entry to gradescope.
- 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 Canvas questions, then you must cite the source along with your Honor Code statement. *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.
  Obviously you must not put your code or solutions anywhere that would be accessible to any other student.

Failure to adequately keep your work private would itself be an Honor Code violation.
Do not discuss the exam with anyone until the entire exam period is over.

- Read and think about all questions before asking any questions on Piazza.
  Posting questions about how to *solve* a problem is inappropriate, and may lead to point deductions.
  Only post a (private) question if you are sure that an exam question needs to be clarified.

- 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*.
  Solutions must be as efficient as possible to earn full credit for any problem involving JULIA.
- 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.
  The very first line of your `.jl` file must be a comment line that has *your* unique name in it like this:
  `# fessler`
  That first comment line must be visible in any code that you submit to the autograder and to gradescope.

For any prohibited function, you may not use the "in place" version either. For example, if `sort` is not allowed, then the corresponding in-place version `sort!` is also not allowed.
Unless you are told otherwise, the autograder is set up only with the packages that you have used on autograded homework problems. You cannot use `Pkg` to add other packages. Unlike in Midterm3, submissions with misspelled package names, uninstalled package names, etc., will count towards your attempts. Test carefully!
The following packages are built-in and available with any Julia installation:

```
LinearAlgebra
SparseArrays
Statistics
```

The following packages are installed on the auto-grader:

```
AbstractAlgebra v0.11.1
FFTW v1.2.4
Polynomials v1.1.5
```

Submitting code that tries `using` with other packages will produce an error that may prevent you from earning full credit. If you think this list overlooks something you have used, then check on Piazza well in advance of submitting.
For all autograder problems on the Final exam, scores vs # of attempts were (9,9,5,3,1).

(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. (4 points)

```
0. For a given $N \in \nats$,
   the set of positive definite $N \times N$ matrices
[ ] is a subspace
%... No, does not include 0
[*] is a convex set
[ ] contains every invertible $N \times N$ matrix
[ ] contains at least one singular matrix
[*] contains no singular matrices
[*] is an infinite set
[ ] contains every $N \times N$ diagonal matrix
[*] contains at least one diagonal matrix
[ ] None of these
```

(HW ?) [71% correct] $>$ 90% chose each of the 4 correct answers

2. (4 points)

```
0. If $\A \in \FMN$ and $\b \in \fF^K$, then
   $\nullspace(\b \otimes \A) = \nullspace(\A)$
%... They are the same whenever $\b \neq \0$ or $\A = \0$.
[ ] always
[ ] whenever $K \leq \min(M,N)$
[ ] whenever $\A$ has rank 1
[*] whenever $\A = \0$
[*] whenever $\| \b \|_2 \neq 0$
[*] whenever $\b \neq \0$
[ ] whenever $\b = \0$
[ ] whenever $K=M$ and $\b \in \range(\A)$
[ ] whenever $K=N$ and $\b \in \range(\A')$
%[*] whenever $K=M$ and $\b \notin \range(\A)$
[ ] whenever $K=N$ and $\b \in \nullspace(\A)$
%[*] whenever $K=N$ and $\b \notin \nullspace(\A)$
[ ] never
%[ ] Insufficient information
```

(HW ?) [49% correct] ($>$ 70% chose each of the 3 correct answers

3. (4 points)

```
0. We know that any $N \times N$ idempotent matrix $\P$ is diagonalizable
   in the form $\P = \V \D \V^{-1}$
   where $\V$ is a matrix of eigenvectors of $\P$.
   For any such factorization, the matrix $\D$ is always
%... Diagonal with 0s and 1s along diagonal
[*] a projection matrix
```

```
[*] an orthogonal projection matrix
[ ] a unitary matrix
[ ] an identity matrix
[*] a matrix whose determinant is either 0 or 1
[ ] a matrix whose trace is N
[*] normal
```

.....

(HW ?) [81% correct] (> 87% chose each of the 4 correct answers

---

4. (4 points)

```
0. Define $f(\x) = \|\A \x\|_2$ where $\x \in \CN$
   and $\A \in \CMN$.
   Which of the following individual conditions on the matrix $\A$
   are sufficient *by themselves* to ensure that $f(\x)$ is a vector norm on
   the vector space $\CN$?
... It is necessary and sufficient for \A to have linearly independent columns.
[*] invertible
[*] linearly independent columns
[ ] linearly independent rows
[*] orthonormal columns
[ ] orthonormal rows
[ ] orthogonal columns
%... no, could all be zero
[ ] orthogonal rows
[*] unitary
[ ] upper triangular
[*] lower triangular with all positive diagonals
[ ] diagonal
[ ] Hermitian symmetric
[ ] tight frame
[ ] positive semidefinite
[*] positive definite
[ ] None of these
```

.....

(HW 7.13) [15% correct] 42% chose orthogonal columns, only 66% chose lin. ind. columns

---

5. (4 points)

```
0. Let $\B$ denote the Vandermonde eigenvector matrix for the companion matrix
   of a polynomial of degree $n \geq 2$
   whose roots are all real and nonnegative.
   The weighted directed graph corresponding to $\B$
%... The roots could be all 0 so the matrix is all 0 except bottom row.
%    Or if the roots are all nonzero, then no transient or absorbing states
%[ ] irreducible
[ ] always is strongly connected
[ ] always is fully connected
[ ] always contains at least one transient state
```

```
[ ] always contains at least one absorbing state
[*] None of these
```

.....

(HW ?) [82% correct]

---

6. (4 points)

```
0. Let $\P = \I_2 \otimes \G_3$ denote
   the transition matrix of a Markov chain,
   where $\G_3$ denotes the generator for $3 \times 3$ circulant matrices.
   Which of the following statements are true?
%... The graph is reducible and $\P$ has period $3$ so only (a),(b) are true.
[*] The chain has a stationary distribution.
[*] The matrix $\P$ has a left eigenvector with all positive elements.
[ ] The chain has a unique stationary distribution.
[ ] The corresponding graph is strongly connected.
[ ] The matrix $\P$ is primitive.
[ ] The matrix $\P$ is irreducible.
%[ ] The matrix $\P$ is aperiodic.
[ ] There is exactly one eigenvalue of $\P$ equal to its spectral radius.
[*] There are exactly two eigenvalues of $\P$ equal to its spectral radius.
%... Yes, one for each 3 by 3 block.
[ ] There are six eigenvalues of $\P$ equal to its spectral radius.
[ ] Applying the power iteration to $\P$ converges
    for any positive initial vector of length $6$.
```

.....

(f19 student) [27% correct] only 42% chose the left eigenvector; 19% chose six eigs = ρ

---

7. (4 points)

```
0. The low-rank matrix completion problem for $\Y \in \FMN$ involves the set
   $\cD \defequ \{\X \in \FMN \, : \, \M \odot \X = \M \odot \Y\}$.
   Assuming that the mask $\M$ is in the set $\{0,1\}^{M \times N}$,
   this set $\cD$
%... D is convex and always contains Y but is not a subspace unless Y=0
[*] is a subset of $\FMN$
[ ] is equal to $\FMN$
[*] is convex
[ ] is always a subspace
[ ] is never a subspace
[*] is nonempty
[*] contains $\Y$
[ ] has cardinality larger than 1
[ ] none of these
```

.....

(HW ?) [54% correct] > 89% chose each of the 4 correct answers; 24% said $|\mathcal{D}| > 1$

`spec/chain-r1`

1. (8 points)

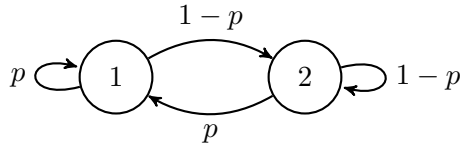
(a) Draw a Markov chain whose 2×2 transition matrix is rank-1, primitive, and asymmetric.

(b) Specify the corresponding transition matrix.

(c) Specify the equilibrium distribution of this Markov chain.

. . . . . .

A rank-1 matrix has the form $\boldsymbol{P} = \boldsymbol{x}\boldsymbol{y}'$ and here we need $\mathbf{1}'\boldsymbol{P} = \mathbf{1}'$ so $(\mathbf{1}'\boldsymbol{x})\boldsymbol{y}' = \mathbf{1}' \Longrightarrow \boldsymbol{y} \propto \mathbf{1}$. WLOG, choosing $\boldsymbol{y} = \mathbf{1}$ then we need $\mathbf{1}'\boldsymbol{x} = 1$, so $\boldsymbol{x} = (p, 1-p)$. Thus the general rank-1 form is $\boldsymbol{P} = \begin{bmatrix} p & p \\ 1-p & 1-p \end{bmatrix}$ with $0 < p < 1$ so that $\boldsymbol{P}$ is positive, and we also require $p \neq 1/2$ so that $\boldsymbol{P}$ is asymmetric.
The (unique) equilibrium distribution is $\boldsymbol{\pi}_* = (p, 1-p)$.



(f19 student)
[83% correct]

`norm/lip-nuc`

2. (8 points)
Define $f : \mathbb{F}^{M\times N} \mapsto \mathbb{R}$ by $f(\boldsymbol{X}) = \|\boldsymbol{AX}\|_*$, where $\boldsymbol{A}$ is an invertible $M \times M$ matrix. This function is Lipschitz continuous; determine a (not necessarily best) Lipschitz constant for it. (You may pick any norms you want for defining Lipschitz continuity, just specify your choice.) Explain the Lipschitz constant you determine.
Optional challenge (not for points): discuss whether it is the best (smallest) Lipschitz constant.

. . . . . .

Because the co-domain of f is $\mathbb{R}$, we always use the absolute value function $|\cdot|$ as the norm for the LHS, *i.e.*, the 1D Euclidean norm. BTW, "best" constant means for any $\boldsymbol{A}$ there is some $\boldsymbol{X}$,$\boldsymbol{Y}$ that achieves equality.

Method 1
Use the reverse triangle inequality and the fact the nuclear norm is sub-multiplicative:

$$|f(\boldsymbol{X}) - f(\boldsymbol{Y})| = |\|\boldsymbol{AX}\|_* - \|\boldsymbol{AY}\|_*| \le \|(\boldsymbol{AX}) - (\boldsymbol{AY})\|_* = \|\boldsymbol{A}(\boldsymbol{X} - \boldsymbol{Y})\|_* \le \|\boldsymbol{A}\|_* \|\boldsymbol{X} - \boldsymbol{Y}\|_*.$$

Thus if we pick the **nuclear norm** to define Lipschitz continuity, then a Lipschitz constant for f is $L = \|\boldsymbol{A}\|_*$.
This is not the best possible constant because, for example, when $\boldsymbol{A} = a\boldsymbol{I}$, we do not need the last inequality above and we have $L = |a|$.
I believe that the best Lipschitz constant (at least for this norm) is $L = \|\boldsymbol{A}\|_2$, but I am unsure how to prove it. If it is correct, it is best for any $\boldsymbol{A} = \boldsymbol{U\Sigma V}'$ because it is achieved when $\boldsymbol{Y} = \boldsymbol{0}$ and $\boldsymbol{X}$ is matrix of the form $\boldsymbol{X} = \boldsymbol{v}_1\boldsymbol{q}'$ where $\boldsymbol{q} \in \mathbb{F}^N$ has unit norm.

Method 2
Using the Ch. 5 inequality $\|\boldsymbol{B}\|_* \le \min(M, N)\|\boldsymbol{B}\|_2$ and continuing from Method 1 yields

$$\|\boldsymbol{A}(\boldsymbol{X} - \boldsymbol{Y})\|_* \le \min(M, N)\|\boldsymbol{A}(\boldsymbol{X} - \boldsymbol{Y})\|_2 \le \min(M, N)\|\boldsymbol{A}\|_2\|\boldsymbol{X} - \boldsymbol{Y}\|_2 \Longrightarrow L = \min(M, N)\|\boldsymbol{A}\|_2,$$

w.r.t. the **spectral norm**. There are other ways to derive this same bound.
It is bound is tight when $\sigma_k(\boldsymbol{A}) = \sigma_1(\boldsymbol{A})$ for all k, but not in general. Consider $\boldsymbol{A} = \boldsymbol{xy}'$ for which
$\|\boldsymbol{AX}\|_* = \|\boldsymbol{xy}'\boldsymbol{X}\|_* = \|\boldsymbol{x}\|_2 \|\boldsymbol{X}'\boldsymbol{y}\|_2 \le \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 \|\boldsymbol{X}\|_2 = \|\boldsymbol{A}\|_2 \|\boldsymbol{X}\|_2$.

Method 3
A HW problem showed that $\|\boldsymbol{B}\|_* \le \sqrt{\mathsf{rank}(\boldsymbol{B})}\|\boldsymbol{B}\|_{\mathrm{F}}$. So another bound, this time for the **Frobenius norm**, is

$$\|\boldsymbol{A}(\boldsymbol{X} - \boldsymbol{Y})\|_* \le \sqrt{\min(M, N)}\|\boldsymbol{A}(\boldsymbol{X} - \boldsymbol{Y})\|_{\mathrm{F}} \le \sqrt{\min(M, N)}\|\boldsymbol{A}\|_{\mathrm{F}}\|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}} \Longrightarrow L = \sqrt{\min(M, N)}\|\boldsymbol{A}\|_{\mathrm{F}}.$$

This is definitely not the best possible constant; consider $\boldsymbol{A} = \boldsymbol{xy}'$ for which
$\|\boldsymbol{AX}\|_* = \|\boldsymbol{xy}'\boldsymbol{X}\|_* = \|\boldsymbol{x}\|_2 \|\boldsymbol{X}'\boldsymbol{y}\|_2 \le \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 \|\boldsymbol{X}\|_2 = \|\boldsymbol{A}\|_2 \|\boldsymbol{X}\|_2 = \|\boldsymbol{A}\|_{\mathrm{F}} \|\boldsymbol{X}\|_2 \le \|\boldsymbol{A}\|_{\mathrm{F}}\|\boldsymbol{X}\|_{\mathrm{F}}$.

(HW 7.6) [69% correct, mean 5.9/8. 10% tried to use ∇f even though it is not differentiable.]

`misc/argmin1b`

3. (8 points)
Complete the following JULIA function so that it computes $\alpha_* = \arg\min_{\alpha\in\mathbb{R}} \|\boldsymbol{I} - \alpha\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}}$ when $\boldsymbol{A} \in \mathbb{F}^{M\times N}$ is a nonzero matrix. Your code should be as efficient as possible. Show your work to explain your math.

```
function alphastar(A::Matrix)

end
```

This is not an autograder problem! Just submit a screenshot to gradescope.

. . . . . .

The ratios below are all well defined because $\boldsymbol{A}$ is a nonzero matrix.
If $\boldsymbol{A}$ is zero, then the set of minimizing α values is all of $\mathbb{R}$.

Method 1
$\|\boldsymbol{I} - \alpha\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}}^2 = \mathsf{trace}\{(\boldsymbol{I} - \alpha\boldsymbol{A}'\boldsymbol{A})(\boldsymbol{I} - \alpha\boldsymbol{A}'\boldsymbol{A})\} = \mathsf{trace}\{\boldsymbol{I}\} - 2\alpha\,\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\} + \alpha^2\,\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\boldsymbol{A}'\boldsymbol{A}\}$.
Differentiating: $0 = -2\,\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\} + 2\alpha\,\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\boldsymbol{A}'\boldsymbol{A}\} \implies \alpha_* = \frac{\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\}}{\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\boldsymbol{A}'\boldsymbol{A}\}} = \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2}{\|\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}}^2}$.
A variation of this method uses $\|\boldsymbol{I} - \alpha\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}}^2 = \|\mathsf{vec}(\boldsymbol{I} - \alpha\boldsymbol{A}'\boldsymbol{A})\|_2^2$, leading to the same answer more painfully.

Method 2
Use vec trick:

$$\alpha_* = \arg\min_{\alpha\in\mathbb{R}} \left\|\mathsf{vec}\left(\boldsymbol{A}'\boldsymbol{A}\right)\alpha - \mathsf{vec}(\boldsymbol{I})\right\|_2 = \left(\mathsf{vec}\left(\boldsymbol{A}'\boldsymbol{A}\right)\right)^{+}\mathsf{vec}(\boldsymbol{I})$$
$$= \frac{(\mathsf{vec}(\boldsymbol{A}'\boldsymbol{A}))'\,\mathsf{vec}(\boldsymbol{I})}{(\mathsf{vec}(\boldsymbol{A}'\boldsymbol{A}))'(\mathsf{vec}(\boldsymbol{A}'\boldsymbol{A}))} = \frac{\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\}}{\|\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}}^2} = \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2}{\|\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}}^2}.$$

Method 3
Using unitary invariance of the Frobenius norm and the Hermitian symmetry of $\boldsymbol{A}'\boldsymbol{A} = \boldsymbol{V}\,\mathsf{Diag}\{\boldsymbol{\lambda}\}\,\boldsymbol{V}'$, where $\boldsymbol{\lambda} = (\sigma_1^2(\boldsymbol{A}), \ldots, \sigma_N^2(\boldsymbol{A}))$:

$$\|\boldsymbol{I} - \alpha\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}} = \|\boldsymbol{I} - \alpha\,\mathsf{Diag}\{\boldsymbol{\lambda}\}\|_{\mathrm{F}} = \|\boldsymbol{1} - \alpha\boldsymbol{\lambda}\|_2 \implies \alpha_* = (\boldsymbol{\lambda}'\boldsymbol{\lambda})^{-1}\boldsymbol{\lambda}'\boldsymbol{1} = \frac{\sum_k \sigma_k^2}{\sum_k \sigma_k^4} = \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2}{\|\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}}^2} = \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2}{\|\boldsymbol{A}\boldsymbol{A}'\|_{\mathrm{F}}^2}.$$

Using $\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\}$ in the numerator requires matrix multiplications not needed by $\|\boldsymbol{A}\|_{\mathrm{F}}^2$.
Likewise, using $\mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\boldsymbol{A}'\boldsymbol{A}\}$ in the denominator requires more matrix multiplications than $\|\boldsymbol{A}'\boldsymbol{A}\|_{\mathrm{F}}^2$.
So an efficient solution is:

```
using LinearAlgebra: norm
function alphastar(A::Matrix)
    return (norm(A) / (size(A,1) > size(A,2) ? norm(A'A) : norm(A*A')))^2
end
```

(HW 8.1) [80% had mathematically correct expressions for α_*, but only 19% had efficient code.
51% used `svdvals` *and 9% used* `tr`*. mean 4.8/8.]*

`code/lr/lr_ur`

4. (15 points)
The compact SVD of a matrix $\boldsymbol{Y} \in \mathbb{F}^{M\times N}$ is $\boldsymbol{Y} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r'$, where $r = \mathsf{rank}(\boldsymbol{Y})$ and $\boldsymbol{U}_r$ consists of the first r left singular vectors of an SVD of $\boldsymbol{Y}$. Complete the following JULIA function so that it returns

$$\hat{\boldsymbol{X}}_K \triangleq \underset{\boldsymbol{X}\in\mathbb{F}^{M\times N}\,:\,\mathsf{rank}(\boldsymbol{X})\le K}{\arg\min} \|\boldsymbol{Y}-\boldsymbol{X}\|_{\mathrm{F}},$$

when given $\boldsymbol{Y}$, its corresponding $\boldsymbol{U}_r$, and $K \in \{0,1,2,\ldots\}$, *without* calling any matrix decomposition functions like `svd`, `eigen`, `qr`, etc., or their relatives. Hint. Be sure to consider *all* nonnegative K values.
Template:

```
"""
    Xh = lr_ur(Y, Ur, K::Int)

Compute the rank-at-most-K best approximation of `Y = U_r Σ_r V_r'`

In:
- `Y` : `M × N` matrix
- `Ur` : `M × r` matrix with first `r` left singular vectors of `Y`
- `K` maximum rank of `Xh` (any nonnegative integer)

Out:
- `Xh` : `M × N` best approximation, having rank <= `K`
"""
function lr_ur(Y::AbstractMatrix, Ur::AbstractMatrix, K::Int)
```

See autograder submission instructions near front of exam.
Remember to put your uniquename in a comment on the first line!

.....

For $K > r$, the solution is simply $\hat{\boldsymbol{X}}_K = \boldsymbol{Y}$.
For $K = 0$, the solution is simply $\hat{\boldsymbol{X}}_K = \boldsymbol{0}$.
Otherwise, the solution is:

$$\hat{\boldsymbol{X}}_K = \boldsymbol{U}_K\boldsymbol{\Sigma}_K\boldsymbol{V}_K' = \boldsymbol{P}_{\mathcal{R}(\boldsymbol{U}_K)}\boldsymbol{Y} = \boldsymbol{U}_K(\boldsymbol{U}_K'\boldsymbol{Y}).$$

The parentheses above are *essential* for efficiency because this expression is used only when $K < r \le \min(M,N)$ so $\boldsymbol{U}_K$ is tall. The autograder could have used cases where K and N are small and M is very large and this would have caused solutions without the parentheses to time out.
An alternate approach that is less efficient is to
An alternate approach that is far less efficient is to note that $\boldsymbol{\Sigma}_r^2 = \boldsymbol{U}_r'\boldsymbol{Y}\boldsymbol{Y}'\boldsymbol{U}_r$ so $\boldsymbol{\Sigma}_r = (\boldsymbol{U}_r'\boldsymbol{Y}\boldsymbol{Y}'\boldsymbol{U}_r)^{1/2}$ and then $\boldsymbol{V}_r' = \boldsymbol{\Sigma}_r^{-1}\boldsymbol{U}_r'\boldsymbol{Y}$, and then finally $\hat{\boldsymbol{X}}_K = \boldsymbol{U}_K\boldsymbol{\Sigma}_K\boldsymbol{V}_K'$. Not only is it inefficient, it also requires careful implementation because $\boldsymbol{U}_r'\boldsymbol{Y}\boldsymbol{Y}'\boldsymbol{U}_r$ will not be exactly diagonal in practice due to numerical precision.
Solution:

```
"""
    Xh = lr_ur(Y, Ur, K::Int)

Compute the rank-at-most-K best approximation of `Y = U_r Σ_r V_r'`

In:
```

```
- `Y` : `M × N` matrix
- `Ur` : `M × r` matrix with first `r` left singular vectors of `Y`
- `K` maximum rank of `Xh` (any nonnegative integer)

Out:
- `Xh` : `M × N` best approximation, having rank <= `K`
"""
function lr_ur(Y::AbstractMatrix, Ur::AbstractMatrix, K::Int)
    K == 0 && (return zeros(eltype(Y), size(Y)))
    r = size(Ur, 2)
    K >= r && (return Y)
    return Ur[:,1:K] * (Ur[:,1:K]' * Y)
end
```

Test:

```
using LinearAlgebra: svd, rank, Diagonal
include(ENV["hw551test"] * "lr_ur.jl")
Y = randn(7,5)
U, s, V = svd(Y)
r = rank(Y)
Ur = U[:,1:r]
@assert lr_ur(Y, Ur, 0) == 0*Y # test K=0
@assert lr_ur(Y, Ur, r+1) == Y # test K>r
@assert lr_ur(Y, Ur, maximum(size(Y))+9) == Y # test big K
K = 3
@assert lr_ur(Y, Ur, K) ≈ U[:,1:K] * Diagonal(s[1:K]) * V[:,1:K]'
```

(HW 8.) [90% arrived at correct mathematical expressions for the "typical case" where $0 < K < r$.*
33% also described mathematically how to handle $K = 0$ and $K \geq r$, meaning most students did not.
16% described inefficient methods using $\boldsymbol{\Sigma}_r$.
mean 2.7/3

20% had fully efficient code with consideration of $K = 0$ and $K \geq r$.
22% used $\boldsymbol{U}_r(\boldsymbol{U}_r'\boldsymbol{Y})$ when $K \geq r$ instead of just $\boldsymbol{Y}$
21% did not consider $K = 0$. Just -0.1 points because JULIA handles it pretty efficiently.
15% used `(Uk * Uk') * Y`*. sigh.*
19% used `U_K * (U_r' Y)[:,1:K]` *entailing unnecessary multiplies that are discarded*
mean 1.9/3

98 attempted autograder and 95 passed with tries [57, 27, 9, 1, 1]
Take away: math was pretty good here but still a lot to learn about coding efficiency.]

`code/opnorm/neartf`

5. (15 points)
Consider the set of $N \times M$ tight frames: $\mathcal{T} = \left\{ \boldsymbol{T} \in \mathbb{F}^{N \times M} \ : \ \boldsymbol{T} \text{ is a tight frame} \right\}$.
Write a JULIA function that, given a nonzero $N \times M$ matrix $\boldsymbol{X}$, returns the closest (in the Frobenius norm sense) tight frame to $\boldsymbol{X}$, *i.e.*, $\hat{\boldsymbol{T}} = \underset{\boldsymbol{T} \in \mathcal{T}}{\arg\min} \left\| \boldsymbol{T} - \boldsymbol{X} \right\|_{\mathrm{F}}$. Assume $N \leq M$.
Optional (not graded): think about why the problem statement assumes $\boldsymbol{X}$ is nonzero.
Template:

```
"""
    T = neartf(X)

Find nearest (in Frobenius norm sense) tight frame to matrix `X`.

In:
* `X` : `N × M` nonzero matrix with `N ≤ M`

Out:
* `T` `N × M` matrix that is the nearest tight frame to `X`
"""
function neartf(X::AbstractMatrix)
```

See autograder submission instructions near front of exam.
Remember to put your uniquename in a comment on the first line!

.....

Another way of writing $\mathcal{T}$ is $\mathcal{T} = \left\{ \boldsymbol{T} \in \mathbb{F}^{N \times M} \ : \ \boldsymbol{T}\boldsymbol{T}' = c\boldsymbol{I}_N, \ c \geq 0 \right\}$.
This problem is closely related to the (generalized) orthogonal Procrustes problem (with scaling) considered in a HW problem except for a transpose.
Let $\mathcal{Q} = \left\{ \boldsymbol{Q} \in \mathbb{F}^{M \times N} \ : \ \boldsymbol{Q}'\boldsymbol{Q} = \boldsymbol{I}_N \right\}$ so

$$\hat{\boldsymbol{T}} = (\hat{\alpha}\hat{\boldsymbol{Q}})', \quad \hat{\boldsymbol{Q}}, \hat{\alpha} = \underset{\boldsymbol{Q} \in \mathcal{Q}, \ \alpha \in \mathbb{F}}{\arg\min} \left\| (\alpha \boldsymbol{Q})' - \boldsymbol{X} \right\|_{\mathrm{F}} \Longrightarrow \hat{\boldsymbol{Q}} = (\boldsymbol{U}\boldsymbol{V}_N')'$$

$$\hat{\alpha} = \frac{\mathsf{vec}(\boldsymbol{Q})' \, \mathsf{vec}(\boldsymbol{X}')}{\|\mathsf{vec}(\boldsymbol{Q})\|_2^2} = \frac{\mathsf{trace}\{\boldsymbol{Q}\boldsymbol{X}'\}}{\|\boldsymbol{Q}\|_{\mathrm{F}}^2} = \frac{\mathsf{trace}\{(\boldsymbol{V}_N\boldsymbol{U}')(\boldsymbol{U}\boldsymbol{\Sigma}_N\boldsymbol{V}_N')\}}{N} = \frac{1}{N}\mathsf{trace}\{\boldsymbol{\Sigma}_N\},$$

where the economy SVD is: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}_N\boldsymbol{V}_N'$. So $\hat{\boldsymbol{T}} = \hat{\alpha}\boldsymbol{U}\boldsymbol{V}_N' = \frac{1}{N}\mathsf{trace}\{\boldsymbol{\Sigma}_N\}\boldsymbol{U}\boldsymbol{V}_N'$.
If $\boldsymbol{X}$ were zero, then $\boldsymbol{\Sigma}_N$ would be zero and we would end up with $\hat{\boldsymbol{T}}$ as zero, which is not a tight frame.
Solution:

```
using LinearAlgebra: svd

"""
    T = neartf(X)

Find nearest (in Frobenius norm sense) tight frame to matrix `X`.

In:
* `X` : `N × M` nonzero matrix with `N ≤ M`
```

```
Out:
* `T` `N × M` matrix that is the nearest tight frame to `X`
"""
function neartf(X::AbstractMatrix)
    U, s, V = svd(X) # economy SVD has V = V_N here
    return ((sum(s)/length(s)) * U) * V' # efficiency: scale U not U*V'
end
```

Test:

```
using LinearAlgebra: svd, opnorm, I
using Random: seed!

include(ENV["hw551test"] * "neartf.jl")

seed!(0)
for T in [Float32, ComplexF64] # test both real and complex
    for X in [zeros(T,5,5), rand(T,5,7)] # square and wide
        neartf(X)
    end

    N,M = 7,9
    Q1 = 5 * svd(randn(N,M)).U
    @assert Q1 ≈ neartf(Q1)

    Q2 = svd(randn(N,M)).U
    X = [3*Q1 2*Q2]
    @assert X ≈ neartf(X)

    X = ones(T, 4, 8) # rank 1
    Y = neartf(X)
    @assert Y * Y' ≈ opnorm(Y)^2 * I
end
```

(HW 6.2 7.8)
45% correct math for $\hat{\boldsymbol{T}}$ with economy SVD (mean 2.4/3)
22% used full SVD $\boldsymbol{UV}'$ which is mathematically incorrect but the idea still works in code because `svd` *returns economy SVD not full SVD.*
This same issue was mentioned in midterm3 solutions, so 1/5 of class did not study or learn from those solutions.

78% had efficient code using economy SVD (mean 2.7/3)
but 51% did not include parentheses, relying implicitly on compiler to do left-right parsing.

97 attempted autograder and 91 passed with tries [69, 15, 5, 1, 1]

`code/ls/ux_xv`

6. (18 points)

Let $\boldsymbol{X}$, $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{Y}$ all denote matrices in $\mathbb{F}^{N\times N}$, and define the following optimization problem

$$\hat{\boldsymbol{X}} = \underset{\boldsymbol{X}\in\mathbb{F}^{N\times N}}{\arg\min} \|\boldsymbol{U}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{V} - \boldsymbol{Y}\|_{\mathrm{F}}.$$

(a) Derive an analytical solution for $\hat{\boldsymbol{X}}$ in terms of the other matrices given in problem. Explain.
Hint: trying to differentiate would be unwise; we never used that strategy for problems with this norm in 551.

(b) Specify and explain necessary and sufficient conditions on the problem matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{Y}$ for ensuring that the above optimization problem has a unique minimizer.

(c) Complete the following JULIA function so that it returns $\hat{\boldsymbol{X}}$ given $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{Y}$. You may assume that the input matrices are square and have the same size and satisfy appropriate sufficient conditions for uniqueness of $\hat{\boldsymbol{X}}$. The target matrix size is small enough that you can use your analytical solution; do not use any iterative method.

To earn full credit, the code must be as efficient as possible, and should work for both real and complex inputs. Passing the autograder with inefficient code will not earn full credit.

Template:

```
"""
    Xh = ux_xv(U, V, Y)
Compute `\\hat{X} = \\argmin_X \\| UX + XV - Y \\|_F`
assuming `U` and `V` are such that the minimizer is unique.

In:
* `U, V, Y` : `N × N` matrices

Out:
* `Xh` `N × N` matrix; minimizer of the optimization problem
"""
function ux_xv(U::AbstractMatrix, V::AbstractMatrix, Y::AbstractMatrix)
```

See autograder submission instructions near front of exam.
Remember to put your uniquename in a comment on the first line!

. . . . . .

As discussed in class for Ch. 5, in general for minimization problems involving Frobenius norms, the two strategies we have are to consider $\|\boldsymbol{A}\|_{\mathrm{F}}^2 = \mathsf{trace}\{\boldsymbol{A}'\boldsymbol{A}\}$ or $\|\boldsymbol{A}\|_{\mathrm{F}} = \|\mathsf{vec}(\boldsymbol{A})\|_2$. The latter equality is more helpful here.

- Use the **vec trick** from Ch. 1 (and linearity) to rewrite the problem in terms of $\boldsymbol{x} = \mathsf{vec}(\boldsymbol{X})$, $\boldsymbol{y} = \mathsf{vec}(\boldsymbol{Y})$ and apply the Kronecker sum from Ch. 7 as follows:

  $$\mathsf{vec}(\boldsymbol{U}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{V} - \boldsymbol{Y}) = \mathsf{vec}(\boldsymbol{U}\boldsymbol{X}\boldsymbol{I}) + \mathsf{vec}(\boldsymbol{I}\boldsymbol{X}\boldsymbol{V}) - \boldsymbol{y} = (\boldsymbol{I}\otimes\boldsymbol{U})\boldsymbol{x} + (\boldsymbol{V}^T\otimes\boldsymbol{I})\boldsymbol{x} - \boldsymbol{y} = (\boldsymbol{V}^T\oplus\boldsymbol{U})\boldsymbol{x} - \boldsymbol{y}.$$

  So an equivalent problem statement is

  $$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{F}^{MN}}{\arg\min}\, f(\boldsymbol{x}), \quad f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2, \quad \boldsymbol{A} = \boldsymbol{V}^T\oplus\boldsymbol{U}.$$

  From Ch. 4, the general set of solutions is $\hat{\boldsymbol{X}} = \boldsymbol{A}^{+}\boldsymbol{y} + \mathcal{N}(\boldsymbol{A})$.
  If $\boldsymbol{A}$ has full column rank, the unique solution is is $\hat{\boldsymbol{x}} = \mathsf{vec}\left(\hat{\boldsymbol{X}}\right) = (\boldsymbol{A}'\boldsymbol{A})^{-1}\boldsymbol{A}'\boldsymbol{y}$.

- Here, $\boldsymbol{A}$ is square, so having full column rank is equivalent to being invertible.
  Because of the **Kronecker sum** form, from Ch. 7 $\boldsymbol{A}$ is invertible iff $\boldsymbol{U}$ and $-\boldsymbol{V}$ do not share any eigenvalues.
- From Ch. 4, the most efficient LS solver is to use backslash.
  Solution:

```
using LinearAlgebra: I

"""
    Xh = ux_xv(U, V, Y)
Compute `\\hat{X} = \\argmin_X \\| UX + XV - Y \\|_F`
assuming `U` and `V` are such that the minimizer is unique.

In:
* `U, V, Y` : `N × N` matrices

Out:
* `Xh` `N × N` matrix; minimizer of the optimization problem
"""
function ux_xv(U::AbstractMatrix, V::AbstractMatrix, Y::AbstractMatrix)
    N = size(Y,1)
    A = kron(I(N), U) + kron(transpose(V), I(N))
    return reshape(A \ vec(Y), N, N)
end
```

The code `kron(I(N),U)` requires $O(N^3)$ multiplies, which more efficient than `kron(Matrix(I,N,N),U)` that requires $O(N^4)$ multiplies, because `I(N)` is diagonal whereas `Matrix(I,N,N)` is full.
Test:

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

T = ComplexF64
N = 5
U = randn(N,N)
V = randn(N,N)
Y = randn(N,N)
Xh = ux_xv(U, V, Y)

A = kron(I(N), U) + kron(transpose(V), I(N))
Xp = reshape(pinv(A) * vec(Y), N, N)

@assert Xh ≈ Xp

U = I(N); V = I(N)
Xh = ux_xv(U, V, Y)
@assert Xh ≈ Y/2
```

BTW, this problem is closely related to the **Sylvester equation** in control theory.

(HW 6.6) [60% had correct or nearly correct answers (mean 2.3/4)
Most common small error was writing $\boldsymbol{U} \oplus \boldsymbol{V}'$ (5%)

23% correctly identified the eigenvalue conditions on $\boldsymbol{U}$ and $-\boldsymbol{V}$ (mean 0.8/3)
30% just said $\boldsymbol{A}$ must be full rank without digging further

27% had perfectly efficient code (mean 1.4/4)
24% had nearly efficient code except for using `Matrix(I)` *. Need to explain* `I(N)` *better apparently.*

78 attemped and 52 passed autograder with tries [30, 14, 3, 4, 1]]

Exam scores

Gradescope part (excluding code) 99 students (2 did not submit) with 2 extra points for submitting Honor Code, so max possible was 47: min 6.5, max 46.4/47, std dev 9.0, mean 34.2, median 34.8
autograder part: range 0-27/27, median 19, mean 20.3, std dev 6.8
Canvas part: min 2, max 28/28, mean 20.7, std dev 1.55

Overall, 99 students: mean=75.1/102, median=74.4, std=18.7, range: 9-100.4