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 3, 2020-11-19 EST (24 hour online)

- There are 6 problems for a total of 100 points.
- This part of the exam has 14 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 $\boxed{\text{box}}$ 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.

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.

Submissions with prohibited functions (like `@show`) or uninstalled packages (like `SpecialMatrices`) are logged by the autograder, but are not counted towards your number of attempts for earning credit (because you do not get any other feedback about your submission).

For all autograder problems on Midterm3, 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.

Last (family) name:____________________

First (given) name:____________________

---

1. (5 points)

```
0. Let $\A$ denote a square $N \times N$ matrix with eigenvalues
   $\lambda_1, \ldots, \lambda_N$.
   A nonzero vector $\x \in \FN$ is an eigenvector of $\A$
   iff, for some $i \in \{1,\ldots,N\}$, we have
%... $\A\x = \lambda\x$ iff $\x \in \nullspace(\A - \lambda \I)$
%    iff $\x \in \range^{\perp}(\A' - \lambda_i^* \I)$
[*] $\A \x = \lambda_i \x$
[ ] $\A' \x = \lambda_i \x$
[ ] $\A' \x = \lambda_i^* \x$
[ ] $\x \in \range^{\perp}(\A' - \lambda_i \I)$
[*] $\x \in \range^{\perp}(\A' - \lambda_i^* \I)$
[ ] $\x \in \range^{\perp}(\A - \lambda_i \I)$
[ ] $\x \in \range^{\perp}(\A - \lambda_i^* \I)$
[ ] None of these
```

....

(HW 2.1, 2.12, 6.12) [61% correct] 24% chose $\mathcal{R}^{\perp}(\boldsymbol{A} - \lambda_i \boldsymbol{I})$

---

2. (5 points)

```
0. Consider the SVD $\A = \U \Sig \V' = U_r \Sigma_r V_r'{}$
   for a nonzero matrix $\A \in \FMN$ with rank $r \ll \min(M,N)$.
   Which of the following is/are equal to $\P^{\perp}_{\range(\A^+)}$
%... Use compact SVD: $\A^+ = \Vr \Sr^{\inv} \Ur' \implies
%    $\P^{\perp}_{\range(\A^+)} = \Vz \Vz' = \I - \Vr \Vr'$
[ ] $\U \U'$
[ ] $\V \V'$
[ ] $\U \V'$
[ ] $\V \U'$
[ ] $\Vr \Vr'$
[ ] $\Ur' \Ur'$
[ ] $\Vr \Ur'$
[ ] $\Ur \Vr'$
[*] $\I - \Vr \Vr'$
[ ] $\I - \Ur \Ur'$
[ ] None of these
```

....

(HW 6.7) [91% correct]

---

3. (5 points)

```
0. Let $\A$ denote a nonzero matrix having
   left singular vectors $\{u_i\}$.
   Define $\cS = \span(u_1)$.
   The cosine of the angle between $\null(\A)$
   and $\span(\A' \P_{\cS})$ is
```

```
%... $\A' \P_{\cS}} = \sigma_1 \v_1$ which is orthogonal to $\null(\A)$
*a) $0$
b) $1$
c) $-1$
d) $\pi/2$
e) None of these
f) Insufficient information
g) $1/\sqrt{2}$
h) $-1/\sqrt{2}$
```

(HW 5.15, 6.11) [79% correct]

4. (5 points)

```
0. Let $M \times N$ matrix $\A$
   have left singular vectors $\{u_i\}$,
   right singular vectors $\{v_i\}$,
   and distinct nonzero singular values $7,6,4,2,1$.
   Define $\X = P_{\span(u_5)}^{\perp}$
   and $\Y = P_{\span(v_3)}^{\perp}$
   and $\B = \X \A \Y$.
   The nuclear norm of $\B$ is:
%... $\B = \X \A \Y = \X \U \Sig \V' \Y = \X \U \Sig \V' \Y' =$
%... $(\X \U) \Sig (\Y\V)' =$
%... $7 \u_1 \v_1' + 6 \u_2 \v_2' + 0\u_3\v_3' + 2 \u_4 \v_4' + 0\u_5\v_5'$
%... so $\| \B_* \| = 7 + 6 + 2 = 15$
=  15
```

(HW 5.15, 7.2) [91% correct]

5. (5 points)

```
0. Every $M \times N$ matrix $\X$ with rank $0 < r < \min(M,N)$
   has a generalized inverse matrix $\G$ whose spectral norm is 3.
%... False.  Just take $X = [1/4 0; 0 0]$
a) True
*b) False
```

(HW 7.4) [93% correct]

6. (5 points)

```
0. Every circulant matrix is
%... Transpose is also circulant so it commutes with its transpose, so normal.
[*] a normal matrix
[*] diagonalizable
```

```
[ ] Hermitian symmetric
[ ] positive definite
[ ] positive semidefinite
[ ] unitary
[ ] invertible
[ ] None of these
```

....

(HW 9.4) [85% correct] 10% chose invertible

ls/pinv6norm

1. (7 points)
Determine $\|\boldsymbol{A}\|_2$ when $\boldsymbol{A}^+ = \boldsymbol{A}'$ and $\mathsf{rank}(\boldsymbol{A}) \geq 1$. (Use SVD components of $\boldsymbol{A}$ as needed and explain.)

. . . .

Using the compact SVD: $\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r'$ for which $\boldsymbol{A}' = \boldsymbol{V}_r \boldsymbol{\Sigma}_r \boldsymbol{U}_r'$ and $\boldsymbol{A}^+ = \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{-1} \boldsymbol{U}_r'$. When $\boldsymbol{A}^+ = \boldsymbol{A}'$ it follows (by multiplying on the left by $\boldsymbol{V}_r'$ and on the right by $\boldsymbol{U}_r$) that $\boldsymbol{\Sigma}_r = \boldsymbol{\Sigma}_r^{-1}$, so it must be the case that $\boldsymbol{\Sigma}_r = \boldsymbol{I}_r$. Thus $\|\boldsymbol{A}\|_2 = 1$.
(If $r = 0$ then $\|\boldsymbol{A}\|_2 = 0$.)
(f19 student) [97% correct]

code/ls/eigtiki

2. (15 points)
Complete the following JULIA function so that it returns all the eigenvalues of the matrix $(\boldsymbol{X}'\boldsymbol{X} + \beta \boldsymbol{I})^+$ given any matrix $\boldsymbol{X} \in \mathbb{F}^{M \times N}$ and any $\beta \geq 0$. Your code must return a vector of the eigenvalues in *ascending* order.
To earn full credit, the code must be as efficient as possible, and should work for both real and complex inputs. You may not use (or replicate) any of the functions `eigen`, `eigvals`, `eigvecs`, `svd`, `inv`, `pinv`, “\”, `sort`, `sortperm`, or `sortslices`. You may use any function *not* listed here. Hint: your code may need an `if` statement or equivalent.
Template:

```julia
"""
    e = eigtiki(X, beta)
Return the eigenvalues in ascending order of `(X'X + beta I)^+` for `β ≥ 0`,
without calling (or duplicating) functions listed in the problem statement.
The output should be a Vector.
"""
function eigtiki(X::AbstractMatrix, beta::Real)::Vector
```

See submission instructions near front of exam.

. . . .

$$(\boldsymbol{X}'\boldsymbol{X} + \beta \boldsymbol{I}_N)^+ = (\boldsymbol{V}(\boldsymbol{\Sigma}'\boldsymbol{\Sigma} + \beta \boldsymbol{I}_N)\boldsymbol{V}')^+ = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}'$$

$$\boldsymbol{\Lambda} = (\boldsymbol{\Sigma}'\boldsymbol{\Sigma} + \beta \boldsymbol{I}_N)^+ = \begin{bmatrix} \boldsymbol{\Sigma}_r^2 + \beta \boldsymbol{I}_r & \boldsymbol{0} \\ \boldsymbol{0} & \beta \boldsymbol{I}_{N-r} \end{bmatrix}^+ = \begin{bmatrix} (\boldsymbol{\Sigma}_r^2 + \beta \boldsymbol{I}_r)^+ & \boldsymbol{0} \\ \boldsymbol{0} & \beta^+ \boldsymbol{I}_{N-r} \end{bmatrix}.$$

If $\beta > 0$, then the eigenvalues (diagonal elements of $\boldsymbol{\Lambda}$) are simply $1/(\sigma_k^2 + \beta)$ for $k = 1, \ldots, \min(M, N)$ and then $1/\beta$ for $k = \min(M, N) + 1, \ldots, N$. In ascending sorted order:

$$\Big(\frac{1}{\sigma_1^2 + \beta}, \ldots, \frac{1}{\sigma_{\min(M,N)}^2 + \beta}, \underbrace{\frac{1}{\beta}, \ldots, \frac{1}{\beta}}_{N - \min(M,N)}\Big) = \Big(\frac{1}{\sigma_1^2 + \beta}, \ldots, \frac{1}{\sigma_r^2 + \beta}, \underbrace{\frac{1}{\beta}, \ldots, \frac{1}{\beta}}_{N-r}\Big).$$

If $\beta = 0$, then the eigenvalues (diagonal elements of $\boldsymbol{\Lambda}$) are simply $1/\sigma_k^2$ for $k = 1, \ldots, r = \mathsf{rank}(X)$ and then 0 for $k = \mathsf{rank}(K) + 1, \ldots, N$. In ascending sorted order:

$$(\underbrace{0, \ldots, 0}_{N-r}, \frac{1}{\sigma_1^2}, \cdots \frac{1}{\sigma_r^2}).$$

Solution:

```julia
using LinearAlgebra: rank, svdvals, Diagonal
"""
    e = eigtiki(X, beta)
Return the eigenvalues in ascending order of `(X'X + beta I)^+` for `β ≥ 0`,
without calling (or duplicating) functions listed in the problem statement.
The output should be a Vector.
"""
function eigtiki(X::AbstractMatrix, beta::Real)::Vector
    M,N = size(X)
    s = svdvals(X)
    e = 1 ./ (s.^2 .+ beta)
    if beta > 0
        return [e; ones(N - min(M,N)) ./ beta]
    else
        r = rank(Diagonal(s)) # needed only for β=0
        return [zeros(N - r); e[1:r]]
    end
end
```

Some students included β when estimating the rank, which could be preferable to the above solution when β is positive but tiny. We designed the grader to only test with $\beta = 0$ and $\beta > 0.1$ to avoid any issues with tiny β values.

Alternate Approach:
The solution above uses `svdvals(X)`. Because $\sigma_i^2(\boldsymbol{X}) = \sigma_i(\boldsymbol{X}'\boldsymbol{X})$ for $i = 1, \ldots, \min(M, N)$, an alternative approach is to use a carefully chosen combination of `svdvals(X'X)` with `svdvals(X*X')`, depending on which is smaller, as shown in the code below. Mathematically that approach is correct (if the other details are implemented properly), but it has a drawback it requires an extra matrix-matrix multiplication *and* it ends up being less numerically accurate because all of the singular values get squared. That is one reason why some students had solutions that did not pass the autograder tolerance initially.
As noted in this JULIA forum, the most efficient approach when $\boldsymbol{X}$ is a tall matrix is to do a QR decomposition of $\boldsymbol{X}$ (not covered in of EECS 551). Surprisingly, that efficient way is not built into JULIA's `svdvals` command. Solutions that use a combination of `svdvals(X'X)` and `svdvals(X*X')` properly (in the first 2 tries) will earn full credit, but will lose a point for efficiency (due to the extra matrix multiplication) *unless* the student submits work to gradescope showing that they timed both ways and found empirically (like we did) that `svdvals(X'X)` is faster in the current JULIA version. (Future JULIA versions are likely to overcome this inefficiency.)
Solutions that use only one of `svdvals(X'X)` or `svdvals(X*X')` lose 2/3 efficiency points.

```julia
using LinearAlgebra: rank, svdvals, Diagonal

function eigtiki_alt1(X::AbstractMatrix, beta::Real)::Vector
    M,N = size(X)
    s2 = M > N ? svdvals(X'X) : svdvals(X*X') # squares the condition number
    e = 1 ./ (s2 .+ beta)
    if beta > 0
        return [e; ones(N - min(M,N)) ./ beta]
    else
        r = rank(Diagonal(s2)) # needed only for β=0
        return [zeros(N - r); e[1:r]]
    end
end
```

Yet another approach is to use the **power iteration** to find each eigenvalue and corresponding eigenvector of $\boldsymbol{X}'\boldsymbol{X} + \beta\boldsymbol{I}$ one by one. This approach is highly inefficient. It could work *if* the eigenvalues have distinct magnitude, but not in general. (Unintentionally, the autograder used random $\boldsymbol{X}$ having distinct singular values.)
Test:

```
using LinearAlgebra: eigvals, pinv, I
include(ENV["hw551test"] * "eigtiki.jl")
include("eigtiki-alt1.jl")

T = ComplexF64
Xs = [randn(T,4,3), randn(T,2,3), randn(T,3,3), # tall wide square
    rand(T,4)*rand(T,3)', rand(T,2)*rand(T,3)', rand(T,3)*rand(T,3)'] # non-full rank
for X in Xs
    for β in [0, 5] # zero and nonzero
        ev = eigvals(pinv(X'X + β*I))
        @assert ev ≈ real(ev)
        e0 = sort(real(ev))
        e1 = eigtiki(X, β)
        @assert e0 ≈ e1
        e2 = eigtiki_alt1(X, β)
@show size(X), e0, e2
        @assert e0 ≈ e2
    end
end
```

(HW 6.1, 5.16)
43% correct and complete math explanations that properly addressed cases where $\beta = 0$ *and* $\sigma_k = 0$. *(mean 2.3/3)*
8% efficient code with proper consideration of numerical precision (mean 1.1/3).
19% used `svdvals(X'X)` *even for wide* $\boldsymbol{X}$.
21% used `isapprox` *with arbitrary absolute tolerance (8% with no tolerance).*
15% used $\sigma_k \neq 0$*, disregarding HW 3#7. Must discuss!*

Initially (88 students attempted):
69 passed autograder with tries (23, 20, 12, 6, 2, 4, 1@10, 1@37)
Ignoring submissions with illegal words in them:
69 passed autograder with tries (23, 21, 12, 5, 3, 3, 1@10, 1@31)
Setting tolerance to 2e-6 for all submissions:
69 passed autograder with tries (29, 24, 6, 3, 3, 3, 1@10) (power iter...)
Setting tolerance to 1e-2:
71 passed autograder with tries (36, 23, 5, 4, 2, 1)

`lr/st-vs-ht`

3. (9 points)
Let $\boldsymbol{Y}$ be a 6×7 matrix with singular values $0, 2, 3, 8, 9, 13$ and define

$$\hat{\boldsymbol{X}} = \underset{\boldsymbol{X} \in \mathbb{R}^{6\times 7}}{\arg\min} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2 + 8\|\boldsymbol{X}\|_*$$
$$\hat{\boldsymbol{Z}} = \underset{\boldsymbol{Z} \in \mathbb{R}^{6\times 7}}{\arg\min} \frac{1}{2}\|\boldsymbol{Z} - \boldsymbol{Y}\|_{\mathrm{F}}^2 + 8\,\mathsf{rank}(\boldsymbol{Z}).$$

Determine the ratio $\|\hat{\boldsymbol{X}} - \boldsymbol{Y}\|_* / \|\hat{\boldsymbol{Z}} - \boldsymbol{Y}\|_*$. Explain.

. . . .

$\hat{\boldsymbol{X}}$ is SVST with singular values $[\sigma - \beta]_+$ so $0, 0, 0, 8-8, 9-8, 13-8$ and $\|\hat{\boldsymbol{X}} - \boldsymbol{Y}\|_* = 2+3+8+8+8 = 29$.
$\hat{\boldsymbol{Z}}$ is SVHT with singular values $\sigma\mathbb{I}_{\{\sigma > \sqrt{2\beta}\}} = \sigma\mathbb{I}_{\{\sigma>4\}}$ so $0, 0, 0, 8, 9, 13$ and $\|\hat{\boldsymbol{Z}} - \boldsymbol{Y}\|_* = 2+3 = 5$.
So the ratio is $29/5$.
(f18 student) [86% correct (mean 8.1/9)]

`code/lr/lr_circ`

4. (15 points)
Let $\boldsymbol{A} \in \mathbb{F}^{N\times N}$ denote a circulant matrix and define

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

for $K \in \{0, 1, 2, \ldots\}$. Complete the following function so that it returns $\hat{\boldsymbol{A}}_K$ given $\boldsymbol{A}$ and K, *without* calling or duplicating `svd` or `eigen` or their relatives.
Template:

```
"""
    Ah = lr_circ(A, K::Int)

Compute the rank-at-most-`K` best approximation to circulant matrix `A`

In:
- `A` : `N × N` circulant matrix
- `K` : rank constraint (nonnegative integer: 0, 1, 2, ...)

Out:
- `Ah` : `N × N` best approximation to `A` having rank ≤ `K`
"""
function lr_circ(A, K::Int)::Matrix
```

Hint. To make complex circulant test matrices, you may use
`using SpecialMatrices: Circulant`
`Circulant(rand(ComplexF64, N))`
See submission instructions near front of exam.

. . . .

Because $\boldsymbol{A}$ is circulant, it has eigendecomposition $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}'$ where $\boldsymbol{Q}$ is the orthonormal DFT matrix. Permuting so that the eigenvalues are in descending magnitude order *and* accounting for the signs of the eigenvalues leads to an SVD of $\boldsymbol{A}$:

$$\begin{aligned}\boldsymbol{A} &= \boldsymbol{Q\Lambda Q}' = \boldsymbol{Q}\,\mathsf{Diag}\{\mathrm{sign}(\lambda_n)\}\,\mathsf{Diag}\{|\lambda_n|\}\,\boldsymbol{Q}' \\ &= (\boldsymbol{Q}\,\mathsf{Diag}\{\mathrm{sign}(\lambda_n)\}\,\boldsymbol{P})(\boldsymbol{P}'\,\mathsf{Diag}\{|\lambda_n|\}\,\boldsymbol{P})(\boldsymbol{P}'\boldsymbol{Q}') = \boldsymbol{U\Sigma V}'.\end{aligned}$$

As usual, the rank-at-most-K approximation is

$$\hat{\boldsymbol{A}}_K = \sum_{k=1}^K \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k' = \sum_{k=1}^K \left|\lambda_{(k)}\right| \boldsymbol{q}_{(k)}\,\mathrm{sign}\big(\lambda_{(k)}\big)\,\boldsymbol{q}_{(k)}' = \sum_{k=1}^K \lambda_{(k)} \boldsymbol{q}_{(k)} \boldsymbol{q}_{(k)}'.$$

In words, keep the components of the K largest magnitude eigenvalues.
As usual, the solution is not unique if $\sigma_K = \sigma_{K+1}$ and this happens quite often if $\boldsymbol{A}$ is real because the DFT of a real array satisfies a conjugate symmetry property where DFT coefficients come in complex conjugate pairs. Testing with random complex arrays avoids this issue.
Solution:

```julia
using FFTW: fft
using LinearAlgebra: I, Diagonal

"""
    Ah = lr_circ(A, K::Int)

Compute the rank-at-most-`K` best approximation to circulant matrix `A`

In:
- `A` : `N × N` circulant matrix
- `K` : rank constraint (nonnegative integer: 0, 1, 2, ...)

Out:
- `Ah` : `N × N` best approximation to `A` having rank ≤ `K`
"""
function lr_circ(A, K::Int)::Matrix
    N = size(A,1)
    K == 0 && return zeros(eltype(A), N,N)
    ((K < 0) || (K > N)) && throw("bad K") # error check
    eigs = fft(A[:,1])
    perm = sortperm(abs.(eigs) ; rev=true) # descending magnitude order
    # Get the K relevant columns of the orthonormal DFT matrix.
    # (Ch1 illustrated this "outer product" technique):
    k = perm[1:K] .- 1 # the relevant "k" values in {0,1,...,N-1}
    QK = exp.(2im*π * (0:(N-1)) * k' / N) / sqrt(N) # exp(2ıπ k/N)^n
    return QK * Diagonal(eigs[perm[1:K]]) * QK'
end
```

Alternate approach:

```julia
using FFTW: fft, ifft
using SpecialMatrices: Circulant
```

```
function lr_circ(A, K::Int)::AbstractMatrix # note change in return type here!
    N = size(A, 1)
    eig_vals = fft(A[:,1])
    svd_vals = abs.(eig_vals)
    index = sortperm(svd_vals, rev=true)[1:K]
    new_eigs = zeros(ComplexF64, N)
    new_eigs[index] = eig_vals[index]
    return Circulant(ifft(new_eigs))
end
```

This alternate way would be preferable in practice because it returns an efficient `Circulant` type. Note that I had to change the return type to `AbstractMatrix` for this approach to be effective.
Test:

```
using LinearAlgebra: I, svd, Diagonal
using SpecialMatrices: Circulant
include(ENV["hw551test"] * "lr_circ.jl")
N = 5
@assert lr_circ(I(N), 0) == zeros(N,N)
@assert lr_circ(I(N), N) ≈ I(N)
@assert lr_circ(ones(N,N), 1) ≈ ones(N,N)

T = ComplexF32
#T = Float32 # tricky because of conjugate eigenvalues (DFT property)
N = 6
A = Circulant(rand(T, N))
U, s, V = svd(Matrix(A))
for K=0:N
    AK = U[:,1:K] * Diagonal(s[1:K]) * V[:,1:K]'
    @assert lr_circ(A, K) ≈ AK
end
```

(HW 9.4)
57% completely correct (mean 2.6/3)
32% efficient (mean 2.3/3)
47% formed full $\boldsymbol{Q}$ instead of $\boldsymbol{Q}_K$
7% used `diagm` *instead of* `Diagonal`
7% used `Q'A[:,1]` *instead of* `fft(A[:,1])`

Initially (97 students attempted):
76 passed autograder with tries (38, 25, 8, 4, 0, 1)
After ignoring submission with illegal keywords and setting tolerance = 1e-2:
76 passed autograder tries (39, 28, 8, 0, 1)

`lr/frob_rank_err`

5. (9 points)
A $M \times N$ matrix $\boldsymbol{B}$ has nonzero singular values $\{1, 2, 4, 8, 16, 24\}$.
Determine $\|\hat{\boldsymbol{B}} - \boldsymbol{B}\|_{\mathrm{F}}^2$, where $\hat{\boldsymbol{B}} \triangleq \underset{\boldsymbol{X} \in \mathbb{C}^{M \times N}}{\arg\min} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{B}\|_{\mathrm{F}}^2 + \frac{9}{2}\,\mathsf{rank}(\boldsymbol{X})$. Explain.

This is SVHT with cutoff $\sqrt{2\beta} = \sqrt{2(9/2)} = 3$ so the weights for $\boldsymbol{Y}$ are $(0, 0, 4, 8, 16, 24)$ and the approximation error squared is $1^2 + 2^2 = 5$.
(HW 8.3) [82% correct.]

`code/opnorm/nearptf`

6. (15 points)
Consider the set of $N \times M$ Parseval tight frames: $\mathcal{T} = \left\{\boldsymbol{T} \in \mathbb{F}^{N \times M} \;:\; \boldsymbol{T} \text{ is a Parseval tight frame}\right\}$.
Write a JULIA function that, given a $N \times M$ matrix $\boldsymbol{X}$, returns the closest (in the Frobenius norm sense) Parseval tight frame to $\boldsymbol{X}$, *i.e.*, $\hat{\boldsymbol{T}} = \underset{\boldsymbol{T} \in \mathcal{T}}{\arg\min} \|\boldsymbol{T} - \boldsymbol{X}\|_{\mathrm{F}}$. Assume $N \leq M$.
Template:

```
"""
    T = nearptf(X)

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

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

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

See submission instructions near front of exam.

Another way of writing $\mathcal{T}$ is $\mathcal{T} = \left\{\boldsymbol{T} \in \mathbb{F}^{N \times M} \;:\; \boldsymbol{T}\boldsymbol{T}' = \boldsymbol{I}_N\right\}$.
This problem is essentially the same as the (generalized) orthogonal Procrustes 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 $\hat{\boldsymbol{T}} = \hat{\boldsymbol{Q}}'$, $\hat{\boldsymbol{Q}} = \arg\min_{\boldsymbol{Q} \in \mathcal{Q}} \|\boldsymbol{Q}' - \boldsymbol{X}\|_{\mathrm{F}} = (\boldsymbol{U}\boldsymbol{V}_N')'$,
where the economy SVD is: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}_N\boldsymbol{V}_N'$. So $\hat{\boldsymbol{T}} = \boldsymbol{U}\boldsymbol{V}_N'$.
Solution:

```
using LinearAlgebra: svd

"""
    T = nearptf(X)

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

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

Out:
```

```
* `T` `N × M` matrix that is the nearest Parseval tight frame to `X`
"""
function nearptf(X::AbstractMatrix)
    U, _, V = svd(X) # economy SVD has V = V_N here
    return U*V'
end
```

Test:

```
using LinearAlgebra: svd
using Random: seed!

include(ENV["hw551test"] * "nearptf.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
        nearptf(X)
    end

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

    Q2 = svd(randn(N,M)).U
    T = [Q1 Q2]/sqrt(2)
    @assert T ≈ nearptf(T)
end
```

One student discovered that the solution is in Theorem 2 of this 2005 paper.

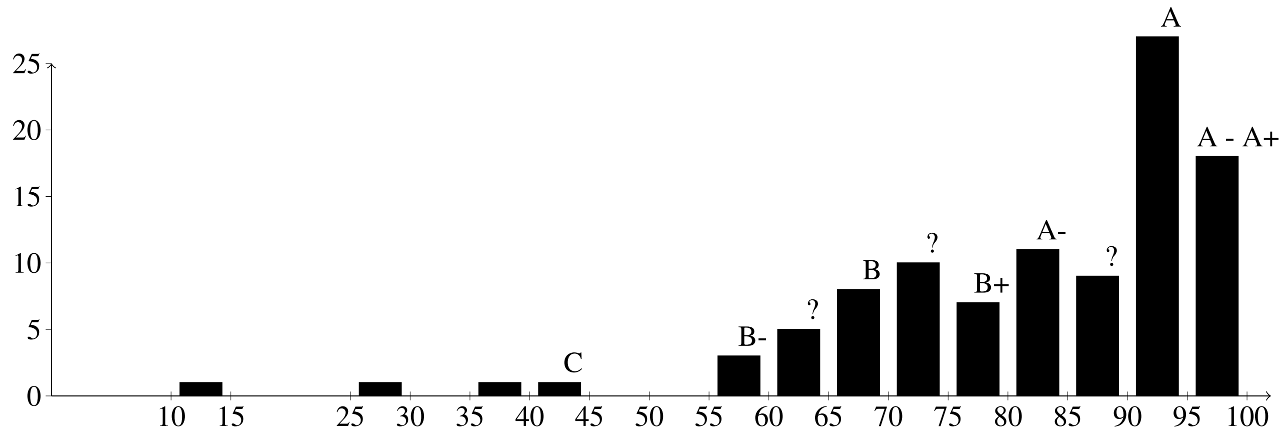
(HW 6.2 7.8)
40% correct with economy SVD (mean 2.1/3)
30% used full SVD UV' which is mathematically incorrect but the idea still works in code because `svd` returns economy SVD not full SVD.
10% incorrectly used rank(X)
48% efficient using economy SVD (mean 2.4/3)
16% had unnecessary `V[:,1:N]` indexing
16% had unnecessary multiply by `I`
7% called `rank(X)`

Initially (97 students attempted):
93 passed autograder with tries (70, 16, 6, 1)
After ignoring submissions with illegal keywords and using tolerance = 1e-6:
97 passed autograder with tries (77, 17, 3)
Some submissions should not have passed due to `rank`.

Exam scores with *approximate* grades.

Gradescope part (excluding code) 102 students: min 5.5, max 43/43, std dev 5.8, mean 36.0, median 38
autograder part: range 0-27/27, mean 20.3, std dev 7.8
Canvas part: min 5, max 30/30, mean 25.5

Overall, 102 students: mean=82.0/100, median=87, std=15.9, range: 10.5-100


A
A - A+
A-
?
B+
?
B
?
B-
C
0
5
10
15
20
25
10
15
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100