Pr. 1.
Show that for any **vector norm** on $\mathbb{F}^N$, the ball of radius $r \geq 0$ with respect to that norm

$$\mathcal{B}_r \triangleq \left\{ \boldsymbol{x} \in \mathbb{F}^N \,:\, \|\boldsymbol{x}\| \leq r \right\}$$

is a **convex set**.

Pr. 2.
A previous problem showed that if $\boldsymbol{A}$ is an $M \times N$ matrix, then $\|\boldsymbol{A}\|_2 \leq \sqrt{N} \, \|\boldsymbol{A}\|_1$.

(a) Find a simple (but nonzero) 1×2 or 2×1 matrix where this inequality is equality. (When an inequality is achievable, we call it a **tight** bound.)

(b) This bound is not tight for **normal** matrices. If $\boldsymbol{A}$ is normal and $N \times N$, then $\|\boldsymbol{A}\|_2 \leq c \, \|\boldsymbol{A}\|_1$ for a constant c that is much smaller than N in general. Determine (and prove) the best value for c.
Hint. Think about **spectral radius**.

(c) Give a nonzero 2×2 matrix where the inequality in (b) is an equality, so it is also tight.

Pr. 3.
Enter the following `Julia` code that makes some **companion matrix** examples.

```
using LinearAlgebra: eigvals, I
n = rand(4:7)
a = randn(n+1) # random polynomial coefficients (a_0,a_1,...,a_n)
b = reverse(a) # reverse the coefficient order
# companion matrix maker:
compan = c -> [-transpose(reverse(c)); [I zeros(length(c)-1)]]
A = compan(a[1:end-1] / a[end])
B = compan(b[1:end-1] / b[end])
[eigvals(A) eigvals(B) 1 ./ eigvals(B)] # study array columns
```

What pattern do you see? Use `rand(ComplexF64, n)` to also try *complex* coefficients. Repeat multiple times to form a conjecture on how the **eigenvalues** of `A` and `B` are related. Prove your conjecture theoretically.

Pr. 4.
Let $\boldsymbol{A}$ denote an $N \times N$ **Vandermonde** matrix corresponding to the roots of the polynomial $z^N - 1$.
Let $\boldsymbol{B}$ denote an $N \times N$ **Vandermonde** matrix corresponding to $\left\{ \mathrm{e}^{-\imath 2\pi (n+1/2)/N} \,:\, n = 0, \ldots, N-1 \right\}$.
Define the matrix $\boldsymbol{X} \triangleq \begin{bmatrix} a\boldsymbol{A} & b\boldsymbol{B} \end{bmatrix}$, for $a, b \in \mathbb{C}$.

(a) Determine the singular values of $\boldsymbol{X}$. Hint. Examine $\boldsymbol{X}\boldsymbol{X}'$.
Optional:
Determine whether the matrix $\boldsymbol{X}$ is a **frame**, a **tight frame**, a **Parseval tight frame**, or none of the above.

(b) Determine $\boldsymbol{X}^+$. (Simply as much as possible.) Hint. Think about the rank of $\boldsymbol{X}$.
Optional:
If $\boldsymbol{X}$ is some type of frame, determine its **frame bound** (s).

(c) Optional. Write a `Julia` function with inputs a, b and $\boldsymbol{y} \in \mathbb{C}^N$ that computes efficiently the minimum-norm solution to the LS problem

$$\underset{\boldsymbol{x} \in \mathbb{C}^{2N}}{\arg\min} \, \|\boldsymbol{X}\boldsymbol{x} - \boldsymbol{y}\|_2 .$$

Hint. For maximum efficiency, use a Vandermonde matrix form corresponding to the **DFT**. Aim for a solution with $O(N \log_2 N)$ computation! The code `phase = cis.(pi / N * (0:N-1))` should be useful.

Pr. 5.

(a) Show that all **circulant matrices** (of the same size) **commute**.

(b) Show that all **circulant matrices** are **normal**.

(c) Determine the eigenvalues of the following $N \times N$ circulant matrix:

$$\boldsymbol{C} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & \ddots & & & \vdots \\ 0 & 0 & 0 & \dots & 0 & -1 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 & -1 \end{bmatrix}.$$

Give a simple expression that holds for any $N \in \mathbb{N}$. Multiplying this matrix by a vector performs **finite differences** of the vectors elements with **periodic end conditions**.

(d) Check your expression for the previous part numerically for the case $N = 4$.
Report the eigenvalues from your expression and from the `eigvals` command in `Julia`.

(e) Write a very short `Julia` function that computes the **nuclear norm** of any **circulant matrix** `C` *without* calling expensive functions like `eig` or `svd`. Suggestion: check your function using the previous part.

(f) Optional. In image processing, matrices that are **block circulant with circulant blocks** (BCCB) are particularly important. Such matrices have the form $\boldsymbol{B} = \boldsymbol{C}_2 \otimes \boldsymbol{C}_1$, where $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are both circulant matrices.

Express the eigenvalues of $\boldsymbol{B}$ in terms of the first columns of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$.

Challenge. Express the eigenvalues of $\boldsymbol{B}$ in terms of the first column of $\boldsymbol{B}$.

Pr. 6.

Low-rank matrix denoising using non-convex Schatten p-"norm"

The **Schatten p-norm**: $\|A\|_{S,p} = \left(\sum_{k=1}^r \sigma_k^p\right)^{1/p}$, where σ_k denotes the singular values of A, is a proper matrix norm for $p \geq 1$. Values of $p \in (0, 1)$ are also useful as regularizers for **low-rank matrix denoising** problems even though it is not a proper norm for $p < 1$.

(a) For $Y = X + \varepsilon \in \mathbb{C}^{M \times N}$, where we think X is low rank, find an expression for the solution of the regularized low-rank matrix denoising method that uses the following **Schatten-norm regularizer** for $p = 1/2$:

$$\hat{X} = \underset{X \in \mathbb{C}^{M \times N}}{\arg\min} \frac{1}{2} \|Y - X\|_F^2 + \beta R(X, 1/2), \quad R(X, p) = \|X\|_{S,p}^p = \sum_{k=1}^r \sigma_k^p(X).$$

Hint: a previous problem will be helpful. Submit your written answer to this part to gradescope.

(b) Write a `Julia` function that performs regularized low-rank matrix denoising using the above cost function.

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

```
"""
    lr_schatten(Y, reg::Real)

Compute the regularized low-rank matrix approximation as the minimizer over `X`
of `1/2 ‖ Y - X ‖_F^2 + reg R(X)`
where `R(X)` is the Schatten p-norm of `X` raised to the pth power, for `p=1/2`,
i.e., `R(X) = \\sum_k (σ_k(X))^{1/2}`

# In:
- `Y` :    `M × N` matrix
- `reg`    regularization parameter

# Out:
- `Xh` : `M × N` solution to above minimization problem
"""
function lr_schatten(Y, reg::Real)
```

Email your solution as an attachment to `eecs551@autograder.eecs.umich.edu`.

Think about your solution and consider whether it seems to be a good method for denoising.

(c) Apply your denoising method to the noisy 100×30 Block M image `Y` in the demo notebook `07_optshrink1.ipynb` at `http://web.eecs.umich.edu/~fessler/course/551/julia/demo` using $\beta = 1000$.

Report the NRMSE of your estimate $\hat{X}$ and submit to gradescope a picture of $\hat{X}$ and a scatter plot of the singular values of $\hat{X}$, X and Y. All of the plotting commands are in the notebook already; you simply need to `include` your `lr_schatten.jl` solution.

Pr. 7.

Testing for common roots of polynomials

(a) Given two polynomials $p(z) = \sum_{i=0}^{m} \alpha_i z^i$ and $q(z) = \sum_{j=0}^{n} \beta_j z^j$, for $n \geq 0$ not necessarily equal to $m \geq 0$, where $\alpha_m \neq 0$ and $\beta_n \neq 0$. How would you check whether p and q have a common (possibly complex) root?

You may use the determinant and/or the trace, but you may *not* explicitly compute the eigenvalues or eigenvectors of the respective companion matrices. Other matrix decompositions (QR, LU) are not allowed either.

Hint. Use the relationship between the eigenvalues of the **Kronecker sum** of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{B}$.

(b) Write a function called `common_root` that takes as input two vectors (not necessarily of equal dimension) whose elements represent the coefficients of the polynomials $p(z)$ and $q(z)$ and returns `true` if the two polynomials share a common root and `false` otherwise.
Caution: the leading coefficient is not necessary 1 here, but you can assume it is nonzero.
In `Julia`, your file should be named `common_root.jl` and should contain the following function:

```julia
"""
    haveCommonRoot = common_root(a, b ; atol)

Determine if the polynomials described by input coefficient vectors `a`
and `b` share a common root, to within an absolute tolerance parameter `atol`.
Assume leading coefficients `a[end]` and `b[end]` are nonzero.

# In:
- `a` : vector of length `m + 1` with `a[m+1] != 0` and `m ≥ 0`
  defining a degree `m` polynomial of the form:
`p(z) = a[m+1] z^m + a[m] z^(m - 1) + ... + a[2] z + a[1]`

- `b` : vector of length `n + 1` with `b[n+1] != 0` and `n ≥ 0`
  defining a degree `n` polynomial of the form:
  `q(z) = b[n+1] z^n + b[n] z^(n - 1) + ... + b[1] z + b[1]`

# Option:
- `atol::Real` absolute tolerance for calling `isapprox`

# Out:
- `haveCommonRoot` = `true` when `p` and `q` share a common root, else `false`
"""
function common_root(a::AbstractVector, b::AbstractVector ; atol::Real=1e-6)
```

Email your solution as an attachment to `eecs551@autograder.eecs.umich.edu`.
Hint 1: Use the `Julia` function `kron`.
Hint 2: To consider finite numerical precision, use the `isapprox` function with the input absolute tolerance parameter `atol` to decide whether some value is "close enough" to zero. See the `Julia` manual for `isapprox` to see why `atol` is essential here.
Test your code with some polynomials of your own design before submitting to the autograder.

- Be sure to test some cases that have roots that are zero or complex.
- Be sure to handle cases where $m = 0$ and/or $n = 0$.
- For randomly chosen Gaussian polynomial coefficients, there is an exceedingly small probability of having common roots, so check more cases than just random coefficients.

Pr. 8.
The following output is from chatGPT in Dec. 2022 when asked to write SVST code:

FE Write a singular value soft thresholding function in Julia

Here is a function that implements singular value soft thresholding in Julia:

Copy code

```
function svt(A::Array, lambda::Real)
    U, S, V = svd(A)

    for i in 1:length(S)
        if S[i] > lambda
            S[i] -= lambda
        elseif S[i] < -lambda
            S[i] += lambda
        else
            S[i] = 0
        end
    end

    return U * diagm(S) * V'
end
```

(a) Does this code have the appropriate input type declarations? If not, describe how to improve.

(b) Does this code work correctly? If not, describe how to fix any errors.

(c) Is this code reasonably concise? If not, describe how to simplify.

(d) Is this code efficient computationally? If not, describe how you would make it more efficient. Hint. Remember that `Julia` is a compiled language.

(e) Do you think ChatGPT would work better with python? See this study.

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

Pr. 9.
For $1 \leq N \leq M$, define the spectral-norm unit ball as

$$\mathcal{B} = \left\{ \boldsymbol{A} \in \mathbb{F}^{M \times N} : \|\boldsymbol{A}\|_2 \leq 1 \right\}.$$

Now define $\mathcal{X}$ to be the subset of matrices in $\mathcal{B}$ having maximal Frobenius norm:

$$\mathcal{X} = \{ \boldsymbol{X} \in \mathcal{B} : \|\boldsymbol{X}\|_{\mathrm{F}} \geq \|\boldsymbol{Y}\|_{\mathrm{F}}, \ \forall \boldsymbol{Y} \in \mathcal{B} \}.$$

Show, or disprove with a counterexample, that $\mathcal{X} = \mathcal{V}^{M \times N}$, where $\mathcal{V}^{M \times N}$ denotes the **Stiefel manifold** of $M \times N$ matrices.

Pr. 10.
The set $\left\{G_N^0, G_N^1, \ldots, G_N^{N-1}\right\}$ forms a **basis** for the subspace of $N \times N$ circulant matrices in the vector space $\mathbb{F}^{N \times N}$ of all $N \times N$ matrices, where G_N denotes the "generator" for circulant matrices.

(a) Is this set an **orthogonal basis** for that subspace? Explain.

(b) Is this set an **orthonormal basis** for that subspace? Explain.

Pr. 11.
Prove or disprove (by a counterexample) the following statement. If T is any matrix for which $T^k = I$ for some natural number $k > 1$, then T is **normal**.

Pr. 12.
Let $t_k = k\Delta$, for some integer k, and $\Delta \in \mathbb{R}$. Consider the sum of sinusoids signal $y(t_k) = \sum_{i=1}^{r} b_i e^{\imath w_i t_k}$, where $y(t_k) \in \mathbb{C}^N$, $b_1, \ldots, b_r \in \mathbb{C}^N$ are linearly independent vectors, and $w_i \in [0, 1]$ are distinct, and $\imath = \sqrt{-1}$. Express the recursion in the following simple matrix form: $y(t_{k+1}) = A\, y(t_k)$.

(a) Determine the matrix A.

(b) Determine the eigenvalues and eigenvectors of A.
Hint. For a matrix B of full column rank r, we have that $B^+ B = I_r$, where I_r is an $r \times r$ identity matrix.

Pr. 13.
Define $\mathbb{R}_+ \triangleq [0, \infty)$. A **symmetric gauge** function $\phi(x)$ is a mapping from $\mathbb{R}^n$ into $\mathbb{R}_+$ that satisfies the following four properties for all $x, y \in \mathbb{R}^n$.

- $\phi(x) > 0$ if $x \neq 0$ (positivity)
- $\phi(\alpha x) = |\alpha|\, \phi(x)$ for all $\alpha \in \mathbb{R}$ (homogeneity)
- $\phi(x + y) \leq \phi(x) + \phi(y)$ (triangle inequality)
- $\phi(s_1 x_{[1]}, \ldots, s_n x_{[n]}) = \phi(x)$ for all $s_k = \pm 1$ and for any permutation $\left(x_{[1]}, \ldots, x_{[n]}\right)$ of the elements of x. (symmetry)

Some examples are:

- $\phi(x) = \|x\|_1$
- $\phi(x) = \max_i |x_i| = \|x\|_\infty$
- $\phi(x) = 7|x_{(1)}| + 5|x_{(2)}|$, where $x_{(1)}$ and $x_{(2)}$ denote the first and second largest elements of x in magnitude.

Such symmetric gauge functions are at the heart of many matrix norm properties.
Let $\phi(\cdot)$ be any symmetric gauge function and define a matrix norm by

$$\|A\|_\phi = \phi(\sigma_1, \ldots, \sigma_{\min(M,N)}),$$

where $\{\sigma_k\}$ denote the singular values of an $M \times N$ matrix A.

(a) Verify that this definition indeed defines a proper matrix norm.

(b) Prove or disprove (by counterexample) that any such matrix norm $\|A\|_\phi$ is **unitarily invariant**.
