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

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

is a **convex set**.

Pr. 2.
An alternative to the logistic loss function for **binary classifier design** is the ℓ_2-hinge **loss function**:

$$\psi(t) \triangleq (\max(1-t, 0))^2.$$

(a) Show that this loss function has a Lipschitz continuous derivative and determine its Lipschitz constant $L_{\dot{\psi}}$.

(b) Now consider **logistic regression** using this loss function for binary classifier design using the optimization problem

$$\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad f(\boldsymbol{x}) = \sum_{m=1}^{M} \psi(y_i \langle \boldsymbol{x}, \boldsymbol{v}_i \rangle) + \beta \frac{1}{2} \|\boldsymbol{x}\|_2^2 \,.$$

Determine a Lipschitz constant L for the gradient of the overall cost function $f(\boldsymbol{x})$.

Pr. 3.
The following matrix is known to be some constant plus a rank-1 matrix, but some values are missing.

$$\boldsymbol{A} = \begin{bmatrix} 20 & 14 & w & 17 \\ 44 & x & 16 & y \\ 8 & 6 & z & 7 \end{bmatrix}.$$

Determine the values of w,x,y,z.

Pr. 4.
Make a clearly stated problem that could be used on an EECS 551 exam based at least in part on the material in Ch. 7 and beyond, and provide your own solution to the problem. Your problem could include something about `Julia`, but in that case should be about concepts, not mere syntax, and should not require submitting code to an autograder.
If you make a good problem (not too trivial, not too hard) then it might be used on an actual exam.
To earn full credit:

- Your problem and solution must fit on a single page with a horizontal line that clearly separates the question (above) from the answer (below).
- Your solution to the problem must be correct.
- Your problem should not be excessively trivial (nor beyond the scope of the course).
- Your problem must not be identical to any quiz or homework or clicker or sample exam problem.
- Your problem should not be of the form "prove that ..." but it is fine to pose a True/False problem with an answer that needs some proving to explain.
- You must upload your problem/solution to gradescope as usual for grading.
- You must submit your problem/solution in pdf format to Canvas for distribution by the deadline for this HW. (We can export from Canvas easily but not from gradescope.)
- Do not submit any other parts of your HW assignment to Canvas; only this 1 page pdf file.
- Do not put your name on the problem/solution that you upload to Canvas because we plan to export all problems/solutions to distribute as practice problems.

- We may make an announcement on Canvas about some additional formatting requirements, so be sure to check the announcements before submitting.

Pr. 5.

Low-rank matrix completion from noisy data

In this problem you will use the **Schatten p-norm** regularizer for **matrix completion**, and show that it works better than the **nuclear norm** regularizer. In matrix completion, instead of being given a full matrix $\boldsymbol{Y} = \boldsymbol{X}_{\text{true}} + \boldsymbol{\varepsilon}$, where $\boldsymbol{X}_{\text{true}}$ is low rank and $\boldsymbol{\varepsilon}$ is additive noise, we are given $\boldsymbol{Y} = \boldsymbol{M} \odot (\boldsymbol{X}_{\text{true}} + \boldsymbol{\varepsilon})$, where $\boldsymbol{M}$ is a binary matrix (a sampling mask or pattern) that is 1 where we have data and 0 where we do not, and $\odot$ denotes element-wise multiplication.

We want to estimate the matrix $\boldsymbol{X}_{\text{true}}$ by solving the following optimization problem:

$$\hat{\boldsymbol{X}} = \underset{\boldsymbol{X} \in \mathbb{C}^{M \times N}}{\arg\min} \frac{1}{2} \|\boldsymbol{M} \odot (\boldsymbol{Y} - \boldsymbol{X})\|_{\text{F}}^2 + \beta R(\boldsymbol{X}),$$

where the regularizer $R(\boldsymbol{X})$ is the one associated with the Schatten p-norm for $p = 1/2$ as discussed in a previous HW problem.

(a) Implement a `Julia` function that performs a specified number of iterations of FISTA to solve for $\hat{\boldsymbol{X}}$. As a guide for this, use code from the FISTA method in the notebook `09_lrmc_nuc` in `http://web.eecs.umich.edu/~fessler/course/551/julia/demo/` This notebook uses the nuclear norm regularizer whereas here you use the Schatten-based regularizer. Just like was done in the notebook, initialize with $\boldsymbol{X}_0 = \boldsymbol{Y}$, the "zero-filled" data.

You might want to debug this in a Jupyter notebook, but then extract your FISTA iteration as a function for submitting to the autograder.

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

```
"""
`Xh = fista_schatten(Y, M, reg::Real, niter::Int)`

Perform `niter` FISTA iterations to perform matrix completion
by seeking the minimizer over `X` of
`1/2 \|M .* (Y - X)\|^2 + reg R(x)`
where `R(X)` is the Schatten p-norm of `X` raised to the `p`th power, for `p=1/2`,
i.e., `R(X) = sum_k (sig_k(X))^(1/2)`

In:
- `Y`       `M by N` matrix (with zeros in missing data locations)
- `M`       `M by N` binary matrix (with ones in sampled data locations)
- `reg` regularization parameter
- `niter`   # of iterations

Output
- `Xh`  `M by N` estimate of `X` after `niter` FISTA iterations
"""
function fista_schatten(Y, M, reg::Real, niter::Int)
```

Submit your solution to the autograder by emailing it as an attachment to `eecs551@autograder.eecs.umich.edu`.

(b) Incorporate your `fista_schatten` code into the Jupyter notebook `09_lrmc_nuc` into the appropriate places at the end of the notebook, either by cut-and-paste or by `include("file.jl")` statements. (Be sure your FISTA method passes the autograder test first.) Apply your Schatten-based FISTA iteration to the "UM" data used in that notebook using $\beta = 120$ for the Schatten case.

Make images of the estimate $\hat{\boldsymbol{X}}$ and the error $\hat{\boldsymbol{X}} - \boldsymbol{X}_{\text{true}}$ for both the FISTA/SVST solution (with $\beta = 0.8$ per the notebook) and for your FISTA/Schatten solution (with $\beta = 120$ per the notebook). Use the xlabel command in the notebook for the error images to show the NRMSE $\left\|\hat{\boldsymbol{X}} - \boldsymbol{X}_{\text{true}}\right\|_{\text{F}} / \|\boldsymbol{X}_{\text{true}}\|_{\text{F}} \cdot 100\%$ for the two methods. Submit those four clearly labeled images to gradescope.

Optionally you can experiment with the regularization parameter β for one or both methods to see if you can make better images.

This is the final segment of the three-part matrix completion problem. You will have implemented a method using quite contemporary methods. Note that FISTA was derived for convex problems whereas for $p < 1$ the Schatten regularizer is a non-convex “quasi-norm” so convergence is (to my knowledge) an open question.

An interesting (but probably nontrivial) extension of this work would be to generalize SURE to select the regularization parameter β automatically.

Pr. 6.

(a) Derive the optimal **step size** α_* for classical **gradient descent** of the **least-squares** cost function $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$, in terms of the singular values of the (full column rank) matrix $\boldsymbol{A}$. Here, optimal means $\rho(\boldsymbol{I} - \alpha\boldsymbol{A}'\boldsymbol{A})$ is as small as possible.

(b) For that optimal step size α_*, determine the **spectral radius** $\rho(\boldsymbol{I} - \alpha_*\boldsymbol{A}'\boldsymbol{A})$ that governs the **convergence rate** of GD.

Express your answer in terms of the **condition number** of $\boldsymbol{A}'\boldsymbol{A}$.

Pr. 7.
Algebraic numbers and polynomials with integer-valued coefficients

- The zeros of any **monic polynomial** are the **eigenvalues** of a corresponding **companion matrix**.
- The **Kronecker sum** of two square matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{m\times m}$ is defined as

$$\boldsymbol{A} \oplus \boldsymbol{B} = \boldsymbol{A} \otimes \boldsymbol{I}_m + \boldsymbol{I}_n \otimes \boldsymbol{B}.$$

  The mn **eigenvalues** of $\boldsymbol{A} \oplus \boldsymbol{B}$ are $\lambda_i(\boldsymbol{A}) + \lambda_j(\boldsymbol{B})$ for $i = 1, \ldots, n$ and $j = 1, \ldots m$.
- An **algebraic number** is a value in $\mathbb{C}$ that is a zero of some polynomial having *integer-valued coefficients*. For example, $x = \sqrt{5}$ is an algebraic number because it is a zero of the polynomial $p(x) = x^2 - 5$.

(a) Consider $x_1 = 3+\sqrt{3}$ and $x_2 = 5-\sqrt{7}$. These are algebraic numbers because they are zeros of some corresponding polynomials $p_1(x)$ and $p_2(x)$, respectively, having integer-valued coefficients. Now define $x_3 \triangleq x_1 + x_2$ and $x_4 \triangleq x_1 x_2$. A **theorem in algebra** says that x_3 and x_4 are also **algebraic** because algebraic numbers form a **field**. In other words, there is a polynomial $p_3(x)$ with integer-valued coefficients that has one zero equal to x_3, and a polynomial $p_4(x)$ with integer coefficients that has one zero equal to x_4.

Use the **Kronecker product** and/or **Kronecker sum** to find those polynomials $p_3(x)$ and $p_4(x)$ without using the `poly_coeff` function described below.

Hint: Start by finding the companion matrix with integer valued coefficients that has an eigenvalue equal to x_1 and another companion matrix with eigenvalue x_2.

You may use a symbolic toolbox to evaluate determinants. To evaluate the determinant $\det\left(z\boldsymbol{I} - \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix}\right)$, one can use the `MATLAB` command `det(sym('z')*eye(2) - [3 2; 2 1])`.

(b) Let $a, b, c, d \in \mathbb{N}$ be positive integers, and set $x_1 = a+\sqrt{b}$ and $x_2 = c-\sqrt{d}$. Write a function called `poly_coeff` that takes (a, b, c, d) as input and returns vectors $p, q \in \mathbb{Z}^5$ of integer-valued coefficients defining the monic ($p_1 = q_1 = 1$) quartic polynomials $p_3(x) = \sum_{i=1}^5 p_i x^{5-i}$ and $p_4(x) = \sum_{j=1}^5 q_j x^{5-j}$ with the properties that $p_3(x_3) = 0$ and $p_4(x_4) = 0$. Use your “hand” solution from part (a) as a test of your function.

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

```
"""
    p, q = poly_coeff(a, b, c, d)

Use Kronecker products to construct integer-valued polynomials
with the given (algebraic) numbers as zeros.

In:
```

```
* `a, b, c, d` are positive integers

Out:
* `p` and `q` are vectors of length 5 with `p[1] = 1` and `q[1] = 1`
such that, in the notation defined below, `P3(x3) = 0` and `P4(x4) = 0`

Notation:
* `x1 = a + sqrt(b)`
* `x2 = c - sqrt(d)`
* `x3 = x1 + x2`
* `x4 = x1 * x2`
* `P1(x)` a quadratic monic polynomial with integer-valued coefficients with a zero at `x1`
* `P2(x)` a quadratic monic polynomial with integer-valued coefficients with a zero at `x2`
* `P3(x) = x^4 + p[2] x^3 + p[3] x^2 + p[4] x + p[5]`
* `P4(x) = x^4 + q[2] x^3 + q[3] x^2 + q[4] x + q[5]`
"""
function poly_coeff(a, b, c, d)
```

Submit your solution to the autograder by emailing it as an attachment to `eecs551@autograder.eecs.umich.edu`.

Pr. 8.

The lecture notes show that the set $\{\boldsymbol{G}_N^0, \boldsymbol{G}_N^1, \ldots, \boldsymbol{G}_N^{N-1}\}$ form a **basis** for the subspace of $N \times N$ circulant matrices in the vector space of all $N \times N$ matrices, where $\boldsymbol{G}_N$ denotes the "generator" for circulant matrices.
Is this set an **orthogonal basis** for that subspace? Explain.
Is this set an **orthonormal basis** for that subspace? Explain.

Pr. 9.

Please fill out the online course evaluation. Your evaluation is very important to me for improving the course next time I teach it. Your evaluation is entirely anonymous. Thank you for your feedback.

Optional problem(s) below

(not graded, but solutions will be provided for self check; do not submit to gradescope)

Pr. 10.

Prove, or disprove by counter example, the following conjecture. If $\|\cdot\|_{\mathrm{UI}}$ denotes any **unitarily invariant** matrix norm, then the solution to the **low-rank matrix approximation** problem

$$\hat{\boldsymbol{X}} = \arg\min_{\boldsymbol{X}} \|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{UI}} + \beta \, \mathrm{rank}(\boldsymbol{X})$$

has the form

$$\hat{\boldsymbol{X}} = \sum_{k=1}^{r} w_k \boldsymbol{u}_k \boldsymbol{v}_k', \qquad \boldsymbol{Y} = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r'$$

where $w_k = \begin{cases} \sigma_k, & \sigma_k > f(\beta) \\ 0, & \text{else,} \end{cases}$ where $f(\beta)$ is some function of β and of the particular UI norm used.
