Pr. 1.
Let $\boldsymbol{\pi}$ be a N-dimensional vector having nonnegative elements that sum to one.

(a) Prove (by example) that there exists a **Markov chain** with $\boldsymbol{P} \neq \boldsymbol{I}$ for which $\boldsymbol{\pi}$ is an equilibrium distribution. Specify the **transition matrix** $\boldsymbol{P}$ for your example.

(b) For that transition matrix $\boldsymbol{P}$, discuss whether the given $\boldsymbol{\pi}$ is the *unique* equilibrium distribution.

(c) For that transition matrix $\boldsymbol{P}$, discuss whether the Markov chain **power iteration** $\boldsymbol{\pi}_{k+1} = \boldsymbol{P}\boldsymbol{\pi}_k$ is guaranteed to converge to $\boldsymbol{\pi}$ for any initial probability distribution $\boldsymbol{\pi}_1$.

Pr. 2.
Consider the **Markov chain** that has **transition matrix** $\boldsymbol{P} = \boldsymbol{G}_N + (1-p)(\boldsymbol{e}_3 - \boldsymbol{e}_2)\boldsymbol{e}_1'$ for $N \geq 3$, where $\boldsymbol{G}_N$ denotes the circulant generator matrix, and $p \in (0,1)$. Determine the equilibrium distribution $\boldsymbol{\pi}_*$ of this Markov chain and discuss whether it is unique. This is a generalization of the Markov chain shown in the demo.
Hint. If needed, solve it numerically for a few small values of $N \geq 3$ to form a conjecture that you can then verify analytically. Your solution must be general for $N \geq 3$ and $p \in (0,1)$.

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

Pr. 3.
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. 4.
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

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

Determine a Lipschitz constant L for the gradient of the overall cost function $f(\boldsymbol{x})$, where $\boldsymbol{x}, \boldsymbol{v}_m \in \mathbb{R}^N$ and $y_m = \pm 1$.

Pr. 5.
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. This is a simple variation on the **matrix completion** problem.

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 the sequence $\{\boldsymbol{x}_k\}$ produced by the GD method.

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

Pr. 7.

Practical preconditioners are often built on approximations of a cost function's Hessian inverse. The initial design of such preconditioners often does not ensure convergence of PGD, so a step size α is usually needed. Specifically, for a LS problem we might have $\boldsymbol{P}_0 \approx (\boldsymbol{A}'\boldsymbol{A})^{-1}$ and then let $\boldsymbol{P} = \alpha \boldsymbol{P}_0$, where $\boldsymbol{P}_0$ is positive definite.

(a) Determine the optimal (fixed) **step size** α_* that makes the **preconditioned gradient descent** (PGD) iteration $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \boldsymbol{P}_0 \boldsymbol{A}'(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$ have the fastest possible **convergence rate**.
Hint. The answer depends on both $\boldsymbol{A}$ and the preconditioner $\boldsymbol{P}_0$.

(b) Simplify your expression for α_* as much as possible in the special case where we use the ideal preconditioner $\boldsymbol{P}_0 = (\boldsymbol{A}'\boldsymbol{A})^{-1}$. Does your simple expression (a number) make sense?

Pr. 8.

On HW2 we reminded you that your graduate school experience should be more than just taking courses, and we encouraged you to write down some of your own non-course-related goals for this semester.
For 10 points, reflect on how it went and write down some of your thoughts about what you accomplished and did not complete of your original non-course goals. The graders will assign credit for anything that looks sincere. Prof. Fessler will read these after finals are graded to get a sense of your experience in this somewhat unusual semester.
Submit your reflection to this google form `https://forms.gle/GeeKvFRdCGESrqsY6` and also upload it to gradescope.
Thanks! We encourage you to keep this practice of planning and self reflection in future semesters.

Pr. 9.

Please fill out the online course evaluation for *both* the lecture and your discussion section. Then submit a statement to gradescope saying that you did the evaluations. The honor code applies. 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.

Pr. 10.
Logistic Regression Classification (Discussion Task)
Download the `task-6-classify-logistic.ipynb` jupyter notebook file from Canvas under this week's discussion folder and follow all instructions to complete the task. You may work individually, but we recommend that you work in pairs or groups of three.
When you are finished, upload your solutions to gradescope. Note that the submission for the task is separate from the rest of the homework because the task allows you to submit as a group. Only upload one submission per group! Whoever uploads the group submission should add all group members in gradescope, using the "View or edit group" option on the right-hand sidebar after uploading a PDF and matching pages. Make sure to add all group members, because this is how they will receive credit.

As part of this task, you will need to complete the following Nesterov gradient descent function.
In `Julia`, your file should be named `ngd.jl` and should contain the following function:

```
"""
    x, out = ngd(grad, x0 ; nIters::Int = 200, L::Real = 0, fun = (x, iter) -> 0)

Implementation of Nesterov's FGD (fast gradient descent),
given the gradient of the cost function

In:
- `grad` function that takes `x` and calculates the gradient of the cost function with respect to `x`
- `x0` an initial point

Option:
- `nIters` number of iterations
- `L` Lipschitz constant of the derivative of the cost function
- `fun` a function to evaluate every iteration

Out:
- `x` is the guess of the minimizer after running `nIters` iterations
- `out` is an Array of evaluations of the `fun` function
"""
function ngd(grad::Function, x0::AbstractArray ;
    niter::Int = 200, L::Real = 0, fun::Function = (x, iter) -> 0)
```

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

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

Pr. 11.
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 matrix completion by using `niter` FISTA iterations
to seek 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 (\\sigma_k(X))^(1/2)`.

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

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

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

Hint. The solution requires only small changes to the FISTA code in the demo notebook.

(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 $\|\hat{\boldsymbol{X}} - \boldsymbol{X}_{\text{true}}\|_{\text{F}} / \|\boldsymbol{X}_{\text{true}}\|_{\text{F}} \cdot 100\%$ for the two methods.

(c) 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.