Pr. 1.
Describe how the **eigenvalues** and **eigenvectors** of $\boldsymbol{B} = \boldsymbol{A} - 10\boldsymbol{I}$ relate to the eigenvalues and eigenvectors of $\boldsymbol{A}$.

Pr. 2.
Let $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ be **orthonormal** vectors in $\mathbb{R}^n$. Show that $\boldsymbol{A}\boldsymbol{v}_1, \boldsymbol{A}\boldsymbol{v}_2, \ldots, \boldsymbol{A}\boldsymbol{v}_n$ are also orthonormal if and only if $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix.

Pr. 3.
Let $\boldsymbol{A} \in \mathbb{C}^{N \times N} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$ have **rank** N. Express $\boldsymbol{A}^{-1}$ in terms of the **SVD** of $\boldsymbol{A}$.

Pr. 4.
Let $\boldsymbol{B} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}$ for an invertible matrix $\boldsymbol{T}$. Determine the relationship between eigenvalues and eigenvectors of $\boldsymbol{B}$ and the eigenvalues and eigenvectors of $\boldsymbol{A}$. Explain. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, when thus related, are called **similar**. (Use the convention throughout that eigenvectors are normalized to have unit norm.)

Pr. 5.
Let $\boldsymbol{X} \in \mathbb{F}^{M \times N}$. Using an **SVD** only, show that if $\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{0}$, then $\boldsymbol{X} = \boldsymbol{0}$.

Pr. 6.
The **Frobenius norm** of a $m \times n$ matrix $\boldsymbol{A}$ is defined as $\|\boldsymbol{A}\|_{\mathrm{F}} = \sqrt{\sum_{i,j} |a_{ij}|^2}$. Express the Frobenius norm of $\boldsymbol{A}$ in terms of its **singular values**. Hint: First express $\|\boldsymbol{A}\|_{\mathrm{F}}$ in terms of the matrix $\boldsymbol{A}'\boldsymbol{A}$.

Pr. 7.
Let $\boldsymbol{A}$ be an $m \times n$ matrix. Show that

$$\sigma_1 \leq \|\boldsymbol{A}\|_{\mathrm{F}} \leq \sqrt{\min(m,n)}\,\sigma_1.$$

The **operator norm** of a matrix equals its largest singular value σ_1. The norm of a matrix can be measured many ways. The above inequality shows that if the Frobenius norm is small then the operator norm is as well. However, the operator norm being small does not guarantee that the Frobenius norm will be as well. (Think for example the setting when m and n are very large).
Optional challenge: Is the upper bound **tight**?

Pr. 8.
Let $\boldsymbol{A} \in \mathbb{F}^{M \times N}$.

(a) Suppose $\boldsymbol{W} \in \mathbb{F}^{M \times M}$ and $\boldsymbol{Q} \in \mathbb{F}^{N \times N}$ are each **unitary** matrices.
Show that $\boldsymbol{A}$ and $\boldsymbol{C} \triangleq \boldsymbol{W}\boldsymbol{A}\boldsymbol{Q}$ have the same singular values.
Consequently, $\boldsymbol{A}$ and $\boldsymbol{C}$ have the same **rank**, the same Frobenius norm and the same operator norm. This is why the Frobenius norm and the operator norm are called **unitarily invariant** norms; their value does not change when the matrix is multiplied from the left and/or the right by a unitary matrix. Any norm that depends only on the singular values of $\boldsymbol{A}$ will, by definition, be unitarily invariant.

(b) Suppose that $\boldsymbol{W}$ and $\boldsymbol{Q}$ are **nonsingular** but not necessarily unitary matrices. Do $\boldsymbol{A}$ and $\boldsymbol{D} \triangleq \boldsymbol{W}\boldsymbol{A}\boldsymbol{Q}$ have the same **rank**? Prove or give a counter-example.

(c) Continuing (b), do $\boldsymbol{A}$ and $\boldsymbol{D} = \boldsymbol{W}\boldsymbol{A}\boldsymbol{Q}$ have the same singular values? Prove or give a counter-example.

Pr. 9.
Let $\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$.

(a) Determine the **nullspace** of $\boldsymbol{A}$, denoted by $\mathcal{N}(\boldsymbol{A})$, and the **range space** or column space of $\boldsymbol{A}$ denoted by $\mathcal{R}(\boldsymbol{A})$.

(b) Are they equal? Is this true in general? If not, provide a counterexample.

Pr. 10.

This class uses the notation $\mathbb{F}$ to denote either the **field** $\mathbb{R}$ of real numbers or the **field** $\mathbb{C}$ of complex numbers. The purpose of this problem is to review what a field is. Refer to the appendix of the Ch. 1 notes as needed.

Determine whether the following sets (with the usual senses of multiplication, addition, etc.) are fields or not. If the answer is Yes then you may say so without proof. If the answer is No then give a concrete counter-example for one of the defining properties of a field that is violated.

(a) The set of numbers that are irrational or zero, *i.e.*, the set $(\mathbb{R} - \mathbb{Q}) \cup \{0\}$

(b) The set of $N \times N$ diagonal matrices (where the "1" element is $\boldsymbol{I}_N$, and the "0" element is $\boldsymbol{0}_{N \times N}$.

(c) The set of $N \times N$ diagonal matrices whose diagonal elements are either all zero or all non-zero.

(d) The set of **rational functions**, *i.e.*, functions of the form $P(x)/Q(x)$ where P and Q are both polynomials and Q is not zero.

(e) The set of $N \times N$ invertible matrices.

Pr. 11.

This problem develops a tool that will be used in a later HW for an application called **photometric stereo**.

To approximate the derivatives of a function $f(x)$ that is sampled on the grid $x_1, \ldots, x_n$ where $x_{i+1} = x_i + \delta$, a typical **finite difference** approach is:

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x=x_i} \approx \frac{f(x_{i+1}) - f(x_i)}{\delta}.$$

When the sample spacing is $\delta = 1$, this approximation simplifies to

$$f'(x_i) \triangleq \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_i} \approx f(x_{i+1}) - f(x_i).$$

We can express this relation for all x_i samples via the matrix-vector product

$$\begin{bmatrix} f'(x_1) \\ f'(x_2) \\ \vdots \\ f'(x_n) \end{bmatrix} \approx \boldsymbol{D}_n \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix},$$

where $\boldsymbol{D}_n$ is the so-called first difference matrix defined by

$$\boldsymbol{D}_n = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ 1 & & & & -1 \end{bmatrix}.$$

Here we choose to set $\boldsymbol{D}_n[n,1] = 1$, which corresponds to the (perhaps unexpected) approximation $f'(x_n) \approx f(x_1) - f(x_n)$. This choice is called a **periodic boundary condition** because essentially we are assuming that the domain wraps around. We make this assumption because the resulting $\boldsymbol{D}_n$ is a **circulant matrix** so its eigenvectors can be computed in closed form!

The goal of this problem is for you to derive and implement the analog of $\boldsymbol{D}_n$ for 2D differentiation. Let $f(x,y)$ be a function of two variables. We can approximate its partial derivatives using finite differences as follows:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{(x+1) - x} = f(x+1,y) - f(x,y) \tag{1}$$

$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x,y+1) - f(x,y)}{(y+1) - y} = f(x,y+1) - f(x,y). \tag{2}$$

To simplify notation, define the $m \times n$ matrices `FXY`, `DFDX`, and `DFDY` having elements as follows:

$$\begin{aligned} \texttt{FXY}[i,j] &= f(i,j) \\ \texttt{DFDX}[i,j] &= \frac{\partial f(i,j)}{\partial x} \\ \texttt{DFDY}[i,j] &= \frac{\partial f(i,j)}{\partial y} \end{aligned}$$

The x coordinate is along the column of `FXY` and the y coordinate is along the row of `FXY`, so we can think `FXY[x,y]`. Define corresponding $mn \times 1$ vectors `fxy`, `dfdx`, and `dfdy` to be vectorized versions[1] of `FXY`, `DFDX`, and `DFDY`. With this notation, we can succinctly express equations (1) and (2) as

$$\begin{bmatrix} \texttt{dfdx} \\ \texttt{dfdy} \end{bmatrix} = \boldsymbol{A} \; \texttt{fxy},$$

where $\boldsymbol{A}$ is a $2mn \times mn$ matrix.

(a) Find an expression for $\boldsymbol{A}$ in terms of the first difference matrices $\boldsymbol{D}_n$, $\boldsymbol{D}_m$, appropriately sized identity matrices, and appropriate **Kronecker products** of these matrices. Use periodic boundary conditions.

Hint: Start with $m = n = 3$. Look for ways to use Kronecker product(s).

(b) Once you have determined $\boldsymbol{A}$, write the function `first_diffs_2d_matrix` that takes as input the dimensions m and n of `FXY` and returns the appropriate $\boldsymbol{A}$ matrix, stored in **sparse** format.

In `Julia`, your file should be named `first_diffs_2d_matrix.jl` and should contain the following function:

```julia
"""
`A = first_diffs_2d_matrix(m, n)`

In:
- m and n are positive integers

Out:
- `A` is a `2mn x mn` sparse matrix such that `A * X[:]` computes the
first differences down the columns (along x direction)
and across the (along y direction) of the `m x n` matrix `X`
"""
function first_diffs_2d_matrix(m, n)
```

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

The matrix $\boldsymbol{A}$ can be gigantic. For example, suppose $m = 550$ and $n = 430$, then $\boldsymbol{A}$ is a $473{,}000 \times 236{,}500$ matrix that would require 833 GB of RAM if stored as a full double precision matrix! However, $\boldsymbol{A}$ has only $4mn$ non-zero entries, so it is very **sparse**.
Matrices with many zero-valued elements are quite common in applications. For normal arrays, `Julia` (and other languages) stores zeros in the same way it stores other numeric values, so having many zero elements can use memory space unnecessarily and can sometimes require extra computing time.
Sparse matrices provide an efficient way to store data that has a large percentage of zero elements. While full matrices internally store every element in memory regardless of value, sparse matrices data structures store only the nonzero elements and their row indices. Using sparse matrices can significantly reduce the amount of memory required for data storage.
In `Julia`, to create a sparse matrix somewhat similar to $\boldsymbol{D}_n$ above one can use either the `spdiagm` command:

```julia
using SparseArrays
n = 5
A = spdiagm(0 => 1:n, -1 => ones(n-1))
```

or a loop:

[1] In `Julia`, `fxy = FXY[:]` and `fxy = vec(FXY)`.

```
n = 5
A = spzeros(n,n)
for i=1:n-1
    A[i,i], A[i,i+1] = i, 1
end
```

You should try one or both of these out and modify one of them to prepare your answer. For documentation, see: `https://docs.julialang.org/en/latest/stdlib/SparseArrays`

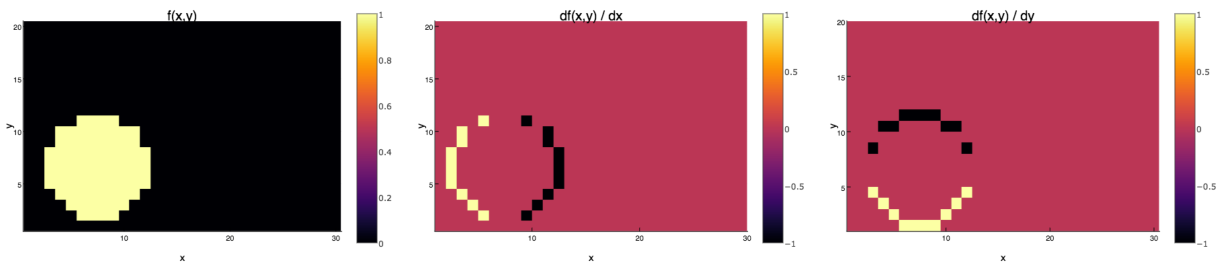
For any `Julia` assignment, you should always try to think of your own ways of testing your function before submitting to the auto-grader. You do get unlimited tries with the auto-grader, but using the feedback from the auto-grader is not the best way to debug. If you design your own test then you can examine the output of it interactively and fix bugs more intelligently.

In this problem your function is designed to compute finite difference approximations to derivatives along x and y. If you create a $m \times n$ array `X` that is, say, a picture of a disk, then the finite derivatives will be mostly zero except near the edges of the disk. (This property is related to an image processing application called edge detection that is discussed in EECS 556.)

Here is `Julia` code for making a (digital/sampled) disk:

```
m = 30; n = 20; X = Float64.([(x-m/4)^2+(y-n/3)^2 < 5^2 for x=1:m, y=1:n])
```

and here are pictures of `FXY DFDX DFDY` for this case.



You should make similar examples to test your function.

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

Pr. 12.
Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$ be an **SVD** of $M \times N$ matrix $\boldsymbol{A}$. We can write

$$\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i',$$

where r is the **rank** of $\boldsymbol{A}$, $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the ith columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively, and σ_i is the (i,i) entry of $\boldsymbol{\Sigma}$.
Recall that the squared **Frobenius norm** of $\boldsymbol{A}$ is $\|\boldsymbol{A}\|_F^2 = \operatorname{trace}(\boldsymbol{A}'\boldsymbol{A}) = \operatorname{trace}(\boldsymbol{A}\boldsymbol{A}')$.
This problem explores some properties of the **singular vectors** of $\boldsymbol{A}$.

(a) (1) $\boldsymbol{A}\boldsymbol{v}_i = ?$
  (2) $(\boldsymbol{A}\boldsymbol{v}_i)'(\boldsymbol{A}\boldsymbol{v}_j) = ?$
  (3) $\boldsymbol{A}'\boldsymbol{u}_i = ?$ when $i \in \{1, \ldots, r\}$?
  What if $i \in \{r+1, \ldots, M\}$?
  (4) $(\boldsymbol{A}'\boldsymbol{u}_i)'(\boldsymbol{A}'\boldsymbol{u}_j) = ?$
  (5) $\|\boldsymbol{A}\boldsymbol{v}_i\|_2^2 = ?$

(b) (1) $\boldsymbol{A}\boldsymbol{v}_i\boldsymbol{v}_i' = ?$
  (2) $\boldsymbol{A}'\boldsymbol{u}_i\boldsymbol{u}_i' = ?$ when$i \in \{1, \ldots, r\}$?
  What if $i \in \{r+1, \ldots, M\}$?
  (3) $\|\boldsymbol{A}\boldsymbol{v}_i\boldsymbol{v}_i'\|_F = ?$
  (4) $\|\boldsymbol{A}'\boldsymbol{u}_i\boldsymbol{u}_i'\|_F = ?$
  (5) $\|\boldsymbol{A}\boldsymbol{v}_i\boldsymbol{u}_j'\|_F = ?$
  (6) $\|\boldsymbol{A}\boldsymbol{v}_i\boldsymbol{v}_j'\|_F = ?$
  (7) $\boldsymbol{A}\boldsymbol{A}' = ?$
  (8) $\boldsymbol{A}'\boldsymbol{A} = ?$
  (9) $\|\boldsymbol{A}'\boldsymbol{A}\|_F = ?$
  (10) $\|\boldsymbol{A}\boldsymbol{A}'\|_F =$

(c) (1) For $1 \le k \le r$, $\boldsymbol{U}[:, 1{:}k]'\boldsymbol{A} = ?$
  (2) For $1 \le k \le r$, $\boldsymbol{A}\boldsymbol{V}[:, 1{:}k] = ?$
  (3) For $1 \le k \le M$, $\boldsymbol{U}[:, 1{:}k]\boldsymbol{U}[:, 1{:}k]'\boldsymbol{A} = ?$
  (4) For $1 \le k \le N$, $\boldsymbol{A}\boldsymbol{V}[:, 1{:}k]\boldsymbol{V}[:, 1{:}k]' = ?$
  (5) For $1 \le k \le r$, $\boldsymbol{U}[:, 1{:}k]'\boldsymbol{A}\boldsymbol{V}[:, 1{:}k] = ?$