Math 110 - Application Project

Crowdmark submissions due by 11:59pm on Friday, December 3

Optional group work

You may choose whether to complete this project alone or in a group. If you choose to work in a group, your group must have **no more than four** members. Your group may include members from any combination of lecture sections. You must form the group in Crowdmark (instructions are included in the email you received from Crowdmark), so that all of the group members Crowdmark accounts will be attached to the submission. Your group then only needs to upload one set of solutions. All members of the group will receive the same grade on the project. The project is designed to be reasonable for an individual to complete alone. Working in a group is permitted, but not required for success.

Part I: Applications

Part I of the project is divided into two options. You should submit answers to **only one of the two options for Part I** - it is up to you which option you would prefer to work on.

The two options in Part I concern two different types of applications of the material from this course to problems that arise in the real world. Option 1 is about heat transfer. Option 2 is about image processing. Information about what to submit to each question number in each option is provided on the next pages of this document.

Part II: Cautionary tales

Part II of the project explores some examples where MATLAB produces incorrect results, as a reminder that whenever you use software like MATLAB you need to be very cautious to make sure that the results it gives you make sense. **Complete all of Part II**. Information about what to submit to each Crowdmark question for Part II is included on later pages of this document.

Give yourself enough time to submit

You have more than two full weeks to complete this project. It is due at 11:59pm on Friday, December 3, but please do not wait until the last moment to submit it. You have plenty of time to submit it before the deadline, so submissions that are late, even by just a little bit, will not be accepted.

Read the instructions and explain your work

Each question below has a label telling you which Crowdmark question to submit your answer to. Please read the questions carefully and make sure you are uploading to the correct part of Crowdmark.

Don't forget that in this course we are **always** more interested in your explanations than the numerical answers! Explanations don't necessarily need to be long, but you should write enough that someone who wasn't there when you did the work can understand how you arrived at your answer.

Part I Option 1: Application to heat transfer

This project option is about applications of linear algebra to physical systems, and specifically the transfer of heat.

Suppose that we are given a thin square plate, and we know the temperature at the edges of the plate, and that the plate is at equilibrium (that is, the temperature at each point of the plate has become constant). Assuming appropriately idealized physical conditions, the temperature of any particular point on the plate will be the average of the temperatures of nearby points. While the mathematics involved in finding exact temperature values is beyond the scope of this course, we can use the tools of linear algebra to find *approximate* temperature values.

The key idea is that we will divide the square plate into regions, as in the image below. The temperature of any region will then be the average of the temperatures of the adjacent regions. To simplify matters even further, we will only consider two regions to be "adjacent" if they share an edge of the grid (so regions that are diagonal to one another do not count as adjacent). **Note**: In these approximations we are only assigning temperature values to regions; we assume that all points within a given region have the same temperature.

For this exercise, we will consider a plate where along the left edge the temperature is 100, along the right edge it is 0, along the bottom edge it is 80, and along the top edge it is 40. The top and bottom left corners are 100 and the top and bottom right corners are 0. Here is what the grid then looks like when divided into a 4×4 grid:

100	40	40	0
100	x	y	0
100	z	w	0
100	80	80	0

In the image we have labelled each region with a temperature; x, y, z, w represent unknown temperatures. The marked point will be used in a later part of this question. Whenever we refer to the marked point, you should think of it as being just slightly up and to the right of the centre of the plate (but remember that we only assign temperatures to squares in the grid, not to individual points).

(P1O1Q1): For each of the interior regions of the diagram above, write an equation that expresses the fact that the temperature at that region is the average of the temperatures of the four adjacent regions. Pick one of your equations and write a brief explanation of how you came up with it.

(P1O1Q2): Use MATLAB to solve the system of equations that you wrote in part (a). In your submission, include both the MATLAB code that you wrote and the output that it produced (the easiest way to do this is probably to simply take a screenshot of your MATLAB window). Use the result you got from MATLAB to estimate the temperature at the marked point, and explain how you got your estimate from the MATLAB output.

(P1O1Q3): Draw a picture of the plate divided into a 6×6 grid. Repeat P1O1Q1 for this new grid.

(P1O1Q4): Repeat P1O1Q2 for the 6×6 grid. The marked point should still be just slightly up and to the right of the centre of the plate.

(P1O1Q5): You now have two estimates for the temperature at the marked point. Which do you think is more accurate? Why?

Part I Option 2: Application to computer images

The two main ways of storing images digitally are *vector images* and *raster images*. This application will focus on raster images.

In a *raster image* the image is stored as a grid of pixels. Each pixel is represented by three numbers, corresponding to the intensity of Red, Green, and Blue needed to colour that pixel. Raster images are the kind of computer image you are likely most familiar with; common formats include BMP, JPG, and PNG. The goal of this project is to use linear algebra to implement a simplified version of an image compression scheme for raster images.

Our main simplification is to consider only greyscale images. In this case each pixel only needs to be represented by a single number, representing the intensity of black at that pixel. In the case of an image measuring $n \times m$ pixels, we can encode the image as an $n \times m$ matrix.

The main tool that we will use is the *Singular Value Decomposition*. The information you need about it is provided here, but you can also find more information online if you want to learn more. The key fact is that any matrix (even one that isn't square!) can be expressed in the form $A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$, with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, for some vectors $\mathbf{u}_i, \mathbf{v}_i$. MATLAB can calculate these values; specifically, the command $[\mathtt{U}, \mathtt{S}, \mathtt{V}] = \mathtt{svd}(\mathtt{A})$ produces a matrix U whose columns are the $\mathbf{u}$-vectors, a matrix V whose columns are the $\mathbf{v}$-vectors, and a matrix S whose diagonal entries are the σs.

During these exercises you will also need to know how to extract just some rows or columns from a matrix. If A is a matrix in MATLAB, then $\mathtt{B} = \mathtt{A}(2:4, 3:7)$ produces a matrix B whose entries are those in rows 2 through 4, columns 3 through 7 of A. To get columns 3 through 7 (all rows), use $\mathtt{B} = \mathtt{A}(:, 3:7)$. To get rows 2 through 9 (all columns), use $\mathtt{B} = \mathtt{A}(2:9, :)$.

(P1O2Q1): Let $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$. The singular value decomposition of A has the form $A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \sigma_3 \mathbf{u}_3 \mathbf{v}_3^T$. Use MATLAB to find $\sigma_1, \sigma_2, \sigma_3, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$. Your submission should include the MATLAB code that you entered, as well as the output (probably the easiest way to do this is by taking a screenshot of your MATLAB window). The MATLAB code isn't the whole answer, though! You need to tell us the values of $\sigma_1, \sigma_2, \sigma_3, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ based on that output.

(P1O2Q2): Using the same matrix A as in P1O2Q2, use the MATLAB commands to extract the appropriate rows and columns of U, S, and V to calculate $\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T$.
Note: If the result you get isn't fairly close to A, then you've selected incorrect rows/columns. Be sure to get this part correct before moving on!

(P1O2Q3): Choose an image in jpg format. Read the image into MATLAB in greyscale by using the command $\mathtt{A} = \mathtt{double}(\mathtt{rgb2gray}(\mathtt{imread}(\text{`}\mathtt{path}'\text{)}))$;, where 'path' is the path to your image. Write this greyscale version of your image to a new jpg file using $\mathtt{imwrite}(\mathtt{uint8}(\mathtt{A}), '\mathtt{path}')$. Your submission for this question should include the original image, the greyscale image, and the file size of the greyscale image.

(P1O2Q4): Let B be the matrix obtained from the matrix of your image by keeping only the first 50 terms of the singular value decomposition. Use MATLAB to create a jpg file containing the image coded by B. Your submission for this question should include the MATLAB code you used, the image output, and the file size of the new image.

(P1O2Q5): Experiment with the number of terms of the singular value decomposition that you keep until you find a number of terms which produces an image that looks fairly close to the original, while still reducing the file size as much as possible ("fairly close" is intentionally vague - we leave it up to you to decide what is close enough). Your solution to this question should have the same information as the solution to the previous question, but for the new number of terms, and you should also tell us how many terms you kept.

Part II: Cautionary Tales

In Part I you saw some applications of linear algebra to real-world problems, and also saw that MATLAB can very quickly handle the kinds of computations that arise in solving those problems. As useful as MATLAB (or other computer algebra software) is, it must always be used with some caution. In this part of the project you will see some examples where MATLAB produces incorrect results. The moral here is that whenever you use MATLAB, you must check whether the output you are getting is reasonable!

The questions P2Q1, P2Q2, P2Q3 are related to each other. The question P2Q4 is not related to the earlier ones.

(P2Q1): Run the following sequence of commands, in order, in one MATLAB session. Your submission for this question should just be a screenshot of the MATLAB window where you ran the commands.

```
rng shuffle
A = 30000 * rand(2, 4)
B = A' * A
det(B)
rref(B)
```

What you have done is created a random 2×4 matrix A with entries between 0 and 30000, set $B = A^T A$, and then had MATLAB compute both the determinant and the RREF of B.

(P2Q2) Based on the RREF given by MATLAB, what is the determinant of B? Briefly explain your answer.

(P2Q3) Your answer to P2Q2 will be different than the determinant MATLAB produced from the `det` command. What is the correct determinant of B? Justify your answer by making reference to the theory we learned in class.
Note 1: Keep in mind that you **know** MATLAB has done something incorrect here, so your answer to this part should **not** make any reference to the MATLAB output.
Note 2: Please do *not* calculate the determinant of B by hand. It is possible to answer this question without knowing which random matrix A was generated.

(P2Q4) Let $C = \begin{bmatrix} 51 & 52 \\ 100 & 101 \end{bmatrix}$. Run the MATLAB command `eig(C^10)` to have MATLAB compute the eigenvalues of C^{10}. The answer you get will be incorrect. Use the theory we developed in class to show that MATLAB has given you the wrong answer (you don't necessarily need to find the correct answer, but you could if you wish). Your solution to this question should include your MATLAB code and output, as well as your explanation of how you know that the MATLAB output is wrong.