MAT 167 Final Project
Summer 2025

Due 8/1/25

1 Instructions

Form a group of two or three. Read the two options below and choose one as your project focus. You must submit your group and your topic by Monday July 14. The purpose of this project is to allow you to use the linear algebra knowledge you have gained in this class in projects that are relevant in today's society.

Due Date: Please have your final report upload to Gradescope by 10 am on Friday, August 1. We will present the posters in class on Friday, August 1.

Project Requirements: Your completed project should include:

1. A poster board sized 22" x 28" (size may vary slightly but this should be the approximate size) including the following items:
   (a) The title to your project
   (b) Your group members' names
   (c) A short explanation of the idea of the solution and the key concepts from linear algebra used in solving the problem
   (d) Important figures to the project

   The poster should be organized in a logical manner. You should be able to give a brief overview in 2-5 minutes of your presentation to any onlooker and answer questions about your project.
2. A report detailing:
   (a) An introduction detailing what your project is about
   (b) A section describing all of the code that you wrote
   (c) A section (or two) answering questions about your project detailed below
   (d) Any necessary figures
   (e) A conclusion summarizing your results
   (f) References
3. `MATLAB` code that details the solution to the problem. I should be able to run your submitted code.
4. A survey detailing what work you and your fellow project members contributed to the project. Multiple group members should contribute to each task. This will address the following:

(a) Writing the code

(b) Writing the report

(c) Preparation of tables/figures/graphics

(d) Assembly of the presentation

(e) Any other items relevant to the project completion

Grading Criteria: Points will be awarded accordingly:

1. Report Content and Quality (50 points). The mathematics should be correct. You should explain what questions you are answering.

2. Report Grammar, Clarity, and Organization (20 points). Your report should be readable, meaning it includes minimal grammatical errors and be organized in a way that makes sense when reading.

3. Code is Executable (20 points). Your instructor should be able to run your code without any alterations. You will need to include instructions for any files that need to be in the working directory to run your code file inside your report.

4. Poster Content and Quality (40 points). Your poster should be visually appealing and easy to read. It should include all relevant figures and be organized in a logical manner.

5. Poster Presentation (20 points). I can follow and understand the mathematics being described. You can answer reasonable questions about your poster when asked. All group members must be present for this presentation.

2 Project 1: PCA and Shape Mode Analysis

Principal component analysis is used to quantify biological organisms' shape changes and dynamics through what is called shape mode analysis. We can apply principal component analysis to these biological data sets to analyze and quantify patterns in movement in 2D image data sets. You will be assigned a data set of images of flagella from *Chlamydomonas reinhardtii*, which is a type of algae. Your job is to analyze the data in the following manner. This project is adapted from the paper by Werner: https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0113083&type=printable and will be a helpful resource when completing this project.

We are going to reproduce some of the results from this paper and learn how to use principal component analysis to understand flagellar shapes.

1. In MATLAB, create several plots to showcase the data set which you are given. This could include looking at all of the data all at once, looking at the first few time steps of data, or pinning the data to the origin and looking at the data there. Also, decide how many time points and spatial points are in your data. Inside your data set, there should be a space scale and a time scale. These tell you the dimensional spacing between data points in both space and time.

2. Create a matrix of all of the data using the tangent angle ϕ. Recall that $\phi = \arctan\left(\frac{y}{x}\right)$. You will need to use the function `unwrap` and `atan2` in MATLAB to avoid jumps of π in data, due to the range of arctangent. Then, using this matrix, recreate Figure 2B from the Werner paper using your own data, which is a kymograph of the tangent angles. Use the command `pcolor`. What does this figure show?

3. We are going to analyze the data slightly different than in the Werner paper. They use eigenvalue decomposition of the covariance matrix in that paper. We will instead look at SVD of our tangent angle matrix. After you demean the matrix (subtract spatial mean), perform SVD on the matrix. Report out the following things from your SVD computation and give a short description of each figure.

   - A graph of the first 4 singular vectors which describe the shape. We call these the shape modes.
   - A graph of the first 4 singular vectors which relate to the beat of the flagella across time. We call these the temporal modes.
   - A graph of the "strength" of the singular vectors via the singular values, similar to Figure 2D from the Werner paper. Use the MATLAB function `cumsum`.

4. For this step we will just consider the first two dimensions of shape space. Denote $\mathbf{V}_i$ as what Werner calls the "shape score", which is really just the ith singular value times the ith temporal mode. Plot $\mathbf{V}_1$ against $\mathbf{V}_2$ using dots. You should see that these data points form a closed loop, like in Figure 2F in the Werner paper. We want to fit a low dimensional Fourier series

   $$A\cos(\theta) + B\sin(\theta)$$

   to both V_1 and V_2 to describe this loop. We will do this as follows:

   - Compute the angle in the plane with the loop between $\mathbf{V}_1$ and $\mathbf{V}_2$. If you consider $x = \mathbf{V}_1$ and $y = \mathbf{V}_2$, you are essentially computing the polar angle θ. You can compute this as `theta = unwrap(atan2(V(:,2),V(:,1)))`.

- Now use least squares to fit
$$\mathbf{V}_i = A_i \cos(\theta) + B_i \sin(\theta)$$
using the θ from the previous step. You are solving for A_i and B_i in your least squares, and you should do this procedure once for $\mathbf{V}_1$ and once for $\mathbf{V}_2$.
- Now you have an approximation of your $\mathbf{V}_i$'s from least squares. Plot that on top of the data points from earlier in this section.

5. Finally, let's build a lower dimensional reconstruction of the data. Consider the reconstruction as follows:
$$\phi = \phi_0 + \sum_{i=1}^{2} \mathbf{V}_i \mathbf{S}_i$$
where ϕ_0 is the spatial mean, $\mathbf{V}_i$ is the least squares fit from part 4, and $\mathbf{S}_i$ are the first two shape modes.
   - To form the positions X and Y from this reconstruction, at each point in time $\mathbf{x}_j$, compute
$$\begin{bmatrix} x_j \\ y_j \end{bmatrix} = \Delta s \sum_{i=1}^{j} \begin{bmatrix} \cos(\phi_i) \\ \sin(\phi_i) \end{bmatrix}.$$
   - Plot and compare the first few strokes from the reconstruction to those of the original data. Try to plot approximately the strokes across one period.

6. Respond fully and in complete thoughts to the following questions:
   - What do you observe from your least squares fit to the $\mathbf{V}_i$'s? What differences do you notice between the original data points and the least squares fit? How could you improve your least squares fit?
   - What differences do you notice between your original data and the reconstruction?
   - How could you improve your reconstruction?
   - What purposes do you think the decomposition of the data and then the subsequent reconstruction might serve in the real world?

3 Project 2: Image Processing

When we pass information from one computer to another, it would be ideal for us to pass that information in its entirety. However, it is often the case that this data packet is so large that this process would either take an impractically long time or large amount of storage to accomplish. Therefore, we want to find a way to pass as much of the "essence" of the data as we can without having to pass the entire physical amount of data. This is the idea of compression. We will create an elementary compression algorithm in this project.

MATLAB has the ability to display both JPEG and PNG files. When they are imported to MATLAB, these pictures are stored in three $m \times n$ `uint8` variables, where $m \times n$ is the resolution, or pixel dimensions, of the file. Each of these matrices corresponds to how much of each color red, blue, and green, respectively, are used in generating each pixel. So for any file M, $M(:, :, 1)$ is a $m \times n$ matrix with `unit8` entries corresponding to the amount of red (a value from 0 - 255) in each pixel. The green and blue amounts come from replacing the '1' with a '2' or a '3' respectively.

1. Use the MATLAB function `imshow()`, create a function `DisplayPicture()` that takes in an $m \times n \times 3$ matrix of `double` entries and produces a figure displaying that picture. (Don't forget to scale the pixel values by 255 when using `imshow()`.

   Now that we can view the picture files in MATLAB, we want to find a way to compress them.

2. SVD can be used to reconstruct most of the picture from only a small amount of the singular vectors. If we think of the singular values as the "total amount" of picture to be reconstructed, outline an algorithm explaining how to reconstruct $X\%$ of the picture from the singular vectors.

3. Use the MATLAB function `cumsum()` to construct a function `NumSingVecs()` that takes in an $m \times n \times 3$ representing an image and the percentage X of the picture you would like to construct. Have it return a three-dimensional vector `[kRed, kGreen, kBlue]`, where each component is the number of singular vectors needed to reach $X\%$ of that color from the original image. In addition, have this function plot "progress curves", where the x-axis is the number of singular vectors and the y-axis shows the % of the picture those singular vectors represent. Plot one progress curve for each pixel color red, blue, and green. Give a short description what you observe from these figures.

4. Now create a function `CompressPicture()` that takes in an $m \times n \times 3$ matrix and a percentage X of the original picture that you would like to recreate. This function should return an $m \times n \times 3$ tensor of `double` entries representing the compressed image and should also produce a figure displaying the compressed image.

5. Consider the pictures in the Gallery section inside the Projects folder on Canvas. Your job is to rank these pictures, based on their compressibility. It is up to you and your group to define what "compressibility" means. Be sure to explain your criteria carefully and then present your ranking of the pictures from "most compressible" to "least compressible". Include any details and/or figures about your choices when describing this in your report.

6. Respond fully and in complete thoughts to the following questions:

   - What makes an image "easy" or "hard" to compress?

- Go online and find different (but appropriate) images giving at least two examples of an "easily compressed" and an "compression resistant" image.
- On average, what proportion of the singular vectors are required in order to construct an image that you can recognize as the original image? How does this change depending on the ease of compressibility which you have outlined in the previous questions?
- Which features does your algorithm capture well and which features are harder to detect? Make suggestions for how it could be improved, making sure to base your suggestions with mathematical justification.