Linear Algebra
Fall 21
Homework Assignments

Each assignment is due by the end of the day. Upload the theoretical and applied assignments to your folder in Google Drive. Explain your solutions as much as you can. Please work with and discuss these assignments with your professor and classmates, but your submitted written work should be your own.

WeBWorK #1 due Friday September 3.

Theoretical homework #1 due Friday September 10:
Explain all your solutions.

- Section 1.1 Exercise D3
- Section 1.2 Exercises D3, D9, D10
- Section 1.3 Exercises 16, D2, D3, D4
- Section 2.1 Exercises D4, D8, D9

WeBWorK #2 due Monday September 13.

Theoretical homework #2 due Friday September 17:
Explain all your solutions.

- Section 2.2 8, D1, D3, D5, D7, P1 (notice there's a part b)
- Section 3.1 D2, D4, D6, D8, D9cd

Applied homework #1 due Wednesday September 22: Splines!

1. Read Python Numerical Methods section 17.3 to learn about cubic splines.
2. Find a cubic spline through the points $(1, 2)$, $(3, 4)$, $(5, -1)$, and $(7, 0)$. Use the standard requirement that the second derivative of the functions at the first and last points should be 0.
3. Write a Python program to find a cubic spline through any four points (a, b), (c, d), (i, j), and (p, q) in the plane. Again use the requirement that the second derivative of the functions at the first and last points should be 0. Don't use the built in `CubicSpline` function, generate your equations and use Python to solve the corresponding system of linear equations.

   Output the formulas for the cubic functions, and also output the graph of your cubic spline together with the four points.

   Test your program on the points in part 2 to check that it works, and show your output.
4. Modify your program to make the first derivative, not the second derivative, of the functions at the first and last points 0. Write a sentence about the qualitative difference between the graphs you got in this part and in part 3.

WeBWorK #3 due Friday September 24.

Theoretical homework #3 due Wednesday September 29:

Explain all your solutions.

- Section 3.2 D4, D6, D7, D8
- Section 3.3 D2, D5ce, D7, P4
- Section 3.4 D3, D4

Applied homework #2 due Monday October 4: Markov chains and predictive text!

1. Use Python to complete problem 5.1: T2. Report your calculations from Python and explain what they mean.
2. Open the Python notebook *Markov chain predictive text* in the Wise resources folder. Use your own text file of at least 100 words to generate some lines of predictive text. Share your favorites, and then explain how the code works.
3. Find the steady-state vector for your transition matrix from part 2, by approximation or direct calculation.
4. Pick five words from your text that appear frequently. Using your transition matrix from part 2, draw a state diagram of these five words / similar to the diagrams in Figures 5.1.1 and 5.1.3. You can leave out arrows with probability 0.
5. *Extra credit:* Modify the code in the *Markov chain predictive text* notebook to generate each word based on the previous two words, instead of the previous one word. Again generate some lines of predictive text and share your favorites, along with your modified code.

WeBWorK #4 due Wednesday October 6.

WeBWorK #5 due Wednesday October 13.

Midterm due Wednesday October 20.

Applied homework #3 due Wednesday October 27: Error correcting codes!

1. Read https://math.ryerson.ca/ danziger/professor/MTH108/Handouts/codes.pdf to learn about error correcting codes. In this document, the author uses the notation $\mathbb{F}_2$, which is the numbers 0 and 1 in binary.
2. Complete exercise 3 in the reading.
3. Using the tools in the Python notebook *Error correcting codes*, implement the $(4, 3, 3)$ Hamming code described in part 1 in Python. Encode a message of between 10 and 20 characters, and use syndrome decoding to decode it.
4. Any matrix of 0s and 1s with linearly independent columns is the generator matrix of a linear code. Make your own generator matrix, find the minimum distance d and the information rate R like you did in part 2, and compare your d and R with the $d = 3$, $R = 4/7$ of the $(4, 3, 3)$ Hamming code given in the reading. (You probably won't do as well – it's hard to find good codes!) Then encode a message with your matrix like you did in part 3.

WeBWorK #6 due Monday November 1.

Theoretical homework #4 due Wednesday November 3:
Explain all your solutions.

- Section 6.1 36, D4
- Section 6.2 D6
- Section 6.3 P1
- Section 7.1 6
- Section 7.2 18, P2
- Section 7.3 26, D3
- Section 7.4 22, D10

Applied homework #4 due Monday November 8: Graphics using homogeneous coordinates and homographies!

1. Complete problems 6.5: 26, 28. Using the Python notebook *Image transformations*, use the matrix you found in problem 26 to transform an image in Python. Depending on the size of your image, you probably want to increase the size of the translation to see the effect.
2. Take or find an image taken from a slanted angle, and use the tools in the Python notebook *Image transformations* to straighten it out so it looks like it's taken head on. This process is called *perspective rectification*. For more examples of this in action, see `https://inst.eecs.berkeley.edu/~cs194-26/fa17/upload/files/proj6A/cs194-26-adp/` or pages 17-22 of `http://people.scs.carleton.ca/~roth/comp4900d-12/notes/homography.pdf`.
3. Experiment with different homogeneous transformations AKA homographies using the Python command transform.warp, and share a couple of your favorites.

WeBWorK #7 due Monday November 15.

Theoretical homework #5 due Wednesday November 17:
Explain all your solutions.

- Section 4.1 D1, D2, D3, D5, P1
- Section 4.2 D4, D9
- Section 4.4 14, P1, P2, P3

WeBWorK #8 due Wednesday December 1.

Applied homework #5 due Friday December 3: Singular value decomposition and image processing!

1. Complete problems 8.6: 10, 16.
2. Take or find an image, and using the tools in the Python notebook *Singular value decomposition*, find the rank k approximation of the matrix of your image for a few different values of k.
3. Suppose that the amount of memory taken by a matrix is the number of entries of the matrix. When the computer saves the original image, it saves the whole original matrix A, but when it saves the SVD it saves the matrices U and V and the list of singular values along the diagonal of Σ. Compare the size of your original image file, the size of the full singular value decomposition, and the size of the approximations you found in part 2. How much space did you save? Find a value of k that's a good balance between file size and image quality.
4. The rank k approximation uses the first k columns of the matrix U, the first k rows of the matrix V, and the first k singular values along the diagonal of the matrix Σ. Try approximating by taking a different set of rows and columns, for example, skipping the first 3 columns of U, rows of V, and diagonal values of Σ. How much worse does the image quality get? This gives you an idea of the relative importance of the few largest eigenvectors in representing your image.