340f24 PA1

Due date: Check the schedule table or eclass
Submission: A single **zip file** must be submitted through eClass. Use filename in the format: **`pa1-[firstname]-[lastname].zip`**.
Total: 100 (50 from code submission + 50 from lab demo)

Submission details:
You are expected to submit a **Matlab function/script for each portion of the PA (programming assignment)**; make sure your functions and demo script are easy to both demo and understand.

Your code should run on any machine without needing to be modified by the examining TA. You can test this by unzipping your solution to a new folder and running it from the new location.

In addition to the code, you should also submit written documentation as a **pdf file** to address the non-programming questions that appear throughout the PA description (Q2D this time).

Finally, we intend to browse your code before the demo. So, for consistency, we ask that you name your files something along the lines of **pa1_1A.m, pa1_1B.m,...,** so we can easily identify the question you are answering. For files that contain a specific function, like Q2A, the file name should match the name of the function, for example, **elimMat.m**. The written documentation should be in a file called **writeup.pdf.**

Demo details:
During the demo, you will demonstrate your solution, explain your code and strategy, and answer questions from the examining TA, who will ensure that you did the work yourself and that you understand the material relevant to the assignment. The questions can be considered a small, informal oral quiz where the examining TA draws a few questions from a prepared list. They will test your knowledge of the assignment and related theories. The target meantime for a demo is about 5min (+/-2), so make sure your functions and demo script are prepared to be easy to both demo and understand.

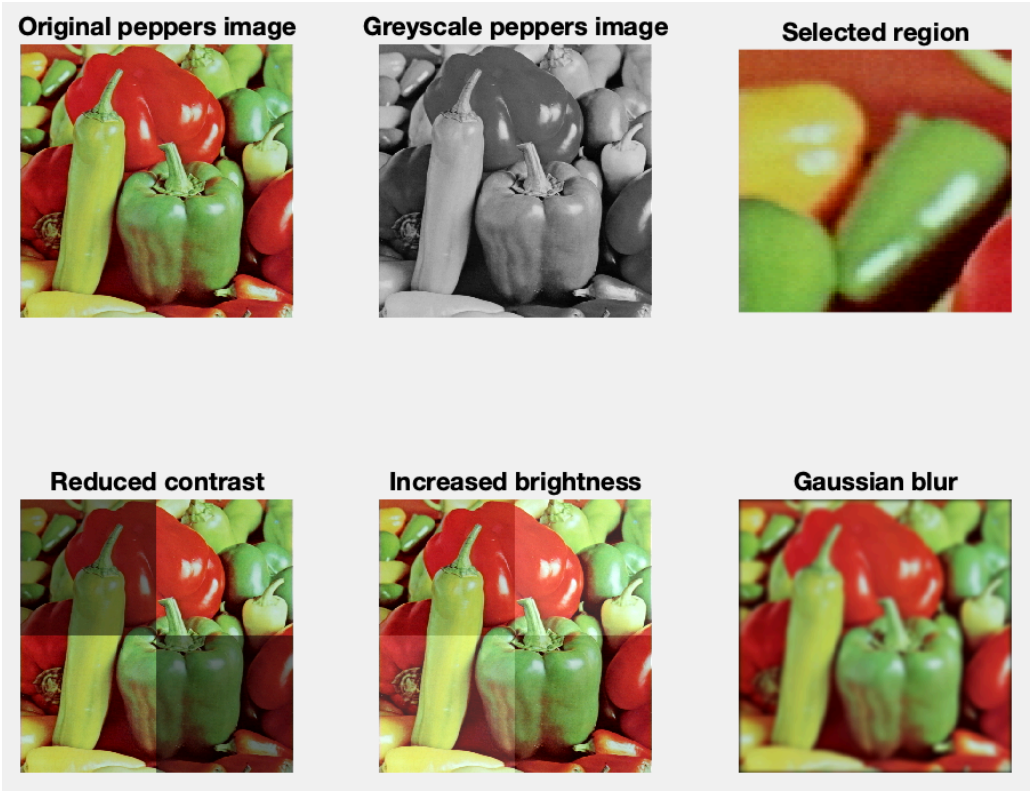
Q1: Intro to Matlab (20)

Objectives

The objective of the first question is to acquire basic Matlab literacy, in particular, familiarity with vector and matrix operations.

Deliverables

A. Vectorize this script and compare the running time of both implementations. Write your comparison as matlab comments (10)
B. Fill in this script details to perform the required image modifications as below. (10)
You **should not** use the `imcrop` function of Matlab. Display all images in one window like the below figure.



Q2: LU factorization (80)

Objectives

1. To practically implement the routines involved in LU factorization and thereby get hands-on knowledge of the numerics of factorization.
2. To write numerical routines in Matlab, particularly to access sub-parts of vectors and matrices.
3. To write clean, correct vectorized code.

Deliverables

The goal is to compose a linear equation solver from the following parts:

A. Write a function `[M_k, L_k] = elimMat(A, k)` that, given matrix `A` and index `k`, computes the elimination matrix `M_k` and `L_k` as specified on page 67 of the textbook. You can assume the matrix `A` is non-singular. (10)

B. Write a function `[L, U] = myLU(A)` that reuses `elimMat` to give an LU factorization of `A`. (20)

C. Write a function `x = backSubst(U, y, k)` that performs back-substitution to solve Ux = y either recursively or using one for loop. (20)
A function for forward-substitution solving Ly = b recursively is provided here.

D. In your written documentation, explain how to use your programs to solve Ax = b using methods described on page 68 of the textbook (10):
   a. Factorization: LU = A
   b. fwdSubst to solve for y in Ly = b
   c. backSubst to solve for x in Ux = y.

E. Test your program with, e.g., example 2.13 and one test of your own design. Write an m-file script to execute the test. (20)

Coding Style to Follow

Show how to use matlab matrix operations to write the above in a clear, compact form with vectorization and a minimum number of for loops (e.g., copying the for-loop-based algorithms 2.2 and 2.3 from the book will not yield good marks). Write your code in a readable, commented, and suitably indented fashion.

Vectorization Requirements

The following table gives the restrictions on the number of loops (for & while) for the deliverable functions:

Function	**Maximum** number of loops
`elimMat(A, k)`	0
`myLU(A)`	1
`backSubst(U, y, k)`	1

Hints

First, understand and experiment with the supplied algorithm for forward-substitution alone. Modify it to do back-substitution. Then write the elimMat and myLU routines. myLU can be written recursively much the same way as the fwdSubst, or with one "for" loop. You need no loop in computing the elimination matrix and, at most, one loop (or recursion) in computing the LU, and forward and back-substitutions. Achieving this depends mostly on skills in linear algebra to do operations on vectors and matrices as a whole instead of element by element and on understanding Matlab's matrix elements addressing as described in the first part of the tutorial. This way, each function (elim matrix, lu, and substitutions) can be written in just 5-10 lines.

Sample demo questions

- What makes vectorized code faster?
- If the loop in myLU goes till n, what would happen if you change it till n-1? If it goes till n-1, why does the loop go till n-1 and not n?
- In the forward substitution function recursive call, explain why we are passing `b-y*l`.
- For the back substitution function, why is the condition `k > 1` used before the recursive call?