**Exercise 1: Fade two images and merged them.**

**Version 1:**

You are going to use Cameraman\_image already loaded in the Part 1.

Load the second image ‘5.1.09.tiff’ which is a surface in the moon. Assign this image to the variable Moon\_image.

Show Moon\_image

Figure (3)

imshow(Moon\_image);

Fade for example Moon\_image by a factor of 0.8 and Cameraman\_image by factor of 0.2 and sum the two matrices in new matrix named Mixte\_image.

Show the mixte\_image

Figure (4)

imshow(Mixte\_image);

Create a new image named First\_part\_image\_1 by selecting the first 100 components of each dimension of image Cameraman\_image.

Create a new image named last\_part\_image\_2 by selecting the last 100 components of each dimension of image Moon\_image.

Fade both images last\_part\_image\_2 by a factor of 0.8 and First\_part\_image\_1 by factor of 0.2 and sum the two matrices in new matrix named last\_part\_Mixte\_image.

**Version 2:**

We will do the same fade and mixing process as version 1 question but this time using matrix and vector multiplication.

The first step will consist of changing both image matrices from size of 256x256 each to vector of size 56536x1; (To create a vector from matrix uses the function reshape?. Use the help to show you how to use reshape function?

Create a new matrix (named Both\_images) by appending both image vectors to form a matrix of size 56536x2.

Create of vector named Fade\_vector of size 2x1 and containing the fade factor values (0.5,0.5) for both image.

Multiply Both\_images matrix and Fade\_vector to obtained the mixing\_image\_vector

Resize using the function reshape again the obtained vector mixing\_image\_vector to create mixing\_image\_mtarix of size (256x256)?

Please see the Figure 1 to show all the steps to solve this section?



Figure 1: mixed to Fade images using matrix and vector multiplication

40000 x 1

56536 x 2

56536 x 1

2 x 1

=

X

**Exercise 2:**

In this project, you will create a demo for Newton’s method graphically. Specifically, you need to create a video of how Newton’s method progresses with iterations. Every step and commands to use is mentioned in comments in Lab2\_NewtonMethod.m file.

The polynomial function is given by

𝑓 (𝑥) = 𝑥3 − 3𝑥 + 1

The polynomial and its derivative should be hardcoded in the files poly.m and poly\_derivative.m. You can load the polynomial using f = @poly. Value of the function at any value, say 0, can be found using f(0). You can load the derivative using fder =@poly\_derivative. Value of the derivative at any value, say -1, can be found using fder(-1)

Test your code(demo) with different initial points.

Note: You are not limited to use same mentioned commands.

For Python:

You can use numpy library.

numpy.polynomial.polynomial.polyval can be used to calc the function at any value.

numpy.polynomial.polynomial.polyder can be used to calc the derivative at any value.

You can plot the animation by matplotlib (examples [here](https://stackoverflow.com/questions/25422073/how-to-animate-matplotlib-function-optimization)) or plotly.