Math 495R Homework 24

In this lab you will perform image compression using singular value decomposition. You will need to import the following:

```
>>> import numpy as NPY
>>> import numpy.linalg as LA
>>> import imageio as IO
>>> import matplotlib.pyplot as PLT
>>> import scipy.misc
```

For this lab you may use an image from the scipy.misc module, which can be accessed by

```
>>> F = scipy.misc.face()
```

You may also use your own `.png` picture if you want. To do so, place it in the same the directory as your Jupyter notebook and read it in using the command

```
>>> F = IO.imread('FILENAME.png')
```

To display the image, use the commands

```
>>> PLT.imshow(F)
>>> PLT.show()
```

This command can only show one image or plot at a time. If you would like to save the image to the directory use the command

```
>>> IO.imwrite('FILENAME.png',F)
```

The command `F.shape` shows that the image is stored as a numpy array of dimensions $m \times n \times 3$. Each entry represents an RGB component of a pixel at given location. Use

```
>>> G=F[:,:,0]
```

to select just the first component per pixel, making G a two dimensional $m \times n$ array. You can use the same commands as above (replacing `F` with `G`) to display `G` or write it to a file. The new image will be a grayscale version of the original. Since `G` is a two-dimensional array we can treat it as a matrix and perform a singular value decomposition:

```
>>> U,S,VT = LA.svd(G)
```

The matrix `U` will be an $m \times m$ matrix with orthonormal columns, and `VT` will be an $n \times n$ matrix with orthonormal columns. `S` will be a list of the singular values of G. The `NPY.diag` function will turn a list into a diagonal matrix. If r is the length of S, the command

```
>>> Gtest = U[:,:r].dot(NPY.diag(S)).dot(VT[:r])
```

will create a matrix `Gtest` that is the same as G up to precision issues. This product is the reduced singular value decomposition of M. If you display or save this as an image it will look indistinguishable from the original picture. Python may complain about a potential loss of precision, but you may ignore this. Now try using the command

```
>>> Gtest = U[:,:s].dot(NPY.diag(S[:s])).dot(VT[:s])
```

with $s < r$. This is the *rank s approximation of M*.

(1) Display or save the rank s approximation of the image for several values of s. What happens when $s = 500$? What about 100? What about 1?
(2) How small can you make s and still have the image recognizable? Don't worry about a little graininess.
(3) How does the choice of s affect the size of the saved `.png` image? Why do you think that happens?