

1. NCM, problem 5.8
2. As discussed in class, the computer representation of an image is just a matrix of pixel values. For a monochrome image, the value is a single intensity of the pixel ranging between black and white. For a color image, there are three numbers stored for each pixel, which are typically the red, green, and blue intensities of the pixel, but other color schemes are possible. A color image of size $m \times n$ therefore consists of 3 matrices of size $m \times n$. Usually the intensities are stored as 8 bits, resulting in integers between 0 and 255.

As discussed in class, the singular value decomposition (SVD) can be used to compress images. For monochrome images stored in a matrix A , the SVD technique consists first decomposing A into the matrices U , Σ , and V^T , (i.e. $A = U\Sigma V^T$), where Σ is a diagonal matrix of the singular values, and U and V^T are orthonormal matrices of size $m \times m$ and $n \times n$, respectively. The rank p matrix $A^{(p)}$ that best approximates A in the 2-norm sense is then given by

$$A^{(p)} = \sum_{j=1}^p U_j (\Sigma_{j,j} V_j^T),$$

where U_j and V_j are the j th columns of U and V respectively. In terms of images, $A^{(p)}$ is now a “compressed” version of the original image A insofar as the space required to store the relevant columns of U and (ΣV^T) is less than that of the image as a whole. This space is $p \times (m + n)$. If no lossless compression methods are applied to U and (ΣV^T) such as Huffman coding, then the compression level of the SVD is $p(m + n)/(mn)$.

For color images, the above procedure can also be applied. However, it is useful to first reshape the 3 intensity matrices of size $m \times n$ into one matrix of size $m \times 3n$. When displayed, this reshaped matrix looks like one monochrome image with the 3 color planes side by side. The compression level for the SVD approach is now $p(m + 3n)/(3mn)$.

- (a) Download the jpeg image file `bsubbronco.jpg` from the course webpage. Load the image into MATLAB and then display the image using the commands

```
>>A = imread('bsubbronco.jpg');  
>>imagesc(A);
```

Note that the image must be stored in your current working MATLAB directory. Upon execution of the first command the variable `A` will contain a $m \times n \times 3$ matrix of the image as discussed above. Reshape `A` into a $m \times 3n$ matrix using MATLAB's `reshape` command. Display the reshaped matrix with the `imagesc` command as given above. Use the gray colormap to make the image grayscale.

- (b) Using the SVD compression technique described above, compress the image to levels 1%, 5%, 10%, 20%, 30%, 40%, 50%, and 75%. For each of these compression levels display the compressed image with the `imagesc` command. The displayed image should be the combined red, green, blue (RGB) color image, not three panels of the same image in grayscale. Comment on the quality of the image as p increases. When do you start seeing an increase in p having little effect. Turn in the plots at the various compression levels.

Some notes:

- i. To use MATLAB's `svd` command on `A` you have to first convert the entries of `A` to double precision floating point. You should also rescale the intensity values to be between 0 and 1. This is done in one command as follows:


```
>>A = double(A)/255;
```
 - ii. The SVD approximation to `A` may cause individual intensity values to fall outside of the range 0 to 1 even if the original intensity was within this range. For display purposes, you may want to fix this with the following command:


```
>>Ap = max(1,min(0,Ap));
```

 where `Ap` contains the p -compressed image.
 - iii. To reshape the `Ap` image to be of size $m \times n \times 3$ for plotting, use the command:


```
>>Ap = reshape(Ap, [m n 3])
```
- (c) Plot the singular values of `A` (i.e. the diagonal entries of Σ). Use a semilogy plot. Mark the compression levels from part (b) on this plot.
3. The initial value problem (IVP) below simulates the trajectory of a small satellite in the Earth-moon system, where all orbits lie in a plane:

$$x_1''(t) = x_1(t) + 2x_2'(t) - \mu^* \frac{x_1 + \mu}{D_1} - \mu \frac{x_1 - \mu^*}{D_2}, \quad x_1(0) = 0.994, \quad x_1'(0) = 0,$$

$$x_2''(t) = x_2(t) - 2x_1'(t) - \mu^* \frac{x_2}{D_1} - \mu \frac{x_2}{D_2}, \quad x_2(0) = 0, \quad x_2'(0) = -2.0015851063790825,$$

where,

$$0 \leq t \leq b = 17.06521656015796,$$

$$\mu = 0.012277471, \quad \mu^* = 1 - \mu,$$

$$D_1 = ((x_1 + \mu)^2 + x_2^2)^{3/2}, \quad D_2 = ((x_1 - \mu^*)^2 + x_2^2)^{3/2},$$

The mass of the satellite is neglected. The coordinate system moves so that the origin is the center of mass of the Earth and moon, and it rotates so that the Earth and moon lie on the x_1 axis a distance 1 apart (the Earth is just left of the origin, and the moon is just left of (1,0)). The constants are chosen so that $\mathbf{x}(b) = \mathbf{x}(0)$.

- (a) Use the `ode45` function to calculate one period of the orbit for each of the relative error tolerances of 10^{-2} , 10^{-4} , 10^{-6} . Create a phase portrait of x_2 vs. x_1 for each case. Try setting the `OutputFcn` of the `options` structure to `odephas2` (i.e. `options = odeset('OutputFcn', @odephas2)`) and then passing this into the `ode45` function to “animate” the orbit.
 - (b) Use `ode45` with 10^{-6} relative error to compute the orbit for *three* periods, and make a phase portrait. Explain what is wrong with the results. Try decreasing the relative error tolerances to see if the problem can be fixed. Describe what happens.
 - (c) Repeat part (b) with the `ode113` function. How do the results from this function compare to those of the `ode45` function?
4. NCM, problem 7.21
5. Use the books `surfer` and `pagerank` functions to compute the PageRanks for some portion of the `www.boisestate.edu` website. Using `surfer`, start at the `www.boisestate.edu` and collect 1000 webpages. Display the connectivity matrix for this collection of webpages using the `spy` command. Compute the PageRank of these 1000 webpages using the `pagerank` command. Do you see anything interesting in the results?