# CS30 Spring 2014
# Lab 9

Use the command `diary` to record your answers and submit them. Submit code for the functions and scripts you write. Submit any figures as indicated.

1. (40 points) SVD based image compression. Given a matrix $A$, the singular value decomposition (SVD) of the matrix is a decomposition

$$A = U\Sigma V^T$$

where $U$ and $V$ are orthonormal matrices and $\Sigma$ is the diagonal matrix of nonnegative sinuglar values. $\Sigma$ is sorted with the larged singular value at $\sigma_{\max} = \Sigma(1,1)$ and the other singular values in decreasing order down the diagonal. Matlab computes the SVD using the function `svd`. This can be used to compress an image as follows. Given an RGB image, compress each of the color planes independently and then recombine them. To compress a color plane give by the 2D matrix $A$, compute the SVD of $A$ as indicated above. Next, zero out the singular values that are smaller than some threshold value to obtain the compressed $\bar{\Sigma}$. Finally, reconstruct the color plane by computing $\bar{A} = U\bar{\Sigma}V^T$.

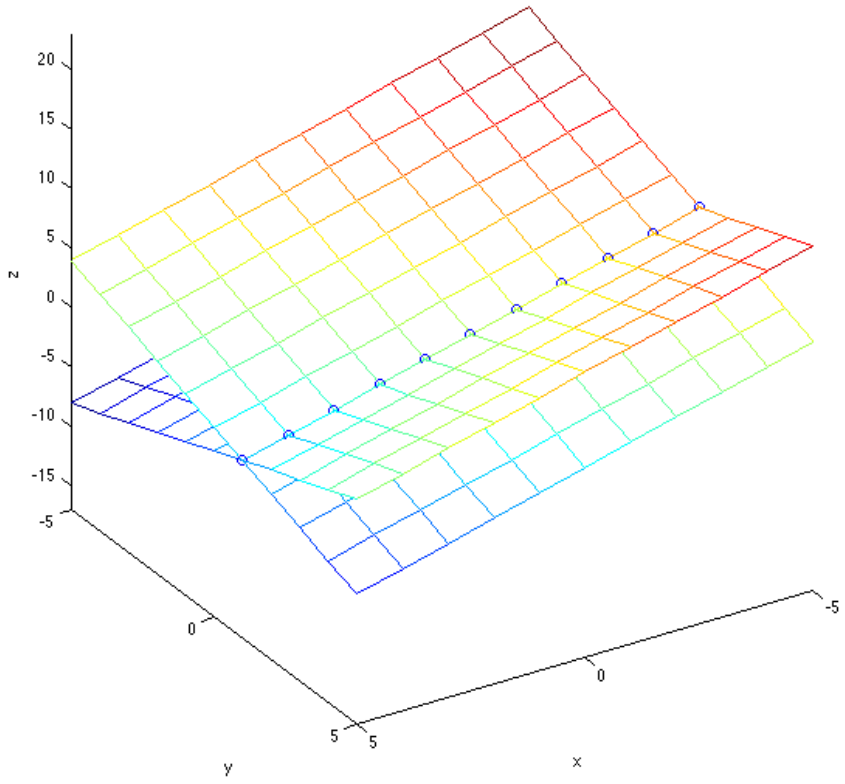
   (a) Write a function `CompressImage` which takes as arguments the input filename of the original RGB image, the output filename to which you will write the compressed image, and a percentage, p. In your compression algorithm, retain any singular value $\sigma > p * \sigma_{\max}$.

   (b) Run your program on the file `0.00-winter.jpg`. Use thresholds of 1%, 2%, ..., 10% and save the results in files `0.01-winter.jpg`, `0.02-winter.jpg`,..., `0.10-winter.jpg`, respectively. Modify the script `RunCompressImages.m` to do this. Submit `0.01-winter.jpg`, `0.05-winter.jpg` and `0.10-winter.jpg`.

   (c) What are the sizes of the resulting compressed files? Make a plot showing threshold percentage vs. resulting image size. You can use the command `ls -l` at the Matlab command prompt to display the image files and their sizes in bytes. Save your plot as a pdf file and submit it.

   (d) Write a Matlab script `DisplayImages.m` to display the compressed files in order starting with the original, 1% compression, 2% compression, pausing for 0.5 seconds after displaying each image.

2. (30 points) Write a Matlab script `PlaneIntersection.m` to recreate the figure below, by doing the following:




(a) Create mesh plots of the two planes

$$x + y + z = 4,$$
$$x - y + z = 2,$$

in a single figure using the Matlab command `mesh`. Use $x = [-5 : 5]$ and $y = [-5 : 5]$.

(b) For $x, y \in [-5, 5]$, find the set of integer triples $(x, y, z)$ that satisfy both plane equations. Also have your script output these integer triples in the command window.

(c) Use `scatter3` to plot these integer triples in your figure.

(d) Choose a view similar to the one shown in the example figure. Have your script automatically set that view by generating code from the figure window and adding it to your script.

3. (30 points) Consider again the following table of grade data from lab 6. Write a script `Lab9Problem3.m` which implements the following instructions.

| Name | HW1 | HW2 | Exam 1 | HW3 | HW4 | Exam 2 |
|---|---|---|---|---|---|---|
| Kermit | 21 | 51 | 76 | 6 | 16 | 69 |
| Piggy | 21 | 20 | 93 | 5 | 87 | 71 |
| Animal | 47 | 14 | 24 | 37 | 3 | 26 |
| Fozzie | 55 | 1 | 51 | 57 | 2 | 32 |
| Gonzo | 76 | 68 | 14 | 44 | 18 | 64 |

(a) Load the data `header`, `names`, and `data` from the file 'datafile.mat'.

(b) Create a cell array containing the data called `cellArray` with one row per muppet and three columns. Columns one, two and three should contain the name, an array of all the homework scores, and an array of all the exam scores, respectively. Verify the following results:

```
>> cellArray{1,2}
ans =
    21    51     6    16
>> cellArray{1,3}
ans =
    76    69
>> cellArray{4,3}
ans =
    51    32
```

(c) Run the command `cellfun(@mean,cellArray(:,2))`. What does this command do? What data type is the result?

(d) Add a fourth column for the average homework score in each row, and a fifth column for the average exam score in each row. Use `mean` and `cellfun` as in the previous step. You can use the function `num2cell` to write the result into the new column.

(e) Convert `cellArray` into a structure array, `structArray`, with field names `name`, `hw`, `exams`, `hwAvg`, and `examAvg` using the matlab function `cell2Struct`. Verify the following results:

```
>> structArray(1)
ans =
       name: 'Kermit'
         hw: [21 51 6 16]
      exams: [76 69]
      hwAvg: 23.5000
    examAvg: 72.5000
>> structArray(5)
ans =
       name: 'Gonzo'
         hw: [76 68 44 18]
      exams: [14 64]
      hwAvg: 51.5000
    examAvg: 39
```