CIEG 675
Lab #4 Due **Monday February 1, 2021 (LONG ASSIGNMENT)**

Please make your lab as a word document now. Include the MATLAB code for the lab and if the output is a figure, save it to a file (`print`), and insert the figure along with the code used to generate it into the document with some description of the figure(s). If the problem asks you to create a function, paste the entire function into the document for that problem as well. However, if you are still working with a script for some problems separate the problems into sections using (%%). Email the word document to me and send functions as requested.

1) The goal at the end of this problem is to create a *.avi* movie file that is a time lapse of the beginning of the Blizzard in Newark, DE of January 2016.

   Download the compressed folder from Canvas called "*BlizzardImages.zip"* and extract the images to your computer. You will need to write a script to find all of the *.JPG* files in the directory where you extracted the images. Recall the dir command discussed in class.

   Then, with a `for` loop, write a code that will create a time lapse of the blizzard (using `imshow`) and write a movie to a *.avi* file. You should also add text somewhere on each image in the time lapse (use a large fontsize) that displays the total amount of time that has elapsed, *in minutes* (you may have to refresh yourself on dates in MATLAB and converting dates from string format to the number of days relative to 0 AD). In addition, you may want to try playing around with the figure window size/position, as well as the position of the image inside the figure window, if you want to get the movie to look more professional without white/grey borders around the image.

   Use a framerate of 8 frames per second (FPS).

   Write your script so that the name of the *.avi* file is as follows:

   'LastName_FirstName.avi'

   So as an example, the resulting movie created by John Jones would be named *'Jones_John.avi'*. Verify that your code works properly and the video is "playable" by opening the movie outside of MATLAB, in a video player (VLC, QuickTime, Windows movie player etc.).

   Finally, submit your code along with the printed Word document. Your video should look something like the first 15 seconds of the video at the following link:
   https://www.youtube.com/watch?v=qCHQn8VW9h4
   Although the time text and other features won't be the same

   HINT: you can maximize an image within a figure window using something like

```
im = get(gcf,'children');
set(im,'position',[0 0 1 1]);
```

2) Write a function (call it **`gridinterp`**) to do an inverse distance weighting algorithm for 2D interpolation. It is a technique for interpolating to a uniform grid. Your function should have 7 input variables: `x, y, z, xg, yg, d and alpha`. Your function should have a single output vector, `zg`,

```
function zg = gridinterp(x, y, z, xg, yg, d, alpha)
```

where z is some measured parameter (e.g. elevation) at scattered horizontal locations x and y; and z_g is the z data interpolated onto the desired uniform grid given by horizontal grid locations x_g and y_g. The parameter d is the allowable radial distance from each uniform grid point that the algorithm should search for measured data points over which to interpolate (sometimes called search radius), and α (`alpha`) is the order to inversely weight each point at x, y based on its distance from x_g, y_g (i.e., for α = 2, the weighting is inverse distance squared). Hint, `xg` and `yg` should come from "vectorizing" the uniform grid you develop using the meshgrid function (see below).

The math way to write the approach looks like

$$zg_j = \frac{\sum_{i=1}^{N} \frac{1}{W_{ij}^{\alpha}} z_i}{\sum_{i=1}^{N} \frac{1}{W_{ij}^{\alpha}}},$$

where W_{ij} are the weights applied to each point. The value α determines how fast the influence of values around the grid point of interest decays. Typical values are 1 or 2, rarely higher.

The weights, W_{ij} are defined as the distance from the grid point to the real value as

$$W_{ij} = \sqrt{\left(xg_j - x_i\right)^2 + \left(yg_j - y_i\right)^2} \quad \text{(the Euclidean distance)},$$

Test your function with the data set called "*Avalon_survey.mat*" located on Canvas which contains scattered x, y, z data as columns. Interpolate to a grid that extends from `xx=-10:dx:100; yy= -180:dx:110`. You should specify an appropriate dx.

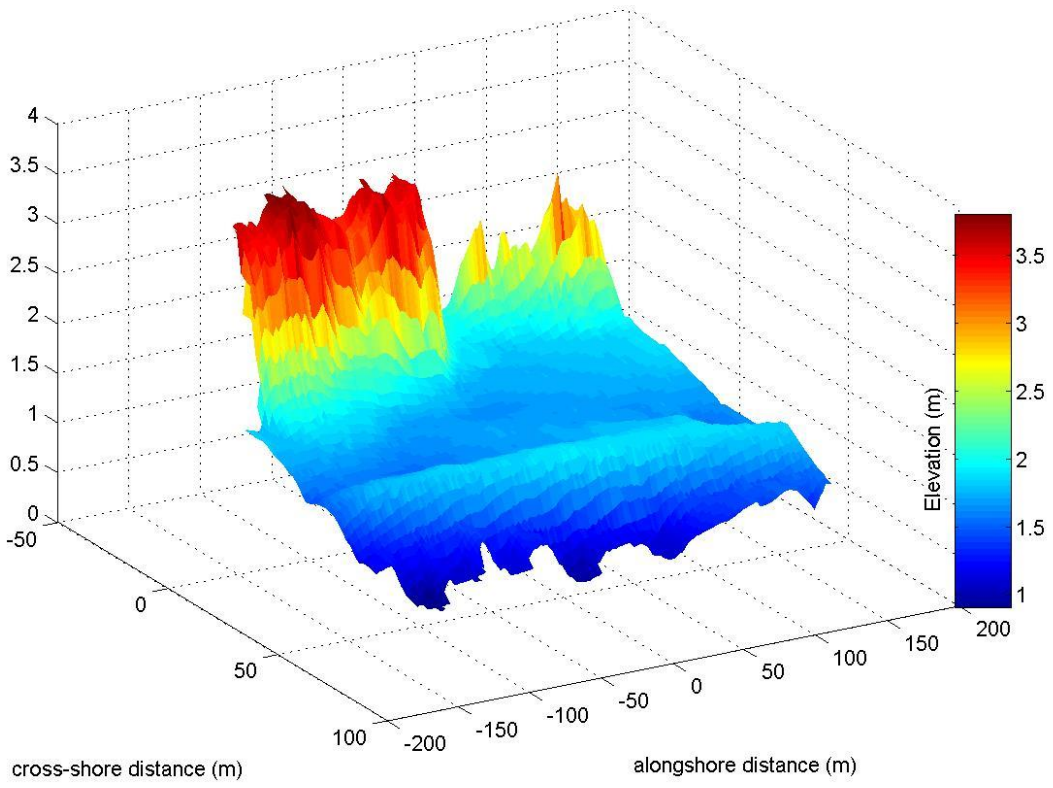
Determine the effect of varying dx, d and α for $1 \leq \alpha \leq 2$. **That means you will have to turn in more than one plot and actually *explain* what you see as differences**. Do not alter more than 1 variable at a time.

For display, it will be easier if you change the shape of the output variable `zg` back into a matrix using something like (too much help):

```
xx=-10:dx:100;
yy=-180:dx:110;

[X,Y]=meshgrid(xx,yy);
xg=X(:);
yg=Y(:);

zg=gridinterp(x,y,z,xg,yg,d,alpha);

ZG=reshape(zg,size(X));
```

Then you can make surface plots, pcolors, etc. using `X,Y,ZG`.

An example may look like this



3) **OPTIONAL**: COVID-19 can be modeled with the SIR model. The model consists of three coupled differential equations

Susceptible (S) : $\frac{dS}{dt} = -bSI$

Infected (I) : $\frac{dI}{dt} = bSI - kI$

Recovered/Removed (R) : $\frac{dR}{dt} = kI$

where S, I, and R are all functions of time (t) in days. Additionally, S, I, and R are the normalized version of the susceptible, infected and recovered/removed population. That is, they are divided by the total population group, N such that $0 \leq S \leq 1$ (and the same applies for I and R). note that $S+I+R = 1$; do you see why?)

The value k is known as the average period of infectiousness, $k = 1/k_1$; where for example $k_1 = 14$ days as the average time to recover.

The value b is defined as $1/b_1$, where b_1 is how often a susceptible – infected contact occurs. For example, if $b_1 = 4$ then every 4 days there is a contact.

Code the SIR model equations as a function using the easiest differencing algorithms you can think of (or matlab ODE solvers if you wish). Test your model with some chosen average time to recover (k_1; I used 14) and b_1 values ranging from 1 to 10. Compare the effect of these changes in b_1 to how the population responds.

NOTE: Your function must additionally take as input an initial VERY SMALL infected rate. Otherwise no one will be infected and the model makes no sense. Your model should have an initial susceptible rate S ($t = 0$) = 1 and an initial R ($t = 0$) = 0. Of course, initially $S+I+R$ will not be exactly one but the initial I is quite small.

4) Solve $\frac{dy}{dx} = -y(x)$ subject to initial condition y(0)=1. Note the real answer is e^{-x}.
Do the solution in matlab two different ways.

A) Solve using a forward difference.
B) Solve using a central difference.

For each solution approach, use three different dx values and make a table showing the root-mean-square error (compared to the real answer) as a function of dx and numerical scheme. Also, show a plot of your solutions.

OPTIONAL: Solve the same equation by exploring the use of the functions ODE23 and/or ODE45. These are matlab's built in differential equation solvers (there are others) for differential equations over a certain range in independent variable and with a known boundary condition. Make a plot of your solutions from 1a (A and B) with the solution from one of these ODE solvers.

5) Load the image speed_limit.JPG from canvas and explore the matlab function called 'edge' to perform edge detection on the simple image. Obtain the output from edge and change the background image color to yellow and the letter and number edges to blue.

There are multiple ways to do this and you will need to figure out how to go from the original RGB image to grayscale for analysis and then back to RGB for output.