

1) Load the data from my webpage called HW8data.mat. It is a 3D matrix of beach elevation changes from Duck, NC on Oct 10, 1994. Each plane (third index in matrix variable called ducksurf) is a map of elevation as a function of cross-shore (variable xi corresponding to the first index of ducksurf) and alongshore distance (variable yi corresponding to the second index of ducksurf). As you move down the matrix, the data planes are a function of time (variable newtime in fractional hours).

Come up with a way to present this data in a neat fashion (I am giving a lot of leeway here; but pretend you were going to submit this as a figure in a publication). So, if you choose subplots, do not leave a lot of space between individual axes.

You will find that you need to use handles repeatedly to really do this correctly.

Good luck

2) Write a **function** to do an inverse distance weighting algorithm for interpolation. I talked about this in class the other day. It is a technique for interpolating to a uniform grid. Your function should take as input x,y,z,xg,yg and output zg of the form

function [zg]=inverse\_distance\_weighting(x,y,z,xg,yg,d,alp)

where x,y,z are the measured values of some parameter z at horizontal locations x and y, xg and yg are the locations on a uniform grid where you want an interpolated value, zg. The parameter d is the allowable distance from each uniform grid point that the algorithm can search for real points over which to interpolate. 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  $\alpha$  determines how fast the influence of values around the grid point of interest decays. Typical values are 1 or 2, rarely higher. I suggest you also have alpha as an input to your function.

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

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

Test your function with the data set called Avalon\_survey.mat (contains irregularly spaced x,y,z data as columns). I want you to interpolate to a grid that extends from xx=-10:dx:100; yy=-180:dx:110. You specify dx.

What I would like is for you to determine the effect of varying dx and varying alpha between 1 and 2. That means you will have to turn in more than one plot and actually explain what you see as differences.

For display, it will be easy if you change the shape of the output variable zg back in to a matrix using something like

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

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

```
[zg]=inverse_distance_weighting(x,y,z,xg,yg,d,alp);
```

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

then you can make surfaces, pcolors etc using X,Y,zg.