

CIEG 675

Homework #7 Due **Wednesday April 29, 2009**

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

```
%%%%%%%%%% MY PLOTS %%%%%%%%%%
```

```
%% make a nice plot from the data given on the website consisting of beach  
%% surface over a small region.
```

```
%% first load the .mat file  
load HW8data.mat
```

```
%% there are 14 surfaces that we want to display.
```

```
% set up frame  
let='abcdefghijklmn'; % letters to label plots  
wid=0.19; % subplot width  
hei=0.19; % subplot height  
sp=0.02; % space between plots
```

```
clf  
figure(4)  
for cf=1:14;
```

```
if cf==1  
pos=[0.1+0*wid 0.18+3*hei wid hei];  
elseif cf==2  
pos=[0.1+wid+sp 0.18+3*hei wid hei];
```

```

elseif cf==3
pos=[0.1+2*(wid+sp) 0.18+3*hei wid hei]; % first row
elseif cf==4
pos=[0.1+3*(wid+sp) 0.18+3*hei wid hei];

elseif cf==5
pos=[0.1+0*(wid+sp) 0.18+2*(hei-sp) wid hei];
elseif cf==6
pos=[0.1+1*(wid+sp) 0.18+2*(hei-sp) wid hei]; % second row
elseif cf==7
pos=[0.1+2*(wid+sp) 0.18+2*(hei-sp) wid hei];
elseif cf==8
pos=[0.1+3*(wid+sp) 0.18+2*(hei-sp) wid hei];

elseif cf==9
pos=[0.1+0*(wid+sp) 0.18+(hei-3*sp) wid hei];
elseif cf==10
pos=[0.1+1*(wid+sp) 0.18+1*(hei-3*sp) wid hei]; % third row
elseif cf==11
pos=[0.1+2*(wid+sp) 0.18+1*(hei-3*sp) wid hei];
elseif cf==12
pos=[0.1+3*(wid+sp) 0.18+1*(hei-3*sp) wid hei];

elseif cf==13
pos=[0.1+0*(wid+sp) 0.18+(0*hei-4*sp) wid hei];
elseif cf==14
pos=[0.1+1*(wid+sp) 0.18+(0*hei-4*sp) wid hei]; % fourth row
end

txt=sprintf('h%d=subplot("position",pos);!,cf); % set handle to each subplot
eval(txt)
pcolor(xi,yi,ducksurf(:, :, cf));
shading interp
hold on

text(13.6,1.56,let(cf));
cax=[0.6 2.6];
caxis(cax);

txt=sprintf('%6.3f',newtime(cf));
hc=text(110,977,[txt 'hr']);
set(hc,'color','k','fontweight','bold')

set(gca,'XTick',[104 112 120]);

```

```

set(gca,'YTick',[960 970 980]);

set(gca,'XTicklabel',[]);
set(gca,'YTicklabel',[]);

grid on
drawnow

end

pos=[0.1+2.5*(wid+sp) 0.18+(0*hei-4*sp) wid hei]; % fourth row
subplot('position',pos)
xlabel('Cross-shore dist. (m)');
ylabel('Alongshore dist (m)');

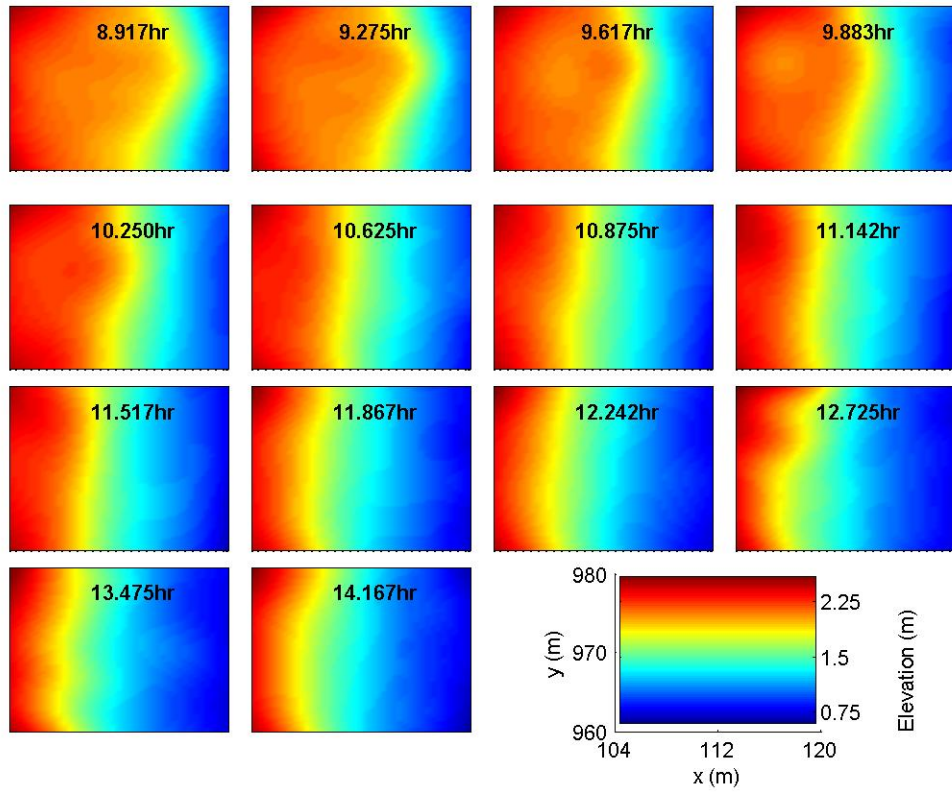
caxis(cax)
hcol=colorbar;
set(hcol,'position',[0.1+2.52*(wid+sp) 0.19+0*hei-4*sp wid-0.02 hei-0.02])
set(hcol,'ytick',[0.75 1.5 2.25])

xlabel('x (m)');
ylabel('y (m)');

set(gca,'ytick',[0 0.5 1]);
set(gca,'yticklabel',[960 970 980])
set(gca,'xtick',[0 0.5 1]);
set(gca,'xticklabel',[104 112 120])

text(1.4,0.0,'Elevation (m)','rotation',90);

```



```
%%%%%%%% a totally different way to present the entire data set with slices and
%%%%%%%% contours
```

```
%% to make a nice 3D image
```

```
%%% try three surf commands, m=cols, n=rows
```

```
figure(1)
[m,n,o]=size(ducksurf); % find size of matrix
surf(xi,yi,newtime(1)*ones(51,41),ducksurf(:, :, 1)); % make surface plot of first
surface
shading interp
colormap(jet)
hold on
```

```
figure(2) % the junk figure
```

```
v=1:.25:2.25; % contours wanted
[c,h]=contour(xi,yi,ducksurf(:, :, 1),v); % put contours on a scratch figure
figure(1);
l=find(c(1,:) < 5); % These are the start indices of the elevations
for k=1:length(l);
hold on
```

```
plot3(c(1,l(k)+1:l(k)+c(2,l(k))),c(2,l(k)+1:l(k)+c(2,l(k))),newtime(1)*ones(1,c(2,l(k)))
,'k--','LineWidth',2);
end
```

```
%% plots the contours back onto figure 1
```

```
% intermediate wall
```

```
mi = 21;
figure(1)
junk=ducksurf(mi, :, :); junk=(reshape(junk,n,o)); % grab intermediate wall of data
& reshape
junk2=ducksurf(mi, :, :); junk2=(reshape(junk2,n,o));
[xg,zg]=meshgrid(xi,newtime); % grid x and newtime
yg=yi(mi)*ones(size(xg)); % fake y vector
surf(xg,yg,zg,junk2); % make surf plot
shading flat
figure(2)
v=1:.25:2.25;
[c,h]=contour(xg,zg,junk,v); % contour on scratch surface
figure(1);
l=find(c(1,:) < 5); % These are the start indices of the elevations
for k=1:length(l);
```

```

hold on
plot3(c(1,l(k)+1:l(k)+c(2,l(k))),yi(mi)*ones(1,c(2,l(k))),c(2,l(k)+1:l(k)+c(2,l(k))),'k--',
'LineWidth',2);
end
%% overlay contours on figure 1

% put some lines to help viewer
plot3([xi(1) xi(end)],[yi(mi) yi(mi)],[newtime(1) newtime(1)],'k-', 'linewidth',1);
plot3([xi(1) xi(end)],[yi(mi) yi(mi)],[newtime(end) newtime(end)],'k-', 'linewidth',1);
plot3([xi(1) xi(1)],[yi(mi) yi(mi)],[newtime(1) newtime(end)],'k', 'linewidth',1);
plot3([xi(end) xi(end)],[yi(1) yi(mi)],[newtime(1) newtime(1)],'k', 'linewidth',1);
plot3([xi(end) xi(end)],[yi(mi) yi(mi)],[newtime(1) newtime(end)],'k', 'linewidth',1);

% now make the backwall
figure(1)
junk=ducksurf(m,,:); junk=(reshape(junk,n,o)); % grab the data along the last row
and reshape
junk2=ducksurf(m,,:); junk2=(reshape(junk2,n,o)); % don't really need both of
these
[xg,zg]=meshgrid(xi,newtime); % meshgrid the xvector and time
yg=yi(m)*ones(size(xg)); % make a fake y vector
surf(xg,yg,zg,junk2); % make a surface plot
shading interp
figure(2)
v=1:.25:2.25;
[c,h]=contour(xg,zg,junk,v); % contour again to our scrtach figure
figure(1);
l=find(c(1,:)<5); %These are the start indices of the elevations

for k=1:length(l);
hold on
plot3(c(1,l(k)+1:l(k)+c(2,l(k))),yi(m)*ones(1,c(2,l(k))),c(2,l(k)+1:l(k)+c(2,l(k))),'k--',
'LineWidth',2);
end
% replot the contour lines on figure 1

% put some lines on the plot to help viewer
plot3([xi(1) xi(end)],[yi(m) yi(m)],[newtime(1) newtime(1)],'k-', 'linewidth',1);
plot3([xi(1) xi(end)],[yi(m) yi(m)],[newtime(end) newtime(end)],'k-', 'linewidth',1);
plot3([xi(1) xi(1)],[yi(m) yi(m)],[newtime(1) newtime(end)],'k', 'linewidth',1);
plot3([xi(end) xi(end)],[yi(1) yi(m)],[newtime(1) newtime(1)],'k', 'linewidth',1);

```

```

plot3([xi(end) xi(end)],[yi(m) yi(m)],[newtime(1) newtime(end)],'k','linewidth',1);

% these go around box
plot3([xi(1) xi(end)],[yi(1) yi(1)],[newtime(1) newtime(1)],'k-','linewidth',1);
plot3([xi(end) xi(end)],[yi(1) yi(1)],[newtime(1) newtime(end)],'k','linewidth',1);
plot3([xi(end) xi(end)],[yi(1) yi(end)],[newtime(end) newtime(end)],'k','linewidth',1);
plot3([xi(1) xi(1)],[yi(1) yi(end)],[newtime(1) newtime(1)],'k','linewidth',1);
axis([103 120 958 980 8.75 14]);

figure(1)
% right wall;
junk2=ducksurf(:,n,:); junk2=reshape(junk2,m,o);
junk=ducksurf(:,n,:); junk=reshape(junk,m,o); % don't really need both of these
[yg,zg]=meshgrid(yi,newtime); % meshgrid the y vector and new time
xg=xi(end)*ones(size(yg)); % make fake x vector
surf(xg,yg,zg,junk2); % make surface plot
shading interp
figure(2)
v=1:.25:2.25;
[c,h]=contour(yg',zg',junk,v); % put contours on our scratch plot
figure(1);
l=find(c(1,:)<5); %These are the start indices of the elevations
for k=1:length(l);
hold on
plot3(xi(end)*ones(1,c(2,l(k))),c(1,l(k)+1:l(k)+c(2,l(k))),c(2,l(k)+1:l(k)+c(2,l(k))),'k--',
'LineWidth',2);
end
%% do the contouring again on figure 1

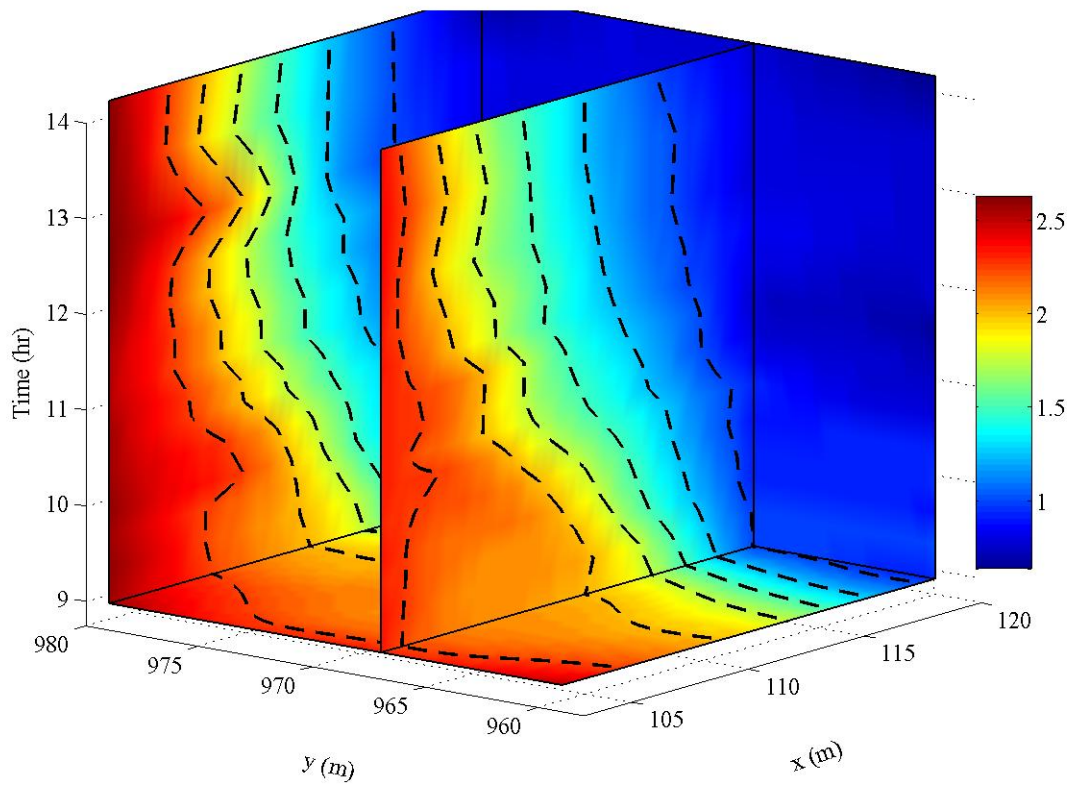
grid on

% set view
az = -51.5000;
el = 16;
view(az,el);

%% work on colorbar size
h=colorbar;
p=get(h,'position'); % present position
set(h,'position',[p(1) .28 p(3) .43]); % new position by trial and error
set(gca,'FontSize',12,'FontName','times');
h=xlabel('x (m)','FontName','times');
set(h,'rotation',20);
h=ylabel('y (m)','FontName','times');

```

```
set(h,'rotation',347);  
zlabel('Time (hr)','FontName','times');
```



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.

```
%%%%%%%%%%%% MY SOLUTION %%%%%%%%%%
```

```
% This is the function, below is some code to call it
```

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

```
%
```

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

```
%
```

```
% function to make an inverse distance weighting interpolation
```

```
%
```

```
% inputs:
```

```
% x,y,z are the irregularly spaced data points
```

```
% xg,yg are the coordinates you want to find new zg values
```

```
% note that xg,yg should be vectorized form of xg,yg data after using
```

```
% meshgrid
```

```
% d is the maximum allowable distance that points can influence a
```

```
% particular zg
```

```
% alp is the alpha value to determine linear (1) or squared (2) weighting
```

```
%
```

```
%
```

```
% output
```

```

% zg, the interpolated values corresponding to location xg,yg

for k=1:length(xg);

    %% catch in case we are right on a real point

    clear p

    p= find ( (abs(x-xg(k)) + abs(y-yg(k))) ==0); % one requested point is right on top of the
    known point (should almost never happen)

    if isempty(p)==0 % it found one

        zg(k)=z(p(1)); % set the value to known condition

    else

        %% find distances to known points

        dists=sqrt ( (x-xg(k)).^2 + (y-yg(k)).^2 ); % Euclidean distance

        clear l

        l=find(dists<=d); % only those within radius of influence

        if isempty(l)==0

            zg(k)= sum( z(l)./(dists(l).^alp) ) ./ sum(1./dists(l).^alp);

        else

            zg(k)=NaN;

```

```

        end

    end % if

end % k loop

%%%%%% IN another m-file I wrote this code to call the function %%%%%%

%% code to call gridding routine from data.

% load the data
load Avalon_survey.mat

% this puts the x,y,z data into matlab

%% the region to be gridded

dx=2;
xx=-10:dx:100;
yy=-180:dx:110;

[X,Y]=meshgrid(xx,yy);
xg=X(:);
yg=Y(:); % the locations of the grid vectorized

% send to gridding routine

d=10; % will get lots of smoothing based on grid spacing
alp=2;

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

Z=reshape(zg,size(X)); % reshape data so we can use surf function

```

```
surf(X,Y,Z)

shading flat

view(60,30)

h=colorbar; % set handle

h1=get(h,'position'); % get position info from handle

set(h,'position',[h1(1) 0.24 h1(3) 0.45])

xlabel('cross-shore distance (m)');

ylabel('alongshore distance (m)');

text(100,180,1,'Elevation (m)','rotation',90) % label the colorbar
```

