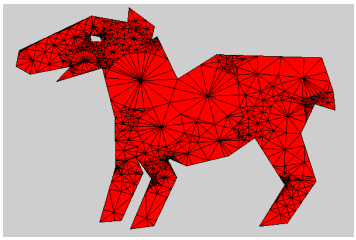
interpolation, color, & light



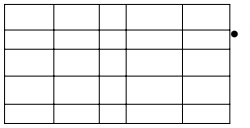
Outline

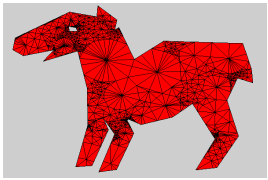
- Announcements
  - HW II--due today, 5PM
  - HW III on the web later today
- HW I: Issues
- Structured vs. Unstructured Meshes
- Working with unstructured meshes
- Interpolation
- colormaps
- lights

HW I

- No issues on the programs--most did well
  - sample solutions are on the web
- No problems figuring out colors or finding handles
  - if you don't understand a question, come find me!
- Only one person got 1 correct
  - This was a bit of a trick question, but …
  - since you have to go to the computer to do the programming, you might as well try the problems

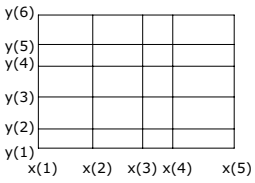
Interpolation & grids





- To plot with surfaces, you need some kind of mesh or grid:
  - a mesh is a collection of non-overlapping polygons that fills a region of space
  - meshes can be structured (all polygons the same size and shape) or unstructured

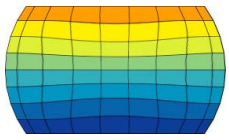
Regular Grids



- Meshes made from quadrilaterals are known as grids
  - A regular grid has only 90° angles (rectangles) and can be defined by vectors x and y
  - if x(j+1)-x(j) and y(j+1)-y(j) are constant, then the grid is uniform

Unstructured Grids

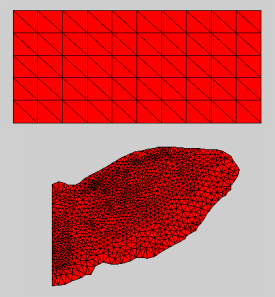
- If the cells are not rectangular, then the grid is irregular or unstructured
- X and Y are now matrices:



Visualizing Grids

- Matlab's core 2D functions want grids:
  - pcolor
  - contour
  - surf
  - mesh

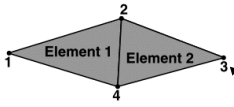
The World is not Square



- Meshes of triangles are common, especially in finite element modlling
- Triangular meshes can also be structured or unstructured
  - unstructured are more common

Triangular Meshes

- Matrices are rectangular, so it is hard to "fit" a triangular mesh into a matrix
- Typically, triangular meshes require 3 arrays:
  - vectors x and y contain the location of the vertices (in no particular order)
  - array tri defines how the vertices are connected
    - Each row contains indexes the three vertices forming a triangle




```
tri=[1 4 2;
     2 4 3];
```

(x(3),y(3))

Plotting Triangular Meshes

- Matlab's trimesh is designed to plot z=t f(x,y) on a triangular mesh
  - trimesh(tri, x,y,z, {c});
- We can do the same thing with patch (or surface)
  - we may not be interested (or have) z and c
  - this is mainly to illustrate the form of x, y, z, and c data fields

Patching Triangular Meshes

- h=patch(X,Y,C) creates polygons for each column of X,Y, and C
  - if our mesh has t triangles, X, Y, and C will be 3-by-t
  - X=[x(tri(:,1)), x(tri(:,2)), x(tri(:,3))]';
- The mesh will be plotted in 2D view with flat color: triangle colors will be set by the first vertex (first row of C);

Patching Triangular Meshes

- Suppose we want to make it 3D with elevation set by C
  - patch(X,Y,C,C) will work (C used for both elevation and color)
- or, if we've already plotted, with h=patch(X,Y,C):
  - set(h,'zdata',C);view(3)

Interpolation

- If we want to plot with surfaces (or patches), we need some kind of mesh
- But, we are rarely able to sample on a grid
  - observations are often made at irregular intervals of time and space due to sampling constraints or equipment error (missing data)
- It is possible to calculate what the observations should've been at locations where we didn't sample
  - This is known as *interpolation*

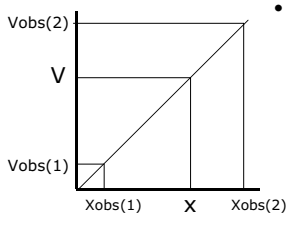
Interpolation

- It is possible to calculate what the observations *should've been* at locations where we *didn't sample*
  - This implies that we know something about the system we're observing
  - But, if we know so darn much, why bother observing?
  - The bottom line is that we are creating data and we have no way of knowing whether or not we've done this correctly
    - All interpolations should be treated with suspicion

Formal Statement of Problem

- Inputs:
  - Xobs= locations where we observed data (time, space, etc., can also have Yobs, Zobs)
  - Vobs= observed values: Vobs=f(Xobs)
    - Remember, we don't know the exact form of f, but we may know something about its structure
  - X=locations where we would like to know the values
- Then,
  - V=INTERPMETHOD(Xobs, Vobs, X)
  - Ideally, we have enough observations and know enough about f so that INTERPMETHOD ≈ f

Linear Interpolation

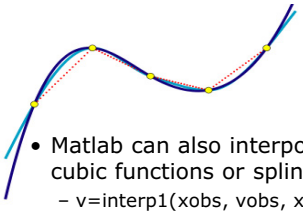


- Linear interpolation is the simplest form of interpolation (other than picking a constant)
  - If we have two observations, we can fit a line between them and use the equation of the line to determine v
  - linear interpolation is used implicitly when plotting with lines or using interpolated shading

Linear Interpolation in Matlab

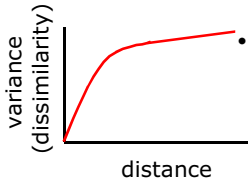
- Matlab's interpolation routines use linear interpolation by default
  - V=interp1(Xobs, Vobs, X)
  - V=interp2(Xobs, Yobs, Vobs, X, Y)
    - Xobs, and Yobs must define a grid (i.e. same form as inputs for pcolor or surface)
    - interp3, interpN work for higher-dimensional data
  - V=griddata(Xobs, Yobs, Vobs, X, Y)
    - observations need not be gridded
    - uses Delaunay triangulation

Higher-order Interpolation



- Matlab can also interpolate using cubic functions or splines
  - v=interp1(xobs, vobs, x, 'spline');
  - the results are smoother, but potentially very wrong

Objective Analysis and Kriging



- Matlab's default interpolation schemes are simple, but stupid
- Kriging (a.k.a objective analysis) is a statistical interpolation technique
  - requires you to know (or guess) the structure of your data's spatial variance

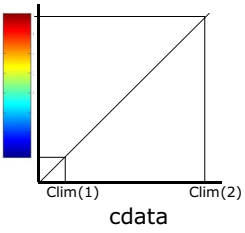
Kriging

- In kriging, Error=f(distance)
  - Assumes your knowledge about v declines as you move away from your observations
  - Can often determine error function from your observations
- v(j)=w1*vobs(1)+w2*vobs(2)+…+wn*vobs(n)
  - The v's are weighted means of the observations, the weights are determined by the distance from v(j) according to the error function
  - In addition to v, we can also get an estimate of the interpolation error

Kriging in Matlab

- Kriging is computationally simple, but there are some statistical condiderations
  - <RECOMMEND BOOK>
- Matlab does not have a built-in kriging function (that I know of)
  - http://globec.whoi.edu/software/kriging/easy_krig/easy_krig.html
  - other software exists

Colormaps



cdata

- Matlab colormaps are m-by-3 matrices, where each row is an RGB vector
- When a color property (face or edge) is set to flat or interp, Matlab will determine the color using Cdata, Clim, and the colormap

Colormaps

- Built in colormaps (help graph3d)
  - map=copper(N);--gets copper colormap with N rows
  - map=colormap--gets current colormap (default is jet)
  - colormap(map);--sets colormap to map
    - map could be a built-in colormap (copper)
- Colormap is a property of the figure, not the axes
  - This means that we can have only one colormap per figure

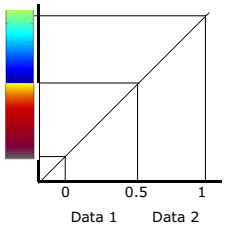
Creating New Colormaps

- Matlab colormaps are usually adequate, but will need to create your own if:
  - You need more than one map/figure
  - You don't like Matlab's

Creating New Colormaps

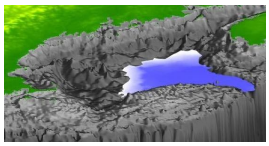
- Simplest approach is modify Matlab's
  - map=colormap(gray);map=flipud(map);
    - map will go from black to white rather than white to black
  - brighten lets you "brighten" or "darken" current colormap
- Create your own with interp1
  - v=[1 3 4]'; col=[0.5 0.5 0.5; .75 0 0; 1 1 0];
  - map=interp1(v,col,linspace(1,4,64)', 'cubic');

Multiple Colormaps



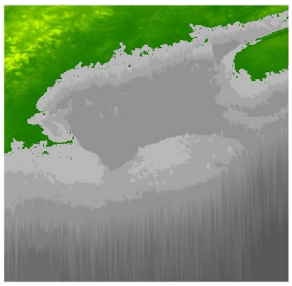
- Working with multiple colormaps gets very complicated
  - requires lots of handle graphics work
- Tips & Things to remember
  - Single Clim-space, so pick something simple [0 1],[-2 1]
  - Transform actual clims to this space

Example: Gulf of Maine Bathymetry



- Today, I'll start leading you through the process of creating my Gulf of Maine visualizations
- We'll start with the bathymetry and add the temp (blue stuff) next week
- This figure has two surfaces and uses 3 colormaps and two light sources

Lighting



- With the colors used, it is impossible to see features in the Gulf of Maine
- Matlab allows you to add light sources
  - Reflections can enhance 3D perspective

Lighting

- Lighting is tough & involves a lot of trial and error
  - 1. Make sure your surface can be lit:
    - Lighting phong (or gourard or flat) sets the 'facelighting' property of your surface
    - It will now reflect light in a "natural" way
    - Setting backfacelighting to lit is also good
  - 2. Add a light
    - L=light(*light options*) creates a light object
      - Control its position, color, and distance (infinite vs. local)
    - camlight(az,el) creates a light source relative to you (the camera)

Lighting

- Lighting is not for the faint of heart, but here are some tips:
  - set(gcf,'renderer','opengl') gives better output and performance
  - Keep track of handles to lights
    - Turn them on or off (change visibility)
    - Move them around