2D Routines in 3D

Outline

- Announcements
  - Wed: 2 Lectures, 8-10
  - bagels or donuts?
  - HW II—due Wed. 5PM
  - Discuss HW I on Wed
- Updated Syllabus
- Representing f(x,y)
- Lines & Surfaces in 3D
- Color & Surfaces
- Example: curtain.m

Syllabus

6. 2D routines in 3D: surfaces
7. Advanced color and lighting
8. Interpolation & unstructured grids
9. GUI’s I
10. GUI’s II
11. Volumetric visualization I
12. VV II & where to go from here
Representing \( f(x,y) \)

- Lots of choices for visualizing 2D data, often depends on properties of data

<table>
<thead>
<tr>
<th>Structure of ( x ) &amp; ( y )</th>
<th>Regular Grid</th>
<th>Irregular</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D view</td>
<td>colormap</td>
<td>colortime</td>
</tr>
<tr>
<td></td>
<td>contour</td>
<td>circleplot</td>
</tr>
<tr>
<td>3D view</td>
<td>surface</td>
<td>bar3</td>
</tr>
<tr>
<td></td>
<td>meshgrid</td>
<td>stem3</td>
</tr>
</tbody>
</table>

3D views

- 3D views on a computer or painting are just illusions
  - Perspective
    - lines converge towards focal point
    - Color and lighting can enhance perspective
    - Optical illusions are possible

Line Objects in 3D

- \( h = \text{plot}(x,y);	ext{get}(h,\text{'zdata'}) \)
  - \( \text{ans} = \)
    - Empty matrix: 1-by-0
- Both patch and line objects have a \( z \)-data field. Plot and patch set this to \([\]\)
- We can plot a line in 3D using \( \text{plot3}(x,y,z) \)
  - could also set \( z \)-data field manually
3D functions will set axes projection to perspective

- The axes are now a box drawn in perspective

**Controlling the 3D view**
- We can control the size of the axes (limits) and the way they are drawn (view)
  - `set(gca,'xlim',[minimum, maximum])`—also for y and z
  - Can also set scale to log or reverse direction (must be done manually)
- Clicking on the circle button allows you to rotate the axes in 3D

**Controlling the 3D view**
- Can also control the view from the command line through `view`:
  - `view(2)` or `view(3)` gets default 2D or 3D views
  - `view([az,el])` sets the azimuth=az (rotates about z) and elevation=el (rotates about line in x-y plane)
**Surfaces in 3D**

- Like lines, patch and surface objects have zdata fields.
- `surf(X,Y,Z)` creates a surface with vertices defined by X,Y, and Z
  - color is proportional to Z
  - `facecolor=flat`
- `mesh(X,Y,Z)` is similar, but doesn’t fill polygons
  - `edgecolor=flat`

---

**Comparing surf and pcolor**

- `pcolor` is a special form of `surf`  
- How can we change cdata?

---

**CDATA**

- We can use surface plots to represent 3D functions c=f(x,y,z)
  - x, y, and z define a surface
  - c is represented by color
- `surf(X,Y,Z,C)` where C is the same size as Z
Example: Curtain plots

- Sample $c$ at every $z$ along an $x,y$ path
- We want to display $c$ on curtain below the path

Curtain plots

- The path is the trajectory of a ship
- The ship has an acoustic transducer that is sending sound into the water
- By measuring the volume of sound returned and when it occurs, we can get $c(z)$
  - this is a rough measure of the concentration of "stuff" in the water
  - depending on the frequency used, this could be shrimp, fish, or submarines

curtain.m

<table>
<thead>
<tr>
<th>name</th>
<th>size</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x$</td>
<td>1-by-$n$</td>
<td>$x$-locations</td>
</tr>
<tr>
<td>$y$</td>
<td>1-by-$n$</td>
<td>$y$-locations</td>
</tr>
<tr>
<td>$z$</td>
<td>m-by-1</td>
<td>depth bins</td>
</tr>
<tr>
<td>BS</td>
<td>m-by-$n$</td>
<td>data</td>
</tr>
<tr>
<td>Outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$h$</td>
<td>scalar</td>
<td>handle to the curtain</td>
</tr>
</tbody>
</table>
curtain.m

• The key to curtain is forming matrices X, Y, and Z from vectors x, y, and z

<table>
<thead>
<tr>
<th>x1, y1</th>
<th>x2, y2</th>
<th>x3, y3</th>
<th>x4, y4</th>
<th>x5, y5</th>
</tr>
</thead>
<tbody>
<tr>
<td>z1</td>
<td>z2</td>
<td>z3</td>
<td>z2</td>
<td>z3</td>
</tr>
<tr>
<td>z1</td>
<td>z1</td>
<td>z3</td>
<td>z2</td>
<td>z3</td>
</tr>
<tr>
<td>z1</td>
<td>z3</td>
<td>z2</td>
<td>z3</td>
<td>z1</td>
</tr>
</tbody>
</table>

• Z = z(:)*ones(1, n);
• X = ones(m, 1)*x(:)'; Y = ones(m, 1)*y(:)';