SECTION 7: THREE-DIMENSIONAL PLOTTING

ENGR 112 – Introduction to Engineering Computing

3-D plots

We'll consider two main categories of 3-D plots:

- ***3-D line plots***
  - A single independent variable – two of the variables are functions of the third
  - A line in 3-D space
- ***Surface plots***
  - A function of two variables – two independent variables, on dependent variable
  - *Height* of the function is dependent on position in the x,y plane

3-D Line Plot – `plot3(…)`

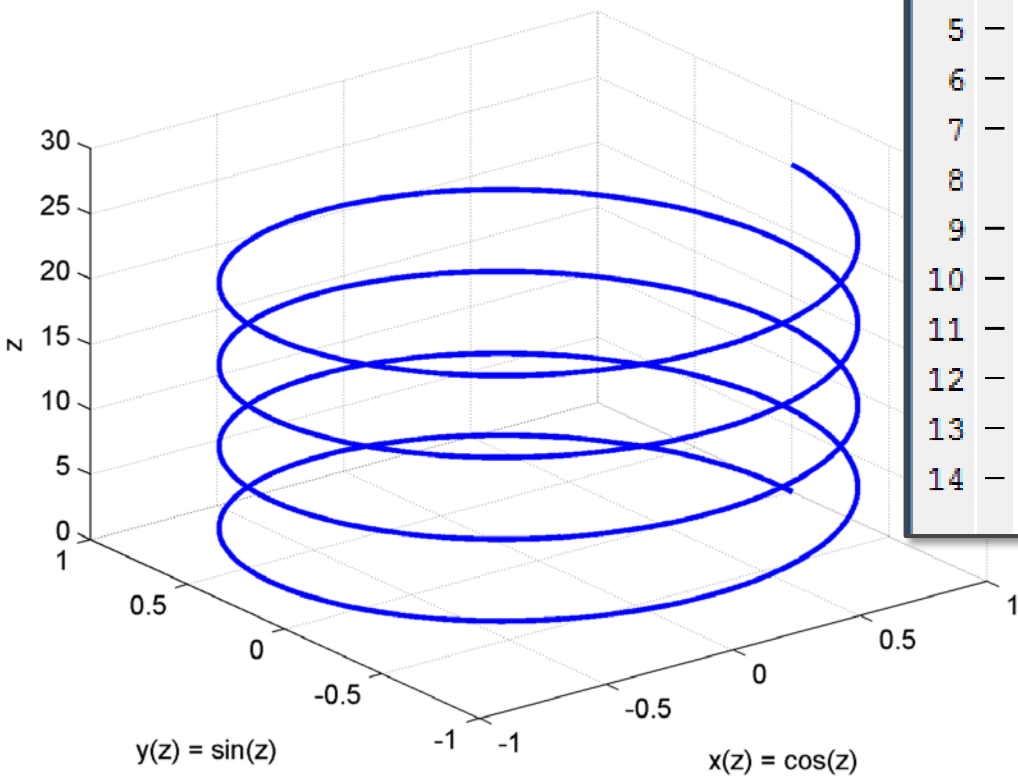
- One independent variable, two dependent variables
  - E.g. $x = f(z), \quad y = f(z)$
  - 3-D line plot – a curve in three-dimensional space

```
plot3(x,y,z,'LineSpec','PropName',PropValue,…)
```

- `x`, `y`, and `z` inputs are ***vectors***
  - One `x` and one `y` value for each `z` value

3-D Line Plot – plot3(…)

$$x = \cos(z), \qquad y = \sin(z), \qquad 0 \le z \le 8\pi$$




```
z = 0:pi/500:8*pi;
x = cos(z);
y = sin(z);

figure(1); clf
plot3(x,y,z,'-b','LineWidth',2)
grid on
xlabel('x(z) = cos(z)')
ylabel('y(z) = sin(z)')
zlabel('z')
```

3-D Surface Plots

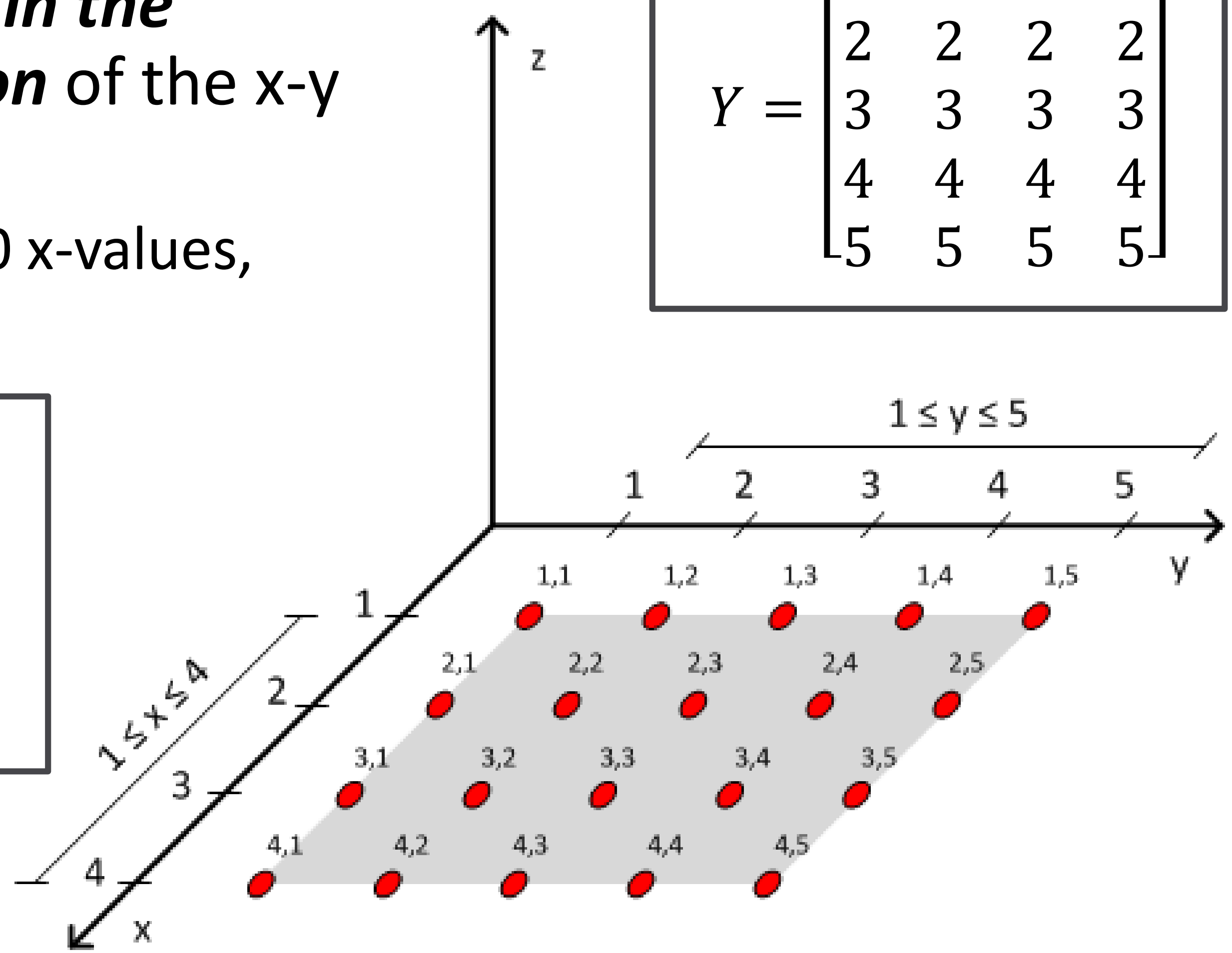
- Functions of two variables can be plotted as ***surfaces*** in 3-D space
  - $z = f(x, y)$
- Functions of one variable get evaluated at each point in the input variable vector
- Say we want to evaluate a function, $z = f(x, y)$, over a range of x and y values, e.g. $0 \leq x \leq 10$ and $0 \leq y \leq 5$
- Must evaluate z not only at each point in x and y, but ***at all possible combinations of x and y***

3-D Surface Plots - Input Matrices

- Must evaluate $z = f(x, y)$ at ***every point in the specified region*** of the x-y plane
  - ***20 points*** – 20 x-values, 20 y-values

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$



3-D Surface Plots – `meshgrid(…)`


```
[Xm,Ym] = meshgrid(x,y)
```

- `x` and `y` are vectors
  - Define ranges of x and y in the x-y plane
- `Xm` and `Ym` are matrices
  - All coordinates in the region of the x-y plane specified by vectors `x` and `y`
  - `size(Xm)=size(Ym)=(length(y),length(x))`
  - Rows of `Xm` are `x`
  - Columns of `Ym` are `y`

Function of Two Variables – Input Matrices

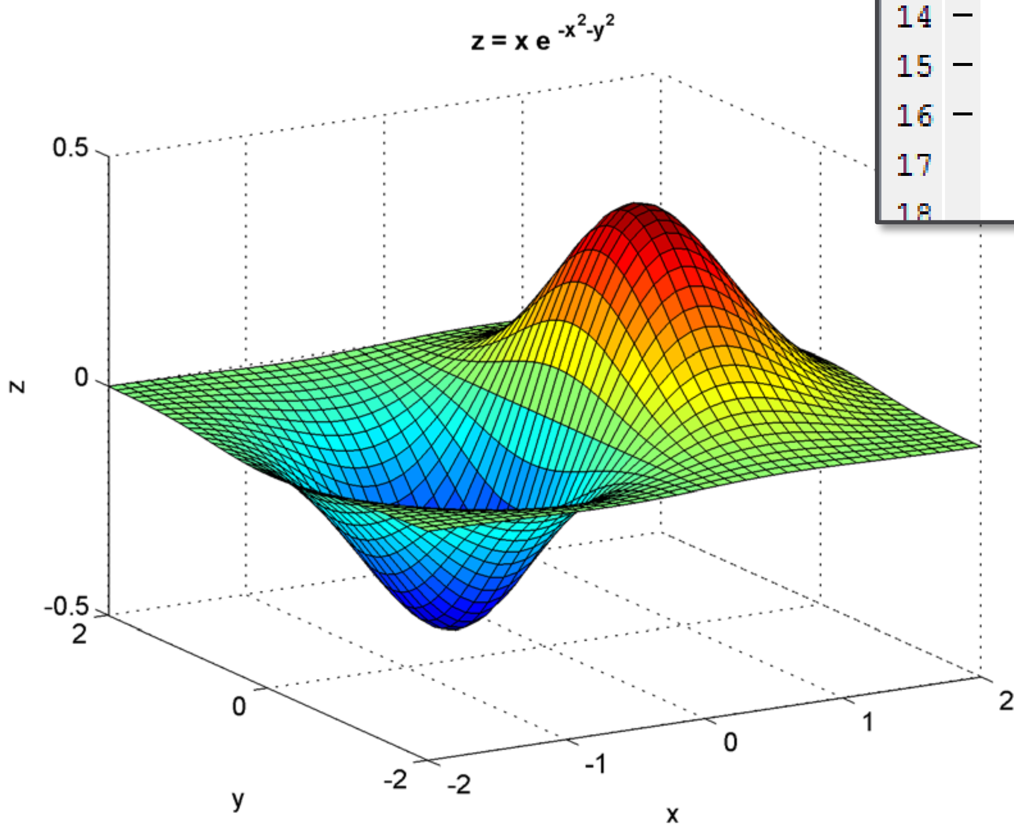
- The inputs to $z = f(x, y)$ are matrices
  - Create from x and y vectors using meshgrid(…)
- Example:
  - Evaluate $z = xe^{-x^2-y^2}$ for $-2 \le x \le 2, \quad -2 \le y \le 2$

```
x = -2:0.1:2;
y = -2:0.1:2;

[Xm,Ym] = meshgrid(x,y);
z = Xm.*exp(-Xm.^2-Ym.^2);
```

Surface Plot – surf(…)

`surf(x,y,z)`



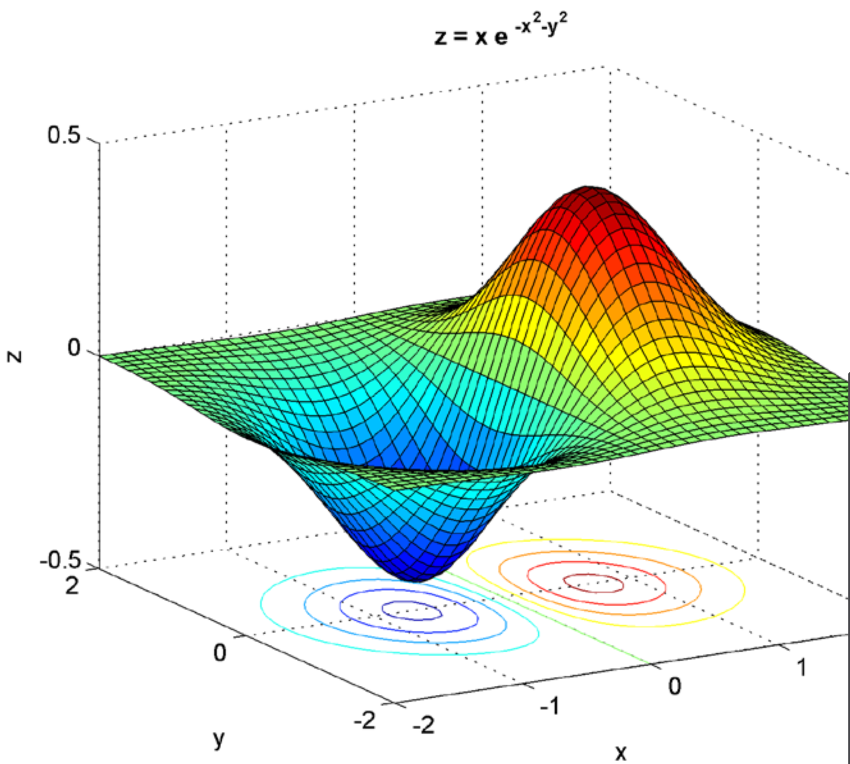

```
figure(1); clf
surf(Xm,Ym,z)
view([-30,20])
xlabel('x'); ylabel('y')
zlabel('z')
title('z = x e^{-x^2-y^2}',...
    'FontWeight','Bold')
```

- Use `view(azim,elev)` to set viewing angle of 3-D plots

Surface Plot with Contours – `surfc(…)`


```
surfc(x,y,z)
```

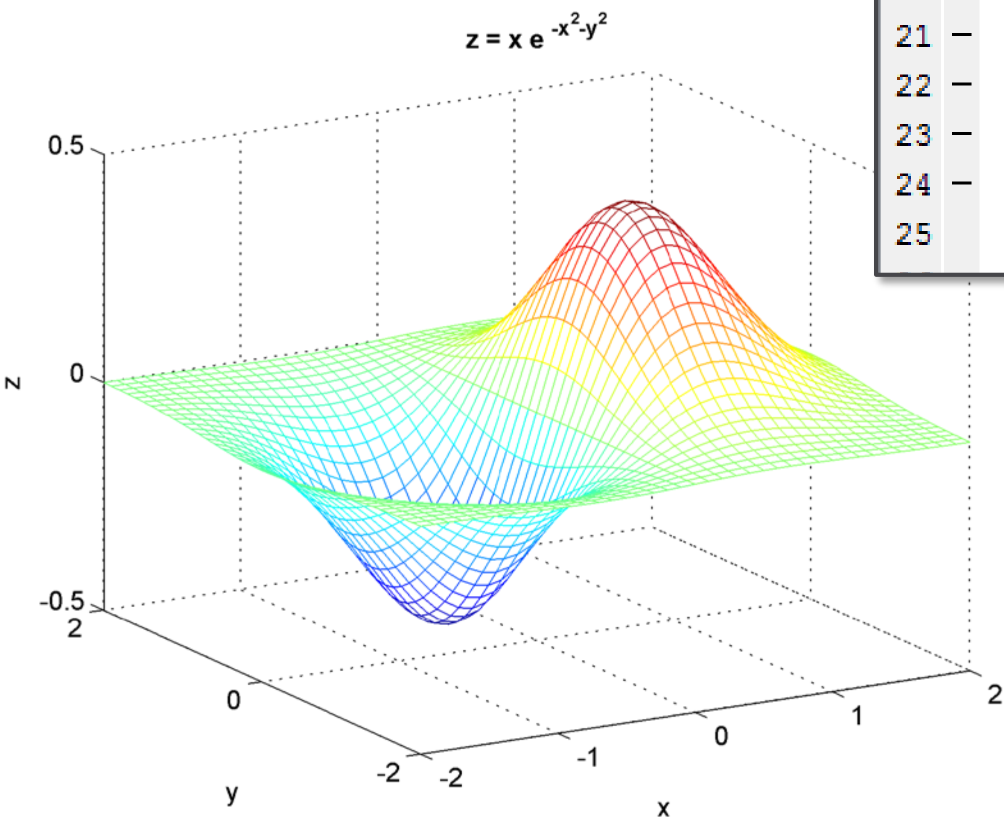
- Same surface as `surf(…)`
- Also draws a contour plot in the x-y plane




```
figure(3); clf
surfc(Xm,Ym,z)
view([-30,20])
xlabel('x'); ylabel('y')
zlabel('z')
title('z = x e^{-x^2-y^2}',...
    'FontWeight','Bold')
```

Mesh Plot – `mesh(…)`


```
mesh(x,y,z)
```



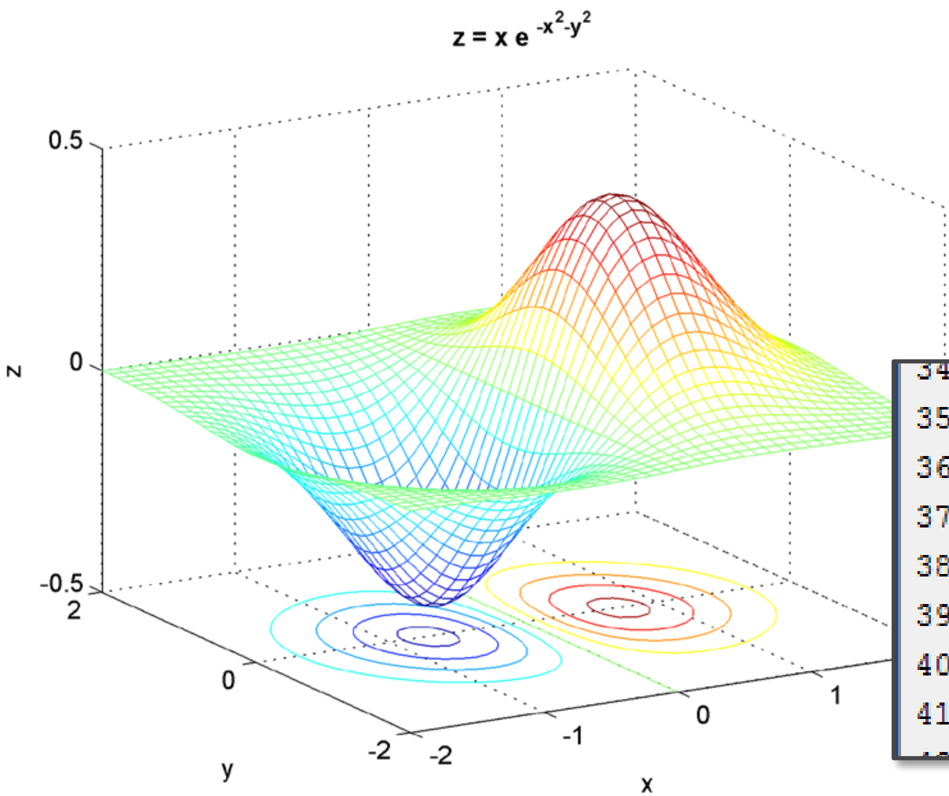

```
figure(2); clf
mesh(Xm,Ym,z)
view([-30,20])
xlabel('x'); ylabel('y')
zlabel('z')
title('z = x e^{-x^2-y^2}',...
    'FontWeight','Bold')
```

- like `surf(…)` but only wireframe is plotted

Mesh Plot with Contours – meshc(…)


```
meshc(x,y,z)
```

- Same wireframe as `mesh(…)`
- Also draws a contour plot in the x-y plane

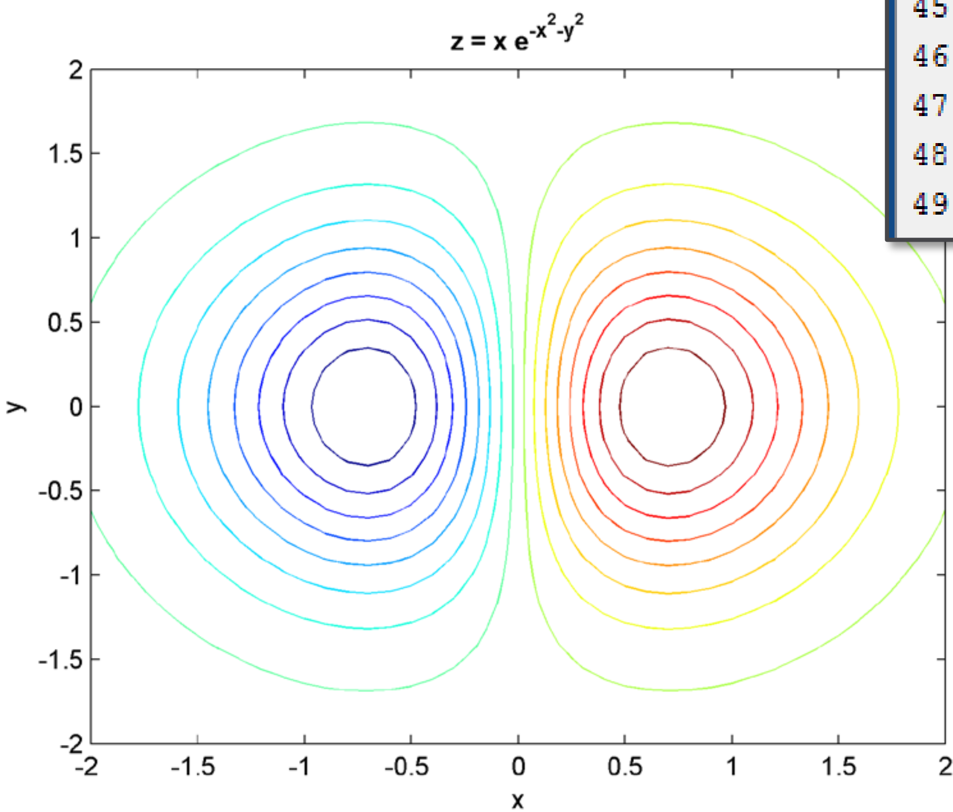



```
figure(4); clf
meshc(Xm,Ym,z)
view([-30,20])
xlabel('x'); ylabel('y')
zlabel('z')
title('z = x e^{-x^2-y^2}',...
    'FontWeight','Bold')
```

2-D Contour Plot – contour(…)


```
contour(x,y,z,N)
```

```
figure(5); clf
contour(Xm,Ym,z,16)
xlabel('x'); ylabel('y')
zlabel('z')
title('z = x e^{-x^2-y^2}',...
    'FontWeight','Bold')
```

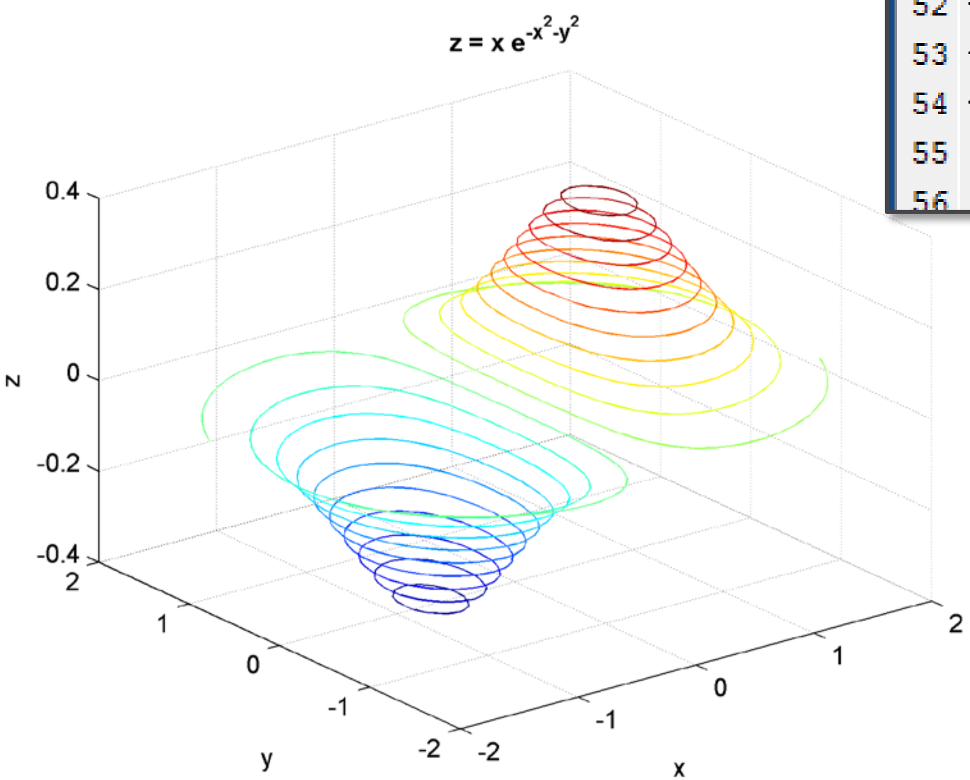


- A 2-D contour plot of the surface defined by z
- N contours drawn

3-D Contour Plot – contour3(…)


```
contour3(x,y,z,N)
```

```
figure(6); clf
contour3(Xm,Ym,z,20)
xlabel('x'); ylabel('y')
zlabel('z')
title('z = x e^{-x^2-y^2}',...
    'FontWeight','Bold')
```

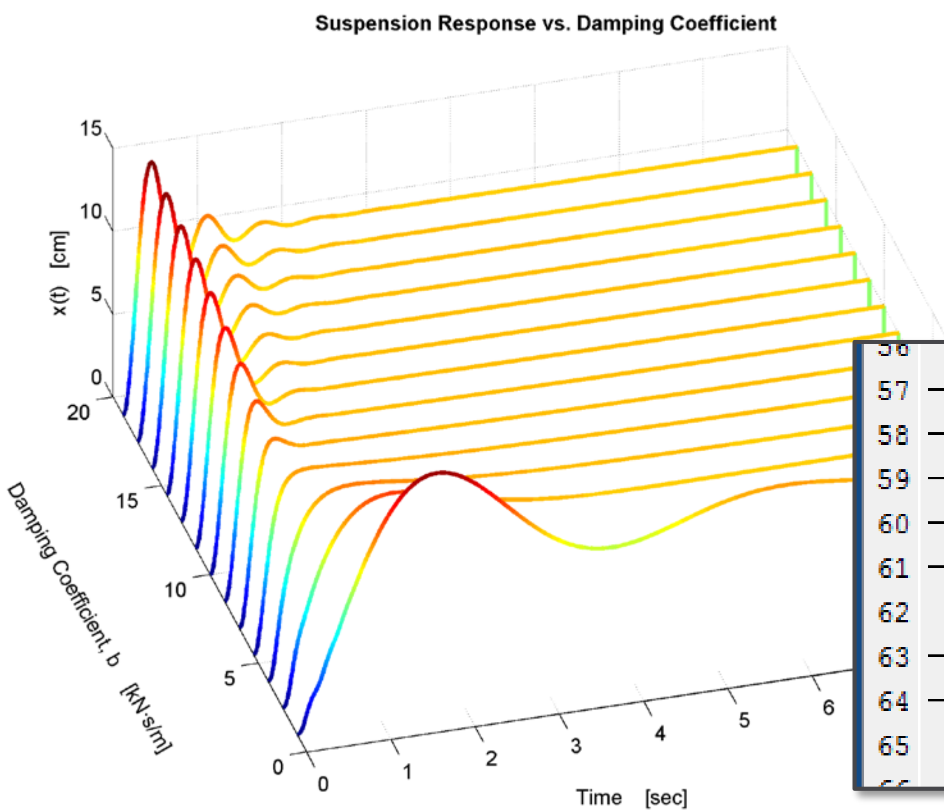


- A 3-D contour plot of the surface defined by z
- N contours drawn at their corresponding z values

Waterfall Plot – `waterfall(…)`


```
waterfall(x,y,z)
```

- Use `z'` for column-oriented data
- Useful for parameterized responses, spectrograms, etc.




```
figure(3); clf
waterfall(tm,bm,ywf')
view([-16 56])
xlabel('Time     [sec]');
ylabel('Damping Coefficient, b     [
    'Rotation',-60)
zlabel('x(t)    [cm]')
title('Suspension Response vs. Damp
    'FontWeight','Bold')
```

Animation – Creating Movies

- To create a movie, generate frames one-at-a-time and write successively to a video file
- Create and open a ***VideoWriterObject*** – where the video is written

```
vidfile=VideoWriter('filename','profile')

open(vidfile)
```

- Plot frames in a loop, grabbing each plot as a single frame

```
frame=getframe(H)
```

  - *H* is a ***handle to a figure or axes***
- Write each frame to `vidfile`

```
writeVideo(vidfile,frame)
```

- Movie stored in pwd in *filename*.avi by default – can specify format

Animation – Setting Video Properties


```
vidfile=VideoWriter('filename','profile')
```

- **`vidfile`** is a ***structure*** – can edit some of its fields to control video properties
  - ***Frame rate*** – for a frame interval of `dt`:

    ```
    vidfile.FrameRate=1/dt;
    ```

  - ***Quality*** – 0…100 – higher value, higher quality, larger file size – (default: 75):

    ```
    vidfile.Quality=90;
    ```

- **`profile`** – specifies the type of video encoding
  - E.g. `'Archival'`, `'Motion JPEG AVI'`(default), `'MPEG-4'`, etc.

Animation – Example

- Animate the motion of a projectile through the earth's gravitational field, neglecting drag
  - Initial velocity: v_0
  - Launch angle: θ_0
  - Gravitational acceleration: $g = 9.81\frac{m}{s^2}$
- Horizontal position: $x = v_0\cos(\theta_0)\cdot t$
- Vertical position: $y = v_0\sin(\theta_0)\cdot t - \frac{1}{2}g\cdot t^2$

Animation – Example

```
x = v0*cos(theta0)*t;
z = v0*sin(theta0)*t - 0.5*g*t.^2;

vidfile = VideoWriter('projectile');
vidfile.FrameRate = 1/dt;
vidfile.Quality = 100;
open(vidfile);

hfig = figure(1); clf
for j = 1:length(t)
    plot(x(j),z(j),'o',...
        'MarkerFaceColor','b',...
        'MarkerSize',8); hold on
    plot(x(1:j),z(1:j),'-b'); hold off
    xlabel('x'); ylabel('z')
    title('Projectile Trajectory',...
        'FontWeight','Bold')
    text(50,17,['t = ',...
        num2str(t(j),'%1.1f'),...
        ' sec'],'FontWeight','Bold')
    axis([0 70 0 20])
    M(j) = getframe(hfig);
    writeVideo(vidfile,M(j));
end

close(vidfile);
```

- Calculate trajectory
- Create and open the video file, specifying frame rate and quality
- Get figure handle when creating figure
- Loop, creating the video frame-by-frame
- Grab each frame and write it to `vidfile`
- Close `vidfile`

Animation – Example