SECTION 7:
THREE-DIMENSIONAL PLOTTING
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 – \texttt{plot3}(...) 

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

\begin{verbatim}
plot3(x, y, z, 'LineSpec', 'PropName', PropValue,...)
\end{verbatim}

- $x$, $y$, and $z$ inputs are \textit{vectors}
  - One $x$ and one $y$ value for each $z$ value
3-D Line Plot – `plot3(...)`

\[ x = \cos(z), \quad y = \sin(z), \quad 0 \leq z \leq 8\pi \]
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 \( y \)
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.

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

\[
X = \begin{bmatrix}
1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
4 & 4 & 4 & 4 \\
5 & 5 & 5 & 5 \\
\end{bmatrix}
\]
3-D Surface Plots – `meshgrid(…)`

```markdown
[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))`
  - **Xm** are **x**
  - **Ym** are **y**
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 $-2 \leq x \leq 2, -2 \leq y \leq 2$

```matlab
5 - x = -2:0.1:2;
6 - y = -2:0.1:2;
7
8 - [Xm, Ym] = meshgrid(x, y);
9 - z = Xm.*exp(-Xm.^2-Ym.^2);
10
```
Surface Plot – $\textit{surf}(\ldots)$

$\textit{surf}(x,y,z)$

$z = xe^{-x^2-y^2}$

Use $\textit{view}(\text{azim}, \text{elev})$ to set viewing angle of 3-D plots

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

- `surf(x,y,z)`

- Same surface as `surf(x,...)`

- Also draws a contour plot in the x-y plane
Mesh Plot – \texttt{mesh(...)}

\texttt{mesh(x,y,z)}

\begin{verbatim}
figure(2); clf
mesh(Xm,Ym,z)
view([-30,20])
xlabel('x'); ylabel('y')
zlabel('z')
title('z = x e^{\{-x^2-y^2\}}', 'FontSize', 'Bold')
\end{verbatim}

\begin{itemize}
\item like \texttt{surf(...)} but only wireframe is plotted
\end{itemize}
Mesh Plot with Contours – \texttt{meshc(\ldots)}

\texttt{meshc(x,y,z)}

- Same wireframe as \texttt{mesh(\ldots)}
- Also draws a contour plot in the x-y plane

```matlab
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 – \texttt{contour(...)}

\texttt{contour(x, y, z, N)}

\begin{itemize}
  \item A 2-D contour plot of the surface defined by \( z \)
  \item \( N \) contours drawn
\end{itemize}
3-D Contour Plot – \texttt{contour3(...)}

\texttt{contour3(x,y,z,N)}

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

\texttt{\begin{verbatim}
figure(6); clf
contour3(Xm,Ym,z,20)
xlabel('x'); ylabel('y')
zlabel('z')
title('z = x e^{-x^2-y^2}',...' '
'FontWeight','Bold')
\end{verbatim}}
Waterfall Plot – \textit{waterfall} (...) 

\begin{center}
\texttt{waterfall(x,y,z)}
\end{center}

- Use \( z' \) for column-oriented data
- Useful for parameterized responses, spectrograms, etc.
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
  
  ```python
  vidfile = VideoWriter(‘filename’, ‘profile’)
  open(vidfile)
  ```

- Plot frames in a loop, grabbing each plot as a single frame
  
  ```python
  frame = getframe(H)
  ```
  
  - *H* is a **handle to a figure or axes**

- Write each frame to **vidfile**
  
  ```python
  writeVideo(vidfile, frame)
  ```

- Movie stored in pwd in **filename.avi** by default – can specify format
Animation – Setting Video Properties

```python
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 $dt$:
    ```python
    vidfile.FrameRate=1/dt;
    ```
  
  - **Quality** – 0...100 – higher value, higher quality, larger file size – (default: 75):
    ```python
    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: $\theta_0$
  - 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

- 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`

Code:

```matlab
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);
```
Animation – Example

![Projectile Trajectory](image)

- **t = 0.0 sec**