CHAPTER 3
Graphs

CHAPTER 3: Graphs

3.1. Introduction

This chapter presents the key concepts necessary for visualizing data using MATLAB's graphical capabilities. Graphical representation is essential for interpreting data and effectively communicating results. You will learn how to create 2D and 3D graphs, manage graphical windows, and customize figures to improve readability. These skills are crucial for both academic and professional work, as they enable the clear and accessible presentation of complex information.

After studying this chapter, you should be able to:

- Create and manage 2D and 3D graphs in MATLAB.
- Understand different types of graphical representations, including Cartesian and polar coordinates.
- Customize graphs to enhance clarity and data communication.
- Use MATLAB's graphical tools to visually analyze and interpret data.

3.2. Managing Graphical Windows

In MATLAB, data visualization using graphs is essential for effectively analyzing, interpreting, and communicating results. Graphical windows serve as the primary interface where these visualizations are displayed. Proper management of graphical windows allows users to control the display, customize figures, and organize multiple graphs coherently.

3.2.1. Creating Graphical Windows

A new graphical window can be created using the figure command, which opens an empty figure where graphs can be drawn.

Matlab

```
figure; %Creates a new empty graphical window
```

3.2.2. Customizing Figures

MATLAB provides extensive flexibility for customizing graphs. You can add titles, axis labels, legends, and adjust other visual properties.

Matlab

```
plot(x, y);
title('Titre du Graphique');
xlabel('Axe X');
ylabel('Axe Y');
legend('Courbe 1');
```

3.2.3. Managing Multiple Figures

When working with multiple graphs, you can manage several figures simultaneously by specifying a figure number.

Matlab

```
1. figure(1);  % Selects or creates figure 1
2. plot(x, y1);

3. figure(2);  % Selects or creates figure 2
4. plot(x, y2);
```

3.2.4. Subplots and Multiple Axes

To display multiple graphs within the same window, MATLAB allows users to divide a figure into subplots using the subplot command.

Matlab

```
1. subplot(2, 1, 1);  % First subplot, 2 rows, 1 column, 1st position
2. plot(x, y1);
3. subplot(2, 1, 2);  % Second subplot, 2 rows, 1 column, 2nd position
4. plot(x, y2);
```

3.2.5. Saving and Exporting Figures

Once a figure has been created and customized, it can be saved in different formats such as PNG, JPEG, or PDF for future use.

```
1. saveas(gcf, 'nom_du_fichier.png');
```

3.2.6. Manipulating Figure Properties

MATLAB also allows modifying specific properties of figures and axes after creation, providing greater control over their appearance.

Matlab

```
1. set(gca, 'XLim', [0 10]);  % Sets X-axis limits
2. set(gca, 'YLim', [0 20]);  % Sets Y-axis limits
```

3.2.7. Example: Creating and Managing Graphs

A practical example of graphical window management in MATLAB using the functions $y_1 = \cos(x)$, $y_2 = \sin(x)$, et $y_3 = 2x + 1$.

Matlab

```
1. % Define the range of x values
2. x = linspace(0, 2*pi, 100);  % Génère 100 points entre 0 et 2*pi

3. % Compute values for y1, y2, and y3
4. y1 = cos(x);       % Cosine function
5. y2 = sin(x);       %  Sine function
6. y3 = 2*x + 1;      % Linear function 2x + 1
```

```
% Create a new graphical window for y1 and y2
figure;

% First subplot for y1
subplot(3, 1, 1);   % Divide figure into 3 rows, 1 column, select 1st part
plot(x, y1, 'b-', 'LineWidth', 2);  % Plot y1 in blue with a line width of 2
title('y_1 = cos(x)');
xlabel('x');
ylabel('y_1');
grid on;  % Enable grid for better readability

% Second subplot for y2
subplot(3, 1, 2);   % Select 2nd part
plot(x, y2, 'r-', 'LineWidth', 2);  % Plot y2 in red with a line width of 2
title('y_2 = sin(x)');
xlabel('x');
ylabel('y_2');
grid on;

% Third subplot for y3
subplot(3, 1, 3);   % Select 3rd part
plot(x, y3, 'g-', 'LineWidth', 2);  % Plot y3 in green with a line width of 2
title('y_3 = 2x + 1');
xlabel('x');
ylabel('y_3');
grid on;

% Save the figure
saveas(gcf, 'graphique_cos_sin_lin.png');  % Save figure as PNG file
```

3.2.7.1. Explanation of the Example

Defining the x Range

The linspace function is used to generate 100 evenly spaced points between

Function Computation :

- $y1 = cos(x)$
- $y2 = sin(x)$
- $y3 = 2x + 1$

Creating Graphs:

The subplot command is used to divide the figure into three subplots, each containing a graph.

- The functions $y_1, y_2, and\ y_3$ are plotted in their respective subplots with distinct colors (blue, red, green) and a line width of 2 for better visibility.

Customization:

Titles, axis labels (xlabel, ylabel), and grids (grid on) are added to each graph to improve readability and presentation.

Saving the Figure :

The complete figure is saved as graph_cos_sin_lin.png in the current working directory.

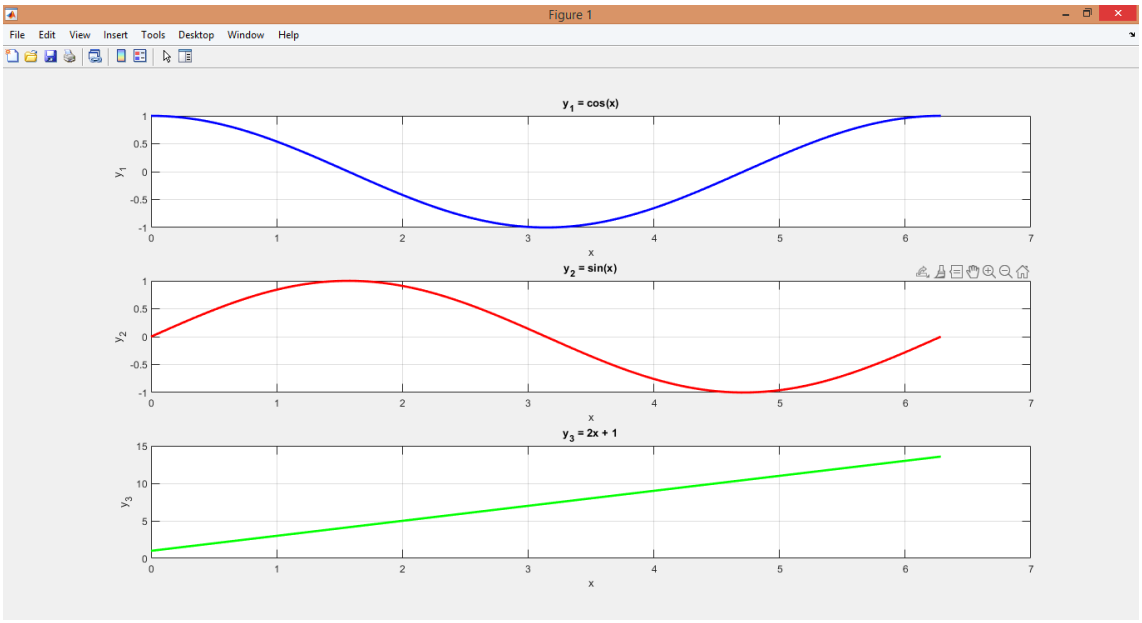

Figure 3.1: Presentation of three curves $y_1, y_2, and\ y_3$.

Result

Running this code will produce a figure consisting of three vertically aligned graphs:

1. The first graph shows the function $y1 = cos(x)$ in blue.
2. The second graph shows the function $y2 = sin(x)$ in red.
3. The third graph shows the linear function $y3 = 2x + 1$ in green.

Each graph is clearly annotated with a title and axis labels, providing a clear and professional visual presentation of the three functions.

3.3. 2D Graphical Representation

2D graphical representation in MATLAB is a powerful tool for visualizing relationships between variables. This type of graph is commonly used to plot curves, scatter plots, and other data forms in a two-dimensional plane.

3.3.1. Simple Curves

The primary function for creating 2D plots is plot. It allows plotting curves by representing the values of two variables (x and y coordinates) on a Cartesian plane.

Matlab

```
x = 0:0.1:10;      % Generates an x vector from 0 to 10 with a step of 0.1
y = sin(x);        % Computes y = sin(x)
figure;            % Creates a new figure window
plot(x, y);        % Plots the curve y = sin(x)
xlabel('x');       % Adds a label for the x-axis
ylabel('y');       % Adds a label for the y-axis
title('Curve of y = sin(x)');  % Adds a title to the graph
grid on;           % Enables the grid
```

3.3.2. Scatter Plots

Scatter plots show the relationship between two variables as individual points. The function used to create a scatter plot is scatter.

Matlab

```
x = randn(100,1);   % Generates 100 random points for x
y = randn(100,1);   % Generates 100 random points for y
figure;
scatter(x, y, 'filled');  % Plots a scatter diagram with filled points
xlabel('x');
ylabel('y');
title('Scatter Plot');
grid on;
```

3.3.3. Multiple Curves

It is possible to plot multiple curves on the same graph by calling the plot function multiple times or by passing multiple pairs of vectors to the function.

Matlab

```
y1 = sin(x);
y2 = cos(x);
figure;
plot(x, y1, '-r', x, y2, '--b');  % Plots y1 in red (solid line) and y2 in blue (dashed line)
legend('sin(x)', 'cos(x)');       % Adds a legend to identify the curves
xlabel('x');
ylabel('y');
title('Curves of sin(x) and cos(x)');
grid on;
```

3.3.4. Subplots

To display multiple graphs in a single figure, use the subplot function.

Matlab

```
1. figure;
2. subplot(2,1,1);      % 2 rows, 1 column, 1st graph
3. plot(x, y1);
4. title('y = sin(x)');
5. subplot(2,1,2);      % 2nd graph
6. plot(x, y2);
7. title('y = cos(x)');
```

3.4. Cartesian Coordinate Graphs

Cartesian coordinate graphs are the most commonly used for representing mathematical functions or empirical data on a 2D plane, where a coordinate pair (x, y) defines each point.

3.4.1. Cartesian Coordinate System

Two perpendicular axes define the Cartesian coordinate system: the x-axis (horizontal) and the y-axis (vertical). Points are placed on the graph based on their coordinates (x, y).

Example: Plotting a Parabola

Matlab

```
1. x = -10:0.1:10;      % Generates an x vector from -10 to 10 with a step of 0.1
2. y = x.^2;            % Computes y = x^2
3. figure;
4. plot(x, y, '-g');    % Plots the parabola y = x^2 in green
5. xlabel('x');
6. ylabel('y');
7. title('Parabola y = x^2');
8. grid on;
```

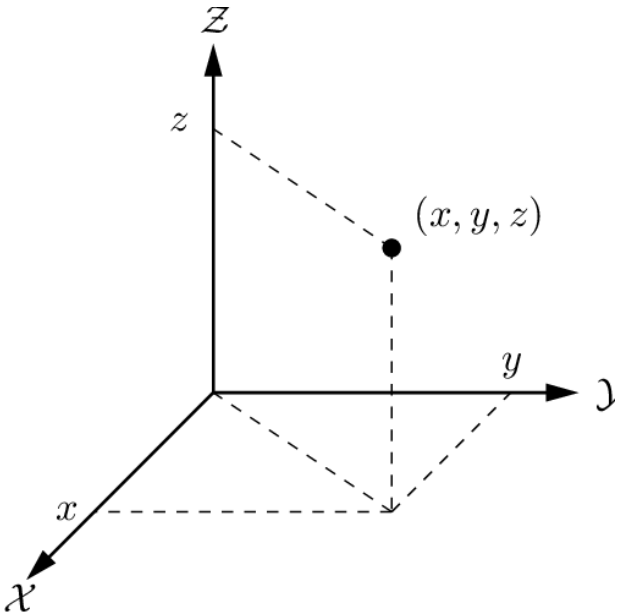

Figure 3.2: Cartesian coordinate system

3.4.2. Adding Reference Lines

You can add horizontal and vertical reference lines to highlight specific points or axes using xline and yline.

Matlab

```
figure;
plot(x, y);
hold on;          % Allows adding multiple elements to the same graph
xline(0, '--r');  % Adds a vertical line at x = 0
yline(0, '--b');  % Adds a horizontal line at y = 0
title('Parabola with Reference Lines');
hold off;
```

3.4.3. 2D Contour Plots

Contour plots in Cartesian coordinates show lines where a function has a constant value (contour lines).

Matlab

```
[X, Y] = meshgrid(-2:0.1:2, -2:0.1:2);  % Creates a 2D grid of points
Z = X.^2 + Y.^2;                % Computes Z = X^2 + Y^2
figure;
contour(X, Y, Z);               % Generates a contour plot
xlabel('X');
ylabel('Y');
title('Contour Plot of Z = X^2 + Y^2');
grid on;
```

3.4.4. Logarithmic Plots

When data varies across several orders of magnitude, logarithmic plots can help visualize these variations more effectively.

Matlab

```
x = logspace(0, 2, 100);  % Generates 100 points between 10^0 and 10^2
y = x.^2;
figure;
loglog(x, y);          % Plots a logarithmic scale graph on both axes
xlabel('x');
ylabel('y');
title('Logarithmic Plot of y = x^2');
grid on;
```

3.5. Improving Figure Readability

When creating graphs in MATLAB, ensuring clarity and readability is crucial for effectively conveying information. MATLAB offers numerous options for improving figure readability by customizing elements such as axes, titles, legends, annotations, and more.

3.5.1. Adding Titles and Labels

Titles and axis labels help contextualize graphs. Using clear and precise names for these elements is essential for understanding the graph.

Matlab

```
x = 0:0.1:10;
y = exp(-0.1*x).*sin(2*x);
figure;
plot(x, y);
title('Damped Oscillation');  % Graph title
xlabel('Time (s)');           % x-axis label
ylabel('Amplitude');          % y-axis label
```

3.5.2. Adjusting Font Sizes

Adjusting font sizes for titles, labels, and legends can improve readability, especially for presentations or prints.

Matlab

```
plot(x, y);
title('Damped Oscillation', 'FontSize', 16);
xlabel('Time (s)', 'FontSize', 14);
ylabel('Amplitude', 'FontSize', 14);
```

3.5.3. Adding a Legend

A legend helps identify different curves or data series on a graph. Using meaningful names in legends makes the graph more understandable.

Matlab

```
y1 = sin(x);
y2 = cos(x);
figure;
plot(x, y1, '-r', x, y2, '--b');
legend('sin(x)', 'cos(x)', 'Location', 'northeast');
```

3.5.4. Adjusting Axis Limits

Manually adjusting axis limits can help focus attention on important data.

Matlab

```
plot(x, y);
axis([0 10 -1 1]);  % Sets x and y axis limits
```

3.5.5. Saving High-Resolution Graphs

Saving a graph in high quality is essential for publications or presentations.

Matlab

```
saveas(gcf, 'my_graph.png');  % Saves the figure in PNG format
print('my_graph', '-dpng', '-r300');  % Saves with a resolution of 300 dpi
```

3.5.6. Adding Grids

Grids help improve the visualization of points relative to the axes. They can be enabled or disabled using grid on or grid off.

Matlab

```
1. plot(x, y);
2. grid on;  % Activates the grid on the plot
```

3.5.7. Annotating the Plot

Adding annotations such as arrows or text can help highlight specific points or explain certain aspects of the graph.

Matlab

```
1. plot(x, y);
2. text(5, 0.5, 'Reference Point', 'FontSize', 12);  % Adds a text annotation
```

3.5.8. Using Subplots to Compare Multiple Graphs

If you need to compare multiple datasets or different visualizations, using subplots in a single figure can facilitate comparison.

Matlab

```
1. figure;
2. subplot(2,1,1);
3. plot(x, y1);
4. title('sin(x)');
5. grid on;
6. subplot(2,1,2);
7. plot(x, y2);
8. title('cos(x)');
9. grid on;
```

3.5.9. Choosing an Appropriate Color Palette

For plots with multiple data series, it is important to choose colors that are easily distinguishable. MATLAB provides color palettes, but you can also define your own for better readability.

Matlab

```
1. figure;
2. plot(x, y1, 'Color', [0.8500, 0.3250, 0.0980], 'LineWidth', 2);  % Red-orange curve
3. hold on;
4. plot(x, y2, 'Color', [0, 0.4470, 0.7410], 'LineWidth', 2);      % Blue curve
5. legend('sin(x)', 'cos(x)', 'Location', 'best');
6. hold off;
```

3.5.10. Saving the Graph with High Resolution

Once you have customized your plot, it is important to save it in high-quality format, especially if you need to include it in publications or presentations.

Matlab

```
1. saveas(gcf, 'my_graph.png');  % Saves the figure in PNG format
2. print('my_graph','-dpng','-r300');  % Saves with 300 dpi resolution
```

3.6. Polar Coordinate Plots

Polar coordinate plots are used to represent data where each point is defined by an angle and a distance from the origin (or pole). MATLAB provides powerful tools for creating and customizing these plots, enabling the visualization of data that varies with angle.

3.6.1. Introduction to Polar Coordinates

In a polar coordinate system, each point is defined by an angle θ (measured in radians or degrees) and a distance r (radius) from the origin. The angle is typically measured from the positive horizontal axis.

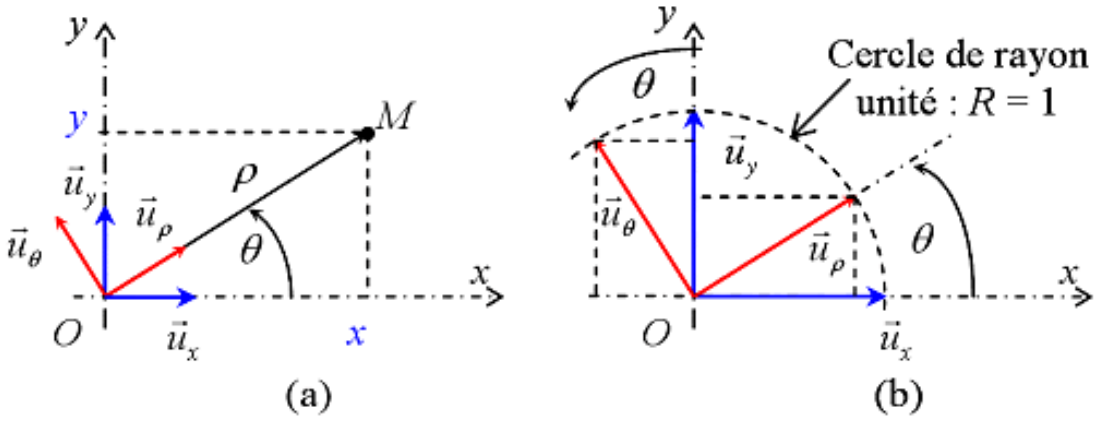

Figure 3.3: Coordinate System (a) Cartesian, (b) Polar

3.6.2. Creating a Polar Coordinate Plot

The basic function for creating a polar coordinate plot in MATLAB is *polarplot*. This function allows plotting curves using angles and radii.

Example: Plotting a Simple Curve

Matlab

```
1. theta = 0:0.01:2*pi;  % Angle from 0 to 2π radians
2. r = 1 + cos(theta);  % Radius r as a function of angle
3. figure;
4. polarplot(theta, r);  % Plots the graph in polar coordinates
5. title('Polar Coordinate Plot of r = 1 + cos(\theta)');
```

3.6.3. Plotting Multiple Curves in Polar Coordinates

Just like Cartesian plots, you can plot multiple curves on the same polar coordinate plot.

Example:

Matlab

```
1. r1 = 1 + cos(theta);
2. r2 = 1 + sin(theta);
3. figure;
4. polarplot(theta, r1, '-r');  % Red curve
5. hold on;
6. polarplot(theta, r2, '--b'); % Blue dashed curve
7. title('Multiple Curves in Polar Coordinates');
8. legend('r = 1 + cos(\theta)', 'r = 1 + sin(\theta)');
9. hold off;
```

3.6.4. Customizing Polar Plots

MATLAB provides several options for customizing polar coordinate plots, including colors, line styles, and annotations.

Matlab

```
1. polarplot(theta, r, 'g--o', 'LineWidth', 2, 'MarkerSize', 6);  % Green dashed line with markers
2. title('Customized Polar Coordinate Plot');
```

3.6.5. Rose Charts (Rose Diagrams)

Rose charts are a variant of polar plots that show the angular distribution of data. MATLAB provides the polarhistogram function to create this type of chart.

Matlab

```
1. data = randn(1, 1000);          % Generates 1000 random data points
2. theta = mod(data, 2*pi);        % Modifies data to be in the [0, 2π] range
3. figure;
4. polarhistogram(theta, 20);      % Creates a polar histogram with 20 sectors
5. title('Rose Diagram');
```

3.6.6. Plotting Rose Curves

Rose curves are a type of curve plotted in polar coordinates that create petal-like patterns. They are defined by the equation:

$$r = a \cdot cos(k\theta) \; or \; r = a \cdot \sin(k\theta)$$

Matlab

```
1. k = 4;
2. r = cos(k * theta);  % Rose curve with 4 petals
3. figure;
4. polarplot(theta, r);
5. title(['Rose Curve with ', num2str(k), ' Petals']);
```

3.6.7. 3D Polar Plots

Although standard polar plots are 2D, it is possible to create 3D polar representations by combining polar coordinates with surface or mesh plots. This requires manually converting polar coordinates to Cartesian coordinates.

Example: Creating a Surface in Polar Coordinates

Matlab

```
[theta, r] = meshgrid(0:0.01:2*pi, 0:0.1:5);  % Grid of points in polar coordinates
X = r .* cos(theta);                % Conversion to Cartesian coordinates
Y = r .* sin(theta);
Z = r;                              % Z depends only on radius
figure;
mesh(X, Y, Z);                      % Creates a 3D surface
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Surface in Polar Coordinates');
```

3.6.8. Annotating Polar Plots

Just like Cartesian plots, you can add annotations to your polar plots to indicate points of interest or specific areas.

Matlab

```
polarplot(theta, r);
title('Annotated Polar Coordinate Plot');
text(pi/4, 1.5, 'Annotation at \theta = \pi/4', 'FontSize', 12, 'Color', 'red');  % Adds an annotation
```

3.7. 2D and 3D Charts

Charts allow data visualization in a clear and intuitive manner. 2D graphs are used to represent relationships between two variables, while 3D graphs enable visualization of relationships among three or more variables, adding depth to data representation.

3.7.1. 2D Charts

2D charts are the most common type of graphs, used to visualize relationships between two variables on a two-dimensional plane.

3.7.1.1. Bar Charts

Bar charts are used to represent categorical data, where each bar represents a category.

Matlab

```
categories = {'A', 'B', 'C', 'D'};
values = [5, 3, 8, 2];
figure;
bar(values);  % Creates a bar chart
```

```
5. set(gca, 'xticklabel', categories);
6. xlabel('Categories');
7. ylabel('Values');
8. title('Bar Chart');
```

3.7.1.2. Histograms

Histograms represent data distribution by grouping values into intervals called "bins."

Matlab

```
1. data = randn(1000,1); % Generates 1000 data points following a normal distribution
2. figure;
3. histogram(data); % Creates a histogram
4. xlabel('Values');
5. ylabel('Frequency');
6. title('Histogram');
```

3.7.1.3. Scatter Plots

Scatter plots display the relationship between two continuous variables as individual points.

Matlab

```
1. x = randn(100,1);
2. y = randn(100,1);
3. figure;
4. scatter(x, y, 'filled'); % Creates a scatter plot
5. xlabel('Variable X');
6. ylabel('Variable Y');
7. title('Scatter Plot');
```

3.7.2. 3D Charts

3D charts allow visualization of relationships among three variables, adding an extra dimension to graphical representation.

3.7.2.1. Surface Plots

Surface plots display a 3D surface where z is a function of x and y.

Matlab

```
1. [X, Y] = meshgrid(-5:0.5:5, -5:0.5:5); % Creates a grid of values
2. Z = X.^2 + Y.^2; % Computes Z values
3. figure;
4. surf(X, Y, Z); % Creates a surface plot
5. xlabel('X');
6. ylabel('Y');
7. zlabel('Z');
8. title('Surface Plot');
```

3.7.2.2. Mesh Plots

Mesh plots are similar to surface plots but only display the mesh lines without filling the surfaces.

Matlab

```
1. figure;
2. mesh(X, Y, Z);  % Creates a mesh plot
3. xlabel('X');
4. ylabel('Y');
5. zlabel('Z');
6. title('Mesh Plot');
```

3.7.2.3. 3D Scatter Plots

Like 2D scatter plots, 3D scatter plots display the relationship among three variables using points in a three-dimensional space.

Matlab

```
1. x = randn(100,1);
2. y = randn(100,1);
3. z = randn(100,1);
4. figure;
5. scatter3(x, y, z, 'filled');  % Creates a 3D scatter plot
6. xlabel('X');
7. ylabel('Y');
8. zlabel('Z');
9. title('3D Scatter Plot');
```

3.7.2.4. 3D Parametric Curves

3D parametric curves are used to plot curves defined by parametric functions in three dimensions.

Matlab

```
1. t = 0:0.01:10;
2. x = sin(t);
3. y = cos(t);
4. z = t;
5. figure;
6. plot3(x, y, z);  % Creates a 3D parametric curve
7. xlabel('X');
8. ylabel('Y');
9. zlabel('Z');
10. title('3D Parametric Curve');
11. grid on;
```

3.7.2.5. 3D Contour Plots

3D contour plots display level curves of a surface, helping visualize altitude variations on a surface.

Matlab

```
1. figure;
2. contour3(X, Y, Z); % Creates a 3D contour plot
3. xlabel('X');
4. ylabel('Y');
5. zlabel('Z');
6. title('3D Contour Plot');
```

3.8. 3D Curves

3D curves represent trajectories or relationships among three variables in a three-dimensional space. MATLAB provides multiple methods for plotting 3D curves depending on the data and desired analysis. These curves can be used to visualize trajectories, solutions of dynamic systems, or complex variable relationships.

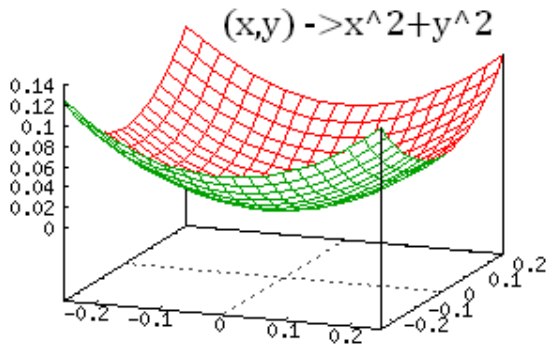

Figure 3.4: curved 3D model, $z = f(x, y) = x^2 + y^2$

3.8.1. 3D Parametric Curves

3D parametric curves are defined by three functions x(t), y(t), and z(t) that depend on a parameter t. These curves allow trajectories to be plotted in three dimensions.

Example: In this example, the curve represents a 3D helix, where x and y are trigonometric functions of t, and z increases linearly with t.

Matlab

```
1. t = 0:0.01:10;
2. x = sin(t);
3. y = cos(t);
4. z = t;
5. figure;
6. plot3(x, y, z);
7. xlabel('X');
8. ylabel('Y');
9. zlabel('Z');
```

10. title('3D Parametric Curve');

11. grid on;

3.8.2. 3D Spiral Plots

3D spirals are another example of parametric curves where the x, y, and z coordinates are all defined in terms of a common parameter.

Example: This curve represents a 3D spiral that wraps around the z axis.

Matlab

1. *t = linspace(0, 10*pi, 1000);*
2. *x = cos(t);*
3. *y = sin(t);*
4. *z = t;*
5. *figure;*
6. *plot3(x, y, z);*
7. *xlabel('X');*
8. *ylabel('Y');*
9. *zlabel('Z');*
10. *title('3D Spiral');*
11. *grid on;*

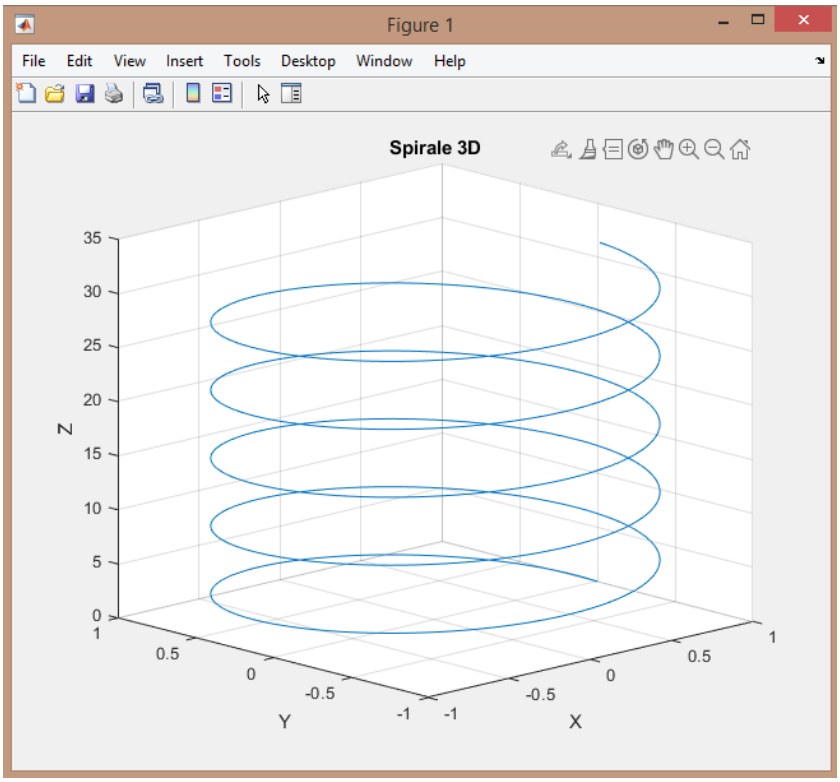

Figure 3.5: Curve represents a 3D spiral

3.8.3. 3D Trajectories

3D trajectories are used to visualize the path of a particle or moving point in three-dimensional space. These curves can be used to model dynamic systems, particle motion, or object trajectories.

Example: In this example, the curve represents a decreasing 3D spiral, where x and y follow exponential functions and z increases linearly.

Matlab

```
t = linspace(0, 10, 500);
x = cos(t) .* exp(-0.1*t);
y = sin(t) .* exp(-0.1*t);
z = t;
figure;
plot3(x, y, z, 'LineWidth', 2);
xlabel('X');
ylabel('Y');
zlabel('Z');
title('3D Trajectory');
grid on;
```

3.8.4. Data Visualization with 3D Curves

3D curves can also be used to visualize complex data sets, where each point on the curve represents a single observation with three dimensions.

Example: Visualizing a Time Series

Matlab

```
t = linspace(0, 10, 100);
x = sin(t);
y = cos(2*t);
z = t;
figure;
plot3(x, y, z, '-o'); % Plot the curve with markers
xlabel('Variable X');
ylabel('Variable Y');
zlabel('Time');
title('3D Time Series Visualization');
grid on;
```

3.8.5. Customizing 3D Curves

MATLAB allows you to customize 3D curves by changing the color, line style, markers, and adding annotations.

Matlab

```
figure;
plot3(x, y, z, '--r', 'LineWidth', 2, 'Marker', 'o'); % Red dashed curve with round markers
xlabel('X');
ylabel('Y');
zlabel('Z');
```

6. *title('Custom 3D Curve');*
7. *grid on;*

3.9. 3D Surfaces

3D surface plots are used to represent relationships between three variables where one variable depends on the other two. These plots are particularly useful for visualizing functions of two variables, plots, parametric surfaces, or complex data distributions. MATLAB offers several functions for creating surface plots, such as surf, mesh, contour, and surfc.

3.9.1. Basic Surface Plots

Surface plotting is one of the most common 3D visualizations. It allows you to represent a colored surface based on x, y, and z coordinates, where $z = f(x, y)$.

Example: In this example, the surface represents a sinusoidal function based on the radial distance from the origin.

Matlab

1. *[X, Y] = meshgrid(-5:0.1:5, -5:0.1:5);*
2. *Z = sin(sqrt(X.^2 + Y.^2));*
3. *figure;*
4. *surf(X, Y, Z);*
5. *xlabel('X');*
6. *ylabel('Y');*
7. *zlabel('Z');*
8. *title('Surface Plot of sin(sqrt(X^2 + Y^2))');*
9. *colorbar;*

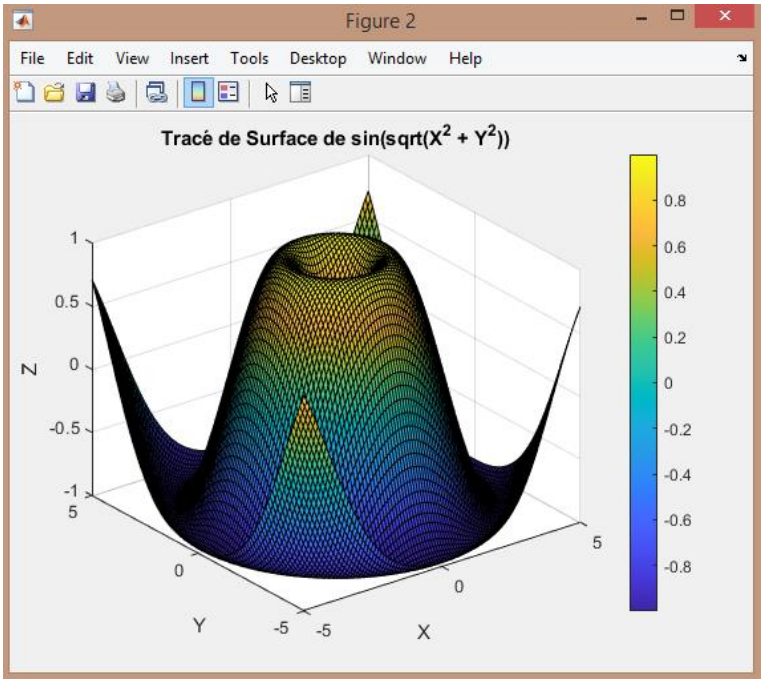

Figure 3.6: Sine curve based on radial distance from the origin

3.9.2. Mesh Plots

Mesh plots are similar to surface plots, but they only show the mesh lines without filling the surfaces. This can be useful for seeing the underlying structure of the surface.

Matlab

```
figure;
mesh(X, Y, Z);
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Mesh Plot of sin(sqrt(X^2 + Y^2))');
grid on;
```

3.9.3. Surface Plots with Contours

Surface plots can be combined with contours to show level lines across the surface base, helping to visualize height variations.

Matlab

```
figure;
surfc(X, Y, Z);
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Surface Plot with Contours');
colorbar;
```

3.9.4. Parametric Surface Plots

Parametric surfaces allow you to plot surfaces where the x, y, and z coordinates are all defined in terms of two other parameters, u and v.

Example: Parametric Surface

Matlab

```
[u, v] = meshgrid(0:0.1:2*pi, 0:0.1:pi);
x = sin(u) .* cos(v);
y = sin(u) .* sin(v);
z = cos(u);
figure;
surf(x, y, z);
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Parametric Surface');
colorbar;
```

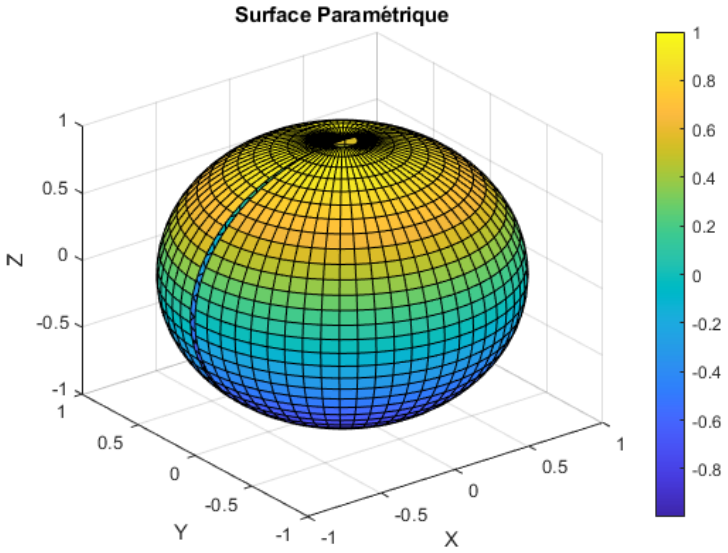

Figure 3.7: Parametric sphere

3.9.5. Colored Surface Plots

You can also modify the surface colors based on height to improve the readability and aesthetics of the graph.

Example: In this example, peaks generates a complex surface, and the shading interp function smooths the color transitions.

Matlab

```
Z = peaks(50);
figure;
surf(Z);
colormap jet;
shading interp;
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Surface with Height-based Colors');
colorbar;
```

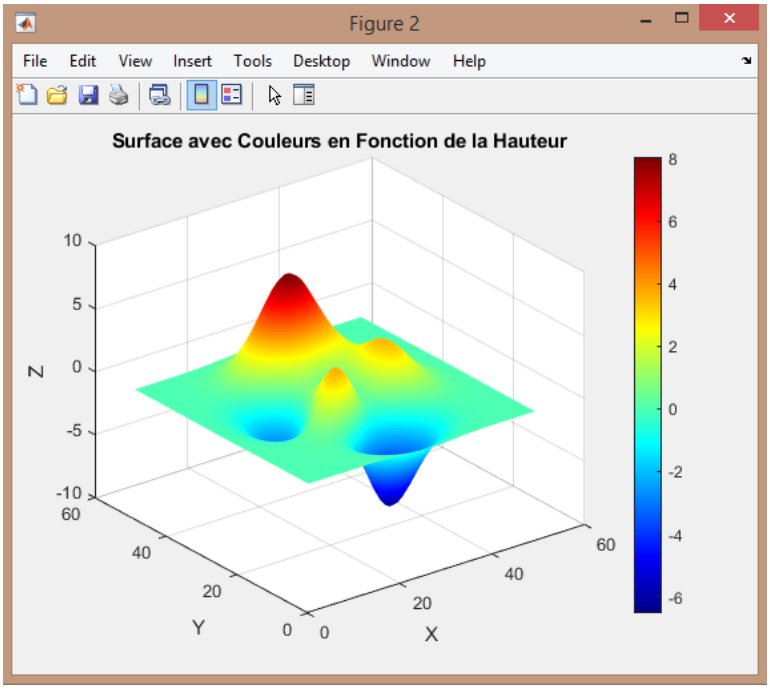

Figure 3.8: Surface with Colors Depending on Height

3.9.6. Surface Plots with Transparency

MATLAB also allows you to add transparency to surfaces to allow for better visualization of multiple overlapping surfaces.

Example: This code overlaps two transparent surfaces, allowing you to see how they interact visually.

```
Z1 = sin(X) .* cos(Y);                % Première surface
Z2 = cos(X) .* sin(Y);                % Deuxième surface
figure;
surf(X, Y, Z1, 'FaceAlpha', 0.5);     % Crée la première surface avec 50% de transparence
hold on;
surf(X, Y, Z2, 'FaceAlpha', 0.5);     % Crée la deuxième surface avec 50% de transparence
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Tracé de Surfaces avec Transparence');
colorbar;
hold off;
```

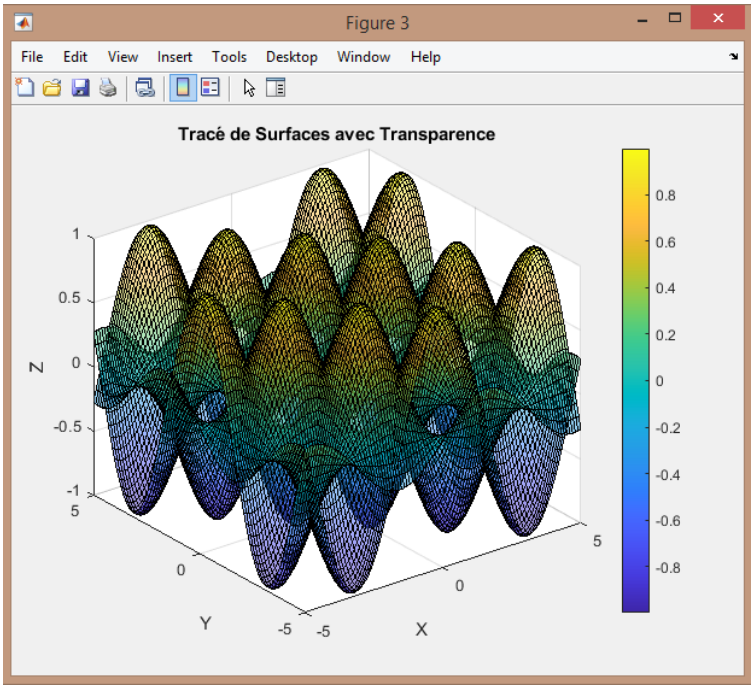

Figure 3.9: Surface Plotting with Transparency

3.10. Conclusion

This chapter has provided you with valuable skills in data visualization using MATLAB's graphical capabilities. By learning how to create and customize 2D and 3D graphs, you will be able to effectively communicate complex information and insights from your data. The ability to represent data graphically is an essential tool in both academic research and professional practice, enabling clearer understanding and better decision-making based on visualized data.