3. Making Plots with MATLAB

3.1 Basic Plotting

Suppose we want to plot the function \( y = \sin(x) \), over the interval \([-\pi, \pi]\). First we have to sample points from this function, say at evenly spaced points at a distance of \( \pi/20 \) apart from each other.

\[
\begin{align*}
>> x &= -\pi : \pi/20 : \pi; \quad \% \text{range where we sample} \\
>> s &= \sin(x); \quad \% \text{generate sample points} \\
>> plot(x, s); \quad \% \text{make the plot}
\end{align*}
\]

By default, MATLAB draws a smooth curve through those sample points. If we wish to plot the cosine function in the same figure, we must use the `hold on` directive:

\[
\begin{align*}
>> hold\ on & \quad \% \text{keep the current plot} \\
>> c &= \cos(x); \quad \% \text{generate sample points for cosine} \\
>> plot(x, c); \quad \% \text{make the plot}
\end{align*}
\]

The plot of the cosine function now appears as a dashed line. We can also work with colors, and plot only the sample points, instead of a smooth line. With the directive `hold off`, MATLAB will start with a new plot in the current figure.

\[
\begin{align*}
>> hold\ off & \quad \% \text{do not keep previous plots} \\
>> plot(x, c,'r+'); \quad \% \text{red + for cosine samples} \\
>> hold\ on & \quad \% \text{keep previous plot} \\
>> plot(x, s,'gx'); \quad \% \text{green x for cosine samples}
\end{align*}
\]

We are not really quite happy with the default view window MATLAB offers us. We can change the default axis scaling and appearance to \([-\pi, \pi]\) for \(x\) and \([-2, 2]\) for \(y\) as follows:

\[
>> axis([-\pi, \pi, -2, 2]) \quad \% \text{change axis scaling}
\]

With `clf` we clear the current figure. If we want to keep the current figure, and make new ones in a new window, you execute the `figure` command. Here is an example of such a session:

\[
\begin{align*}
>> figure & \quad \% \text{open new figure window} \\
>> plot(x, c) & \quad \% \text{plot cosine in new figure} \\
>> figure(1) & \quad \% \text{make figure one current} \\
>> clf & \quad \% \text{clear figure one}
\end{align*}
\]

3.2 Polar Plots

Some plots are better drawn in polar coordinates. Instead of the \((x, y)\) in rectangular coordinates, we use an angle \(t\) and radius \(r\) to define a point in the plane. The angle \(t\) is unique modulo \(2\pi\), and we have the relations \((x = r \cos(t), y = r \sin(t))\).

As example, we plot Galileo’s spiral, defined by \(r = at^2 - l\). The spiral models the trajectory of a point falling near the earth’s equator. We take \(a = 0.01\), \(l = 0.02\) and let \(t\) go from 0 to 10\(\pi\).

\[
\begin{align*}
>> t &= 0 : 0.1 : 10 * pi; \quad \% \text{range to sample} \\
>> r &= 0.01 * t^2 - 0.02; \quad \% \text{sample points} \\
>> polar(t, r) & \quad \% \text{make the polar plot}
\end{align*}
\]

Observe that we use the \(^.\) operator in the definition of \(r\) to tell MATLAB we want the squaring to be done componentwise.
3.3 Three Dimensional Plots

Suppose we wish to plot the function \( z = \cos(xy) \). As with two dimensional plots, we have to generate a grid of sample points to evaluate the function \( \cos(xy) \). For example, if we are interested in the values \((x, y) \in [-2, 2] \times [-2, 2]\), we can evaluate \( \cos \) at the points

\[
x_i = -2 + 0.2(i - 1) \quad 1 \leq i \leq 21 \\
y_j = -2 + 0.2(j - 1) \quad 1 \leq j \leq 21
\]

The generation of the grid of points where we will sample the function is achieved by

\[
\text{>> } xa = -2 : 0.2 : 2; \quad \% \text{ range for } x \\
\text{>> } ya = -2 : 0.2 : 2; \quad \% \text{ range for } y \\
\text{>> } [x, y] = \text{meshgrid}(xa, ya); \quad \% \text{ returns two matrices}
\]

Now that we have our grid, we still must be able to sample. An ordinary multiplication symbol will not do, we have to multiply the grid points componentwise. This is achieved with the .* operator :

\[
\text{>> } z = \cos(x .* y); \quad \% \text{ evaluate at grid points} \\
\text{>> } \text{mesh}(x, y, z) \quad \% \text{ make the 3d plot}
\]

We can change the view point with the command view :

\[
\text{>> view([10, 37])} \quad \% \text{ change horizontal rotation} \\
\text{>> view([-37.5, -20])} \quad \% \text{ change vertical rotation} \\
\text{>> view(3)} \quad \% \text{ back to default 3d view}
\]

The color coding for the surface depends on the height (the values of \( z \)). The mesh command admits a fourth argument which allows the user to define the color coding of the surface. This color coding is often used to make plots of four dimensional objects.

3.4 Special Three Dimensional Plots

To model a spring, we can use the space curve with parametric equations \((\cos(kt), \sin(kt), t)\), for \( t \) going from 0 to \( 2\pi \). The constant \( k \) controls the period. For example, for \( k = 3 \), the spring completes three turns as \( t \) ranges from 0 to \( 2\pi \). The plot3 is the 3-d analogue to plot :

\[
\text{>> } t = 0 : 0.1 : 10 * \pi; \quad \% \text{ range to sample} \\
\text{>> } \text{plot3}(\cos(3*t), \sin(3*t), t); \quad \% \text{ make the plot} \\
\text{>> } \text{rotate3d} \quad \% \text{ adjust view with mouse}
\]

The rotate3d allows us to change the view point: after clicking the mouse button down, with dragging, we can move the axes and set the view into the desired angle. After releasing the mouse button, the data is redrawn. Typing rotate3d again turns of this behavior.

For three dimensional surfaces, we can make contour plots. For example, consider the surface \( z = e^{x+y} \), for \( x \) and \( y \) ranging from \(-2\) to \(+2\).

\[
\text{>> } xr = -2 : 0.1 : 2; \quad \text{>> } yr = -2 : 0.1 : 2; \quad \% \text{ define ranges} \\
\text{>> } [x, y] = \text{meshgrid}(xr, yr); \quad \% \text{ generate grid} \\
\text{>> } z = x .* \exp(-x.^2 - y.^2); \quad \% \text{ sample points} \\
\text{>> } \text{contour}(z) \quad \% \text{ make contour plot}
\]

With the command contour we plot level curves for \( z \). Besides \( z \), we can add the number of contour lines we want to see. Interactively, labels with heights can be added to the level curves:

\[
\text{>> } h = \text{contour}(z, 20); \quad \% \text{ 20 level curves, } h \text{ is handle} \\
\text{>> } \text{clabel}(h,'manual'); \quad \% \text{ add labels manually}
\]
3.4 Subplots

For the three dimensional surface we created above, we will create a new figure with four subplots with different views:

```matlab
>> figure % create new figure
>> subplot(2, 2, 1) % set axes for upper left
>> mesh(x, y, z) % plot with default 3d view
>> subplot(2, 2, 2) % activate second subplot
>> mesh(x, y, z) % plot with default 3d view
>> view([10, 10]) % change view
>> subplot(2, 2, 3) % activate third subplot
>> mesh(x, y, z) % plot with default 3d view
>> view([10, 70]) % change view
>> subplot(2, 2, 4) % activate fourth subplot
>> mesh(x, y, z) % plot with default 3d view
>> view([10, -30]) % change view
```

3.5 Exercises

1. Make a plot of the function \( f(x) = e^{-x^2} \sin(\pi x^3) \) over the interval \([-2, 2]\).

2. The curves \( r = R \sin(k t) \) are called roses. Take \( R = 1 \) and make several plots for different values of \( k \). Make a table of two rows and three columns for the respective values \( k = 4, 5, 6 \) in the top row and \( k = \frac{2}{3}, \frac{4}{3}, \frac{5}{3} \) in the bottom row.

3. The inverse Galileo spiral for \( l = 0 \) is defined by \( r = a t^2 \). Take \( a = 100 \) and choose a suitable range for \( t \). Make several subplots for different ranges of \( t \) to illustrate that the trajectory modeled by this spiral is the inverse of the one we have seen.

4. Consider the surface \( z = x^3 - 3xy^2 \).
   
   (a) Make a plot over the grid \([-1, 1] \times [-1, 1]\).
   
   (b) Make four subplots for different view points.

5. Closed curves on a sphere are defined by \( x = \cos(m t) \cos(nt), y = \sin(m t) \sin(nt), z = \sin(nt) \), for \( t = -\pi, \ldots, \pi \). Make a table of plots, for \( m = 2, 5, 7 \) and \( n = 5, 8, 10 \). Make sure to take enough samples to see smooth curves.