Plotting with MATLAB

Jiseok Chae

Department of Mathematical Sciences
KAIST

Week 11

Before we start...

This week we will see how plotting can be done in MATLAB.

There are a huge variety of types and functionalities for plotting provided by MATLAB[1], while we only have limited amount of time.

We focus on the principles of plotting, while discussing only the most representative plot styles.

[1] See `https://www.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html` for a list of supported plotting types.

Contents

1 Plotting Curves

2 Plotting Surfaces

3 Plotting Vector Fields

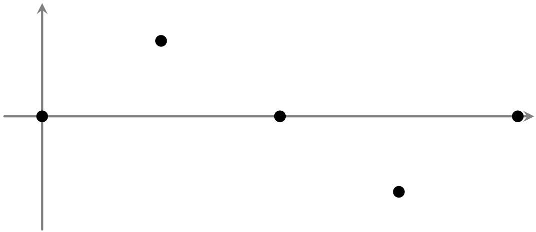
How would you draw the graph of $y = f(x)$, without differentiation?

To give you a concrete example, let us try drawing the graph of $f(x) = \sin x$ on the domain $[0, 2\pi]$.

One would start by sampling several points from the domain, and evaluating $f(x)$ at them.

x	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π
$f(x)$	0	1	0	-1	0

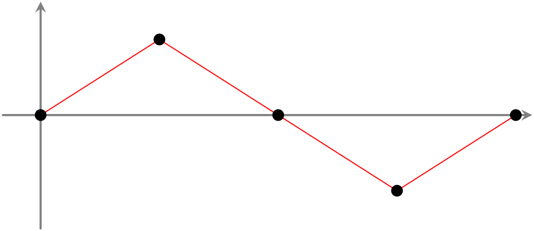
Using this table, you locate the points where the graph goes through, and then connect them.



One would start by sampling several points from the domain, and evaluating $f(x)$ at them.

x	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π
$f(x)$	0	1	0	-1	0

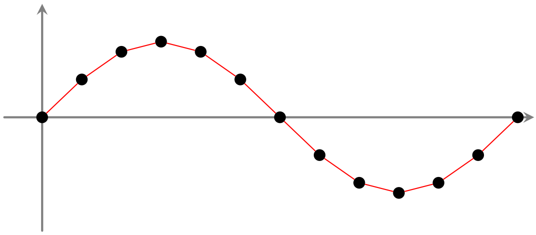
Using this table, you locate the points where the graph goes through, and then connect them.



If you sample more points, you get a better plot:

x	0	$\frac{\pi}{6}$	$\frac{\pi}{3}$	$\cdots$	2π
$f(x)$	0	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\cdots$	0

This table allows us to plot the graph as follows:



Actually, you can do the same thing in MATLAB to plot graphs. There are two ways to do something similar to sampling points from the domain.

The command `linspace(a, b, n)` generates a vector which contains `n` equally spaced points, from `a` to `b`.

```
>> x = linspace(0, 5, 5)

x =

         0    1.2500    2.5000    3.7500    5.0000
```

Note that the space between two points is $\frac{b-a}{n-1}$.

The command `a : dx : b` generates a vector which starts with `a`, is incremented by `dx` every element, and contains elements only $\leq$ `b`.

```
>> x = 0 : 2 : 6

x =

     0     2     4     6

>> x = 0 : 2 : 5

x =

     0     2     4
```

However this method may cause unexpected issues, especially when `dx` is not an integer. Using `linspace` is usually safer.

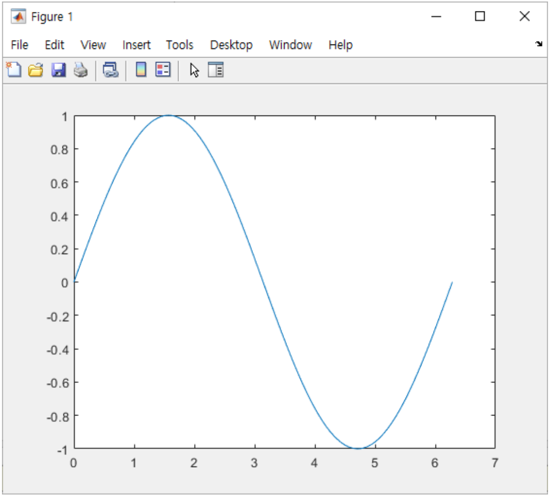
Now let us plot the graph of $f(x) = \sin x$, by sampling 100 points from the domain $[0, 2\pi]$.

We make a new script, named `plot1.m`, as:

plot1.m

```
x = linspace(0, 2 * pi, 100);
y = sin(x);
plot(x, y);
```

Running the script `plot1.m`, a figure window appears:

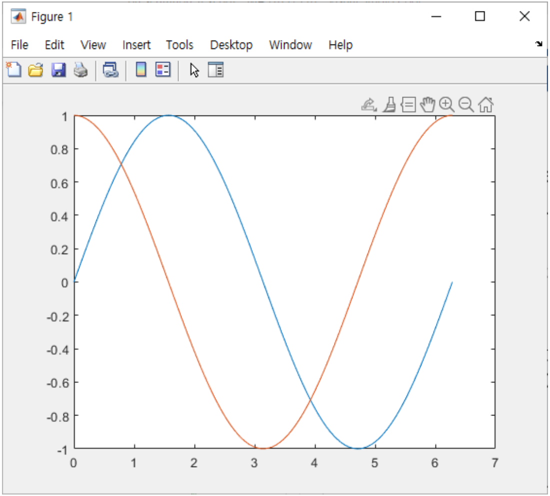


We can plot multiple functions. One option is as follows.

plot2.m

```
x = linspace(0, 2 * pi, 100);
y1 = sin(x);
y2 = cos(x);
plot(x, y1, x, y2); % Draw in one figure
```

Running the script `plot2.m`, a figure window appears:



Recall that, from the data points `x` and `y`, the command `plot(x, y)` generates a 'curve' which passes through the points $(\mathtt{x}_i, \mathtt{y}_i)$, $i = 1, 2, \ldots$

In fact, the data points `x` and `y` do not have to be of the form $\mathtt{y}_i = f(\mathtt{x}_i)$. This suggests a generalization.

For example, a circle cannot be a graph of a function. However, by introducing a parameter t, for example a unit circle can be represented as the set of points

$$\{(\cos t, \sin t) : 0 \le t \le 2\pi\}.$$

So let's try this...

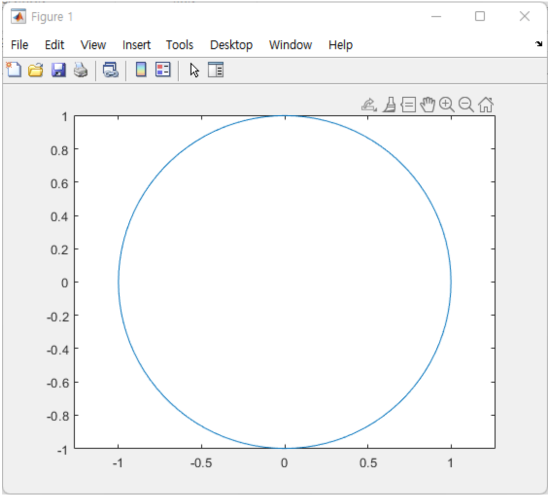
circle.m

```
t = linspace(0, 2*pi, 101);
x = cos(t);
y = sin(t);

plot(x, y)
axis equal % force axis ratio to be 1:1
```

t	0	$\frac{2\pi}{100}$	$\frac{4\pi}{100}$	$\cdots$	2π
$x(=\cos t)$	1	$\cos\left(\frac{2\pi}{100}\right)$	$\cos\left(\frac{4\pi}{100}\right)$	$\cdots$	1
$y(=\sin t)$	0	$\sin\left(\frac{2\pi}{100}\right)$	$\sin\left(\frac{4\pi}{100}\right)$	$\cdots$	0

The result is, as expected, a circle.



Using this technique, we can make two-dimensional parametric plots expressed as $(x, y) = (f(t), g(t))$ for two functions $f, g : \mathbb{R} \to \mathbb{R}$.

There is a three-dimensional counterpart of `plot`, namely `plot3`.

For three vectors `x`, `y`, and `z` of the same size, `plot3(x, y, z)` will draw a curve in $\mathbb{R}^3$ that passes through the points $(\mathrm{x}_i, \mathrm{y}_i, \mathrm{z}_i)$, $i = 1, 2, \ldots$.

Almost everything discussed about `plot` naturally extends to `plot3`.

Sometimes you may want to just see where the data points lie on, instead of a graph.

This would especially be the case when you want to visualize data you have collected.

There is a function `scatter` which is specialized for this task. Let's see what this does.

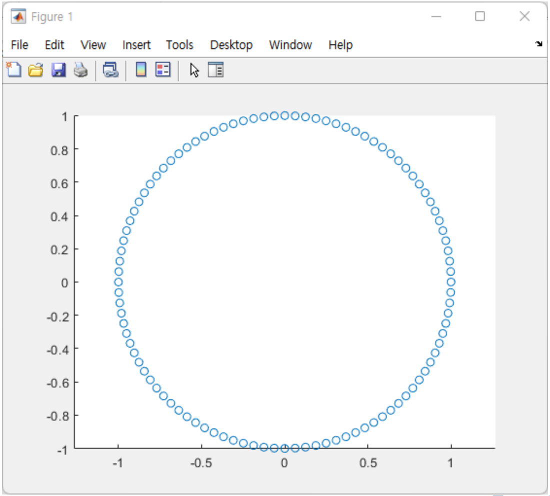
Change the `plot` to `scatter`...

circle_points.m

```
t = linspace(0, 2*pi, 101);
x = cos(t);
y = sin(t);

scatter(x, y)
axis equal % force axis ratio to be 1:1
```

Now the data points are shown with markers, instead of a curve connecting them.



Contents

1 Plotting Curves

2 Plotting Surfaces

3 Plotting Vector Fields

In $\mathbb{R}^3$, we can also consider plotting a scalar-valued two-variable function of the form $z = f(x, y)$.

Then now, we should sample points from a *two-dimensional* domain, instead of an interval.

The function `meshgrid` can be used to make grid points in $\mathbb{R}^n$.

For now, let us generate grid points in $\mathbb{R}^2$.

mesh_points.m

```
s = linspace(1, 2, 4);
t = [0, 1, 2];
[X, Y] = meshgrid(s, t)
```

We use square bracket `[X, Y]` because `meshgrid` returns *two* matrices.

After executing `mesh_points.m`, the outputs `X` and `Y` should be as follows.

```
X =

    1.0000    1.3333    1.6667    2.0000
    1.0000    1.3333    1.6667    2.0000
    1.0000    1.3333    1.6667    2.0000

Y =

     0     0     0     0
     1     1     1     1
     2     2     2     2
```

The vectors $\mathtt{s} \in \mathbb{R}^4$ and $\mathtt{t} \in \mathbb{R}^3$ specify $4 \times 3 = 12$ gridpoints, and the coordinates of those 12 gridpoints are $(\mathtt{X}_{ij}, \mathtt{Y}_{ij})$.

Let us plot the graph of the function $z = \dfrac{1}{x^2 + y^2 + 1}$.

surf(X, Y, Z) draws a surface that passes through the points $(\mathtt{X}_{ij}, \mathtt{Y}_{ij}, \mathtt{Z}_{ij})$.

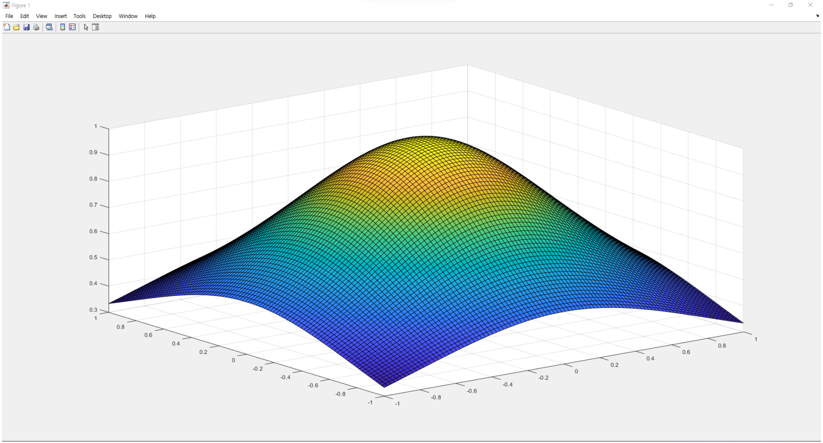
surf_plot.m

```
s = linspace(-1, 1, 100);
t = linspace(-1, 1, 100);

[X, Y] = meshgrid(s, t);
Z = 1 ./ (X.^2 + Y.^2 + 1);

surf(X, Y, Z)
```

Note: by dragging the plot around, you can change the point of view.



`mesh(X, Y, Z)` draws a ‘wireframe’ version of the surface that passes through the points $(\mathtt{X}_{ij}, \mathtt{Y}_{ij}, \mathtt{Z}_{ij})$.

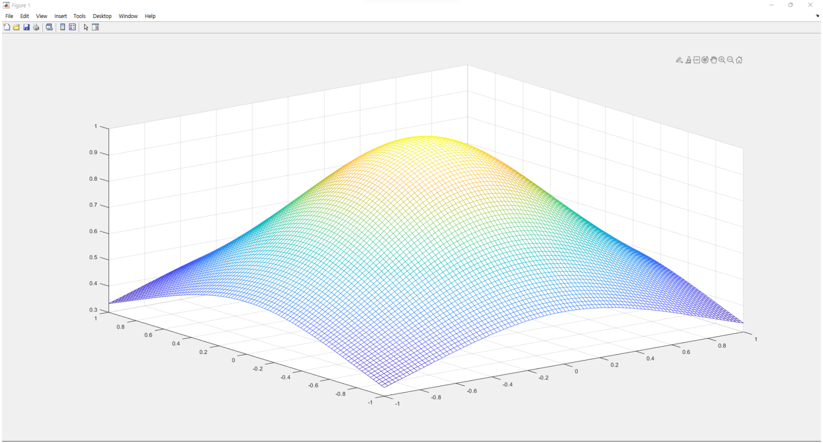
mesh_plot.m

```
s = linspace(-1, 1, 100);
t = linspace(-1, 1, 100);

[X, Y] = meshgrid(s, t);
Z = 1 ./ (X.^2 + Y.^2 + 1);

mesh(X, Y, Z)
```

Now the figure only shows the 'frame' of the surface.



Contents

1 Plotting Curves

2 Plotting Surfaces

3 Plotting Vector Fields

Vector fields are, simply put, functions from $\mathbb{R}^n$ to $\mathbb{R}^n$.

Recall that vectors in $\mathbb{R}^2$ and $\mathbb{R}^3$ can be visualized by an arrow.

So, given a vector field $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$ or $\mathbb{R}^3 \to \mathbb{R}^3$, we can visualize it by:

(i) sample points from the domain,

(ii) for each sample point $\mathbf{x}$, draw an arrow corresponding to the vector $\mathbf{F}(\mathbf{x})$ with the initial point at $\mathbf{x}$.

`quiver(x, y, u, v)` plots the vector field $(u, v) = \mathbf{F}(x, y)$ with arrows.

Let us plot the graph of the vector field $(u, v) = \mathbf{F}(x, y) = (-y, x)$.

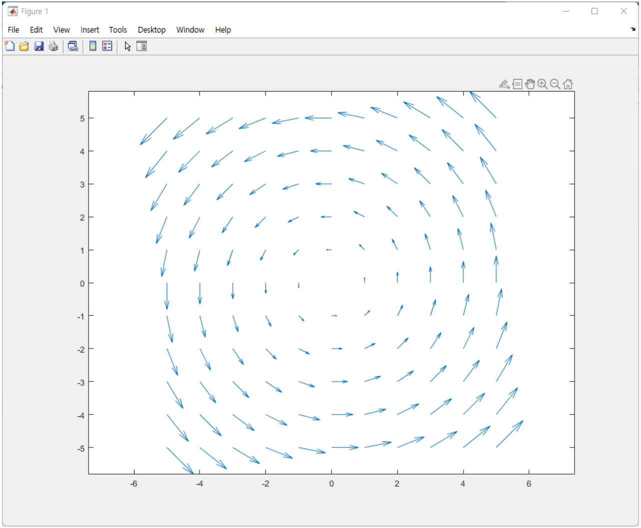
vector_field_2d.m

```
s = -5:1:5;
t = -5:1:5;

[x, y] = meshgrid(s, t);
u = -y; v = x;

quiver(x, y, u, v)
axis equal
```

The arrows are scaled so that they do not overlap.



`quiver3(x, y, z, u, v, w)` plots the vector field $(u, v, w) = \mathbf{F}(x, y, z)$.

Let us visualize the *inverse square law* by plotting the vector field

$$(u, v, w) = \left(-\frac{x}{x^2+y^2+z^2}, -\frac{y}{x^2+y^2+z^2}, -\frac{z}{x^2+y^2+z^2} \right).$$

inverse_square_field.m

```
r = linspace(-1, 1, 7);
s = linspace(-1, 1, 7);
t = linspace(-1, 1, 7);

[x, y, z] = meshgrid(r, s, t);
u = -x ./ (x.^2 + y.^2 + z.^2);
v = -y ./ (x.^2 + y.^2 + z.^2);
w = -z ./ (x.^2 + y.^2 + z.^2);

quiver3(x, y, z, u, v, w)
```

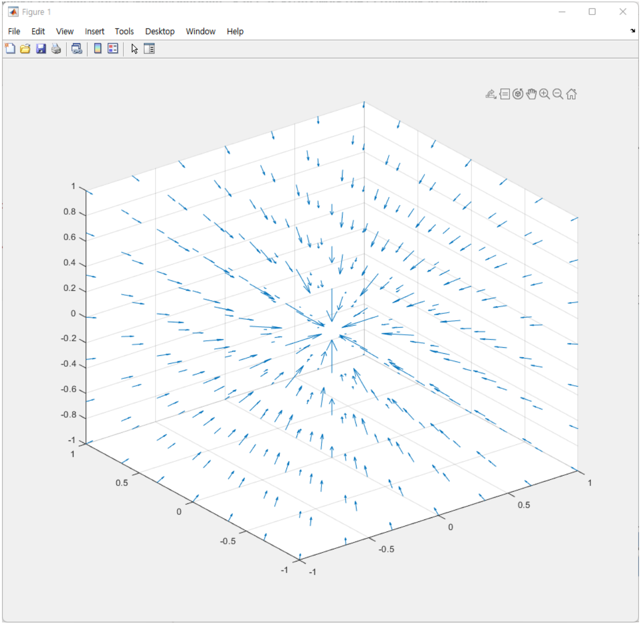


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

Thank you!