Solving Algebraic Equations and Other Symbolic Tools

Solving Basic Algebraic Equations

To solve an algebraic equation in MATLAB we can call upon the *solve command.* We can find the solution in one step. All we do is create a variable and assign it , the value returned by solve in the following way:

$x + 3 = 0$

```
>> x = solve('x + 3 = 0')
x =
–3
```

Note: Now it isn't necessary to include the right-hand side of the equation. As you can see from the following example, MATLAB *assumes that when you pass* $x + 8$ *to* solve that you mean $x + 8 = 0$.

```
>> x = solve('x+8')
x =
–8
```

Solving Quadratic Equations

The solve command can be used to solve higher order equations. Let's consider the equation:

x2 – 6x – 12 = 0

To solve it using MATLAB, we write:,

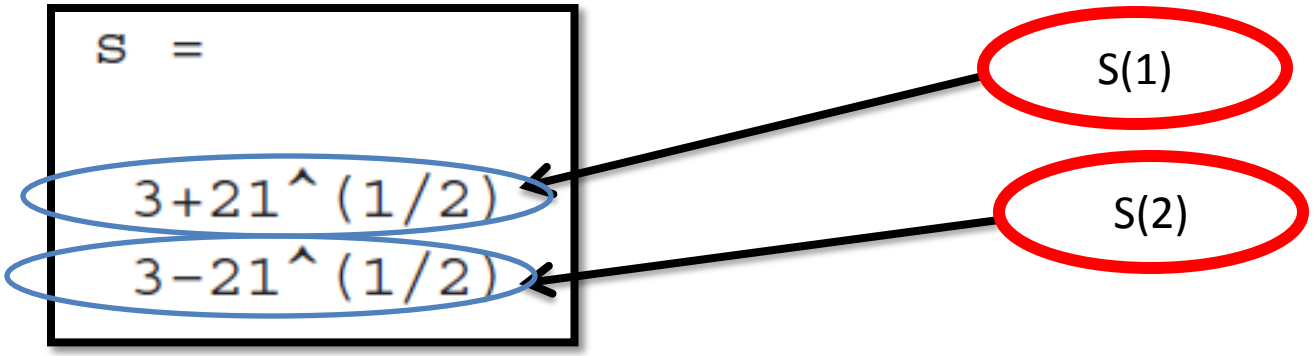
```
>> s = solve('x^2 -6*x -12 = 0')
```

MATLAB responds with the two roots of the equation:

```
s =

  3+21^(1/2)
  3-21^(1/2)
```

- In the case of two roots like this one, the roots are stored as ***s(1)*** *and* ***s(2)***. *So we can use one of the roots to define a* new quantity:




```
>> y = 3 + s(1)

y =

6+21^(1/2)
```

Here is another example

```
>> s(1) + s(2)

ans =

6
```

- **It is possible to assign an equation to a variable and then pass the variable to solve. For instance,**

```
>> d = 'x^2 + 9*x -7 = 0';
```

Now we call solve this way:

```
>> solve(d)
```

MATLAB correctly tells us the two roots of the equation:

```
ans =

 -9/2+1/2*109^(1/2)

 -9/2-1/2*109^(1/2)
```

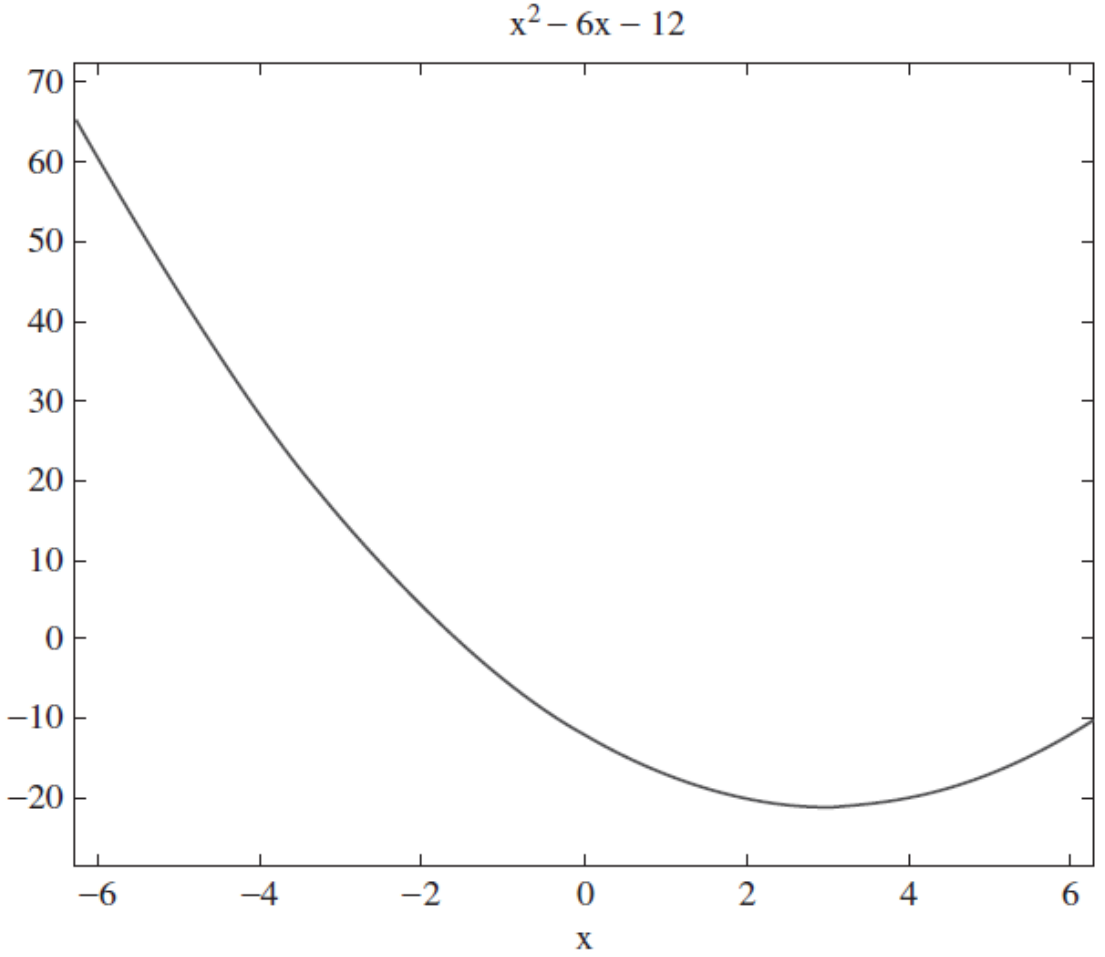
Plotting Symbolic Equations

Let's do that for this example.

```
>> d = 'x^2 -6*x - 12';
```

Now we call ezplot:

```
>> ezplot(d)
```

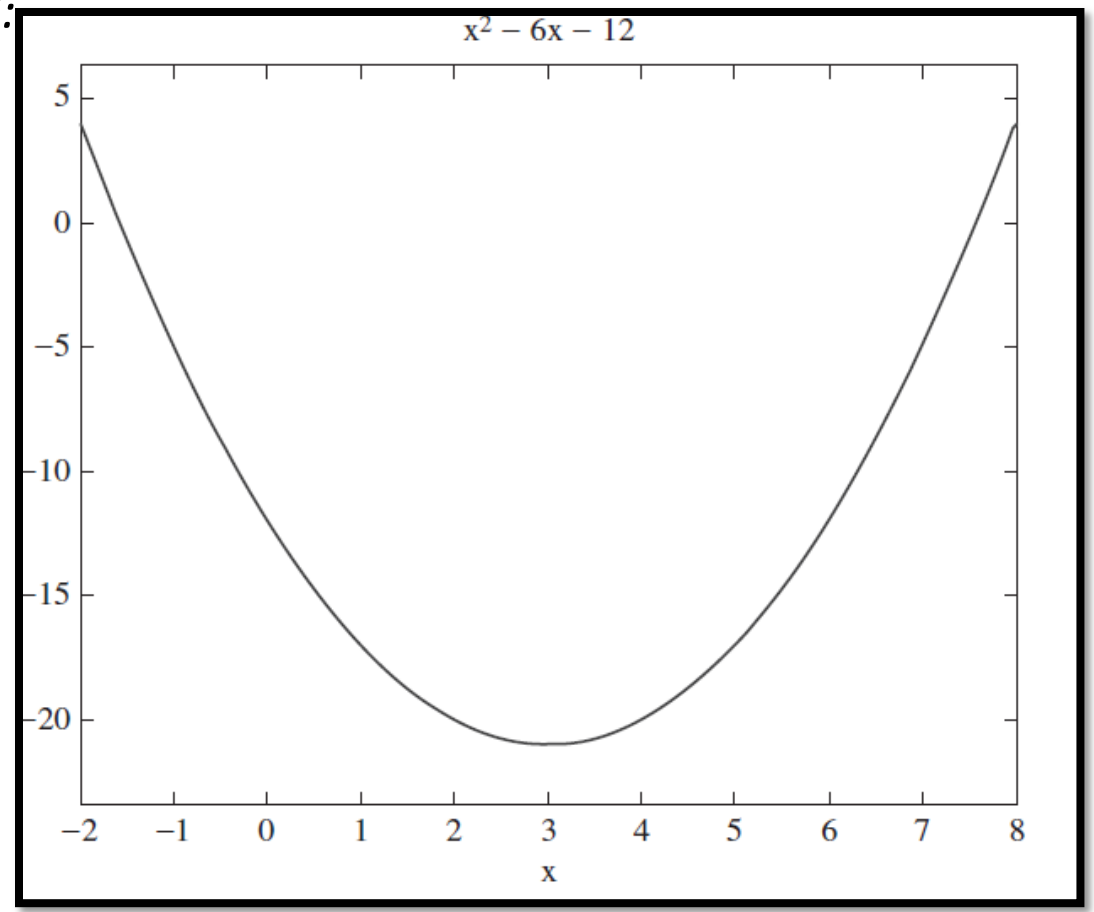


The function also picked what values to use for the domain and range in the plot. Of course we may not like what it picked. We can specify what we want by specifying the domain with the following syntax:

```
ezplot(f, [x1, x2])
```

Returning to the previous example, let's say we wanted to plot it for $-2 < x < 8$. *We can do that using the following command:*

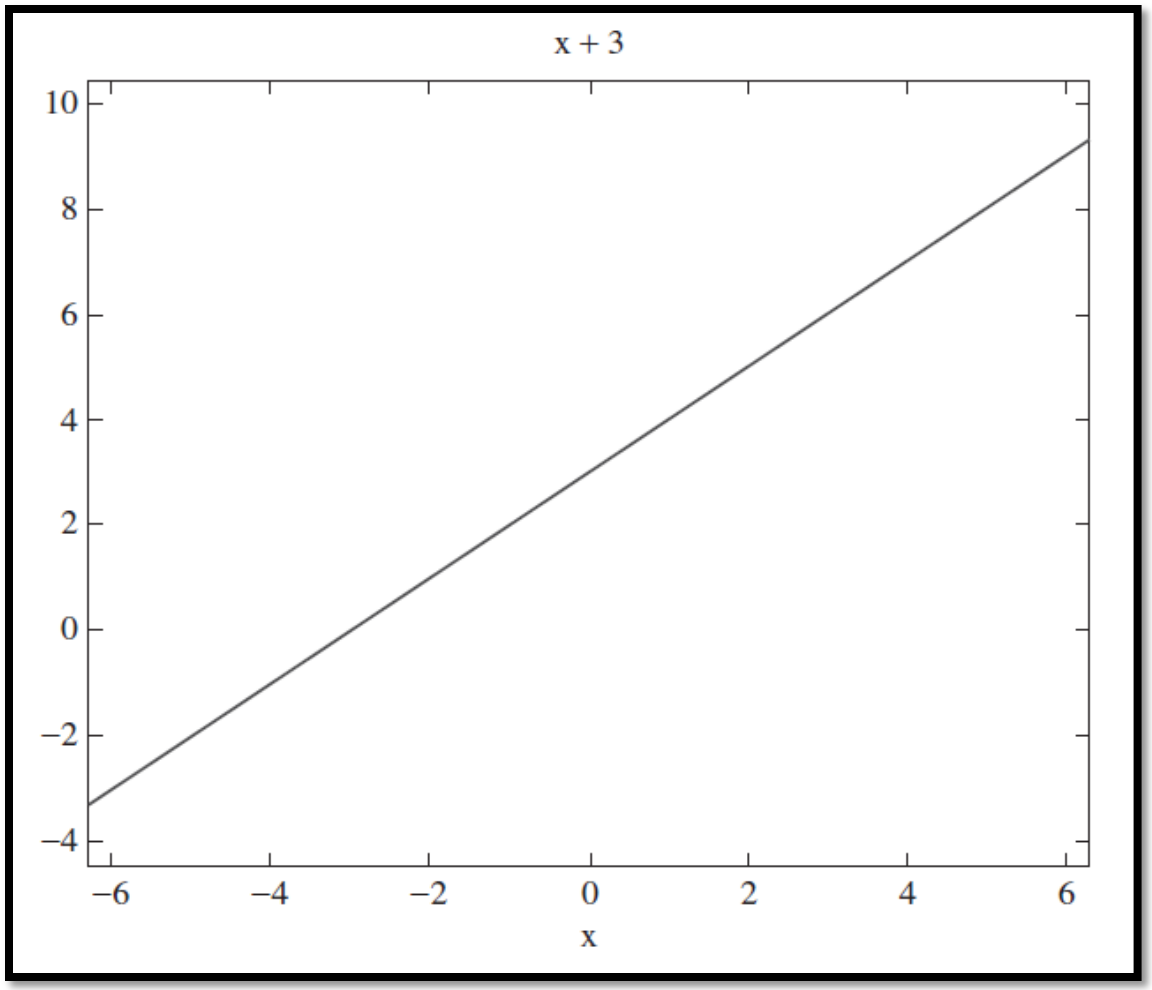
```
>> d = 'x^2 -6*x - 12';
>> ezplot(d,[-2,8])
```



Now, if we instead type:

```
>> ezplot('x+3')
```

It happily generates the straight line shown in Figure



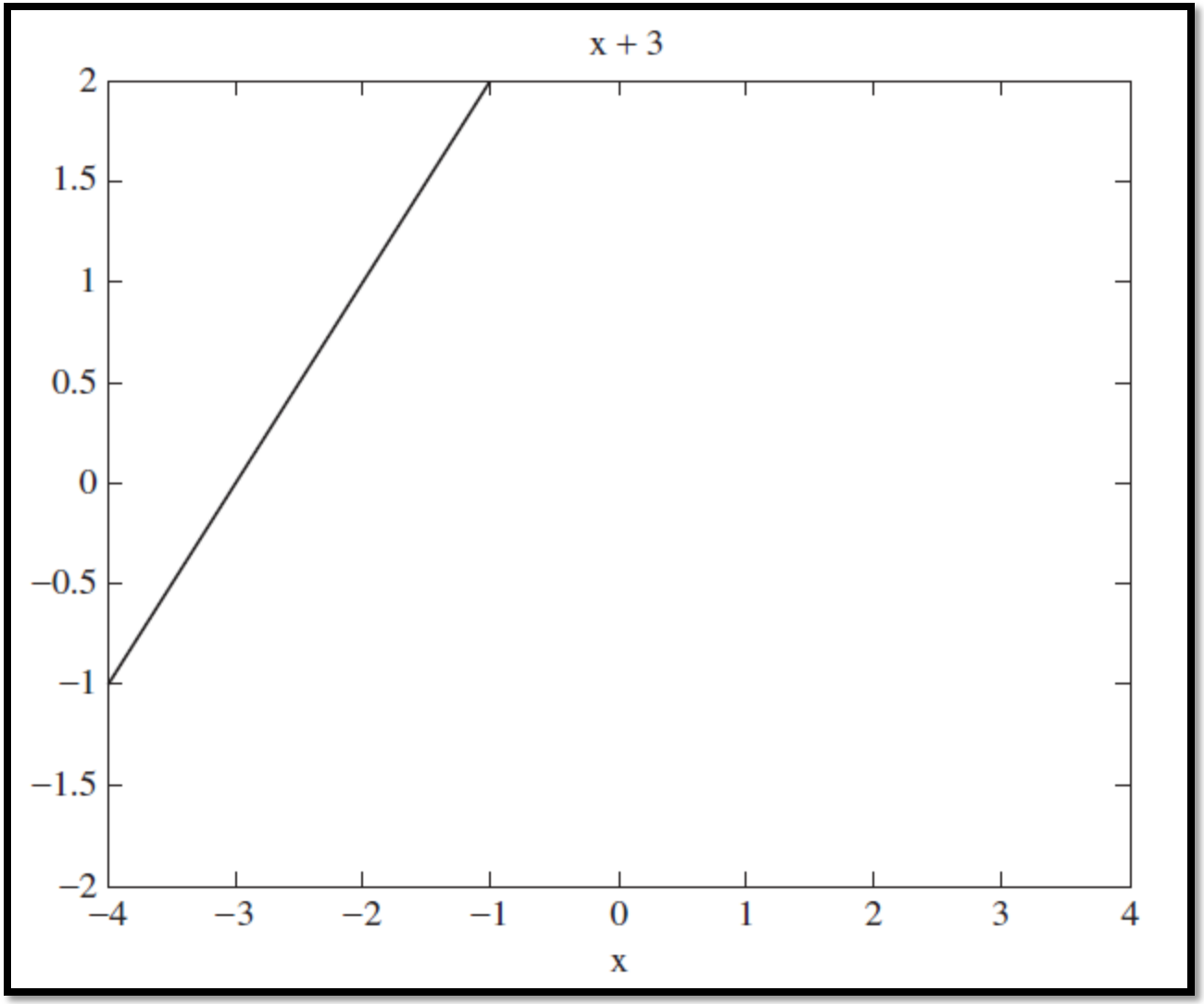
Now we just mentioned a while ago that we could tell ezplot how to specify the domain to include in the plot. Naturally it also allows us to specify the range. Just for giggles, let's say we wanted to plot:

$$x + 3 = 0$$
$$-4 < x < 4, -2 < y < 2$$

We can do this by typing:

```
>> ezplot('x+3',[-4,4,-2,2])
```

The plot generated is shown in Figure So, to specify you want the plot to be over *x1 < x < x2 and y1 < y < y2 include [x1, x2, y1, y1] in your call to ezplot.*


x + 3
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−4
−3
−2
−1
0
1
2
3
4
x

EXAMPLE :

Find the roots of *x2 + x – 2 = 0 and plot the function. Determine the numerical* value of the roots.

```
>> eq = 'x^2 + x - sqrt(2)';
```

Or if you like, you could write:

```
>> eq = 'x^2 + x - 2^(1/2)';
```

Next we call solve to find the roots:

```
>> s = solve(eq)

s =
 -1/2+1/2*(1+4*2^(1/2))^(1/2)
 -1/2-1/2*(1+4*2^(1/2))^(1/2)
```

To determine the numerical value of the roots, we need to extract them from the array and convert them into type double. This is done by simply passing them to the ***double(.) command***. For example, we get the first root out by writing:

```
>> x = double(s(1))

x =

    0.7900
```

And the second root:

```
>> y = double(s(2))

y =

   -1.7900
```

To plot the function, we use a call to ezplot:

>> ezplot(eq)

The result is shown in Figure