# 11 .Numerical analysis and curve fitting

- we have learnt to program ***functions***, we are able to use Matlab's numerical analysis and ***curve fitting tools***. Here, we describe how to numerically solve equations and find local minimum points, as well as how to perform numerical integration.

## 11.1 Solving equations

- There are several ways to define functions in Matlab. In Chapter 9, we learnt how to write user defined functions. Let us use the function

$$y = \sin(x) + e^{-x}$$

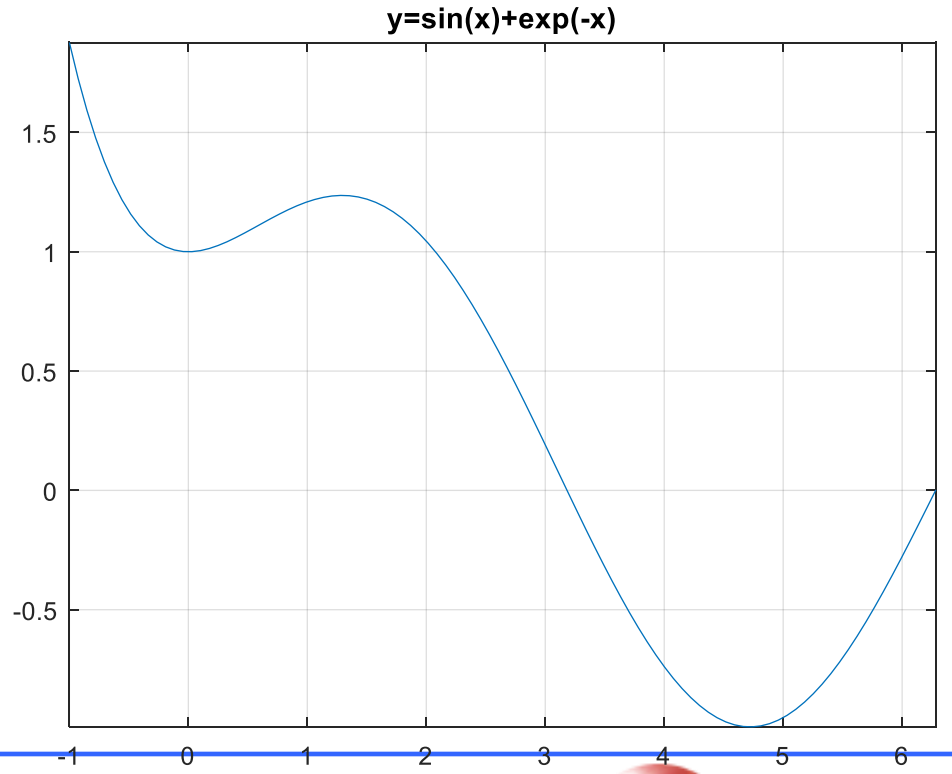
and want to plot the function for X-values between -1 and 2π and then solve it (i.e., find the point where the function equals zero).

# 11 .Numerical analysis and curve fitting

$$y = \sin(x) + e^{-x}$$

- First, we create a vector of X-values, noting that it has to be fine grained enough to represent the changes in Y-values.
- Then we create a vector of the corresponding Y-values and plot the data.

```
>> x=linspace(-1,2*pi,100); y=sin(x)+exp(-x);
>> plot(x,y);
>> title('y=sin(x)+exp(-x)'); grid on, axis tight
>> [x(1:5)' y(1:5)']
ans =
   -1.0000    1.8768
   -0.9264    1.7260
   -0.8529    1.5932
   -0.7793    1.4772
   -0.7057    1.3767
```



# 11 .Numerical analysis and curve fitting

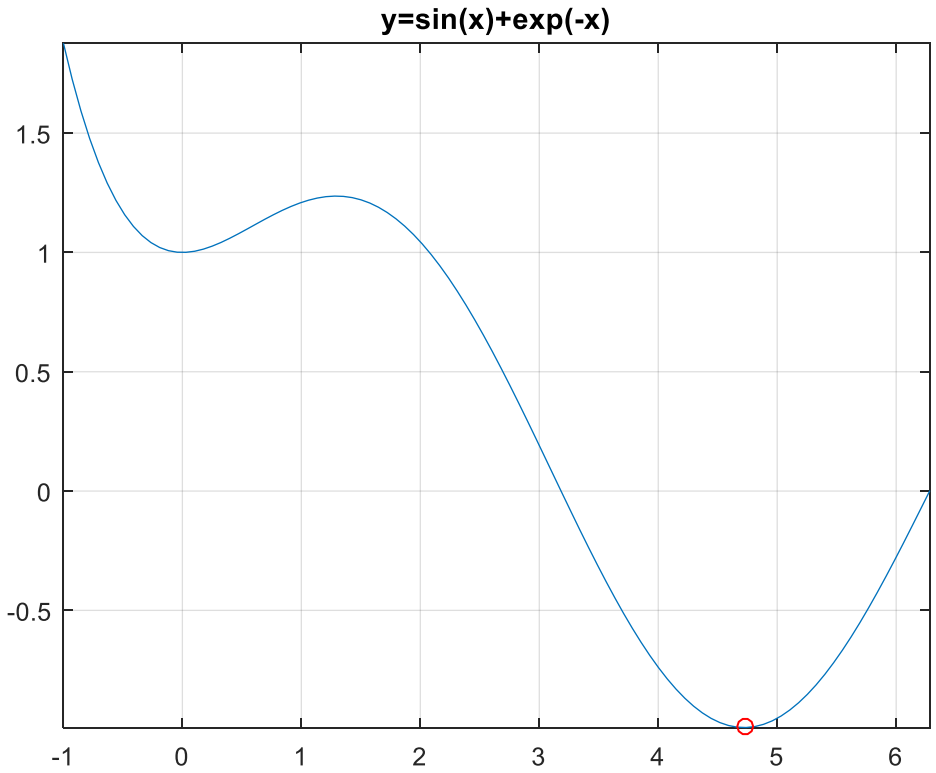
- For a given set of X- and Y-values, it is also easy to add markers at specific values, for example at a function minimum. When using the min() function with two output arguments, it supplies both the minimum value and the location of that number

```
>> [minVal,minLoc] = min(y)
minVal =
        -0.99091
minLoc =
         79
```

- Since the X- and Y-values are paired, the X-value that corresponds to the minimum Y-value must be in the same location in x. We can then put a red o-marker in the correct spot by issuing

# 11 .Numerical analysis and curve fitting

>> hold on; plot(x(minLoc),y(minLoc),'ro')



# 11 .Numerical analysis and curve fitting

- Define that as a user defined function, and then solve it (i.e., find the point where the function equals zero).

- We write the function, named waveFunction, such that we supply an X-value and then get back the corresponding Y-value. For such a simple function, this is easily done.

```
function y = waveFunction(x)
y = sin(x) + exp(-x);
```

# 11 .Numerical analysis and curve fitting

- To start searching for a solution, we need a starting value. Looking at Figure 7-3, we see that the function equals zero where x is roughly equal to 3. Then we issue the command fzero() as

```
>> x_zero = fzero('waveFunction',3)
x_zero =
        3.1831
```

- The first input to fzero() is the name of the function we want to solve, within single quotes.
- And the second is a "best guess" where a solution will be found.

# 11 .Numerical analysis and curve fitting

## 11.2 Finding a function minimum point

- It is equally easy to find a local minimum of a function. We see in Figure 7-3 that there should be a local minimum where the red circle marker is, somewhere between 4 and 5. The command for finding a minimum between two values on the X-axis is fminbnd().

```
>> [x_min,fval] = fminbnd('waveFunction',4,5)
x_min =
        4.7213
fval =
        -0.99106
```

# 11 .Numerical analysis and curve fitting

## 11.2 Finding a function minimum point

- The second and third input arguments are the end values of the region along the X-axis where we want to search for a minimum. One is found at 4.7213, and the corresponding Y-value, fval, at that point is -0.99106. (Note that, this is slightly smaller than the value we found when plotting the minimum point in the figure in Section 7.3. The difference depends mainly on how fine grained the x-vector was there.)

# 11 .Numerical analysis and curve fitting

## 11.3 integration

- To numerically integrate our function between -1 and 2π (i.e., the plotted region), there are several different methods to choose from. For instance, we can issue

```
>> area = quad('waveFunction',-1,2*pi)
area =
        2.2567
```

The area beneath the function is approximately 2.2567.

# 11 .Numerical analysis and curve fitting

## 11.4 Curve fitting

- Matlab also has a convenient tool for curve fitting. If we have two vectors, x and y, with paired observations, we can approximate the functional relation between them with a polynomial of some degree.

- If the degree is 1, the relation is linear;

- if it is 2, the relation is quadratic, etc.

- This can be done with the function polyfit().