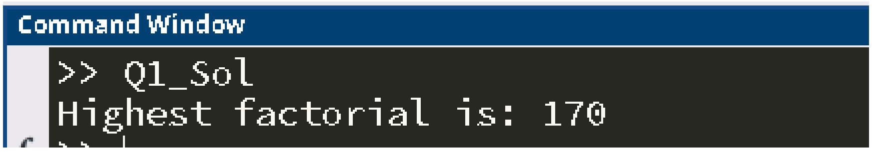
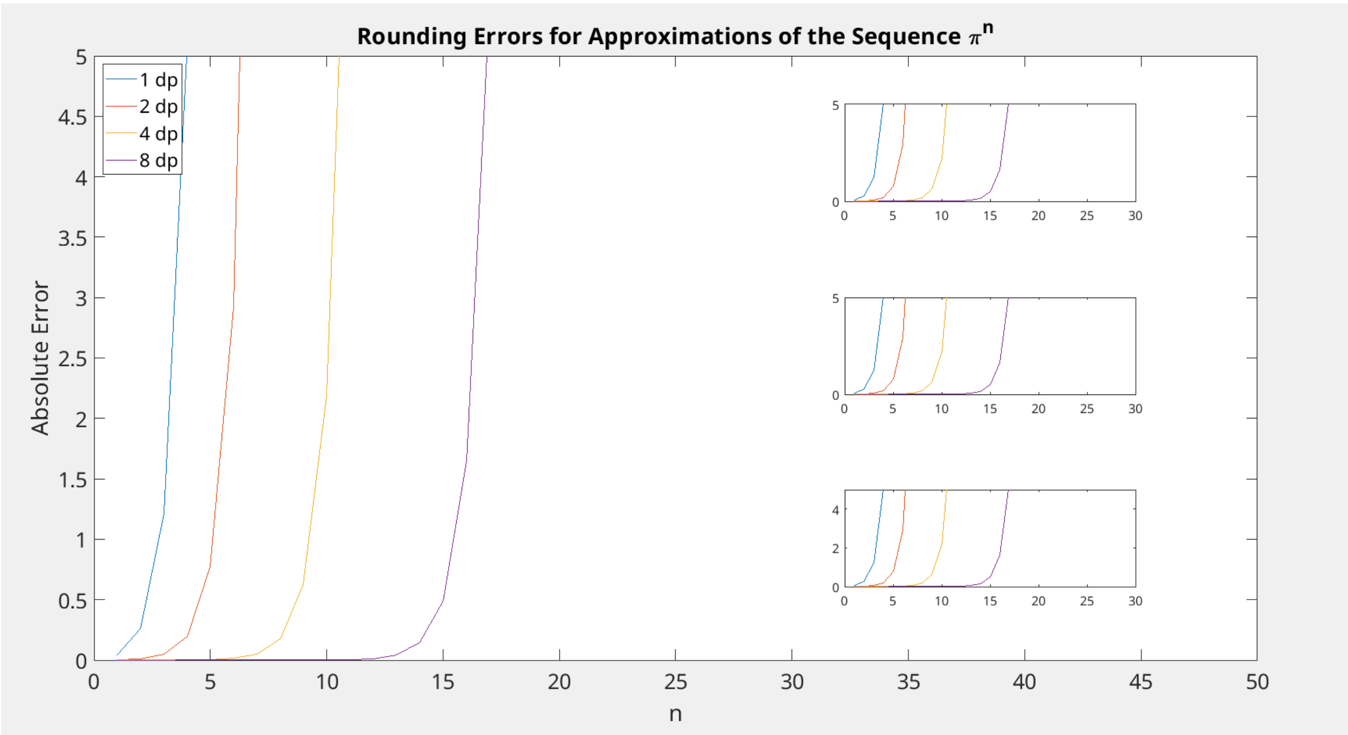
Practical Problems 4 – Errors & Bracketed Root Finding

1. **[Overflow Error]** Matlab returns **Inf** when an overflow error in double precision is encountered. Write a script that starts with $n = 1$, calculates $n!$ and increases n until overflow occurs (number is too big that Matlab can't handle it). Display the value of n that represents the largest factorial that Matlab can handle.



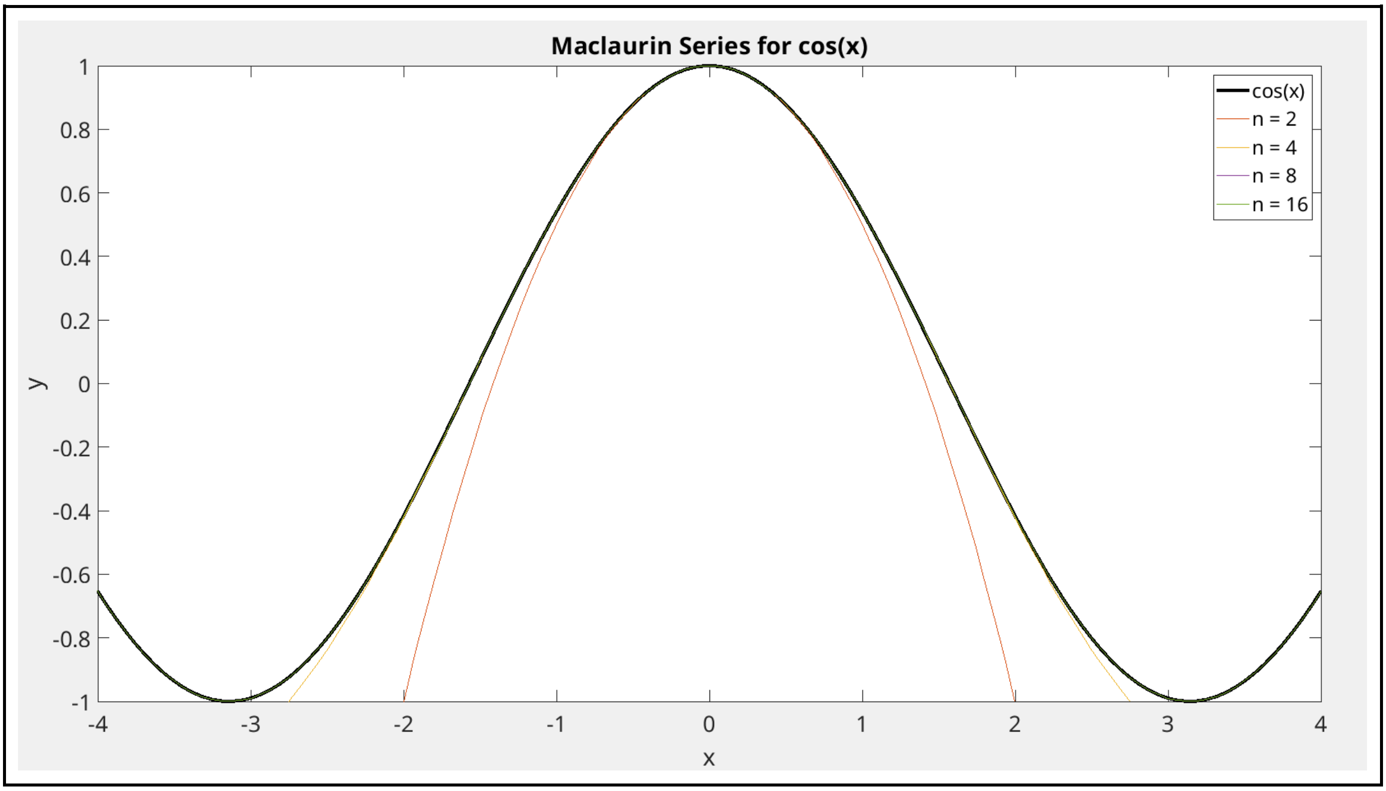
2. **[Rounding Error]** Approximate the sequence $\{\pi^n\}$ for $n = 1$ to 18 by rounding π to 1, 2, 4, and 8 decimal places. Calculate the absolute error for each sequence approximation from the sequence that uses π with double precision. Plot the errors for each approximate sequence on the same plot with y-axis between 0 and 5. See if you can figure out how to embed smaller plots within the main plot that zoom in on each of the sequences.



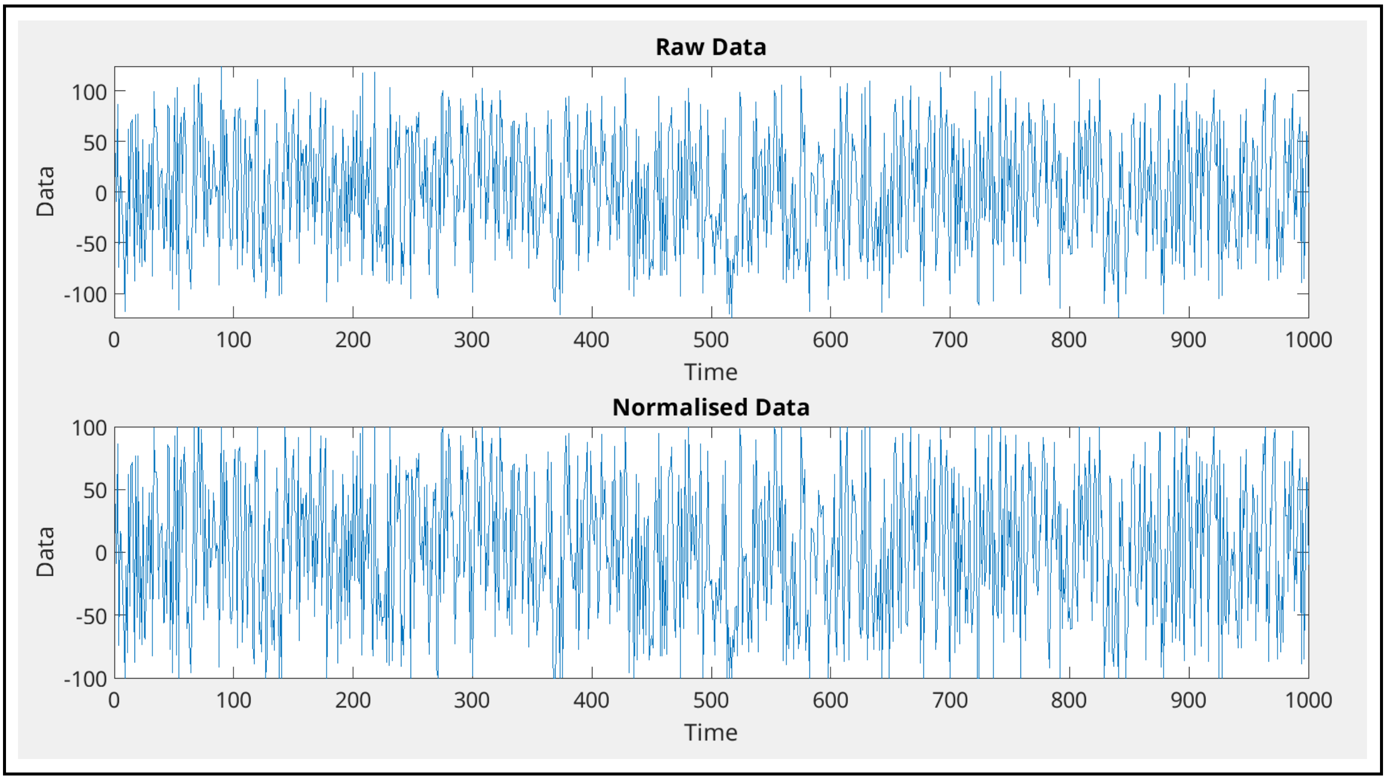
3. **[Truncation Error]** The Maclaurin series for cos(x) is

$$cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$$

Plot the cosine function in a thick black line between -2π and 2π. On the same graph plot the Maclaurin series approximation for n equal to 2, 4, 8 and 16.



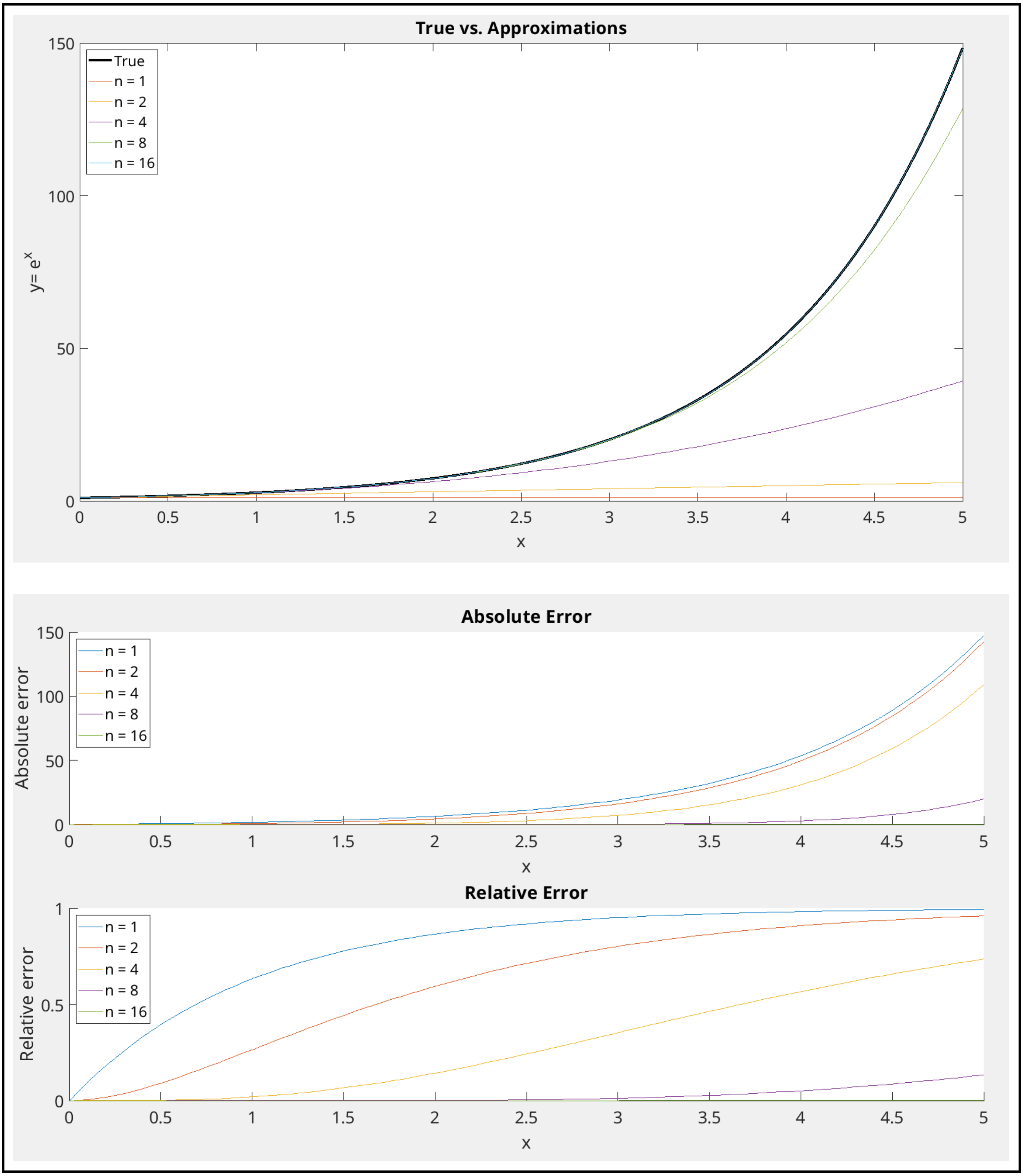
4. **[Correcting Measurement Data]** Import the data from the file **data_pp4_q4.xls** using the import wizard (or using the **xlsread** command) then plot the Data against Time. The data should be corrected so that all values above 100 are normalised to exactly 100, and all values below –100 are normalised to exactly –100. Plot the corrected data on a subplot below the original.



5. **[True vs. Relative Error]** The Maclaurin series for e^x is

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Plot the true function from 0 to 5 then on the same graph plot the Maclaurin approximations for *n* equal to 1, 2, 4, 8, and 16. Calculate the true absolute and true relative errors then plot these together in a separate graph.



6. **[Convergence with Absolute Error Criterion]** The series

$$\sum_{n=0}^{\infty} \frac{4^{n+1}}{(3n+1)!}$$

is convergent. Write a script that finds the value the series converges to by using a loop and the absolute approximate error with a tolerance of 0.0001 (stopping criterion).

```
Command Window
>> Q6_Sol
Absolute error is: 0.66667
Absolute error is: 0.012698
Absolute error is: 7.0547e-05
The sum has converged to 4.6794
The number of terms required is 3
```

7. **[Convergence with Relative Error Criterion]** Write a script that finds the value that the series in question 6 converges to, by using a loop and the relative approximate error with a tolerance of 0.0001 (stopping criterion). Compare the results of this question with question 6.

```
Command Window
>> Q7_Sol
Absolute error is: 0.14286
Absolute error is: 0.0027137
Absolute error is: 1.5076e-05
The sum has converged to 4.6794
The number of terms required is 3
```

8. **[Open Method Root Finding]** Write a script that implements the Incremental Search Method, with a step size of 0.1, to approximate the 3 roots to the following equation:

$$y = x^3 - 6x^2 + 10x - 4$$

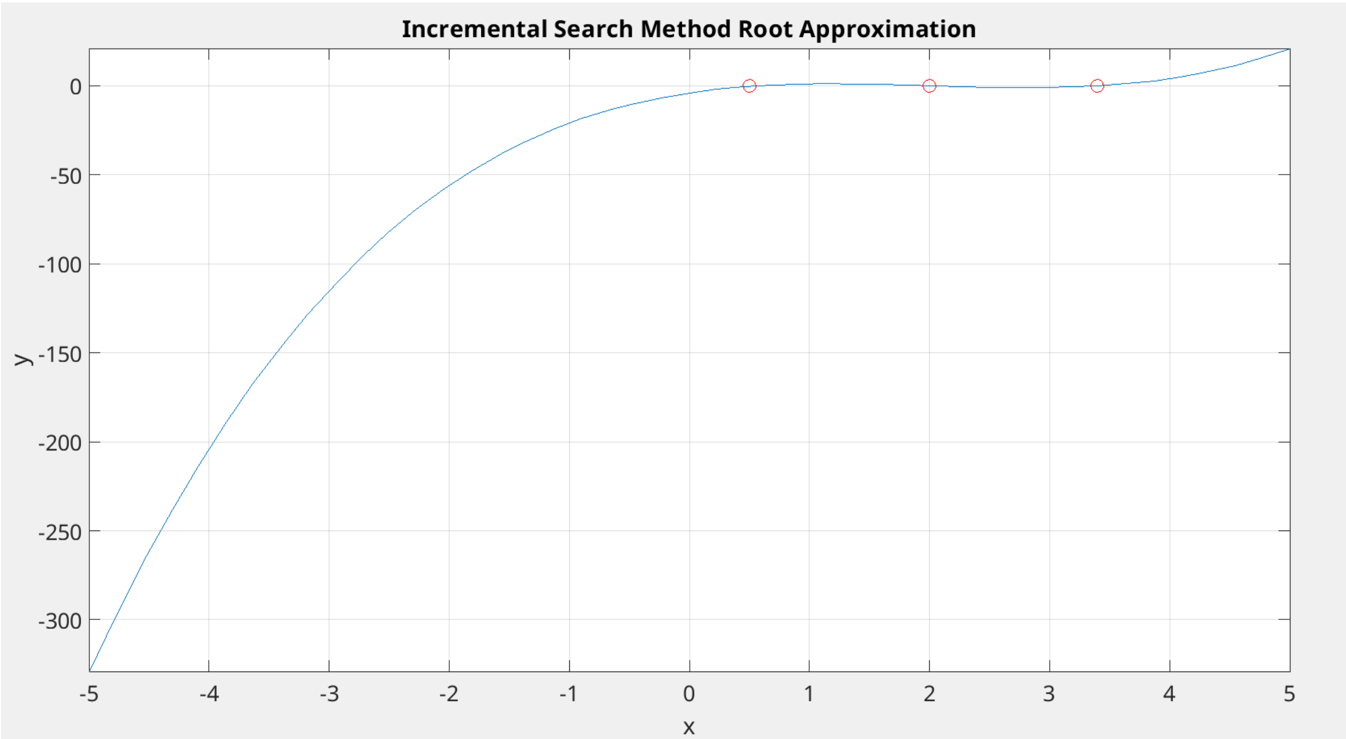
Choose a sensible initial guess for your algorithm (plotting the function might help you decide).

Find a way to test how closely you approximated the roots.
Can you see why this method is only good for initial bracketing?

Command Window

```
>> Q8_Sol
Root approximations:
    0.5000    2.0000    3.4000

Function values at approximate roots:
   -0.3750         0   -0.0560
```



9. **[Preparing Open Method for Use With Closed Method]** Rewrite your script from Q9 to output 3 intervals where the roots are located between.

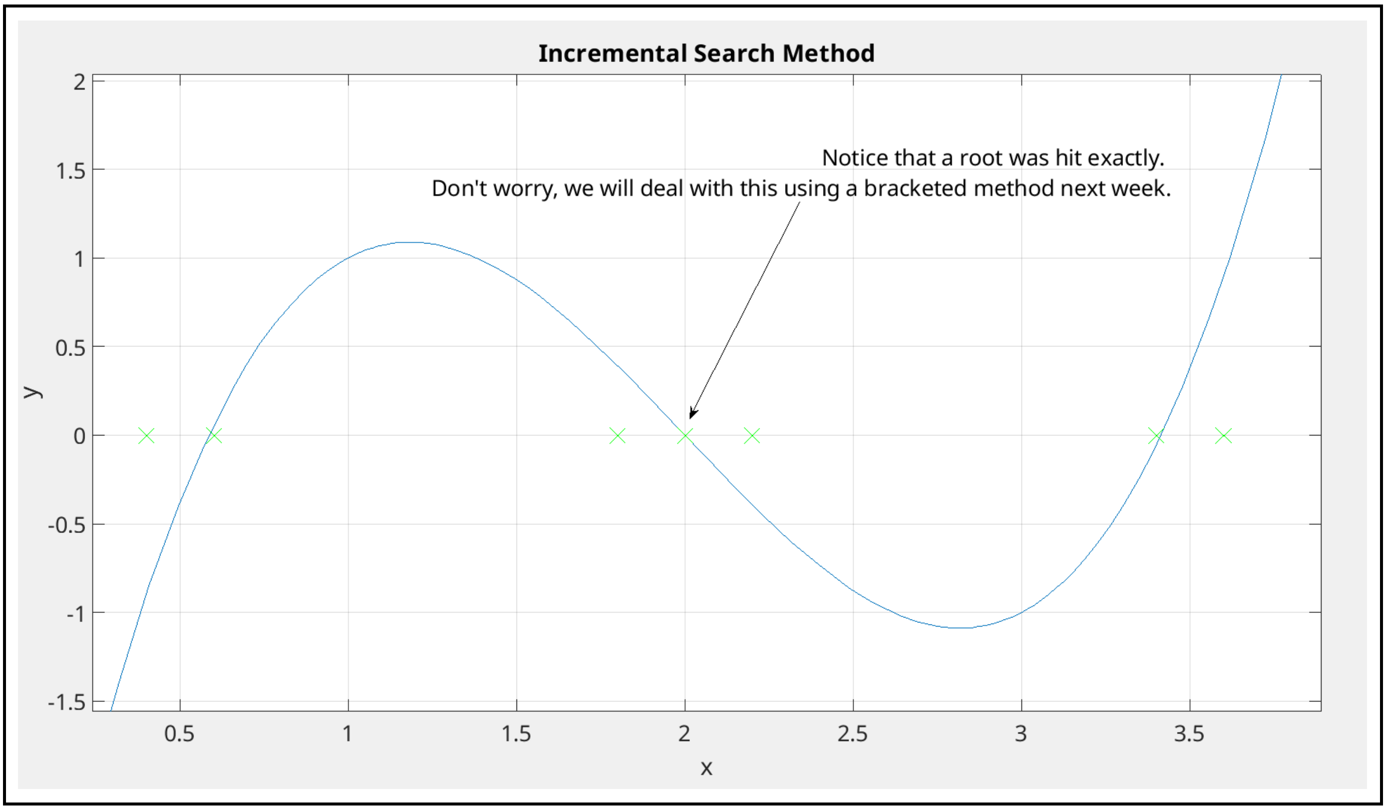
Command Window

```
>> Q9_Sol
The intervals are:
    0.4000    0.6000

    1.8000    2.0000

    2.0000    2.2000

    3.4000    3.6000
```



10. **[Generalising Method to Any Function]** Write a Matlab function file that implements the Incremental Search Method. The user should be able to input a function f(x), step size h, and range of values to search between $[x_0, x_{end}]$. The output should be a list of brackets (intervals) containing each root.

    Test your function with the function

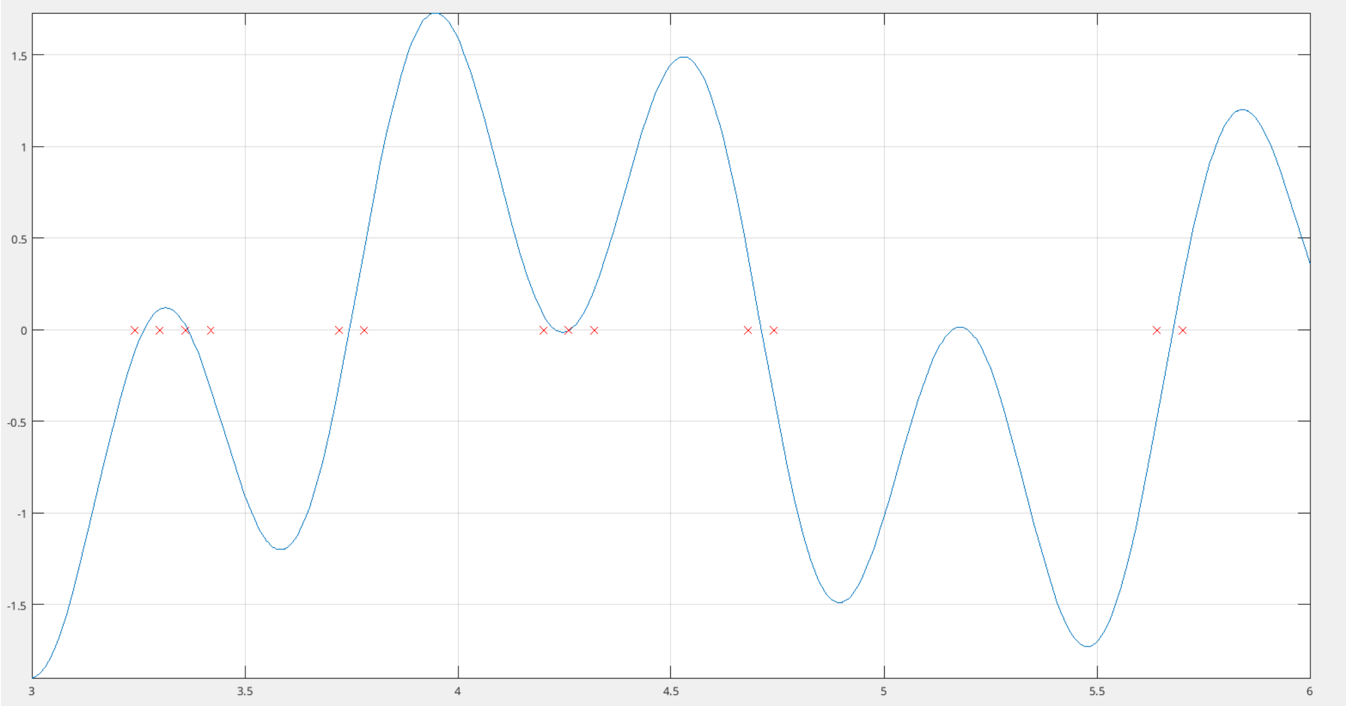
$$f(x) = \sin(10x) + \cos(3x)$$

    on the interval [3, 6] with step sizes of 0.06 and 0.03.

```
Command Window
>> f = @(x) sin(10*x) + cos(3*x);
>> Q10_Sol(f, 0.06, [3 6])

ans =

    3.2400    3.3000
    3.3600    3.4200
    3.7200    3.7800
    4.2000    4.2600
    4.2600    4.3200
    4.6800    4.7400
    5.6400    5.7000
```



Command Window

```
>> f = @(x) sin(10*x) + cos(3*x);
>> Q10_Sol(f, 0.03, [3 6])

ans =

    3.2400    3.2700
    3.3600    3.3900
    3.7200    3.7500
    4.2000    4.2300
    4.2600    4.2900
    4.7100    4.7400
    5.1600    5.1900
    5.1900    5.2200
    5.6700    5.7000
```