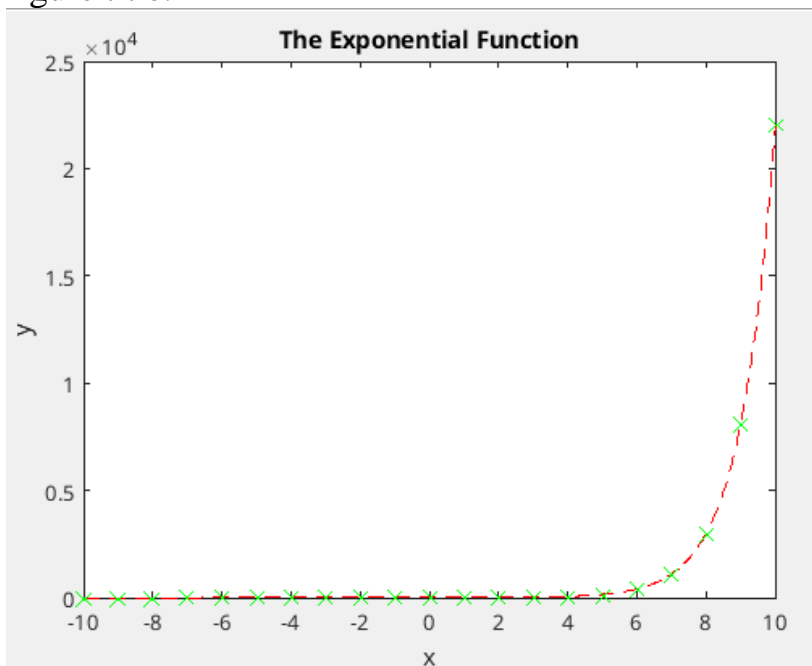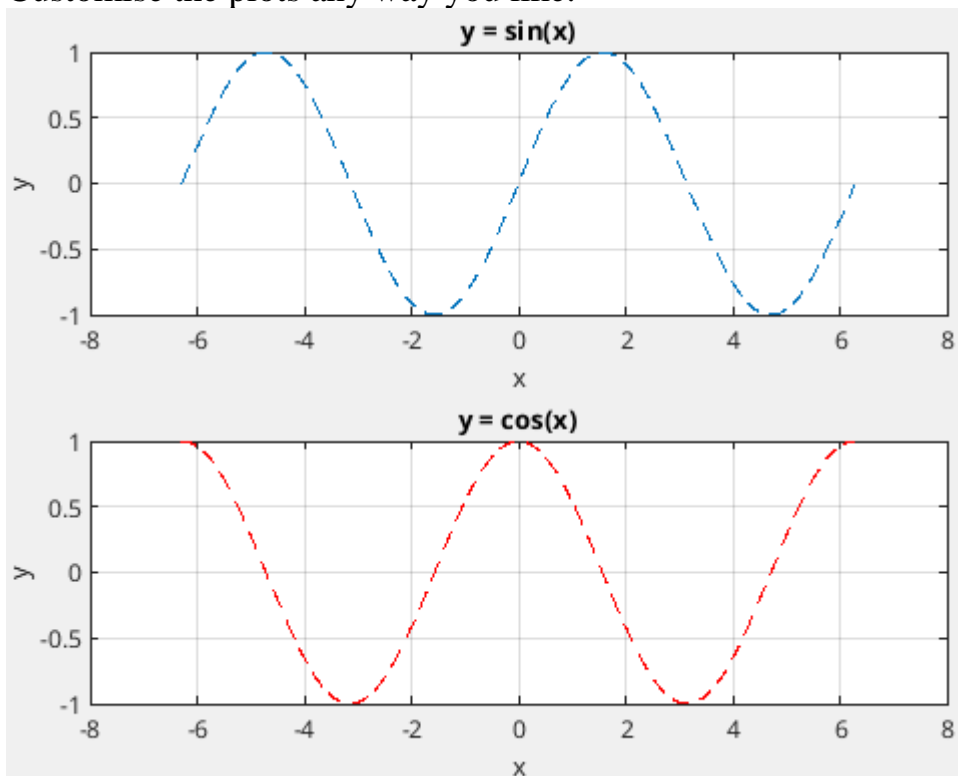# Practical Problems 2 – Plotting & Programming

1. Plot the exponential function between $x = -10$ and 10 using a red dashed line, then plot the points at each integer value of x in large green crosses. Remember to add axes labels and a figure title.
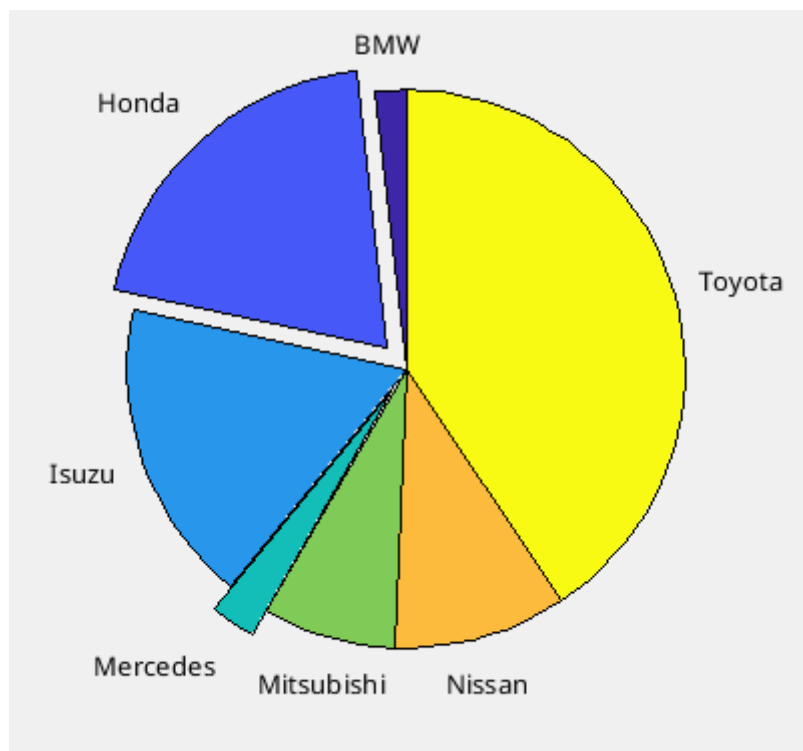

2. Use 2 subplots above and below each other to plot both $\sin(x)$ and $\cos(x)$ between -2π and 2π. Customise the plots any way you like.

3. Plot the following data in a pie chart and explode the Honda and Mercedes segments.
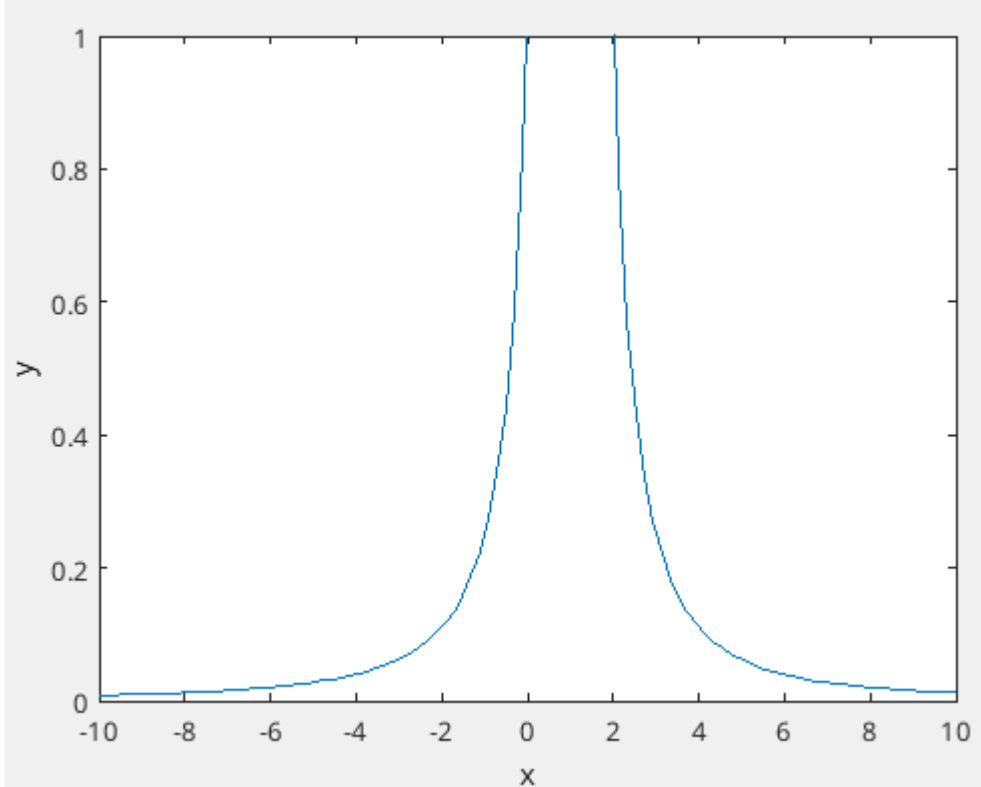
Car Manufacturer	Number of Sales
BMW	37,210
Honda	412,178
Isuzu	366,040
Mercedes	53,442
Mitsubishi	157,803
Nissan	210,000
Toyota	845,213

4. Plot

$$\frac{1}{(x-1)^2}$$

between -10 and 10, but only display the y-axis between 0 and 1.

5. Create a character array called **x** that contains the first 6 letters of the alphabet. The vector should contain 6 separate character elements.

6. Display every odd element in the command window with a 2 second pause in between.

> Matlab should display a c e in the command window with a gap of 2 seconds in between

7. Create a 3 × 3 string matrix called **A** containing the first 9 letters of the alphabet.

8. Print the diagonal entries of **A** with a 3 second pause in between.

> Matlab should display a e i in the command window with a gap of 2 seconds in between

9. Create a 1 × 8 vector of random integers called *y*.

> (example solution - yours may be different)
>
> y =
>
>   6   10   7   10   3   7   3   7

10. Write a for loop that replaces elements of *y* that have an even value with 0 and elements that have an odd value with 1.

> (example solution - yours may be different)

```
y =

   0   0   1   0   1   1   1   1
```

11. Create a 6 x 1 vector of random integers between 20 and 50 called *z*.

```
(example solution - yours may be different)

z =

   41
   22
   27
   26
   40
   46
```

12. Write a for loop that subtracts 12 from an element of *z* that has a value between 27 and 43.

```
(example solution - yours may be different)

z =

   29
   22
   27
   26
   28
   46
```

13. Write a for loop that doubles the 2nd, 4th and 5th entries of *y*.

```
(example solution - yours may be different)

y =

   6   20   7   20   6   7   3   7
```

14. Repeat the task in question 13 **without using a loop** (just vector notation).

15. Write a while loop that multiplies a number by 2 until the answer is greater than 100. Test your code with numbers of your choice.

> For example if you start with 7 then the final number before 100 is reached is 56.

16. Write a while loop that decreases a number by 10 until it is less than 0.

> For example if you start with 27 then the final number before 0 is reached is 7.

17. Write a function that accepts a number as an argument. Check if it is positive or negative. If it is positive then double it, if it is negative then halve it. Return the value. Remember to give your function a descriptive name.

> For example if the input is 18 then the output should be 36, and if the input is -18 then the output should be -9.

18. Write a function that accepts a number as an argument. Check if it is even or odd. If it is even, add 1 to it. If it is odd, subtract 1 from it. Return the value.

> For example if the input is 18 then the output should be 19, and if the input is 17 then the output should be 16.

19. Write a function that accepts 2 numbers as arguments. Check if they are equal to each other. If they are, then add them. If they are not, then return the absolute value of their difference.

> For example if the inputs are 2 and 2 then the output should be 4, iif the inputs are 2 and 3 then the output should be 1.

20. Write a function that accepts 2 numbers as arguments. If their sum is not equal to 10 then return false. If it is equal to 10, return true.

> For example if the inputs are 6 and 4 then the output should be true (or 1), and if the inputs are 3 and 11 then the output should be false (or 0).

21. Write a function that accepts 3 numbers as arguments. Add the largest 2 numbers together. Return the result.

For example if the inputs are 1,-2 and 3 then the output should be 4.

22. Write a function that accepts 3 numbers as arguments. Subtract the smallest number from the largest. Return the result.

For example if the inputs are 1,-2 and 3 then the output should be 5.

23. Write a function that accepts a vector as an input argument. The function should print the number of elements of the input vector as well as its smallest 2 values. If the vector has less than 2 elements the function should tell the user it requires more inputs.

For example if the input vector is [-4] then the function should display "Requires at least 2 inputs" but if the input vector is [-4 -2 0 2 4] then the function should display "No. of elements is 5 and the smallest two values are -4 and -2".

24. Write a script that creates a random vector of 1's and 0's then finds the indexes of the non-zero elements (read about the find function).

For example if the random vector is [1 0 0 1 1 0 0 1 0] then the output should be the vector [1 4 5 8].

25. Write a script which creates a random 4 $\times$ 4 matrix of integers, then finds the even numbers and sets them equal to 0. The final matrix should be only odd numbers and 0 values.

For example if the random matrix is

1 3 4 9
2 2 7 7
3 1 4 5
7 9 8 1

then the output should be the matrix

1 3 0 9
0 0 7 7
3 1 0 5

26. Convert the script into a function which accepts a value, *n*, as an input argument to make an *n* x *n* matrix. Set the even numbers equal to 0.

27. Make a new function which takes two input arguments, *n* and *m*, which define an *n* x *m* matrix. Set the odd numbers equal to 0.

28. Create a function which takes a value, *x*, and matrix, **A**, as inputs. It will set any entries of the input matrix greater than *x* equal to 0 and return the result.

For example if the random matrix is

1 3 4 9
2 2 7 7
3 1 4 5
7 9 8 1

and the value of *x* is 5 then the output should be the matrix

1 3 4 0
2 2 0 0
3 1 4 5
0 0 0 1

29. The following series converges to 1. Write a function that accepts a number specifying a tolerance, *tol*, as input argument. Calculate the sum of the series, until the answer is within *tol* away from 1. Display how many terms are required for the given precision.

$$S = \sum_{i=1}^{n} \frac{1}{2^i}$$

For example if the tolerance is 0.3 then n should be 2. If the tolerance is 0.04 then n should be 5. If the tolerance is 0.0005 then n should be 11.

30. Write a function which accepts a number, *n*, as input argument. Calculate Euler's number up to and including *n* terms *using* the following recursive formula.

$$e = \sum_{i=0}^{n} \frac{1}{i!}$$

For example if n = 3 the approximation is 2.666666666666667. If n = 7 the approximation is 2.718253968253968.

31. **Challenge Problem**

The golden spiral can be approximated using the Fibonacci sequence:

1, 1, 2, 3, 5, 8, 13, 21, 34…

where each number is the sum of the previous 2 numbers.

Recreate the Fibonacci spiral shown on the next page by plotting squares as shown with lengths equal to the numbers in the Fibonacci sequence, and then adding quarter circular arcs, which join opposite corners of each square, to the plot. The file should be a function or script that specifies the number of terms in the Fibonacci sequence and produces the corresponding plot.

Hint: You will probably need to use pen and paper to figure out the coordinates of each of the squares. Also to plot a square you must use 5 points. The first and last coordinate points must be the same!