Over the past few weeks, we have learned a lot about the C++ programming language and its applications in designing simple algorithms. In this assignment, we utilized this knowledge to implement two numerical solvers for Ordinary Differential Equations (ODEs), namely *4th order Runge-Kutta* and *Forward Euler Method*, and to find equilibrium points using the bisection method. To achieve this, we make use of conditional statements, loops, recursion and switch statements.

The Fourth order Runge-Kutta method approximates numerical values ordinary differential equations by setting four variables (k1, k2, k3, and k3) where each is dependent on the preceding one and are evaluated as multiples of a predetermined delta (h) value and the value of the ODE at x_n, x_n+h/2, x_n+h/2 and x_n+ h respectively. The euler method on the other hand estimates the underlying function for the differential by incrementing the preceding value by delta multiples of f(x, y) where x and y are the function variables. My code implements this in detail and descriptions and comments are included.

Many of the concepts we have discussed and from the book are demonstrated in this exercise. The implementation of the Euler method makes use of recursion, a new concept that I learned recently. Moreover, there is widespread use of functions and the power of function reusability is very much evident. The Runge-Kutta approach makes use of the while loop and occasional conditional statements. We have also declared some of the methods as prototypes so that they can be accessed from anywhere in the class. The entry point to the program is the main method which can accept an array of parameters to be used in the code. In the program i have implemented, all the parameters needed to run the code are encapsulated in the main class and there is no need for the user to input values. Ideally, we would expose variables such as the methodology for which to estimate the differential or the number of partitions, the spacing, initial values, function e.t.c. However, for the purpose of this exercise, considering that the requirements are set, we do not need such flexibility.

To run this code, please execute ./run.sh in the differentiator package. This ./run.sh contains the compiler and run commands to run the program. If it is the first time you are running the script, you might need to modify the permission i.e chmod 777 run.sh

Here are my findings:

Question 1:

A. Runge Kutta method

- i.) (f(1) given f(0) = 0
  - Partitions: 4 = 2.78172
  - Partitions: 8 = 2.78172
  - Partitions: 16 = 2.78172
  - Partitions: 32 = 2.78172
  - Partitions: 64 = 2.78172
  - Partitions: 128 = 2.78172
  - Partitions: 256 = 2.78172

Partitions: 512 = 2.78172
Partitions: 1024 = 2.78172
Partitions: 2048 = 2.78172
Partitions: 4096 = 2.78172
Partitions: 8192 = 2.78172

ii.) (f(1) given f(0) = 10

Partitions: 4 = 12.7817
Partitions: 8 = 12.7817
Partitions: 16 = 12.7817
Partitions: 32 = 12.7817
Partitions: 64 = 12.7817
Partitions: 128 = 12.7817
Partitions: 256 = 12.7817
Partitions: 512 = 12.7817
Partitions: 1024 = 12.7817
Partitions: 2048 = 12.7817
Partitions: 4096 = 12.7817
Partitions: 8192 = 12.7817

B. Euler method

i.) (f(1) given f(0) = 0

Partitions: 4 = 2.86256
Partitions: 8 = 2.82437
Partitions: 16 = 2.80361
Partitions: 32 = 2.7928
Partitions: 64 = 2.78729
Partitions: 128 = 2.78452
Partitions: 256 = 2.78312
Partitions: 512 = 2.78242
Partitions: 1024 = 2.78207
Partitions: 2048 = 2.78189
Partitions: 4096 = 2.78181
Partitions: 8192 = 2.78176

ii.) (f(1) given f(0) = 10

Partitions: 4 = 12.8626
Partitions: 8 = 12.8244
Partitions: 16 = 12.8036
Partitions: 32 = 12.7928
Partitions: 64 = 12.7873
Partitions: 128 = 12.7845
Partitions: 256 = 12.7831
Partitions: 512 = 12.7824
Partitions: 1024 = 12.7821

Partitions: 2048 = 12.7819
Partitions: 4096 = 12.7818
Partitions: 8192 = 12.7818

Question 2: Finding equilibrium points for f(x) = x * sin(x) - 1
In the range [-4, 4] for f(x) = x * sin(x) - 1 has the following equilibria

1: Stable between t = {-2.7728, -2.7726}
2: Unstable between t = {-1.1142, -1.114}
3: Stable between t = {1.114, 1.1142}
4: Unstable between t = {2.7726, 2.7728}

Question 3: Finding equilibria using forward euler
f(-4) = -0.270016
f(-3.9375) = -0.50835
f(-3.875) = -0.732971
f(-3.8125) = -0.94361
f(-3.75) = -1.14007
f(-3.6875) = -1.32223
f(-3.625) = -1.49003
f(-3.5625) = -1.64351
f(-3.5) = -1.78274
f(-3.4375) = -1.90789
f(-3.375) = -2.01918
f(-3.3125) = -2.11689
f(-3.25) = -2.20137
f(-3.1875) = -2.27301
f(-3.125) = -2.33227
f(-3.0625) = -2.37965
f(-3) = -2.41569
f(-2.9375) = -2.44098
f(-2.875) = -2.45614
f(-2.8125) = -2.46183
f(-2.75) = -2.45873
f(-2.6875) = -2.44755
f(-2.625) = -2.42902
f(-2.5625) = -2.40387
f(-2.5) = -2.37286
f(-2.4375) = -2.33674
f(-2.375) = -2.29627
f(-2.3125) = -2.25221
f(-2.25) = -2.20529
f(-2.1875) = -2.15626
f(-2.125) = -2.10582
f(-2.0625) = -2.05469

f(-2) = -2.00353
f(-1.9375) = -1.95298
f(-1.875) = -1.90368
f(-1.8125) = -1.85619
f(-1.75) = -1.81107
f(-1.6875) = -1.76881
f(-1.625) = -1.7299
f(-1.5625) = -1.69475
f(-1.5) = -1.66373
f(-1.4375) = -1.63719
f(-1.375) = -1.61539
f(-1.3125) = -1.59858
f(-1.25) = -1.58694
f(-1.1875) = -1.58061
f(-1.125) = -1.57967
f(-1.0625) = -1.58416
f(-1) = -1.59406
f(-0.9375) = -1.60933
f(-0.875) = -1.62986
f(-0.8125) = -1.65549
f(-0.75) = -1.68604
f(-0.6875) = -1.72127
f(-0.625) = -1.76091
f(-0.5625) = -1.80467
f(-0.5) = -1.85218
f(-0.4375) = -1.9031
f(-0.375) = -1.95701
f(-0.3125) = -2.01351
f(-0.25) = -2.07214
f(-0.1875) = -2.13246
f(-0.125) = -2.19399
f(-0.0625) = -2.25624
f(0) = -2.31874
f(0.0625) = -2.381
f(0.125) = -2.44252
f(0.1875) = -2.50284
f(0.25) = -2.56147
f(0.3125) = -2.61797
f(0.375) = -2.67188
f(0.4375) = -2.7228
f(0.5) = -2.77032
f(0.5625) = -2.81407
f(0.625) = -2.85371

f(0.6875) = -2.88894
f(0.75) = -2.91949
f(0.8125) = -2.94513
f(0.875) = -2.96565
f(0.9375) = -2.98092
f(1) = -2.99083
f(1.0625) = -2.99532
f(1.125) = -2.99438
f(1.1875) = -2.98804
f(1.25) = -2.9764
f(1.3125) = -2.95959
f(1.375) = -2.9378
f(1.4375) = -2.91125
f(1.5) = -2.88024
f(1.5625) = -2.84508
f(1.625) = -2.80617
f(1.6875) = -2.76392
f(1.75) = -2.71879
f(1.8125) = -2.67131
f(1.875) = -2.622
f(1.9375) = -2.57146
f(2) = -2.52029
f(2.0625) = -2.46916
f(2.125) = -2.41873
f(2.1875) = -2.36969
f(2.25) = -2.32278
f(2.3125) = -2.27871
f(2.375) = -2.23824
f(2.4375) = -2.20212
f(2.5) = -2.17111
f(2.5625) = -2.14596
f(2.625) = -2.12743
f(2.6875) = -2.11625
f(2.75) = -2.11315
f(2.8125) = -2.11884
f(2.875) = -2.13401
f(2.9375) = -2.15929
f(3) = -2.19533
f(3.0625) = -2.24271
f(3.125) = -2.30197
f(3.1875) = -2.37361
f(3.25) = -2.45809
f(3.3125) = -2.5558

f(3.375) = -2.66709
f(3.4375) = -2.79224
f(3.5) = -2.93147
f(3.5625) = -3.08495
f(3.625) = -3.25276
f(3.6875) = -3.43491
f(3.75) = -3.63137
f(3.8125) = -3.84201
f(3.875) = -4.06663
f(3.9375) = -4.30497

We see that this function has inflection in the points determined above. That is:

1: Stable between t = {-2.7728, -2.7726}
2: Unstable between t = {-1.1142, -1.114}
3: Stable between t = {1.114, 1.1142}
4: Unstable between t = {2.7726, 2.7728}