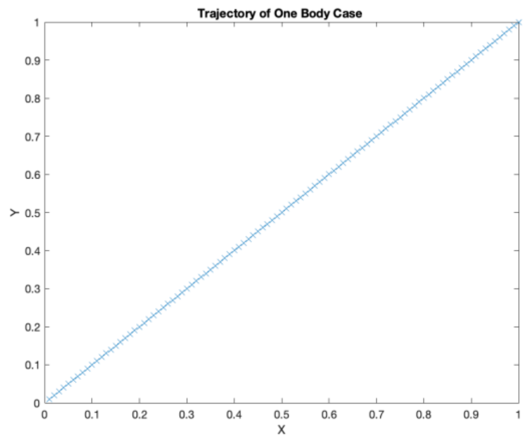
PART 2- Numerical Methods Coursework

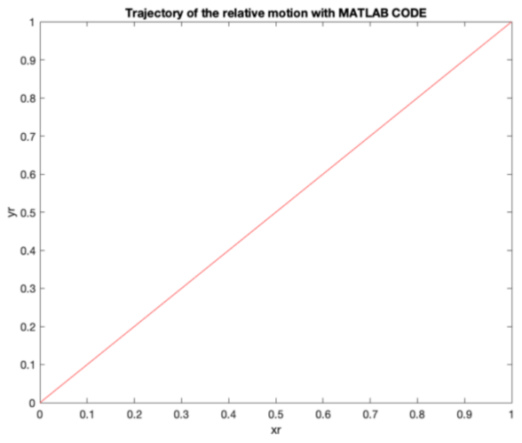
5. (A zip file containing the C++ code has been submitted to Part 2 Submission box)

6.

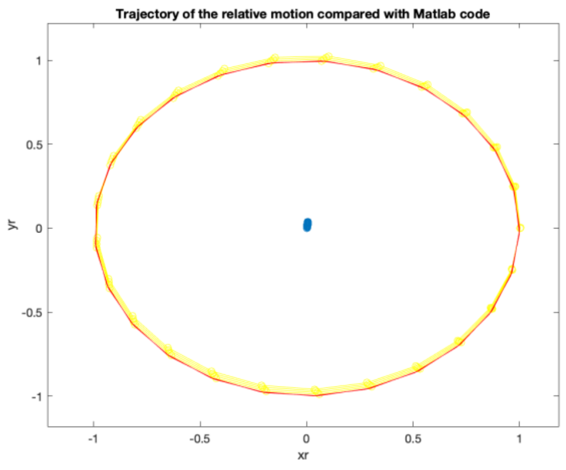
A. The table displays the X and Y components of final positions and velocities of bodies

System	Body	X Position	Y Position	X velocity	Y velocity
1 Body Orbit	1	1	1	1	1
2 Body Orbit	1	0.005002	0.03978	-0.01458	0.0001959
	2	1.004	0.00329	0.9135	24.99
2 Body Deflect	1	0.262	0.1143	0.3078	0.2533
	2	-1.618	2.827	-3.057	2.446
3 Body Orbit	1	-0.0000310	-0.0000238	0.0003167	-0.0003951
	2	-1.093	1.06	-13.93	-9.368
	3	0.9612	-1.135	15.28	7.953

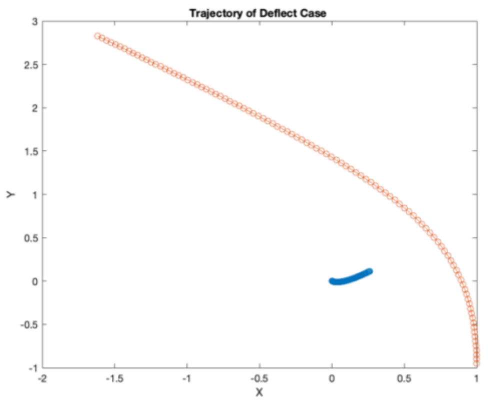


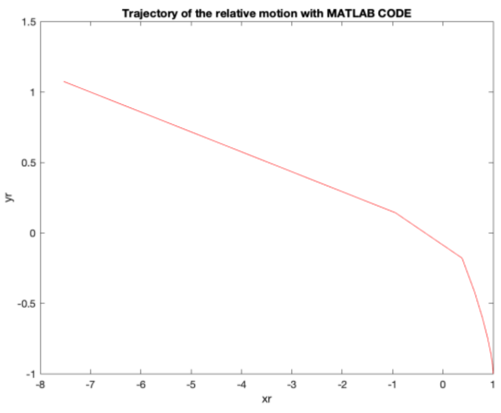


For the case of the one body, as shown on the figure on the left, the trajectory that has been evaluated on C++ seems to be consistent with the trajectory in the figure on the right, which was the trajectory that was computed on MATLAB.

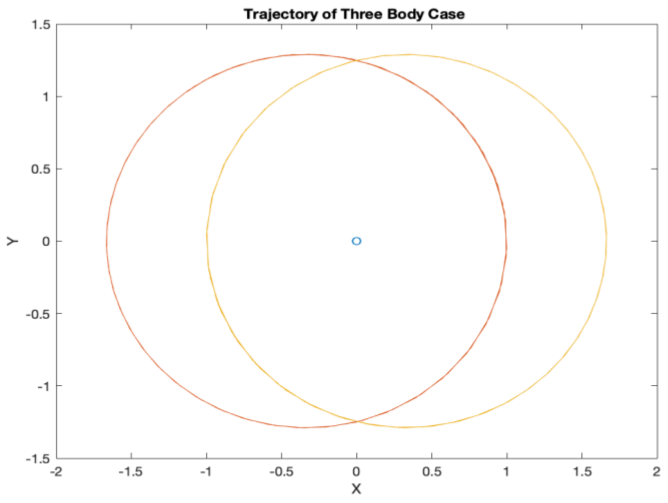


For the orbit of the two-body case, the trajectory seems to be consistent with MATLAB, this is depicted in the figure above where both trajectories seem to coincide with each other. The yellow and blue lines indicate the C++ code, whilst the red indicates the MATLAB code.

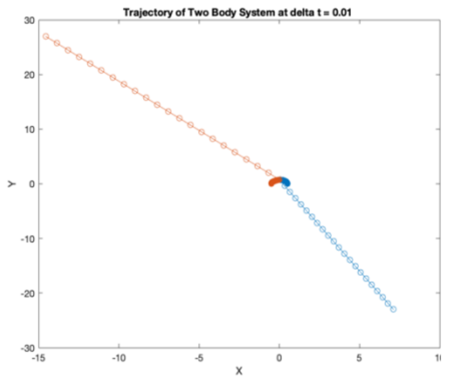


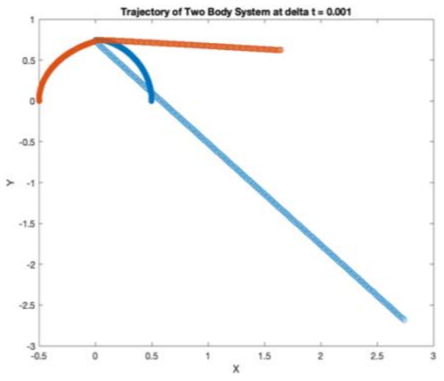


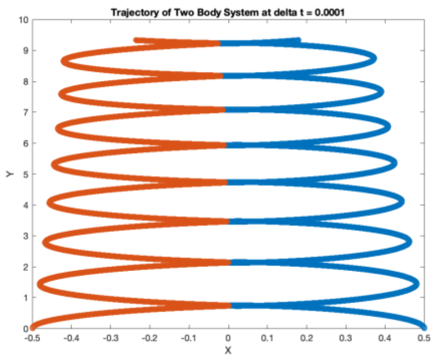
For the deflect case the relative trajectory of the lighter body on the left figure seems to follow the trajectory depicted by that of the MATLAB code on the figure on the right.



B. Investigation on the influence of delta t on the system has allowed me to conclude that a time step of at least 0.0001 is required for there to be convergence, this is displayed below.

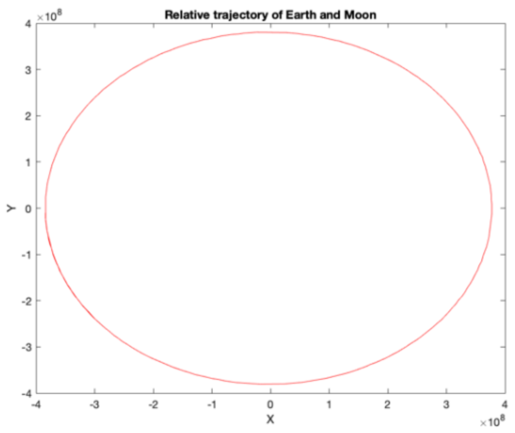


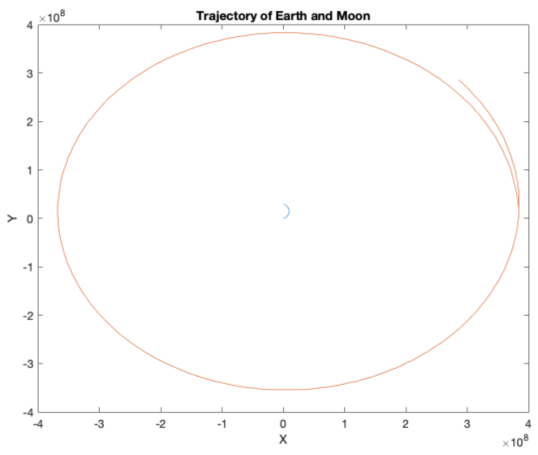




C. The table below displays the parameters of the Earth-Moon system, and the figures display the relative motion and also the trajectory of the Earth and Moon.

Constants	6.67e-11	2.63e+3	10000		
Earth	0	0	0	0	5.97e+24
Moon	3.84e+8	0	0	1020	7.35e+22





D. Advantages:

- ⇒ C++ is more efficient with memory, allowing for the simulation of larger, more complex problems.
- ⇒ C++ is a faster and more efficient language than interpreted languages. It compiles the code before running it, which makes it more efficient and faster than interpreted languages. This is important when dealing with any large and complex simulation.
- ⇒ C++ has a larger set of features than most interpreted languages, such as operator overloading and templates, which makes it easier to write code that is more efficient and easier to understand.

Disadvantages:

- ⇒ C++ may be less portable than an interpreted language, making it more difficult to share programs between different platforms.
- ⇒ C++ is more difficult to learn and use than interpreted languages. It requires more code to accomplish the same tasks, and a deeper understanding of the language is needed to write code that is efficient and easy to understand.
- ⇒ C++ programs are more prone to errors than interpreted languages since type errors can be hard to spot.

E. I used the STL to exploit the C++ STL library for efficient data structures, to store and manipulate the data for the bodies in the simulation. The STL (Standard Template Library) was used in this code in order to include the necessary void x and y acceleration functions of a body due to the interaction of forces with all the other bodies. Within my void Runge Kutta Function, acceleration functions were used so that values of each bodies' acceleration were numerically integrated to obtain the values of x and y components of velocities and positions at each timestep. Specifically, the pow() and sqrt() functions were used from the STL. I also used object-oriented programming to create a struct called n_planets, which allowed me to easily store and access the data for each body in the simulation. This struct allowed for me to keep the data for each body separate and organized for efficient retrieval and manipulation, and held each body's positions and velocities and mass, along with the initial displacements and velocities. Functions have separated the code into distinct sections to allow for greater readability and maintainability. This allowed me to easily identify the different sections of the code and make any changes to the code more quickly and efficiently whenever there was an error with the values. This structure was then used to create an array of Body objects, which were used to store the required parameters for each body; this then allowed me to easily pass in information about each body to the various functions used in the code.