ENG1002 Project - Written Self-Assessment - N-Body Simulator

By Syeam Bin Abdullah | **Total Marks: 95/100**

# Introduction – Brief Overview of Project

I chose to create an n-body simulator, which basically simulates the movement of point masses in 3d space given some initial conditions set by the user. The positions of the particles are displayed as an animated 3d plot in MATLAB, along with the time that has passed since the user started the program. This document is a self-assessment of my ability to demonstrate key skills and leverage key concepts surrounding development process of said project, based on the rubric provided.

# Criterion 1 and 5 – Conceptual Coverage and Styling

## **Marks: 1: 20/20, 5: 20/20**

Although all the following programming concepts were effectively utilized throughout entirety of the project:

* Loops
* Vectors
* Matrices
* Conditional execution
* Functions

One function in particular – which computes the 3d net forces on particles given matrix inputs - wholly demonstrates my understanding and ability to effectively utilize said concepts. The following image provides a succinct overview of how certain concepts are leveraged in the inner workings of the subroutine: Text

Description automatically generated

Figure 1 – A function which is used to compute the net forces on all particles given their masses, radii between each of them (squared) and their corresponding radial vectors (of which is computed in other functions). Notice the consistent indenting and presence of succinct comments highlighting the purpose of certain operations

User-friendly input and output is also present, as shown below:

Graphical user interface, text, application

Description automatically generated

Figure 2 – Friendly user input

A picture containing diagram

Description automatically generated

Figure 3 – Animated 3d plot of all particles loaded from user input – updates over time by computing gravitational forces on every particle and applying them to every point in the scene.

# Criterion 2 – Value-add

## **Marks: 20/20**

The only contributions made to this project were mine (this is made apparent in Section 4**).** Certain snippets of code which have been written by me utilize some of MATLAB’s “rudimentary” operations (i.e., matrix addition and subtraction, concatenations, etc.), can in some cases be mimicked by utilizing some of MATLAB’s in-built frameworks. However, such luxuries were not utilized during development (except for MATLAB’s plotting package), with the intention of demonstrating acute conceptual coverage by implementing certain functionalities from scratch, of which include:

* Computing the radial unit vectors between 3d points in space
* Computing the forces vectors between 3d particles in 3d space given their masses and positions (as shown in Figure 1)
* Calculating the acceleration, velocity, and acceleration vectors of particles in space once their respective force vectors have been computed.

Caching the radial unit vectors between particles for later use when computing forces was a challenge that I overcame by utilizing a multi-dimensional array structure (tensor), the rows of which denoted the particle from which the vector was protruding from, and the “page” corresponded to the particle the unit vector was pointing to (see Figure 4). This in turn made it much easier to write out the algorithm for computing the forces between each particle (since indexing the radial vectors are intuitive) as per the formula:

Diagram

Description automatically generated

Figure 4 – Diagram visualising how radial vectors are stored as a tensor

Figure 5

# Criterion 3 - Incremental Development

## **Marks: 15/20**

Git version control was used throughout the development of the simulator, which in turn displays the different stages of development via commits in the “master” branch of the repository (see: <https://github.com/Shr1ftyy/matlab-project1/commits/master>). Although, it may appear a little more disorganized than ideal.

A screenshot of a computer

Description automatically generated with medium confidence

# Criterion 3 - Testing

## **Marks: 20/20**

Text

Description automatically generatedMy project utilizes MATLAB’s class-based testing framework to run unit tests on every critical component of the simulator and was updated throughout the development process. An example is provided below: