CENG216 – EE204
Joint Interdisciplinary Homework: Two-body Numerical Simulation

Announcement date: 24 May 2021
Due date for team selections: 23:59, 26 May 2021
Due date for submissions: 23:59, 04 June 2021

The two-body problem is to predict the motion of two planetary objects which are viewed as points. See the appendix for the details related to the problem. In this homework, design and implement a two-body simulator that adheres to the **Model-View-Controller** (MVC) design pattern as follows:

1. **Simulation:** Develop two applications one of which is written in **two_body_simulation.c** and the other in **two_body_simulation.py**. These programs must be functionally equivalent and include the following parts separately:
    a. **Model**: a C structure and Python class (or, ideally a data class) named **TwoBodyModel** that stores the necessary data needed to visualize the calculations.
    b. **Controller:** a C structure and a Python class named **TwoBodyController** that stores the parameters such as eccentricity and mass ratio to specify the simulation details, and related functions/methods to perform the simulation by numerically solving the related ordinary differential equations using the Euler's method or the fourth order Runge-Kutta method.
    c. **App:** a simple command-line interface which allows for specifying the parameters of the simulation: T (e.g. 10000), δt (e.g. 0.15), mass ratio (e.g. 0.5), eccentricity (e.g. 1.0) and method (e.g. runge-kutta). It must run the simulation from t = 0 to t = T in steps of δt using TwoBodyModel and TwoBodyController, and then save the location vectors in a text file.

2. **Animation:** Develop an application consists of a single source file written in Python Center and named as **two_body_animation.py**. This program must be completely independent of the simulation part and include the following parts separately:
    a. **View**: a class named **TwoBodyView** that can draw the two-bodies on a window interactively. You may implement the drawing functionality using OpenCV (a computer vision library), TkInter (a built-in GUI toolkit), PyQt (an advanced GUI toolkit), PyGame (a library designed for video games) or a library of your choice. This application must allow users to play/pause, rewind

and exit the animation using keyboard inputs (e.g., hitting space to play/pause) or graphical buttons (e.g., a button whose label toggles between "play" and "pause").

b. **App**: a simple program which reads the text file generated in the simulation step and visualize the data using TwoBodyView.

Please submit your C (one file) and Python (two files) source files and a text file named **readme.txt** in a zip archive named ⟨TeamName⟩.zip (omit the angle brackets). readme.txt should describe the C (e.g., C99), compiler (e.g., gcc 7.5.0) and Python (e.g., 3.9.4) versions, the Python library names (ideally names of the pip packages; for example opencv-python for OpenCV), any information necessary to use the applications (e.g., which keyboard buttons are used for which functionalities, and how the text file should be interpreted), and team members (Student ID, department, full name).

Note: This homework requires an interdisciplinary team of three to four students. At least must be at least one member from each departments. Fill “teams.xlsx” under the section “Files”. Follow the instructions given in the upper left cell.

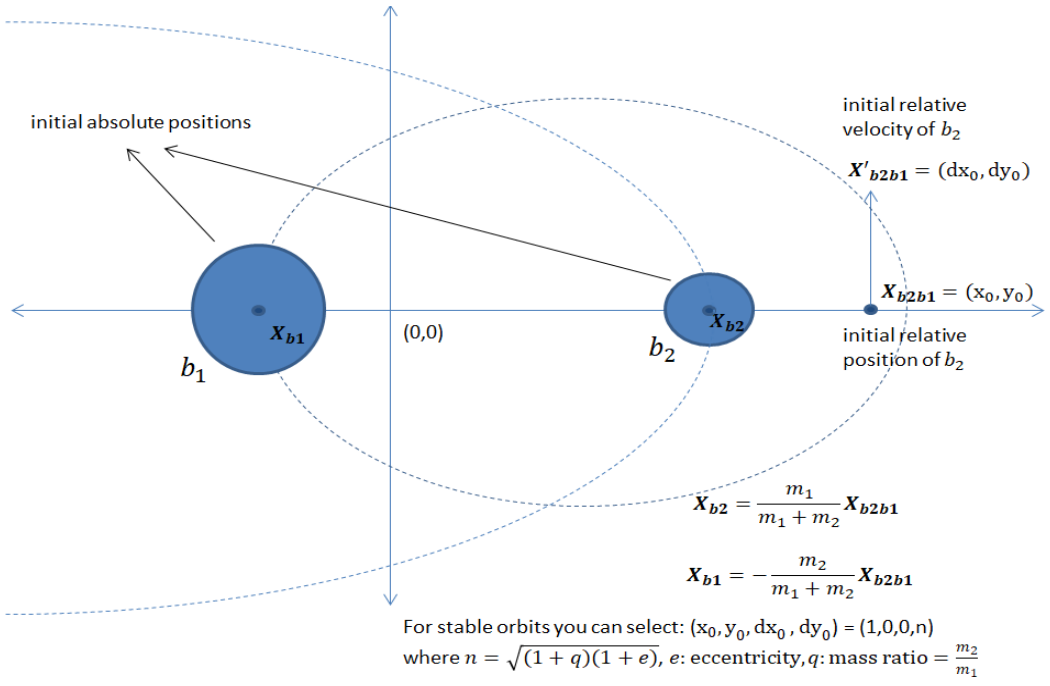
Appendix

You can compute the positions of planetary objects by solving following differential equations:

$$x'' = -x(1+q)/r^3$$

$$y'' = -y(1+q)/r^3$$

where x and y specify the relative positions of body2 with respect to body1, r defines the current distance between objects, q defines the mass ratio m_2/m_1. See the figure below.



Please check Example Simulator and its source code to see how the default values for certain variables are determined and how the second order differential equations are solved. Make sure that your animation looks similar to this online simulator for the same inputs when T is large, δt is small and the method is Runge-Kutta. Euler's method may give different results.

Hints

- If you do not set a meaningful step size the solution method of ODE can have a high error that may lead divergence from the stable orbit limits.
- In C, you should be careful about the precedence and associativity of the arrow (->) operator while dealing with structures and use parentheses whenever necessary. Mistakes regarding this may not result in a syntax error which makes them hard to detect.
- In order to learn the concept of MVC, you can check MVC Design Pattern Tutorial.
- You can learn more about ordinary differential equations: Numerical Analysis (Chapter 6 ODEs), 2nd Ed., T. Sauer.