Two-Body Problem

Michelle Pichardo Munoz
AM 129: Scientific Computing, HW 2
University of California, Santa Cruz
October 14, 2022

Contents

List of Figures

List of Tables

Git Commit Hash: 6e506541e182d530e0ab725dc1af51925c00152e

Introduction

To better understand the physical concept that is the two-body problem, we made edits to an existing computer program that simulates its evolution. The models used are Newton's equations of motion, i.e.,

$$m_1 \frac{d^2\vec{x_1}}{dt^2} = \vec{F_{1,2}} \tag{1}$$

$$m_2 \frac{d^2\vec{x_2}}{dt^2} = \vec{F_{2,1}} \tag{2}$$

Where m_1 and m_2 are point masses of our two bodies, $\vec{F_{1,2}}$ and $\vec{F_{2,1}}$ represent the force experienced by each mass, i.e. the first suffix denotes the mass experiencing the force and the second is the mass imparting the force, and lastly, $\frac{d^2\vec{x_i}}{dt^2}$ for $i = 1, 2$ are the second derivative of position vectors $\vec{x_1}, \vec{x_2}$. By Newton's third law, we can claim,

$$\vec{F_{1,2}} = -\vec{F_{2,1}} \tag{3}$$

such that the magnitudes equal $|\vec{F_{1,2}}| = |\vec{F_{2,1}}| = F$. The model used to describe the force, and ultimately our model for the numerical evolution, is Newton's law of Universal Gravitation i.e.

$$F = G\frac{m_1 \cdot m_2}{r^2} \tag{4}$$

where the radius r is represented formally as,

$$r = ||\vec{x_2} - \vec{x_1}|| = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \tag{5}$$

such that the position vectors are expressed as $\vec{x_i} = [x_i, y_i]$ for $i = 1, 2$. The gravitational constant G is unity and can be obtained by a change of variables. However, this proof is omitted. This leads us to the true expression of our force,

$$F = \frac{Gm_1m_2}{||\vec{x_2} - \vec{x_1}||^2} \cdot \frac{(\vec{x_2} - \vec{x_1})}{||\vec{x_2} - \vec{x_1}||} = \frac{1 \cdot m_1m_2 \cdot (\vec{x_2} - \vec{x_1})}{||\vec{x_2} - \vec{x_1}||^3} \tag{6}$$

With the RHS of Equations 1 and 2 completed, we now focus on the LHS. Rather than working with a second-order differential equation, we consider it's first-order equivalent which utilizes the momentum of the particles.

$$\vec{p_i} = m_i \frac{d\vec{x_i}}{dt} = m_i \cdot \vec{v_i} \quad \text{for } i = 1, 2 \tag{7}$$

Noting $\dot{\vec{p_i}} = m_i \frac{d^2\vec{x_i}}{dt^2} = \vec{F_{i,j}}$ for $i, j = 1, 2$ such that $i \neq j$. Our resulting system, after substitution into Equations 1 and 2, are as follows,

$$\begin{aligned} \frac{d\vec{x_2}}{dt} &= \frac{\vec{p_2}}{m_2} & \qquad \frac{d\vec{x_1}}{dt} &= \frac{\vec{p_1}}{m_1} \\ \frac{d\vec{p_2}}{dt} &= -\vec{F} & \qquad \frac{d\vec{p_1}}{dt} &= \vec{F} \end{aligned} \tag{8}$$

The four equations labeled Equation 8 along with their explicit representations in Equations 4 and 7 are the exact models used to evolve our two-body trajectory.

This report does not include the exact numerical method of discretizing the problem. However, the formal equation is,

$$t^n = n\Delta t \quad , \text{ for } , n \in \mathbb{N} \tag{9}$$

Materials

This short section lists the software used to perform and analyze the model.

Table 1: Software

Name	**Description**
Visual Studio Code	Code editor
Fortran	Main programming language used (compiler based)
Linux	open-source Unix-like operating system
Python	high-level programming language used to display data produced by Fortran
Poetry	Dependency management tool
Git	open source software for distributed version control used to track code changes
Given Files	makefile, orbits.f90, timestep.f90, utility.f90 used to make needed edits or execute the program
Processed Files	orbits.ex, sol.dat used to create the data needed to examine the trajectory
Display Files	plotter.py used to display the plots of our data

Questions

1. Why can `real(fp)` be used throughout the files?

In our program files ass seen in Table 1 we have one called `utility.f90` which holds our module consisting of key declarations; one being `real(fp)`. From here, we can call on the variables in this module from other files by including a call function i.e. `use utility`. The preprocessor is then instructed to view all variable instances as though it was declared in that file.

2. Comment on the separation of duties between `orbits.f90` and `timestep.f90`. Does the distinction make sense?

Yes, the distinction makes perfect sense regarding the file's contents. In the `orbits.f90` file, we find initial and end conditions/parameters along with key variable declarations necessary to the two body problems which result in orbits. The file additionally contains the code required to generate the trajectory, i.e., the orbit of the two bodies. In the `timestep.f90` file, we find our system's actual model, which is needed to create a loop program that essentially increments in discrete time, i.e., a time step. The output of `timestep.f90` is then called into `orbits.f90` and exported to a data file, `sol.dat`.

---

3. What would you need to do to increase this system to evolve 3 particles? What about N particles?

To increase the number of particles, we must add their respective pair of first-order differential equations. However, in theory, if we're only examining Equation 8 this might seem simple, but in reality, I would not know how to account for Equation 4. Each new point mass would have a force on each other preexisting mass, which must be accounted for in our equations.

---

4. The code evolves the dynamics in the plane. What would you need to do to raise this to dynamics in full 3D space? Would this be easier or harder than increasing the number of particles?

To increase the dynamics, we need to extend our position vectors in the plane to have a third dimension. I believe we might do this more thoroughly by incorporating rotational angles θ, ϕ, ψ and essentially creating a six-dimensional vector. However, this is only a theory, and I believe it would be easier to do this with a transformation rather than increasing the number of particles observed.

---

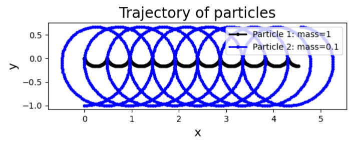
5. Fortran Flags

A list of Fortran flags used in our processing is provided with a brief description of their purpose.

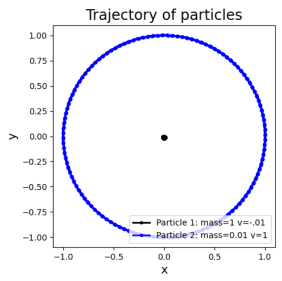
1. `-Wall` : provides warnings for familiar sources of error
2. `-Wextra` : provides more warnings than `-Wall`, specifically to subroutine arguments that are in use
3. `-ffree-line-length-none` : disables the line length limit, i.e., it removes the need to include line continuation where it might be needed
4. `-fPIC` : I could not find information on this flag, however I understand the `-f` represents a flag in Fortran's dialect
5. `-fcheck=all` : this may slow the program; however, it checks that any array is appropriately used, i.e., checks that the indices
6. `-fbacktrace` : provides a warning that the program crashed and gives a trace to the function or subroutine calling the error
7. `-g` : generates extra debugging information for the GDB debugger or the GNU Project Debugger which debugs C, but in this case Fortran

Results

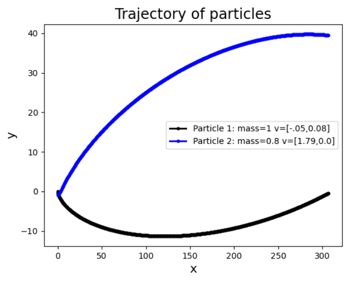
The following Figure consists of three trials run with our program. The images are kept to this size to remain within the 4-page limit. I underestimated the formatting of this report.



(a) A more massive particle. Mass1 = 1 & Mass2 = 0.1. Formally Mass2 = 0.01. Final time step is 50.



(b) Zero Total Momentum. Mass1 = 1 with velocity = [-0.01,0] and Mass2 = 0.01 with velocity =[1,0]. Final time step is 50.



(c) Possibly a plot of the moment just before the escape velocity. Mass1 = 1 with velocity = [-0.05, 0.08] and Mass2 = 0.8 with velocity = [1.79, 0]. Final time step is 400.

Figure 1: Results of three different simulations with varying initial conditions and final times.

Table 2: Trajectory's Initial Conditions

Variable	**Value: Particle 1**	**Value: Particle 2**	**Trial**
Position (x, y) m	(0,0)	(0,-1)	Figure 1.a, 1.b, 1.c
Mass kg	**1.0**	**0.1**	**Figure 1.a**
	1.0	0.01	Figure 1.b
	1.0	0.8	Figure 1.c
Velocity (v_x, v_y) m/s	**(0,0)**	**(1,0)**	**Figure 1.a**
	(-0.01,0)	(1,0)	Figure 1.b
	(-0.05,0.08)	(1.79, 0)	Figure 1.c

Conclusion

We used a provided Fortran program template and base theory of our two-body problem to generate specific visualizations of their trajectories. The ease of editing allowed effortless detection of the known sensitivities to initial conditions. Though, we did not outline that fact in our results. Another pitfall is not accounting for the total momentum of systems shown in Figures 1. a and 1. c., which were requested. Moving forward, we can expect to implement a few improvements and include all requested data, i.e., showing a progression of images revealing the sensitivity of ICs and implementing a potential movie of our data plots. The film will add emphasis to our system's evolution. Cutting down our information to fit four pages will also be coolly considered.