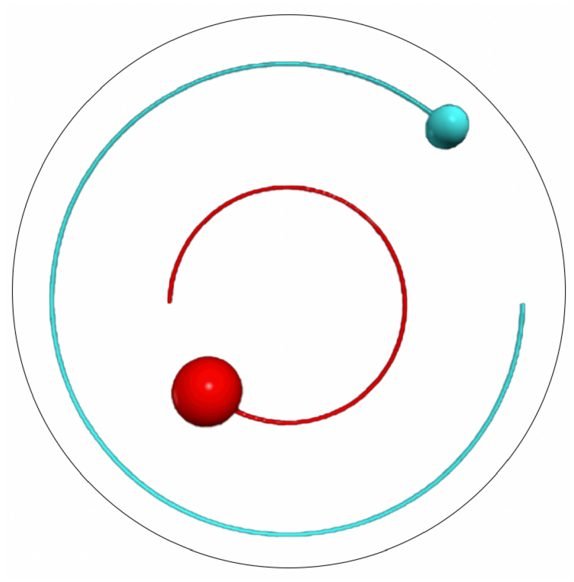
Eton College

Lent Physics Project

2023



Modelling the Two-Body Problem

Matteo Mountain

Modelling the Two-Body Problem

Matteo Mountain

January 31, 2023

Contents

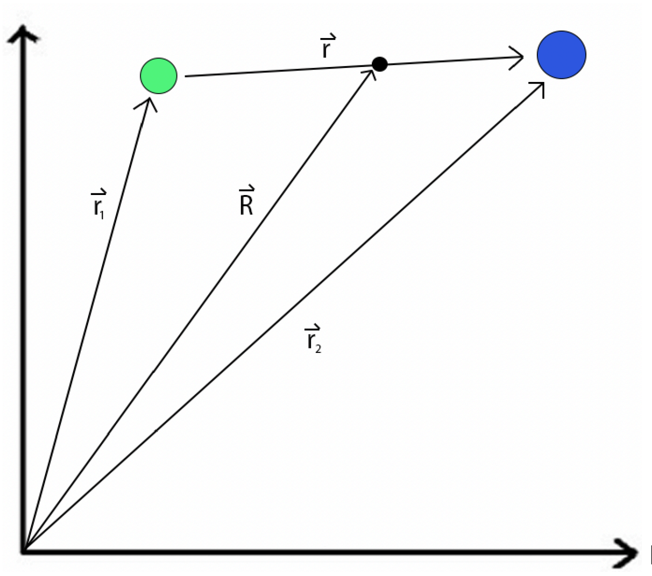
1 Introduction

This paper will explore the Two-Body Problem and derivations of equations needed to simulate it in Python. The Two-Body Problem refers to the motion of massive bodies under the influence of their mutual gravitational attraction, causing them to orbit one another [1]. This problem has been a topic of study for centuries, with early solutions dating back to the 17th century by physicists such as Isaac Newton and Johannes Kepler.

The Two-Body Problem has many practical applications and is recognised as a fundamental problem in celestial mechanics. It is used to predict the orbits of satellites, the motion of binary stars, and to avoid collision with space debris. Before modelling in Python, it is necessary to derive the equations that govern for the Two-Body Problem.

2 Derivation of Relevant Equations

2.1 High Level Overview



[2]

The above graph represents two bodies rotating about a point, $\vec{R}$, from the origin. The bodies are $\vec{r}$ apart from each other and are $\vec{r}_1$ and $\vec{r}_2$ from the origin.

Algebraic expressions for the following need to be derived:

1. The barycenter, $\vec{R}$
2. Position Vectors $\vec{r}_1$ and $\vec{r}_2$ in terms of $\vec{R}$ and mass
3. Circular Velocity

For each time increment of the simulation, Δt, the following needs to happen:

1. The vector $\vec{r}$ (displacement between the two bodies) is determined
2. The gravitational force is determined
3. The momentum is determined
4. The positions of the bodies are determined

2.2 Barycenter

The barycenter is defined as the center of mass of two or more bodies that are orbiting each other. This barycenter also happens to be the point around which both bodies will orbit[3]. The equation for the barycenter is:

$$\vec{R} = \frac{m_1\vec{r}_1 + m_2\vec{r}_2}{M}$$

where M is the total mass of the two bodies, $m_1 + m_2$

2.3 Position Vectors

Both $\vec{r}_1$ and $\vec{r}_2$ are to be expressed in terms of $\vec{R}$ and $\vec{r}$. The purpose of having the equations for the position vectors in this form will make calculating the coordinates of the bodies in the simulation much more simple.

Rearrange the barycenter equation:

$$\vec{R} = \frac{m_1\vec{r}_1}{M} + \frac{m_2\vec{r}_2}{M}$$

Substitute $\vec{r} + \vec{r}_1$ for $\vec{r}_2$:

$$\vec{R} = \frac{m_1\vec{r}_1}{M} + \frac{m_2}{M}(\vec{r} + \vec{r}_1)$$
$$\vec{R} = \vec{r}_1(\frac{m_1 + m_2}{M}) + \frac{m_2}{M}\vec{r}$$
$$\vec{R} = \vec{r}_1 + \frac{m_2}{M}\vec{r}$$
$$\vec{r}_1 = \vec{R} - \frac{m_2}{M}\vec{r}$$

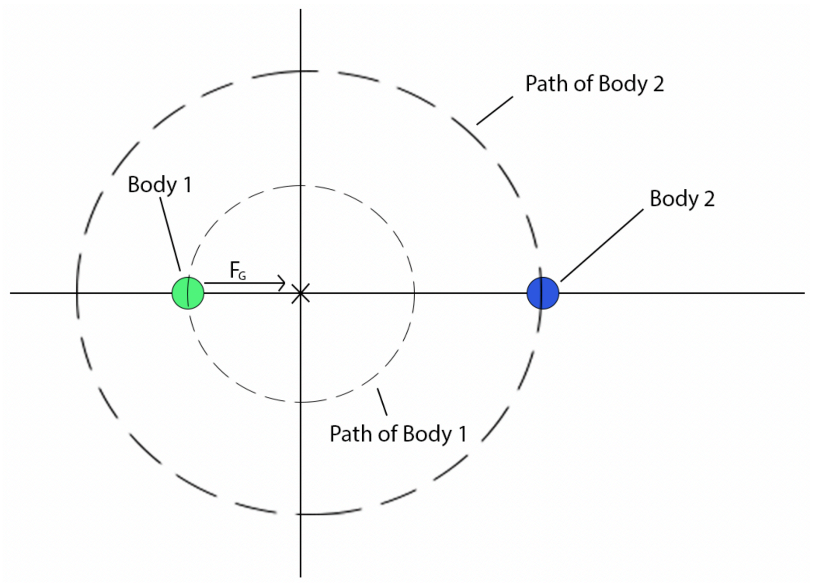
Repeating the above steps for $\vec{r}_2$ gives:

$$\vec{r}_2 = \vec{R} + \frac{m_2}{M}\vec{r}$$

2.4 Circular Velocity

Another factor to consider is the velocity required for a body to have a circular orbit. To derive an equation which describes this, the following equations need to be found:

1. The gravitational force on a body
2. Acceleration in terms of the body's velocity and orbit's radius



[4]

The only force acting on body 1 is F_G, so using Newton's Law of Universal Gravitation, it can be expressed as:

$$F_G = G\frac{m_1 m_2}{r^2}$$

where G is the Newtonian constant of gravitation, $6.67 \times 10^{-11} Nm^2kg^{-2}$, and r is the distance between the bodies.

Given that Newton's Second Law states that:

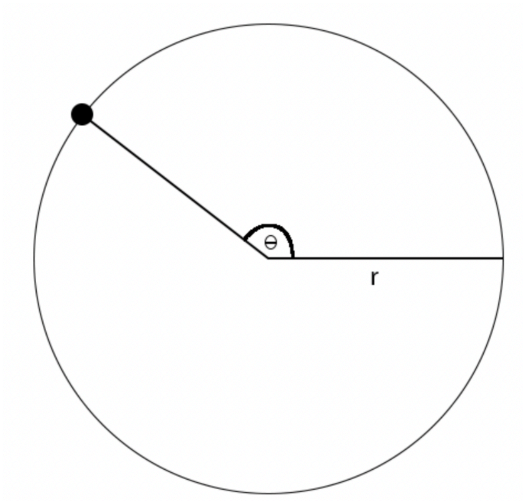
$$F_{net} = m\vec{a}$$

It follows that:

$$F_{net} = G\frac{m_1 m_2}{r^2}$$

Although it is assumed that the body's speed remains constant throughout its (circular) orbit, its direction changes. Therefore, its acceleration also changes as it is a vector quantity.

To derive the next part of the equation, we can use the unit circle to define the displacement of the point on the circle, $\vec{s}$, in terms of r.



In the unit circle, the x coordinate is $\cos\theta$ and the y coordinate is $\sin\theta$. Therefore, $\vec{s}$ can be written as:

$$\vec{s} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}$$

To derive an expression for the velocity, take the derivative of $\vec{s}$ with respect to time:

$$\begin{aligned}\vec{v} &= \frac{d\vec{s}}{dt} \\ &= \frac{d\vec{s}}{d\theta} \times \frac{d\theta}{dt} \\ &= \begin{bmatrix} -r\sin\theta \\ r\cos\theta \end{bmatrix} \times \frac{d\theta}{dt}\end{aligned}$$

As $\frac{d\theta}{dt}$ is a constant, it can be written as angular velocity, ω:

$$\vec{v} = \begin{bmatrix} -r\sin\theta \\ r\cos\theta \end{bmatrix} \times \omega$$

Calculating the magnitude of velocity:

$$\begin{aligned}|\vec{v}| &= \left|\begin{bmatrix} -r\sin\theta \\ r\cos\theta \end{bmatrix}\right| \times \omega \\ &= \sqrt{(-r\sin\theta)^2 + (r\cos\theta)^2} \times \omega \\ &= \sqrt{(\sin^2\theta + \cos^2\theta)r^2} \times \omega \\ &= \sqrt{r^2} \times \omega \\ &= r\omega\end{aligned}$$

Calculating the magnitude of acceleration:

$$\begin{aligned}
\vec{a} &= \frac{d\vec{v}}{dt} \\
&= \frac{d\vec{v}}{d\theta} \times \frac{d\theta}{dt} \\
&= \begin{bmatrix} -r\cos\theta \\ -r\sin\theta \end{bmatrix} \times \omega^2 \\
|\vec{a}| &= \sqrt{(-r\cos\theta)^2 + (-r\sin\theta)^2} \times \omega^2 \\
&= \sqrt{(\sin^2\theta + \cos^2\theta)r^2} \times \omega^2 \\
&= \sqrt{r^2} \times \omega^2 \\
&= r\omega^2
\end{aligned}$$

Substituting $|\vec{a}|$ into $|\vec{v}|^2$:

$$\begin{aligned}
|\vec{v}|^2 &= r^2\omega^2 \\
&= r(r\omega^2) \\
&= r\,|\vec{a}| \\
|\vec{a}| &= \frac{|\vec{v}|^2}{r}
\end{aligned}$$

The above equation can be rewritten with Newton's Second Law:

$$F_{net} = \frac{m_1 v_1^2}{r_1}$$

Note the use of r_1 instead of r. r_1 represents the orbital radius, while r represents the distance between the two bodies.
Given that F_{net} is equal to both the above equation and Newton's Law of Gravitation:

$$G\frac{m_1 m_2}{r^2} = \frac{m_1 v_1^2}{r_1}$$

Simplifying and rearranging for v_1 gives:

$$v_1 = \sqrt{\frac{G m_2 r_1}{r^2}}$$

3 Python Implementation

Using Python in Trinket, the derived equations can be used to simulate the motion of the interacting bodies.

The following constants need to be defined:

1. The Gravitational constant, G
2. The mass of each body
3. The distance between each body
4. The total mass of the system

Python Code:

```
G = 6.674e-11
mass1 = 10e30
mass2 = 5e30
distance_between = 1e10
total_mass = mass1 + mass2
```

Using the equations for vectors $\vec{r}_1$ and $\vec{r}_2$, the starting x-coordinates can be defined. Note that, for this model, the barycenter - $\vec{R}$ - has been defined as the origin. As a result, $y = 0$ and $\vec{R}$ can be neglected from the equation. Furthermore, $z = 0$ because this model is in the xy plane.

```
x1 = -(mass2 / total_mass) * distance_between
x2 = (mass1 / total_mass) * distance_between
```

The 'sphere()' object is initialised to represent each body:

```
body1 = sphere(pos=vector(x1,0,0), radius=1e9)
body2 = sphere(pos=vector(x2,0,0), radius=6e8)
```

To visually represent the barycenter, the same steps are taken as above:

```
barycenter = sphere(pos=vector(0,0,0), radius=1e8)
```

The following code calculates the circular velocity using the previously derived equation:

```
# vdb stands for vector distance between
vdb = body2.pos - body1.pos
circular_velocity = sqrt(G * mass2 * mag(body1.pos) / mag(vdb) ** 2)
```

The velocity and momentum are assigned to the first body:

```
body1.velocity = vector(0,circular_velocity,0)
body1.momentum = mass1 * body1.velocity
```

Using the principle of conservation of momentum, we can say that body2 has the negative amount of body1's momentum:

```
body2.momentum = -body1.momentum
```

The following code creates the time loop required to simulate the motion of the objects:

```
t = 0
dt = 1000
while t < 2e7:
    rate(100) # rate(x) tells Python to perform x calculations per second
```

For each time increment (i.e. iteration in the while loop), $\vec{r}$ is calculated:

```
    vdb = body2.pos - body1.pos
```

The gravitational force is calculated:

```
    # It is negative as the force acts against the vector r
    # norm() -> unit vector
    # It is a unit vector as it is directional
    gravitational_force = -G * mass1 * mass2 * norm(vdb) / mag(vdb) ** 2
```

The momentum is calculated using that fact that $p = \int F_G dt$[6]:

```
    body1.momentum -= gravitational_force * dt
    body2.momentum += gravitational_force * dt
```

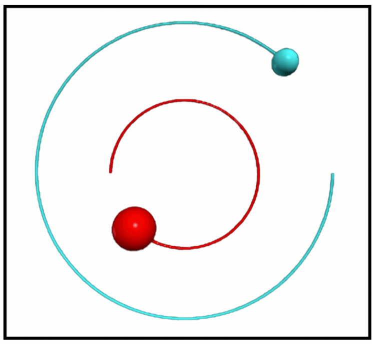
The positions are calculated:

```
    # Momentum = mass x velocity
    # Momentum / mass = velocity
    # Taking the numerical integral of velocity will give the position
    body1.pos += (body1.momentum / mass1) * dt
    body2.pos += (body2.momentum / mass2) * dt
```

The time is incremented:

```
    t += dt
```

That is all the code required to model the Two-Body Problem in Python. Here is a screenshot of the simulation in action:



4 The N-Body Problem

The N-Body Problem, an extension of the well-understood Two-Body Problem, is a more complex concern for physicists as it deals with N bodies orbiting each other. When N is greater than 2, the system becomes impossible to predict with a growing number of interactions and equations. Despite this, numerical methods can be used to approximate the solution for specific initial conditions in closed form. The N-Body Problem plays a crucial role in fields such as celestial mechanics, astrophysics, and planetary science. The Two-Body Problem, however, provides an understanding of what the N-Body Problem entails.

5 Conclusion

This project aimed to present a clear and concise explanation of how to model the Two-Body Problem using Python by outlining the necessary steps and implementing them in code. The accompanying program serves as a practical example of the concepts presented in this project and can be viewed by the reader - https://trinket.io/library/trinkets/566ee77167
Overall, the simulation successfully shows what was expected to occur. Using the derived equations, the two bodies move in clear circular orbits with a constant speed and rotate about the barycenter.

6 References

1. Wikipedia - Two-Body Problem: https://en.wikipedia.org//wiki/Two-body_problem#Displacement_vector_motion_(2nd_one-body_problem) [22/01/23]

2. Medium - Two-Body Problem: https://medium.com/illumination/astrodynamics-two-body-problem-ec2c5e148184 [22/01/23]

3. Wikipedia - Barycenter: https://en.wikipedia.org/wiki/Barycenter [22/01/23]

4. Physics Explained - Modelling Stable Circular Orbits: https://www.youtube.com/watch?v=Ukkr0v7ks54

5. Physics Explained - Central Forces and the 2 Body Problem: https://www.youtube.com/watch?v=d05oyVJpv24