INFORME TAREA 3

Antonia Labarca Sánchez
RUT:20.072.551-4
Github: @antolabarca

1. Introducción

El problema a resolver consistió en modelar la órbita de un planeta cercano al sol, como Mercurio, y calcular su energía en cada momento.

Para estos planetas, el potencial gravitacional es el siguiente:

$$U(r) = -\frac{GM_\odot m}{r} + \alpha \frac{GM_\odot m}{r^2}$$

donde G es la constante de gravitación universal, $M_\odot$ es la masa del Sol, m es la masa del planeta, r es la distancia entre el planeta y el Sol, y α es un número pequeño.

Para resolver el problema, se utilizaron distintos métodos numéricos para resolver ecuaciones diferenciales: Runge Kutta de orden 4, Velocity Verlet y Beeman.

Además, se simplificó el problema utilizando unidades de medida tales que $GM_\odot m = 1$, ya que lo interesante es observar la forma de la órbita y si la energía se conserva, no sus magnitudes exactas.

Para resolver el problema, se utilizó OOP (programación orientada a objeto), y se creó una clase Planeta con métodos para cada método numérico.

2. Desarrollo

Como ya se dijo, se creó la clase Planeta. Al crear un objeto Planeta, se crean los siguientes atributos:

- `y_actual`, un `numpy array` que representa la condición actual del planeta, inicialmente es la condición inicial
- `t_actual`, un `float` que representa el tiempo, inicialmente es 0.
- `alpha`, un `float` que representa el parámetro α. Si no se entrega uno al método que crea el planeta, por default es 0.

Además, en el caso de usarse el método de Beeman, se crea un nuevo atributo, `a_previo`, que representa la aceleración en el instante previo del planeta. Esto es necesario, ya que este método requiere de 2 tiempos anteriores para calcular la condición actual.

La ecuación vectorial de movimiento de un planeta cercano al sol es la siguiente:

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ x(-r^{-3} + 2\alpha r^{-4}) \\ y(-r^{-3} + 2\alpha r^{-4}) \end{pmatrix}$$

con $r = \sqrt{x^2 + y^2}$, es decir, el radio r en el que está el planeta.

La clase Planeta cuenta con los siguientes métodos:

- `ecuacion_de_movimiento(self)`, que recibe un arreglo `[x, y, vx, vy]` y retorna el vector de sus derivadas de acuerdo a la ecuación vectorial anterior.
- `avanza_rk4(self, dt)`, que realiza un "paso" con el algoritmo de Runge-Kutta de orden 4, avanzando su posición y velocidad en un intervalo de tamaño `dt`. Este método actualiza el valor de `y_actual`.
- `avanza_verlet(self, dt)`, que igual que el método anterior realiza un "paso" de tamaño `dt`, solo que en este caso utilizando el algoritmo de Velocity Verlet.
- `avanza_beeman(self, dt)`, que al igual que los métodos anteriores realiza un "paso" de tamaño `dt`, solo que utilizando el algoritmo de Beeman. Como ya se dijo, este método requiere agregar un atributo adicional a la clase Planeta, para en cada instante de tiempo tener el valor actual y el valor anterior de `[x, y, vx, vy]`.
- `energia_total(self)`, que calcula la energía total del sistema, sumando las energías potencial y cinética en un instante dado.

Con los métodos anteriores, en primer lugar se consideró $\alpha = 0$. Se crearon 3 planetas Mercurio, uno por cada método numérico implementado, con las mismas condiciones iniciales en cada caso: `[10, 0, 0, 0.2]`. Estas condiciones iniciales fueron escogidas para que la energía total inicial fuese negativa (con esas condiciones, tiene un valor de aproximadamente -0.08).

Se realizaron aproximadamente 5 "vueltas" del planeta en cada caso. Para esto se usó un contador de "media vuelta", que aumentaba cada vez que se pasaba de negativo a positivo o de positivo a negativo en la coordenada y. Luego, cuando este contador llega a 10, se han dado 5 vueltas, ya que se cruzó el eje x 10 veces.

Luego, se graficaron las órbitas de cada planeta y una comparación de las energías en función del tiempo de cada uno usando `pyplot`.

Para el caso de $\alpha > 0$, se consideró un valor pequeño: $\alpha = 10^{-2.551}$. Se integró durante 30 "vueltas", contadas de la misma forma que antes, y se calculó el radio r en cada instante, almacenando esta información en una matriz, en la que cada columna corresponde a una "vuelta". Luego, se determinó la posición del afelio en cada uan buscando el máximo radio de cada columna. Teniendo esto, se puede calcular la velocidad angular de precesión simplemente viendo cómo cambia el afelio en cada vuelta. Para esto, se escogieron 2 elementos seguidos cualquiera (en este caso, el 0 y el 1), ya que la velocidad angular del afelio se puede suponer constante, y se obtuvo el ángulo en el que estaba cada uno. Luego, se aproximó la velocidad angular como la diferencia entre estos 2 dividida por la diferencia de tiempo.

3. Resultados

En primer lugar, para $\alpha = 0$ la energía total obtenida en cada caso fue la siguiente:

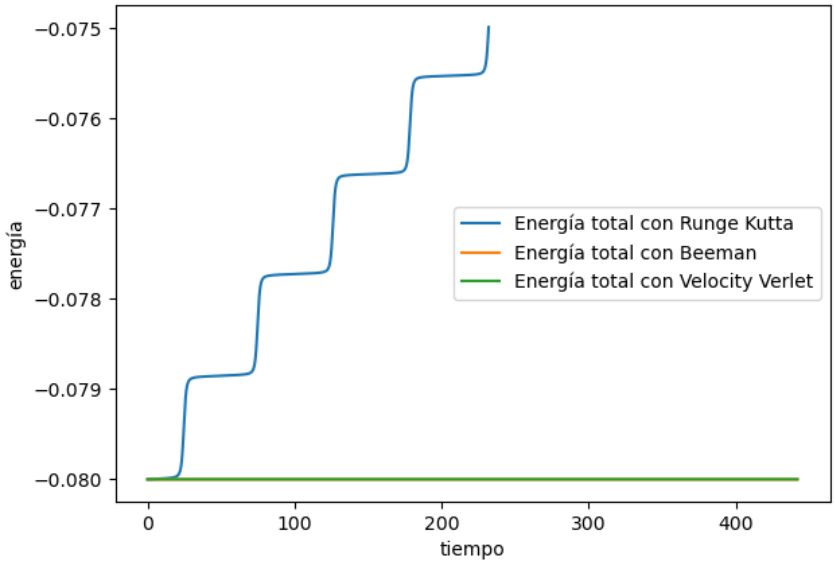

Figura 1: Energía calculada para cada método numérico, considerando un planeta con condiciones iniciales $(10, 0, 0, 0.2)$ y parámetro $\alpha = 0$.

Podemos notar que para el caso de Beeman y Velocity Verlet, este valor es constante (o casi constante), mientras que en el caso de Runge Kutta, si bien la diferencia es bastante pequeña en magnitud, se observa que va aumentando en el tiempo. Esto hace sentido si se considera que la ventaja de los métodos de Beeman y Velocity Verlet es precisamente conservar las cantidades conservativas, tales como la energía.

Para cada método numérico, las órbitas fueron las observadas en las figuras 2, 3 y 4.

A simple vista, la órbita usando Runge-Kutta cambia de forma en cada "vuelta", mientras que las otras 2 no. Esto se relaciona con que al usar Runge-Kutta la energía va aumentando, por lo que tiene sentido que la órbita se "abra", mientras que como Verlet y Beeman aseguran que la energía se conserva, para estos algoritmos tiene sentido que la forma de la órbita también lo haga.

Para el caso de $\alpha = 10^{-2.551}$, se obtuvo la energía graficada en la figura 5. Esta energía no es exactamente constante, pero en magnitud cambia muy poco. Esto puede ser debido a errores de cálculo, o al efecto de otros planetas (lo que se refleja en el parámetro α). En el segundo caso, no se rompe la conservación de la energía, ya que los otros planetas también ejercen fuerza de gravedad, por lo que también habría una energía potencial gravitatoria asociada.

La órbita del planeta corresponde a la figura 6.

Esta órbita no es exactamente constante, debido al efecto de $\alpha > 0$. Se puede observar que el afelio cambia de cada posición en cada vuelta.

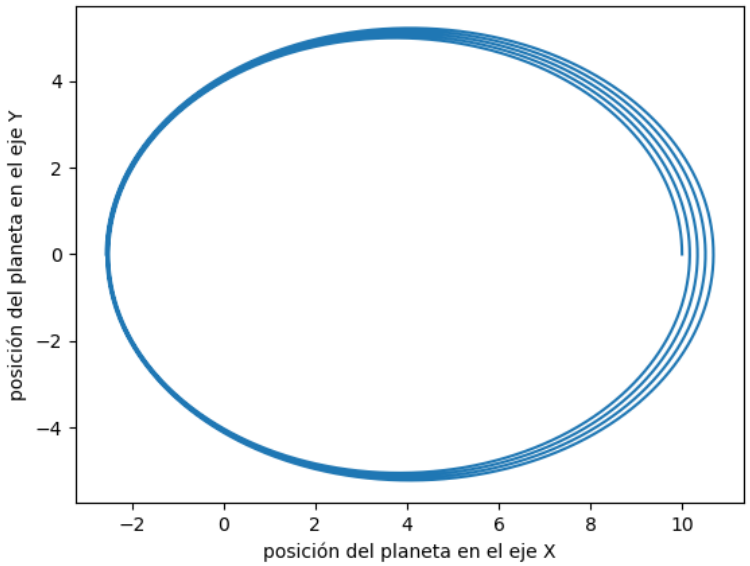

Figura 2: Órbita simulada usando Runge-Kutta de orden 4.

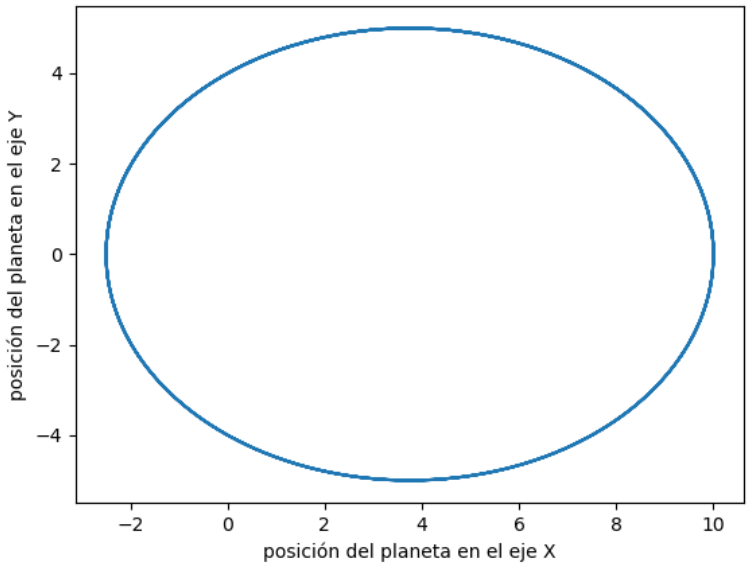

Figura 3: Órbita simulada usando Velocity Verlet.

La velocidad angular (en radianes) de precesión obtenida fue de $5.1115 \cdot 10^{-5}$.

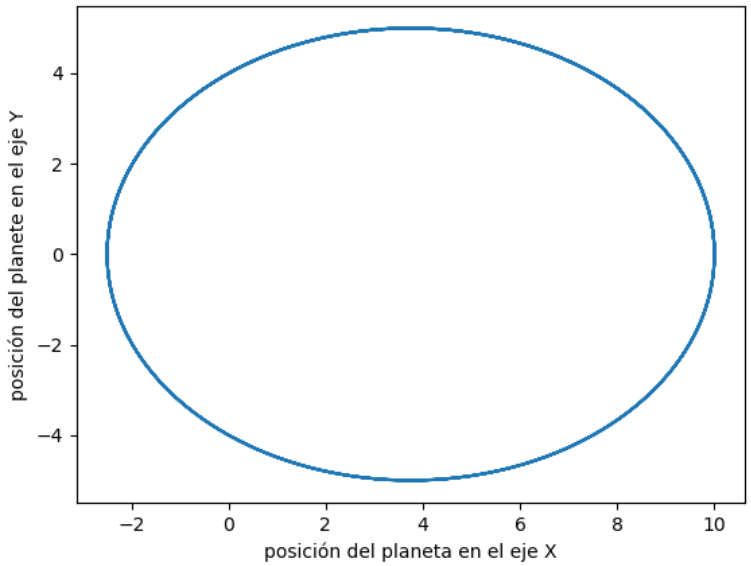

Figura 4: Órbita simulada usando Beeman.

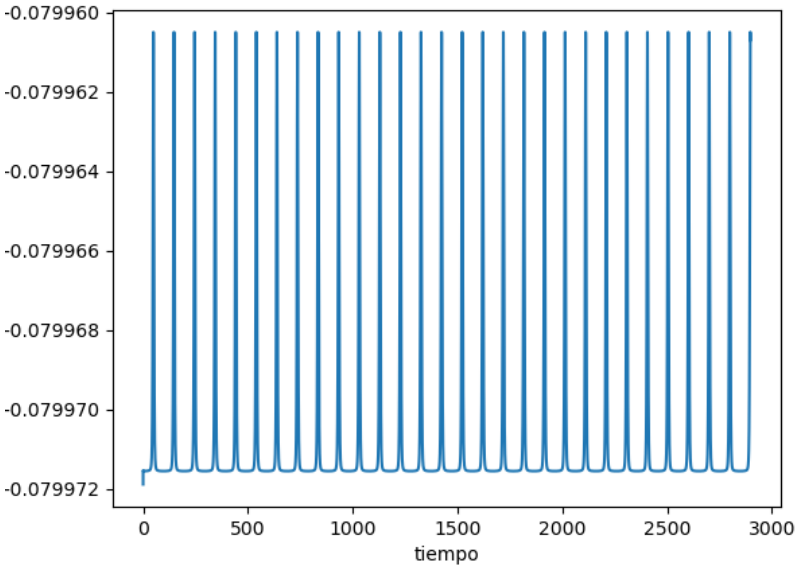

Figura 5: Energía total considerando un planeta con parámetro $\alpha = 10^{-2.551}$.

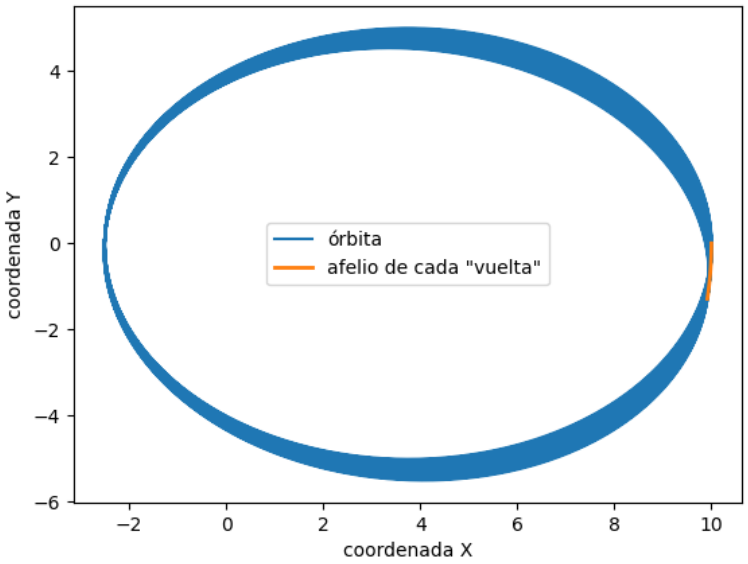

Figura 6: Órbita del planeta con parámetro $\alpha = 10^{-2.551}$, durante 30 vueltas.

4. Discusión y Conclusiones

Una conclusión importante es que si se quiere trabajar con cantidades conservativas, puede ser útil un método numérico que las conserve, especialmente si se desea ocupar esta información para otras cosas. En este problema, con Runge-Kutta la energía aumentó, lo que en la realidad no puede ocurrir, por lo que podría llevar a errores importantes en los resultados. Además la forma de la órbita cambió bastante.

Por otro lado, es interesante ver que a pesar de ser un número muy pequeño, α afecta bastante la posición del afelio, es decir, la forma de la órbita. De esto se puede concluir que la aproximación elíptica perfecta es buena para un número pequeño de "vueltas", pero no para un número mayor.