UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

DEPARTAMENTO DE FÍSICA

MÉTODOS NUMÉRICOS PARA LA CIENCIA E INGENIERÍA

TAREA 4

"Ley de Gravitación Universal y órbitas relativistas"

Braulio Sánchez Ibáñez

16.880.977-8

Profesor: Valentino González

Auxiliar: Felipe Pesce

Santiago, Chile

PREGUNTA 1

1. Describa la idea de escribir el main driver primero y llenar los huecos después. ¿Por qué es buena idea?

Respuesta: El objetivo de escribir primero el main driver es establecer una estructura central jerarquizada clara, que permita ordenar la forma en que el código abordará el problema, definiendo primero las funciones y clases que se utilizarán. De esta forma, el código se ve más claro y ordenado, permitiendo posteriormente completar los segmentos de códigos restantes para tareas específicas. La ventaja de hacer esto es que facilita la detección de posibles errores y además permite optimizar la manera en que se implementa el código.

2. ¿Cuál es la idea detrás de la función `mark_filled`? ¿Por qué es buena idea crearla en vez del código al que reemplaza?

Respuesta: Esta función permite detectar y reparar con mayor rapidez y facilidad errores asociados a índices inexistentes o fuera de rango en una lista o arreglo. La ventaja de esto es que se puede desplegar un mensaje más específico e indicativo del error cometido, permitiendo identificar el lugar donde se cometió y repararlo rápidamente, sin necesidad de revisar el código completo.

3. ¿Qué es refactoring?

Respuesta: Refactoring es, en líneas generales, optimización del código sin alterar su funcionalidad. Es decir, se modifica el código, pero no lo que hace o el resultado que

entrega. El objetivo de esto es “limpiar” el código, es eliminar código muerto, definiciones redundantes, etc, y dar una estructura y diseños más claros, de modo de facilitar el testeo del programa y la detección de errores, además de simplificar la implementación de futuras posibles modificaciones.

4. ¿Por qué es importante implementar tests que sean sencillos de escribir? ¿Cuál es la estrategia usada en el tutorial?

Respuesta: Es importante ir testeando el programa y verificar si entrega los resultados esperados o no. Por ello, es necesario crear tests sencillos y fáciles de entender, de modo de poder detectar cualquier error o falla en el código de manera rápida. La estrategia utilizada en el tutorial es la de usar “fixtures”, tests muy fáciles de leer y escribir y que permiten ingresar “strings” para analizar el funcionamiento del código.

5. El tutorial habla de dos grandes ideas para optimizar programas, ¿cuáles son esas ideas? Descríbalas.

Respuesta: La primera es Asymptotic Analysis, y consiste en reducir al máximo el número de datos del problema, de manera de poder reducir el tiempo de procesamiento y cálculo. Para esto, se utilizan resultados ya calculados por el programa para cálculos posteriores, actuando de forma recursiva.

La segunda idea es Bynary Search, cuya filosofía es reducir la cantidad de datos a la mitad y disminuir así el número de iteraciones y, por lo tanto, el tiempo de cálculo. Para esto, el programa selecciona, mediante algún criterio, sólo una mitad de los datos y los procesa, obteniendo un resultado que le servirá para decidir cuál siguiente mitad

deberá escoger. De esta forma, el número de datos y el tiempo de procesamiento se reducen exponencialmente.

6. ¿Qué es `lazy evaluation`?

Respuesta: Corresponde al proceso de evaluar y calcular datos sólo cuando sea estrictamente necesario, y no al inicio del programa, por ejemplo. Con esto se evita crear información innecesaria y se ahorra espacio de memoria y tiempo de cálculo.

7. Describir la other moral del tutorial.

Respuesta: Si se quiere escribir un código que sea rápido, entonces debe ser simple. El mejor camino es escribirlo, correrlo y verificar que funciona correctamente. Una vez superada esta etapa, se puede hacer refactoring del programa para optimizar el tiempo de cálculo, refinando el código, eliminado variables redundantes o utilizando cualquier otra técnica orientada a simplificar el programa, pero sin alterar su funcionamiento.

PREGUNTA 2

1) El potencial gravitatorio para planetas que orbitan cerca del Sol, como Mercurio, es

$$U(r) = -\frac{GMm}{r} + \alpha\frac{GMm}{r^2}$$

A través de la transformación de coordenadas polares es posible obtener el potencial en coordenadas cartesianas:

$$U(x,y) = -\frac{GMm}{\sqrt{x^2+y^2}} + \alpha\frac{GMm}{x^2+y^2}$$

Las ecuaciones de movimiento, derivadas de la segunda ley de Newton o de las ecuaciones de Euler-Lagrange, son

$$\ddot{x} = -GMx\left(\frac{1}{(x^2+y^2)^{\frac{3}{2}}} - \frac{2\alpha}{(x^2+y^2)^2}\right)$$
$$\ddot{y} = -GMy\left(\frac{1}{(x^2+y^2)^{\frac{3}{2}}} - \frac{2\alpha}{(x^2+y^2)^2}\right)$$

Este sistema de segundo orden puede escribirse como un sistema de 4 ecuaciones diferenciales de primer orden:

$$\dot{x} = v_x$$
$$\dot{y} = v_y$$
$$\dot{v_x} = -GMx\left(\frac{1}{(x^2+y^2)^{\frac{3}{2}}} - \frac{2\alpha}{(x^2+y^2)^2}\right) = f_x(x,y)$$
$$\dot{v_y} = -GMy\left(\frac{1}{(x^2+y^2)^{\frac{3}{2}}} - \frac{2\alpha}{(x^2+y^2)^2}\right) = f_y(x,y)$$

Este sistema será resuelto mediante 3 algoritmos distintos.

a) Método de Euler explícito: dado un sistema de la forma $\dot{\vec{X}} = f(\vec{X})$, la solución está dada por la recurrencia

$$\vec{X}_{n+1} = \vec{X}_n + hf(\vec{X}_n)$$

Donde h es el paso de la recurrencia y $\vec{X} = (x, y, v_x, v_y)$. La función está definida por $f(x, y, v_x, v_y) = (v_x, v_y, f_x, f_y)$

b) Método de Runge-Kutta: en este caso la recurrencia que permite determinar la solución es

$$\vec{X}_{n+1} = \vec{X}_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Donde los vectores k se calculan de la forma

$$k_1 = f(\vec{X}_n)$$
$$k_2 = f\left(\vec{X}_n + \frac{k_1}{2}\right)$$
$$k_3 = f\left(\vec{X}_n + \frac{k_2}{2}\right)$$
$$k_4 = f\left(\vec{X}_n + k_3\right)$$

c) Método de Verlet: las soluciones x, y vienen dadas por las siguientes dos recurrencias

$$x_{n+1} = x_n + hv_{x,n} + \frac{h^2}{2}f_x(x_n, y_n)$$
$$y_{n+1} = y_n + hv_{y,n} + \frac{h^2}{2}f_y(x_n, y_n)$$

Con estos resultados se calculan las velocidades:

$$v_{x,n+1} = v_{x,n} + \frac{h}{2}(f_x(x_n, y_n) + f_x(x_{n+1}, y_{n+1}))$$

$$v_{y,n+1} = v_{y,n} + \frac{h}{2}(f_y(x_n, y_n) + f_y(x_{n+1}, y_{n+1}))$$

Para implementar estos métodos se define la clase Planeta, cuyas variables de entrada son la condición inicial del movimiento y la constante α, que por defecto tiene valor nulo. A partir de esto se genera un objeto, que en este caso fue denominado Tierra. El comando **`Tierra.y_actual`** devuelve el vector $\vec{X}_n$ para el estado actual del objeto Tierra. El estado de partida es la condición inicial ingresada.

Dentro de la clase Planeta se definen los siguientes módulos:

- `Ecuación_de_movimiento:` **`Tierra.ecuacion_de_movimiento`** devuelve el vector $f(\vec{X}_n) = (v_{x,n}, v_{y,n}, f_x(x_n, y_n), f_y(x_n, y_n))$ para poder ser utilizado en los métodos antes descritos.
- `Avanza_euler:` **`Tierra.avanza_euler(dt)`** implementa el método de Euler explícito para resolver el sistema de ecuaciones con un paso h=dt. No devuelve nada, sólo actualiza el estado del objeto Tierra en un paso.

  El objeto comienza en el estado asociado a n=0, que no es más que la condición inicial. Si se aplica el módulo una vez, se obtiene el estado n=1, y ahora **`Tierra.y_actual`** devuelve el vector $\vec{X}_1 = (x_1, y_1, v_{x,1}, v_{y,1})$

  De esta forma, para obtener la solución completa, se debe implementar una recursión **`for`** que vaya completando los vectores soluciones con cada estado actual del objeto Tierra.
- `Avanza_rk4:` Análogo al anterior, pero ahora se resuelve el sistema con el método de Runge-Kutta de orden 4.
- `Avanza_verlet:` Actualiza el estado del objeto Tierra, resolviendo el sistema mediante el método de Verlet.

2) Se resuelve el sistema de ecuaciones diferenciales usando los 3 métodos antes expuestos, para el caso $\alpha = 0$. Los valores de las constantes son $m = 1[kg]$, $G = 6.67 \times 10^{-11}[Nm^2kg^{-2}]$ y $M = \frac{1}{6.67} \times 10^{11}[kg]$, de modo que $GMm = 1$.

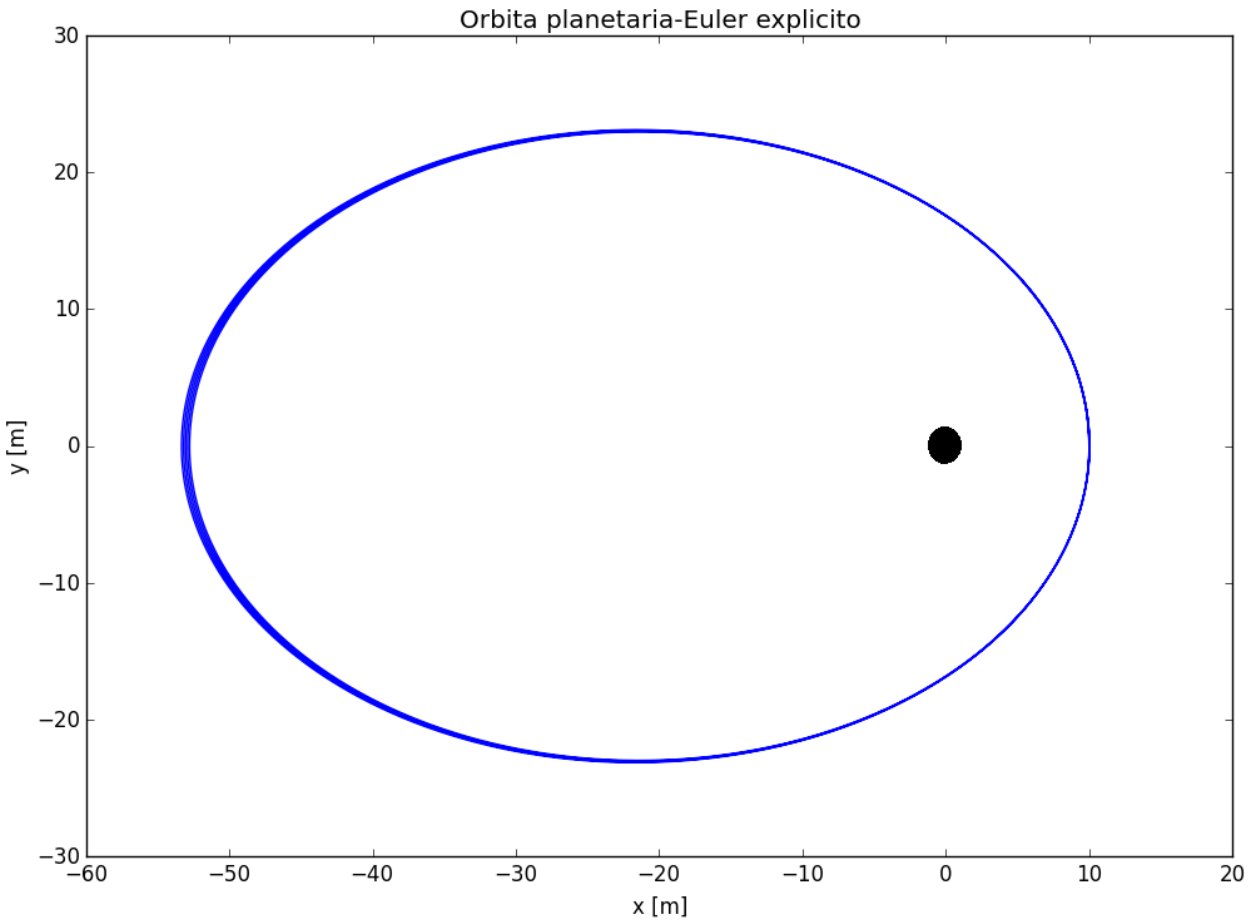
La condición inicial es $(x_0, y_0, v_{x,0}, v_{y,0}) = (10,\ 0,\ 0,\ 0.41)$.

Con esta información es posible calcular el período de la órbita mediante la tercera ley de Kepler $T^2 = \frac{4\pi^2 a^3}{GM}$.

El semieje mayor de la elipse viene dado por $a = -\frac{GMm}{2E}$. Se calcula la energía con las condiciones iniciales y se obtiene $T = 1102.793\ [s]$.

Luego, para obtener aproximadamente 5 órbitas completas, se integrará entre t=0 y t=5600 segundos, con $N = 10^6$, por lo que el paso es $h = dt = 0.0056\ [s]$. Los siguientes gráficos muestran estos resultados:

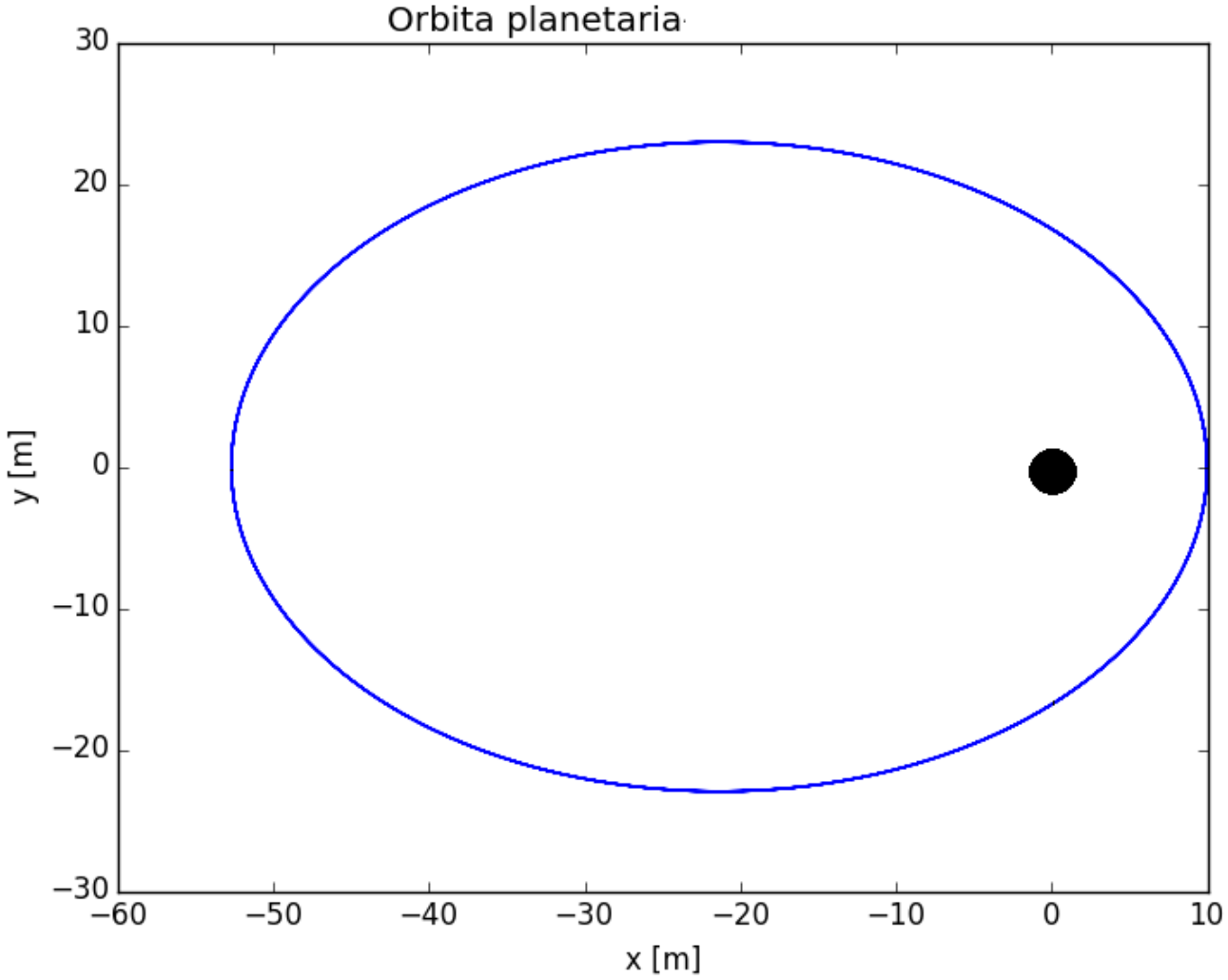
Figura 1: Órbita planetaria mediante el método de Euler explícito



Se ha marcado con un círculo negro el foco en torno al cual orbita la partícula, que corresponde a la posición del Sol o del cuerpo celeste que esté produciendo este movimiento orbital.

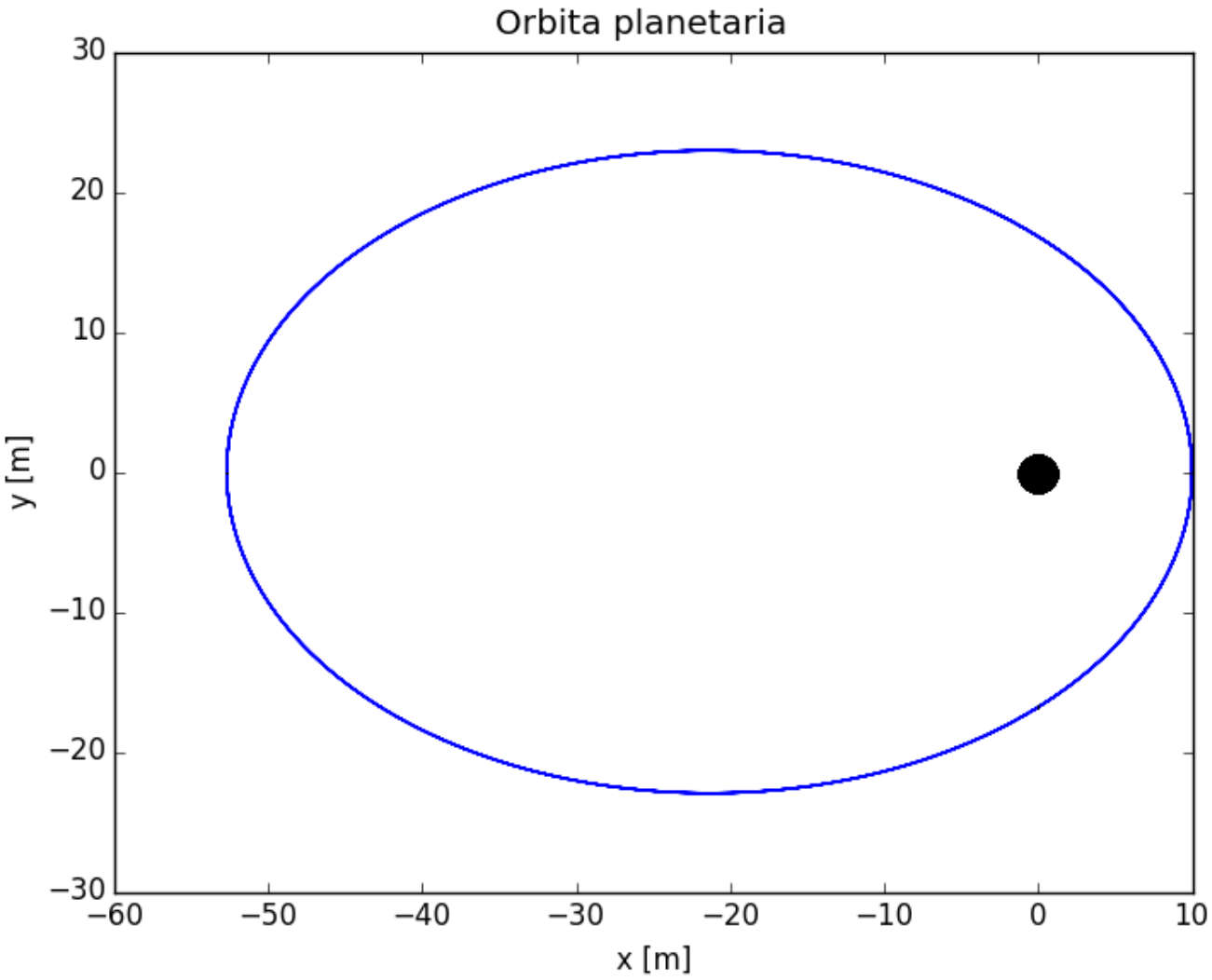
Se puede observar que el método va perdiendo estabilidad ligeramente a medida que avanza el tiempo, lo que se traduce en que el vértice izquierdo de la elipse se traslada hacia la izquierda. La única forma de reducir este efecto es considerando un paso más pequeño.

Figura 2: Órbita planetaria mediante el método RK4



En el caso de RK4 los problemas de estabilidad son menores, ya que no se aprecia, a simple vista, alguna divergencia en la solución y las 5 órbitas parecen corresponder a la misma elipse.

Figura 3: órbita planetaria mediante el método de Verlet



Al igual que con RK4, el método es estable y converge siempre a la misma elipse.

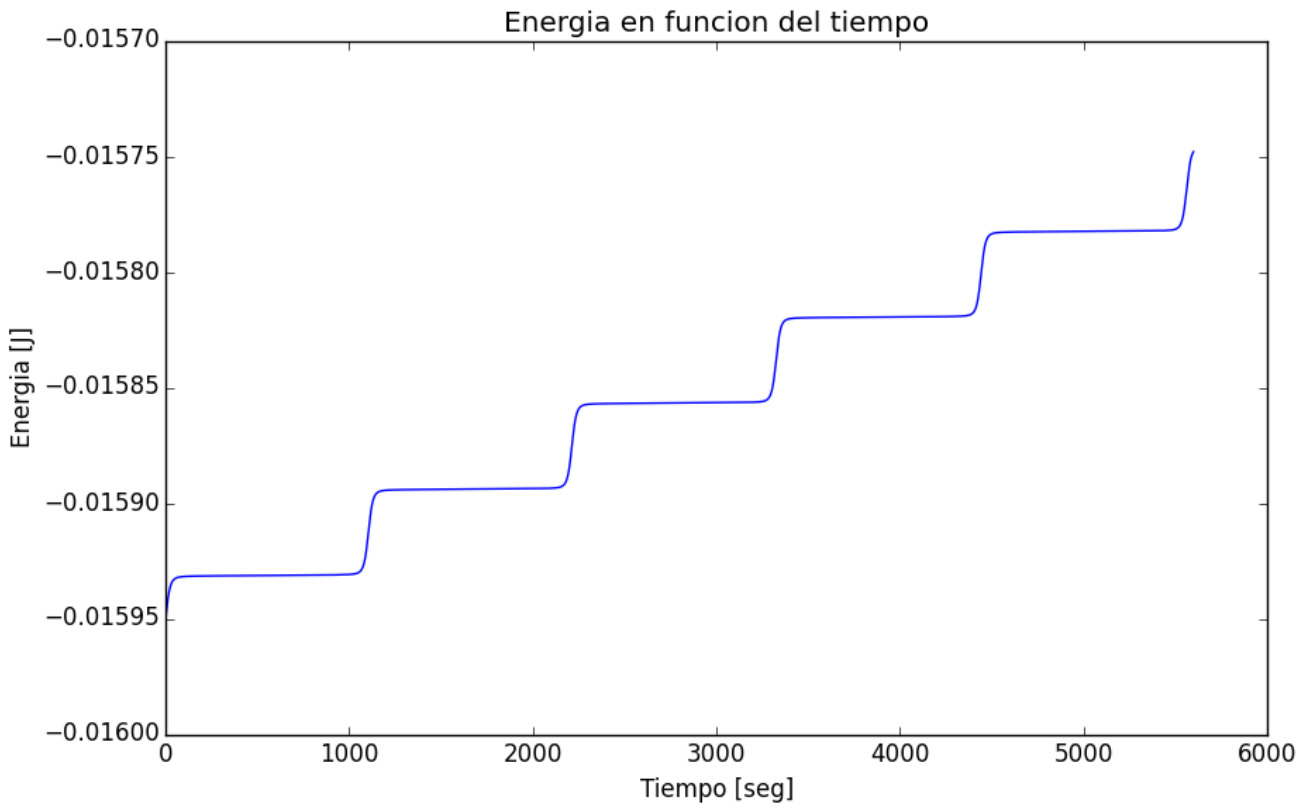
Para calcular la energía en función del tiempo, se usa la expresión

$$E_n = \frac{m}{2}(v_{x,n}^2 + v_{y,n}^2) - GMm\left(\frac{1}{\sqrt{x_n^2 + y_n^2}} - \frac{\alpha}{x_n^2 + y_n^2}\right)$$

Dentro de la clase Planeta se implementa el módulo **`energia_total,`** cuya única función es tomar el estado actual **`Tierra.y_actual`** y calcular la energía asociada a ese estado.

Los siguientes gráficos muestran la energía del sistema en función del tiempo para cada método descrito

Figura 4: Energía mediante el método de Euler



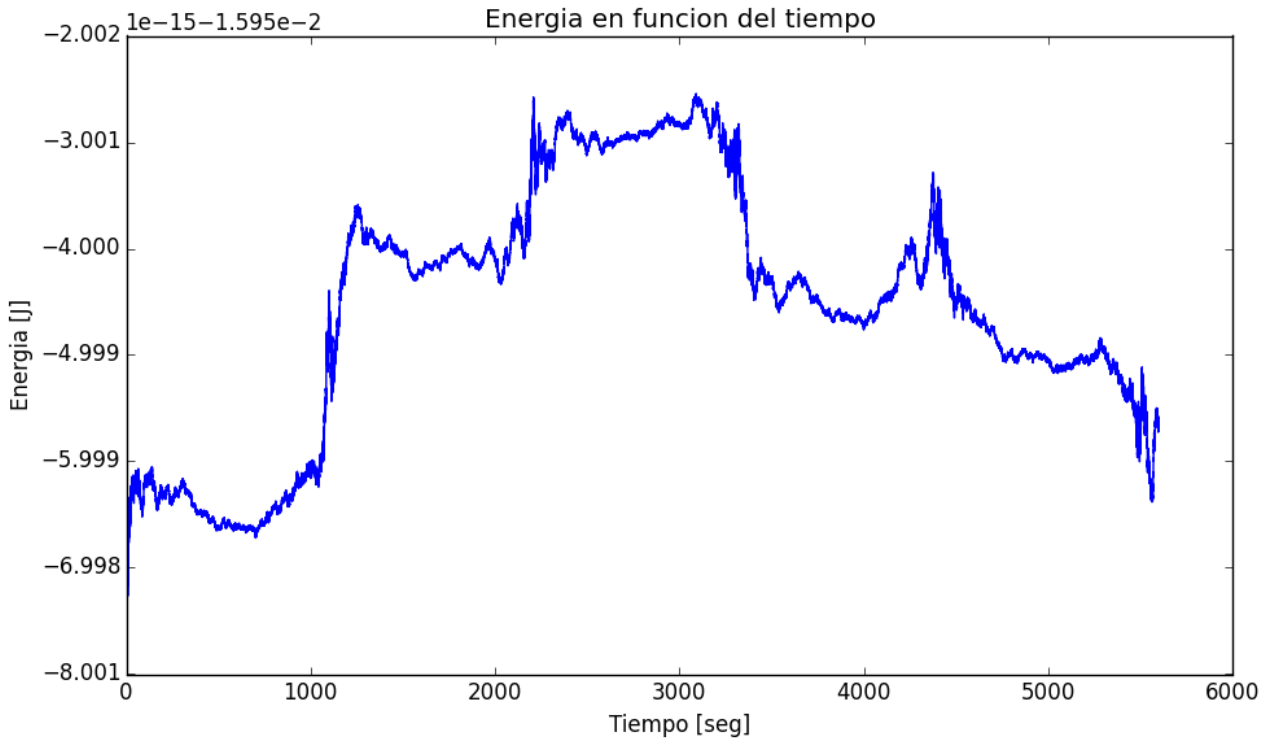
La teoría dice que la energía de este sistema debe conservarse y ser constante en el tiempo, ya que el potencial es conservativo y no existen fuerzas disipativas como el roce. Sin embargo, ya se vio que el método de Euler presenta una marcada propagación de errores, lo que provoca que el extremo izquierdo de la órbita se desplace y, por lo tanto, la energía varíe.

En consecuencia, la variación temporal de la energía se debe exclusivamente a errores propios del algoritmo que se van propagando con cada paso.

Es interesante notar que la curva es prácticamente constante en determinados intervalos y presenta un rápido crecimiento en pequeños instantes situados en torno a múltiplos del período $T = 1102.793\ [s]$. Es decir, durante casi todo el trayecto de la partícula los errores son despreciables y no contribuyen a modificar significativamente la energía, excepto en torno al perihelio. La zona cercana al perihelio es donde se generan los mayores errores, que se traducen en una brusca variación de la energía.

La energía inicial es $E_0 = -0.01595\ [J]$. En 5 revoluciones el aumento de energía fue, aproximadamente, 0.0002 Joules, es decir, 0.00004 Joules por ciclo, lo que equivale a un error de $e = 2.508 \times 10^{-3}\ [J/ciclo]$. Es decir, en cada ciclo, el sistema aumenta su energía en un 0.251% con respecto al valor inicial.

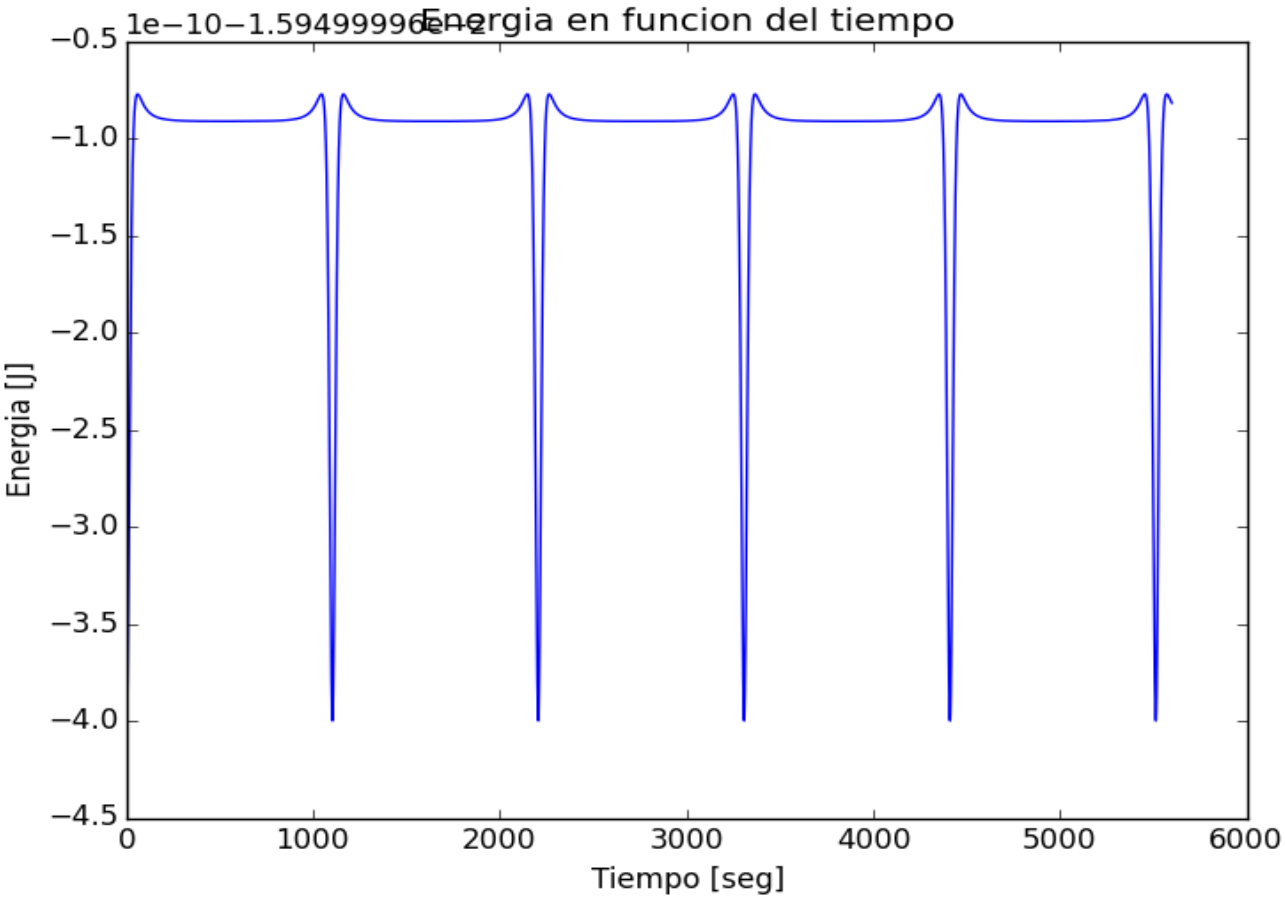
Figura 5: energía mediante el método RK4



Lo primero claro a partir del gráfico es que el error no presenta un patrón definido, a diferencia del método de Euler. Sin embargo, el error asociado es mucho menor, ya que las desviaciones de la energía con respecto al valor inicial son del orden de 10^{-15}, como se ve en el eje OY del gráfico.

La mayor desviación es, aproximadamente, $\Delta E = -7 \times 10^{-15}$, lo que se traduce en un error de $e = \frac{\Delta E}{E_0} = 4.389 \times 10^{-13}$. Este error es muy pequeño, por lo que la energía se mantiene prácticamente constante durante todo el movimiento.

Figura 6: Energía mediante el método de Verlet

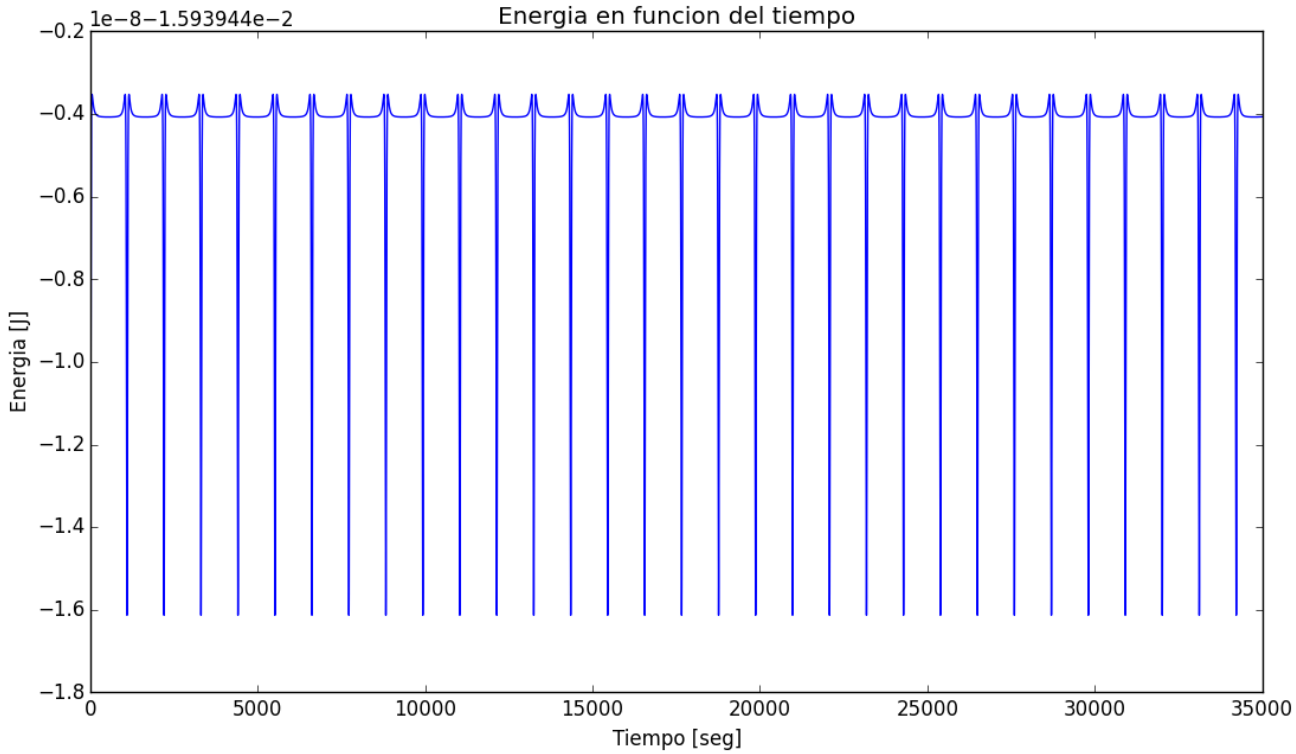


Al igual que con Euler explícito, la energía se mantiene constante en intervalos entre múltiplos del período T. En torno a múltiplos de T, la energía toma valores cercanos a E_0, pero variando bruscamente, y toma exactamente este valor en instantes correspondientes a múltiplos de T.

La diferencia es que, a diferencia de Euler, este error no se propaga, sino que oscila entre 0 y un valor máximo dado por $e = \frac{\Delta E}{E_0} = 2.038 \times 10^{-8}$, que sigue siendo muy pequeño, por lo que la energía se mantiene, para efectos prácticos, constante en el tiempo.

3) Ahora se considera el caso $\alpha = 10^{-2.977}$ y se calculan al menos 30 órbitas, integrando entre t=0 y t=35.000 segundos, por lo que $h = dt = 0.035\ [s]$. El siguiente gráfico muestra la energía de la órbita relativista:

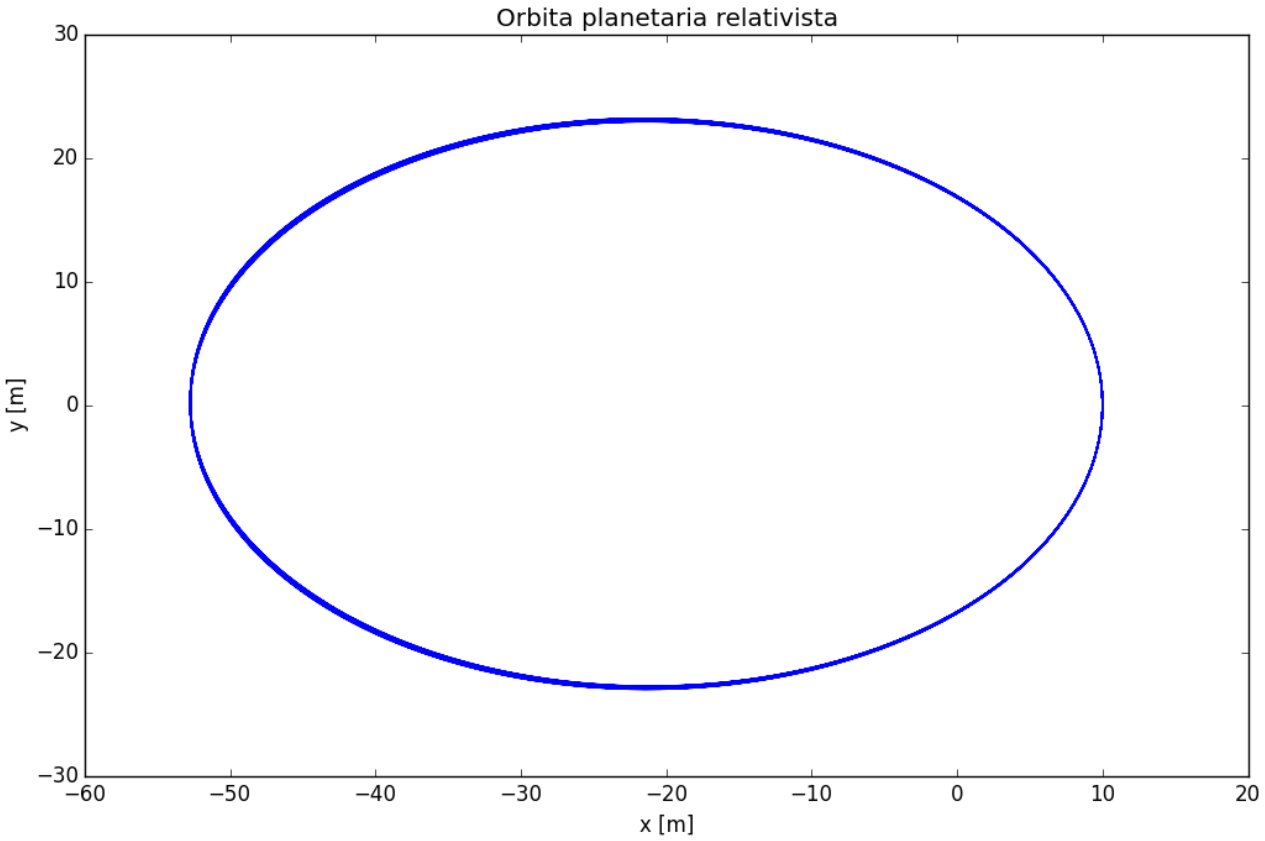
Figura 7: Energía de la órbita relativista



Como se utilizó el mismo método de integración (Verlet), la curva del error tiene la misma forma. En este caso ΔE es del orden de 10^{-8}, lo que es muy pequeño, pero dos órdenes de magnitud más grande que el caso no relativista. Esto se debe exclusivamente a que ahora se está utilizando un paso $dt = 0.035[s]$, a diferencia del caso anterior donde se usó $dt = 0.0056[s]$.

La decisión de integrar con un paso menos fino se debe a que ahora el dominio llega hasta 35000 segundos, y para mantener el mismo paso se debe utilizar $N = 6.25 \times 10^6$, lo que ralentizaría excesivamente el cálculo, tomando un tiempo demasiado largo.

Figura 8: órbita relativista.



En este caso, que considera 30 revoluciones, no se alcanza a apreciar detalladamente el efecto relativista, debido a que $\alpha = 0.00105$ es muy pequeño. Sin embargo, un leve engrosamiento de la elipse es distinguible hacia el lado izquierdo del gráfico, evidencia de la rotación de los ejes de simetría.

Para calcular la velocidad de precesión del eje de la elipse, se define un nuevo vector **r2**, que almacena los valores del cuadrado de la distancia del cuerpo al foco de la elipse, es decir, $r2 = x^2 + y^2$. Cuando el cuerpo se encuentra en el perihelio orbital, r2 tiene un mínimo; y cuando está en el afelio, un máximo. El objetivo es determinar la posición en donde se produce el primer mínimo del vector r2 (sin considerar el valor inicial en t=0). Para ello, y sabiendo que $T = 1102.793[s]$, se integrará sólo entre t=0 y t=1110 segundos, con $N = 2.5 \times 10^6$, por lo que $dt =$

$0.000444[s]$. La posición aproximada del período T dentro de la lista tiempo es $i = \frac{T}{dt} \approx 2.483.000$.

Por lo tanto, para determinar la posición del mínimo (perihelio), se usará el módulo **`numpy.argmin`** sobre el vector r2 truncado desde la posición i=2.400.000 hasta el final, **`numpy.argmin(r2[2400000:]).`** De esta manera, el programa se ahorra una gran cantidad de cálculo innecesario, ya que se le está orientando a buscar el mínimo en un intervalo bien restringido donde se sabe que él existe. Este tiempo ahorrado permite, por lo tanto, utilizar un paso más fino y obtener el resultado con mayor precisión.

Una vez conocida la posición i_{min} de este mínimo, se evaluará en las funciones **`x, y`** , permitiendo calcular el ángulo que forma el eje de la elipse con el eje OX con la expresión

$$\theta = arctan\left(\frac{y[i_{min}]}{x[i_{min}]}\right)$$

El valor arrojado por este método es

$$\theta = -3.946 \times 10^{-4}[rad]$$

$$\theta = 0.023°$$

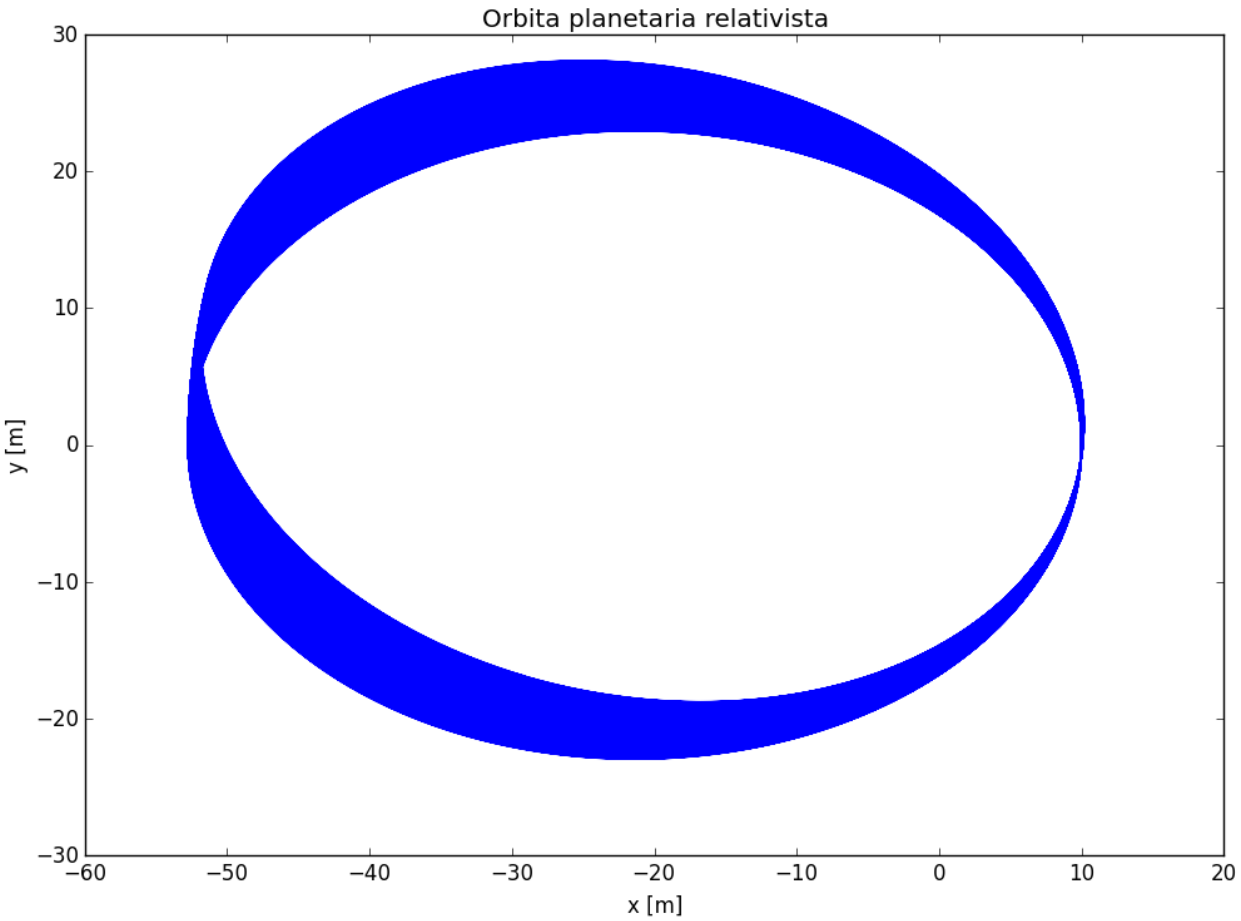
La velocidad angular de precesión, por lo tanto, se puede expresar en unidades de ángulo por ciclo, o en unidades de ángulo por unidades de tiempo, simplemente dividiendo la primera por el período T.

$$\omega = -0.023 \left[\frac{°}{ciclo}\right]$$

$$\omega = -2.050 \times 10^{-5} \left[\frac{°}{s}\right]$$

Para apreciar de mejor forma el efecto relativista, se integró la ecuación hasta 5×10^5 segundos, lo que equivale aproximadamente a 450 ciclos.

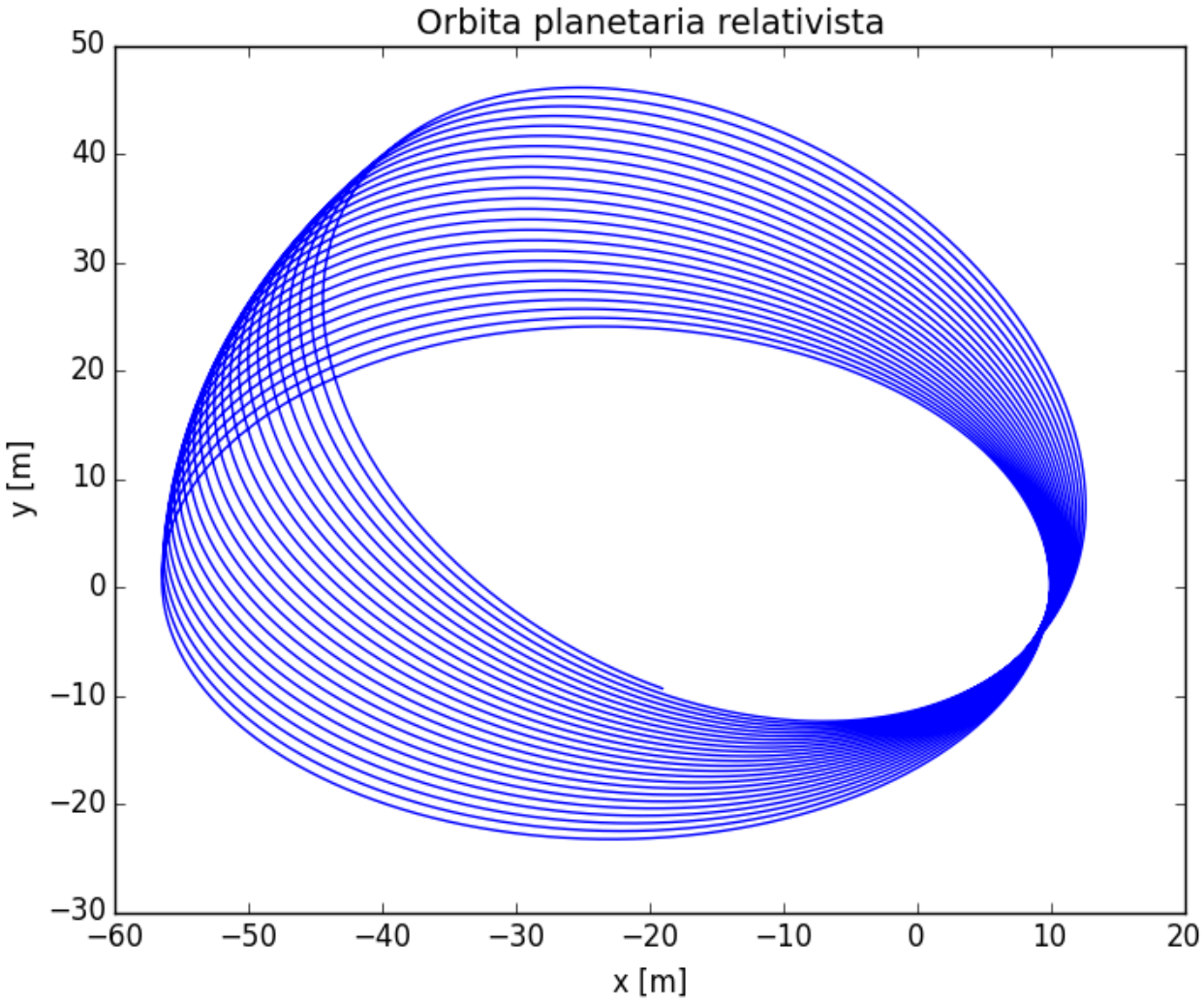
Figura 9: órbita relativista para 450 ciclos



Ahora sí es posible ver la rotación de la elipse, sin embargo, el desplazamiento es tan pequeño que las trayectorias están demasiado cerca como para ser distinguibles, viéndose como una superficie continua.

Para finalizar, y sólo como una forma de apreciar de mejor manera este fenómeno, se integrará la ecuación para $\alpha = 0.09$.

Figura 10: órbita relativista con $\alpha = 0.09$



La precesión de la órbita es evidente, siendo ésta en sentido horario, lo que concuerda con el signo negativo de la velocidad angular calculada.