Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas

FI3104-01 Métodos Numéricos para la Ciencia e Ingeniería

Tarea N°4: Diseño de Software, Ley de Gravitación Universal y órbitas cercanas al Sol

Bruno Scheihing, RUT: 18.954.350-6

20 de Octubre de 2015

Resumen

En la presente tarea se buscó aprender técnicas de diseño de software, para mejorar la implementación de programas, haciéndolos más estructurados y más fáciles de testear. Se respondieron las preguntas planteadas al respecto. La ecuación de Newton para la Ley de Gravitación Universal fue resuelta utilizando los métodos: Euler explícito, Runge-Kutta orden 4, y Verlet, obteniendo resultados satisfactorios con los dos últimos y una desviación importante en el caso de Euler. Se eligieron M, m tales que $GM = 1\,[Nm^2/kg]$ y $m = 1\,[kg]$, y condiciones iniciales nulas salvo por $x(0) = 10[m]$ y $v_y(0) = 0{,}3\,[m/s]$. En el caso donde se introduce una perturbación al potencial gravitatorio caracterizada por $\alpha = 10^{-2{,}350}$, se encontró una velocidad angular de precesión igual a $\omega_{prec} = -1{,}80 \pm 0{,}08\,[10^{-5} \cdot rad/s]$ (es decir, en sentido contrario a la órbita), que es consistente con la predicción algebraica que se obtiene para los parámetros y condiciones iniciales utilizadas.

1. Introducción

Las buenas prácticas de programación son vitales para tener un código funcional y estructurado. Esto último es muy importante a la hora de hacer modificaciones al mismo, pues permite controlar y testear los cambios efectuados al código. En este informe se responderán preguntas relacionadas con este tema (diseño de software), con el objetivo de aprender acerca de buenas prácticas de programación a partir de un tutorial desarrollado por Software Carpentry.

Por otra parte, en esta tarea se buscará resolver numéricamente la ecuación de movimiento para una partícula puntual en un campo gravitatorio central. En primer momento se estudiará la estabilidad de las soluciones, según los diferentes métodos para resolver ecuaciones diferenciales ordinarias (Euler explícito, RK4, Verlet), para la ecuación de movimiento planteada por Isaac Newton, correspondiente al potencial:

$$U(r) = -\frac{GMm}{r}$$

Se analizará el caso donde $GM = 1\,[Nm^2/kg]$, es decir, $M = 1{,}498 \cdot 10^{10}\,[kg]$, y además $m = 1\,[kg]$. El estudio se hará sobre el plano (x, y), y las condiciones iniciales serán:

$$x(0) = 10\,[m] \;\;;\;\; y(0) = 0\,[m] \;\;;\;\; dx/dt(0) = 0\,[m/s] \;\;;\;\; dy/dt(0) = 0{,}3\,[m/s]$$

Es fácil verificar que estas condiciones iniciales deberían generar una órbita elíptica, pues la energía total resulta ser menor que cero:

$$E = K + U = \frac{1\,[kg]}{2}(0{,}3\,[m/s])^2 - \frac{1\,[Nm^2]}{10\,[m]} = (0{,}045 - 0{,}1)\,[J] = -0{,}055\,[J]$$

Luego se procederá a analizar lo que ocurre cuando se introduce una perturbación en el potencial gravitatorio para explicar la precesión de las órbitas cuando se está suficientemente cerca del centro del campo gravitatorio. El potencial queda de la forma:

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

Este análisis se hará utilizando el algoritmo de Verlet, usando $\alpha = 10^{-2{,}350}$, con los demás parámetros y condiciones iniciales tomando los mismos valores que antes. Sin embargo, quizás sea necesario hacer más fina la partición de los intervalos temporales, para estudiar con mejor precisión la velocidad angular de precesión de la órbita.

En general se presentarán los resultados en forma de la trayectoria en el espacio (x, y) (las órbitas) y evolución temporal de la energía del sistema. Se espera poder sacar conclusiones respecto de la estabilidad de las soluciones según los métodos utilizados y determinar la velocidad de precesión de la órbita con el potencial perturbado.

Tal como se dice en el enunciado de la tarea, nos referiremos al perihelio como el punto más lejano de la órbita.

2. Diseño de Software

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

  La idea de escribir primero el *main driver* es crear la estructura del programa en primer lugar, y luego ir rellenando los segmentos de código faltantes, tales como funciones o clases. Es buena idea pues permite definir inmediatamente qué es lo que se espera del programa, caracterizando de mejor forma cuáles son las funcionalidades de los métodos a escribir más tarde. En resumen, permite definir qué es lo que hará el programa.

- ¿Cuál es la idea detrás de la función *mark_filled*? ¿Por qué es buena idea crearla en vez del código original al que reemplaza?

  La idea de la función *mark_filled* es controlar el buen funcionamiento del programa, pues añade mensajes de errores propios al programa, que se muestran en pantalla cuando uno de los índices del casillero a marcar está fuera de rango. Es buena idea hacerlo porque, en caso de que se ingrese un índice no permitido, el programa será capaz de manejarlo y sabremos por qué pasó. Si no lo hiciéramos, y por ejemplo, se ingresa un índice

negativo, python lo permitiría y más tarde en el programa habría algo que no funcionará como debería, y rastrear el error hasta la línea que lo provocó sería más costoso en tiempo.

- ¿Qué es *refactoring*?

  Refactoring es cambiar la estructura de un programa sin modificar su comportamiento ni funcionalidades con el fin de mejorar su calidad, por lo general con el objetivo de hacer que sea más fácil testearlo. Esto incluye hacer el código legible por sí mismo, dando nombres explicativos a las variables y funciones.

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

  Para que su implementación no introduzca errores propios, y permita detectar errores en el programa sin modificar su comportamiento. La simplicidad de los tests es muy importante, pues permite asegurarse de que el test está haciendo lo que debería. La estrategia usada en el tutorial es usar "fixtures", a partir de los cuales ejecutar los tests, junto con los resultados esperados. Luego define un método que ejecuta todos los tests para ir verificando el correcto funcionamiento del programa a medida que se implementan los cambios.

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

  La primera es cambiar espacio de memoria por tiempo, guardando información redundante para evitar re-hacer ciertos cálculos, así optimizando el tiempo de ejecución. La otra gran forma de optimizar programas es usar más tiempo para programar, y así generar código más específico que funcione más rápido para la situación particular a resolver.

- ¿Qué es *lazy evaluation*?

  Lazy evaluation es generar/evaluar/calcular las variables a utilizar sólo cuando se necesita, evitando usar espacio de memoria que finalmente no será utilizado. Esto puede hacer que la escritura del programa sea más complicada, pues se requiere de mayor especificidad.

- Describa la *other moral* del tutorial (es una de las más importantes a la hora de escribir buen código).

  Escribir un programa simple en primer momento, y luego testearlo y mejorarlo parte por parte, de forma que los tests generados en primer momento sigan siendo útiles a medida que el programa vaya creciendo, y usarlos para verificar su funcionalidad. Así se garantiza el buen funcionamiento del programa, pues su proceso de diseño permite corregir los errores de forma temprana (apenas son introducidos en el código) y efectiva.

3. Solución numérica a la Ley de Gravitación universal y órbitas cercanas al Sol

3.1. Procedimiento

Dado el potencial antes definido, tenemos que en general la ecuación de movimiento será:

$$m\ddot{\vec{r}} = -\frac{GMm}{r^2}\hat{r} + \frac{2\alpha GMm}{r^3}\hat{r}$$

por lo que en términos de (x, y) es el sistema:

$$m\ddot{x} = -\frac{GMm}{x^2+y^2}\frac{x}{\sqrt{x^2+y^2}} + \frac{2\alpha GMm}{(x^2+y^2)^{(3/2)}}\frac{x}{\sqrt{x^2+y^2}}$$
$$m\ddot{y} = -\frac{GMm}{x^2+y^2}\frac{y}{\sqrt{x^2+y^2}} + \frac{2\alpha GMm}{(x^2+y^2)^{(3/2)}}\frac{y}{\sqrt{x^2+y^2}}$$

Y se puede expresar como un sistema lineal de primer orden, haciendo la sustitución $v_x = \dot{x}$, $v_y = \dot{y}$:

$$\frac{d}{dt}\vec{y} = \frac{d}{dt}\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ \frac{GMx}{(x^2+y^2)^{(3/2)}}\left(\frac{2\alpha}{\sqrt{x^2+y^2}} - 1\right) \\ \frac{GMy}{(x^2+y^2)^{(3/2)}}\left(\frac{2\alpha}{\sqrt{x^2+y^2}} - 1\right) \end{pmatrix} = \vec{f}(\vec{y}, t) = \vec{f}(\vec{y})$$

En primer lugar, se considera $\alpha = 0$. Para resolver la ecuación mediante el método de Euler explícito, se sigue el proceso:

$$\vec{y}_{n+1} = \vec{y}_n + h\vec{f}(\vec{y}_n)$$

donde h es el paso elegido para discretizar los intervalos temporales. En términos de las unidades del problema, tal como está planteado, se usará $h = 0{,}1\,[s]$. Este paso también aplica para las otras dos resoluciones de la ecuación con $\alpha = 0$.

Por otra parte, se busca resolver la misma ecuación, con $\alpha = 0$, utilizando un método Runge-Kutta orden 4 (RK4). Para ello se calcula $\vec{y}_{n+1}$ como sigue:

$$\vec{K}_1 = h\vec{f}(\vec{y}_n)$$
$$\vec{K}_2 = h\vec{f}(\vec{y}_n + \vec{K}_1/2)$$
$$\vec{K}_3 = h\vec{f}(\vec{y}_n + \vec{K}_2/2)$$
$$\vec{K}_4 = h\vec{f}(\vec{y}_n + \vec{K}_3)$$
$$\vec{y}_{n+1} = \vec{y}_n + [K_1 + 2K_2 + 2K_3 + K_4]/6$$

Similarmente, también interesa resolver el sistema físico utilizando el método de Verlet (todavía con $\alpha = 0$). En este caso no interesarán las velocidades de forma directa, y se resuelve el sistema:

$$\frac{d^2}{dt^2}\vec{y} = \frac{d^2}{dt^2}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{GMx}{(x^2+y^2)^{(3/2)}} \left(\frac{2\alpha}{\sqrt{x^2+y^2}} - 1 \right) \\ \frac{GMy}{(x^2+y^2)^{(3/2)}} \left(\frac{2\alpha}{\sqrt{x^2+y^2}} - 1 \right) \end{pmatrix} = \vec{f}(\vec{y}, t) = \vec{f}(\vec{y})$$

y se calcula $\vec{y}_{n+1}$ como:

$$\vec{y}_{n+1} = 2\vec{y}_n - \vec{y}_{n-1} + h^2 \vec{f}(\vec{y}_n)$$

Para este método se requieren dos instantes iniciales, así que se da el primer paso usando RK4 y desde ahí en adelante aplica lo anterior.

El cálculo de la velocidad (se usará para graficar la energía en función del tiempo), a falta de una mejor alternativa, se hace en cada paso como:

$$\vec{v}_n = \frac{\vec{y}_n - \vec{y}_{n-1}}{h}$$

Finalmente, para estudiar una posible precesión de la órbita se usará $\alpha = 10^{-2,350}$, esto con la idea de modelar órbitas cercanas al Sol. Para ello se usará el método de Verlet, de la misma forma que fue presentado justo antes. Es importante notar que el potencial perturbado admite una solución analítica para la coordenada radial r en función de la coordenada angular ϕ. Definiendo $u(\phi) \equiv 1/r(\phi)$, se tiene la ecuación de Binet:

$$\frac{d^2u}{d\phi^2} + u = -\frac{m}{l^2}\frac{d}{du}(U(1/u))$$

Donde $l = mr_0v_0$ es el momentum angular de la partícula. Desarrollando la expresión anterior, se obtiene:

$$\frac{d^2u}{d\phi^2} + \left(1 + \frac{2\alpha GMm^2}{l^2}\right) u = \frac{GMm^2}{l^2}$$

Resolviendo esta ecuación y despejando en términos de $r(\phi)$, se obtiene:

$$r(\phi) = \frac{1}{Acos(\sqrt{1 + \frac{2\alpha GMm^2}{l^2}}) + C}$$

Con A, C constantes dependientes de las condiciones iniciales y de los parámetros del problema. Lo importante es que la variable r repite sus máximos cuando el argumento del coseno aumenta en 2π. Es decir, si en $\phi = 0$ hubo un máximo, entonces en ϕ_m (descrito por la ecuación siguiente) también hay un máximo:

$$2\pi = \sqrt{1 + \frac{2\alpha GMm^2}{l^2}} \cdot \phi_m \implies \phi_m = \frac{2\pi}{\sqrt{1 + \frac{2\alpha GM}{r_0^2 v_0^2}}}$$

Lo que nos da una expresión para el ángulo de precesión esperado por ciclo de r (en términos de los parámetros y condiciones iniciales):

$$\phi_{prec} = \phi_m - 2\pi = 2\pi \left(\frac{1}{\sqrt{1 + \frac{2\alpha GM}{r_0^2 v_0^2}}} - 1 \right) \approx -0{,}003116\,[rad]$$

Esto será un importante valor de referencia para decidir qué tan buena es la solución encontrada.

Para determinar la posición del punto más lejano de la órbita, se calcula y recorre el arreglo de posiciones radiales a partir de las posiciones resultantes de la solución al sistema, comparando cada distancia radial contra la máxima del arreglo. Si está suficientemente cerca (en este caso, se usará una tolerancia de $10^{-6}\,[m]$) entonces se considera que el planeta llegó a su punto más lejano del cuerpo central. Como el ángulo de precesión se espera pequeño, entonces un modo de asegurar que el planeta ha abandonado su región más lejana a la estrella es esperar a que haya recorrido un ángulo de $\pi/2$, y luego volver a preguntar en cada paso si está suficientemente cerca o no.

Experimentalmente se determinó que para obtener un ángulo de precesión suficientemente regular, es necesario disminuir el paso h a $0{,}01\,[s]$.

3.2. Resultados

Los resultados de la integración de la ecuación diferencial utilizando Euler explícito se presentan en Figura 1. Se observa que la solución es inestable, pues va aumentando su energía con el tiempo, con lo que eventualmente la solución dejará de entregar órbitas completas en torno al cuerpo central.

Como se ve desde Figura 2 y Figura 3, los resultados de la resolución de este sistema de ecuaciones concuerdan con lo esperado (las órbitas son cerradas) y la energía se mantiene esencialmente constante en el rango temporal estudiado, e igual a la energía calculada en *Introducción*. Si se hace zoom en los gráficos (en la ejecución misma de los programas) se encuentra que la energía tiene un comportamiento ligeramente decreciente-oscilante en el caso de RK4 y oscilante en el caso de Verlet. Ambas variaciones son despreciables para efectos del análisis de solución obtenida, en la escala relevante para este problema.

En Figura 4 se muestra la solución obtenida mediante el método de Verlet para el potencial perturbado. Después de 34 órbitas, se aprecia un ligero cambio en la órbita: un giro en sentido horario, indicado por las flechas de la primera imagen. Se muestran solamente la primera y última órbita para que sea posible apreciar la diferencia. Realizando el cálculo de la velocidad angular de precesión, se obtuvo:

$$\omega_{prec} = -1{,}80 \pm 0{,}08\,[10^{-5} \cdot rad/s]$$

que corresponde a una precesión angular por ciclo de r de:

$$\phi_{prec} = -0{,}00311 \pm 0{,}00014\,[rad]$$

que se compara bastante bien con el valor predicho teóricamente en la sección *Procedimiento* ($-0{,}003116\,[rad]$).

Es interesante notar que, si se explora el arreglo de ángulos obtenidos correspondientes a los máximos de la distancia radial, por lo general al variar h (que por cierto en el código se escribe como dt), parecieran haber dos ángulos preferidos para precesar, que son función de h, pero el promedio de los arreglos está siempre bastante cerca del valor predicho algebraicamente. Esto sugiere que dependiendo de la discretización del movimiento se podría observar una mayor desviación estándar en la precesión. También tiene que ver la tolerancia elegida al momento de detectar los máximos de r. Sin embargo, cuando se elige un valor suficientemente bajo de h, como es el caso, los ángulos preferidos se parecen cada vez más entre sí, haciendo que la precesión medida sea más estable.

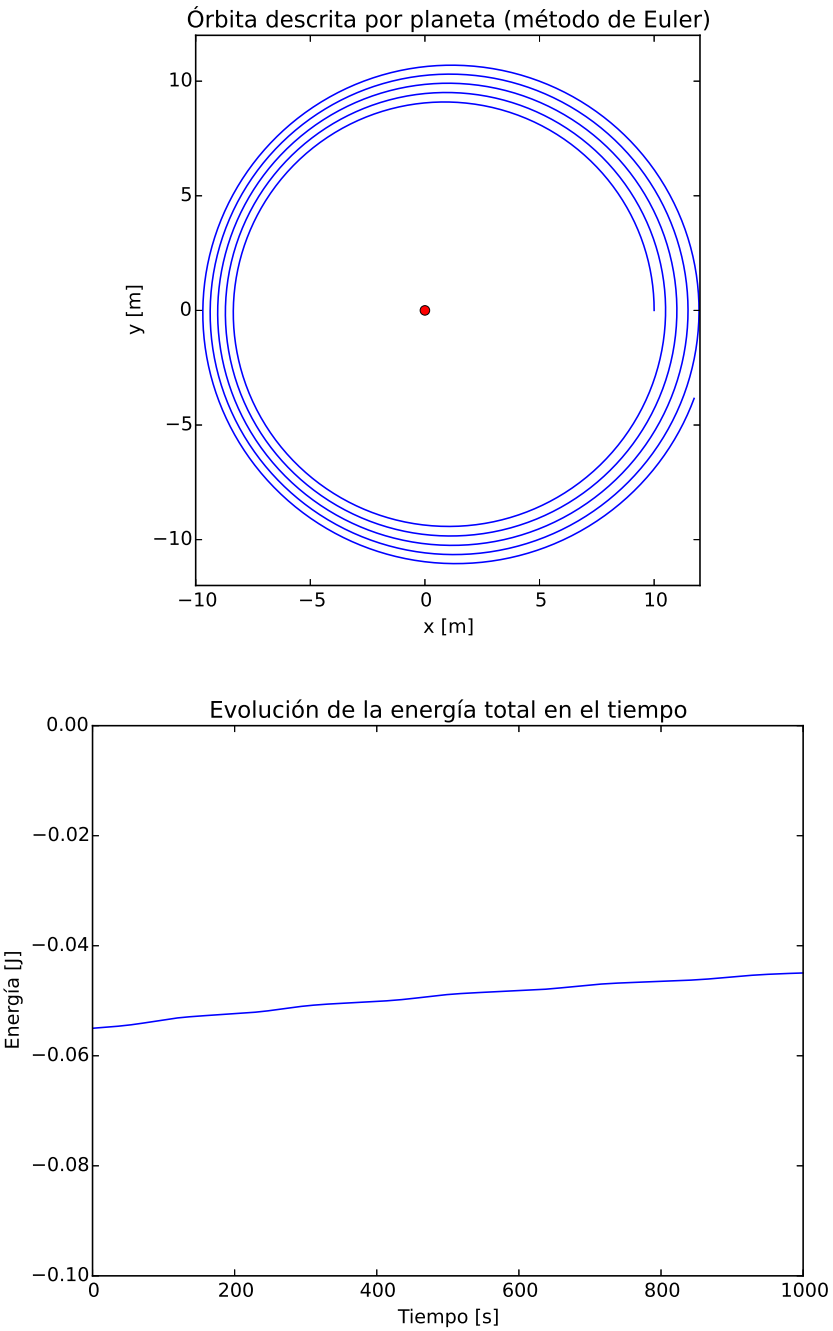

Figura 1: Solución a ley de gravitación universal utilizando el método de Euler explícito. Masa del cuerpo central = $1{,}498 \cdot 10^{10}$ kg, masa del planeta = 1 kg. El punto rojo muestra la posición del "Sol".

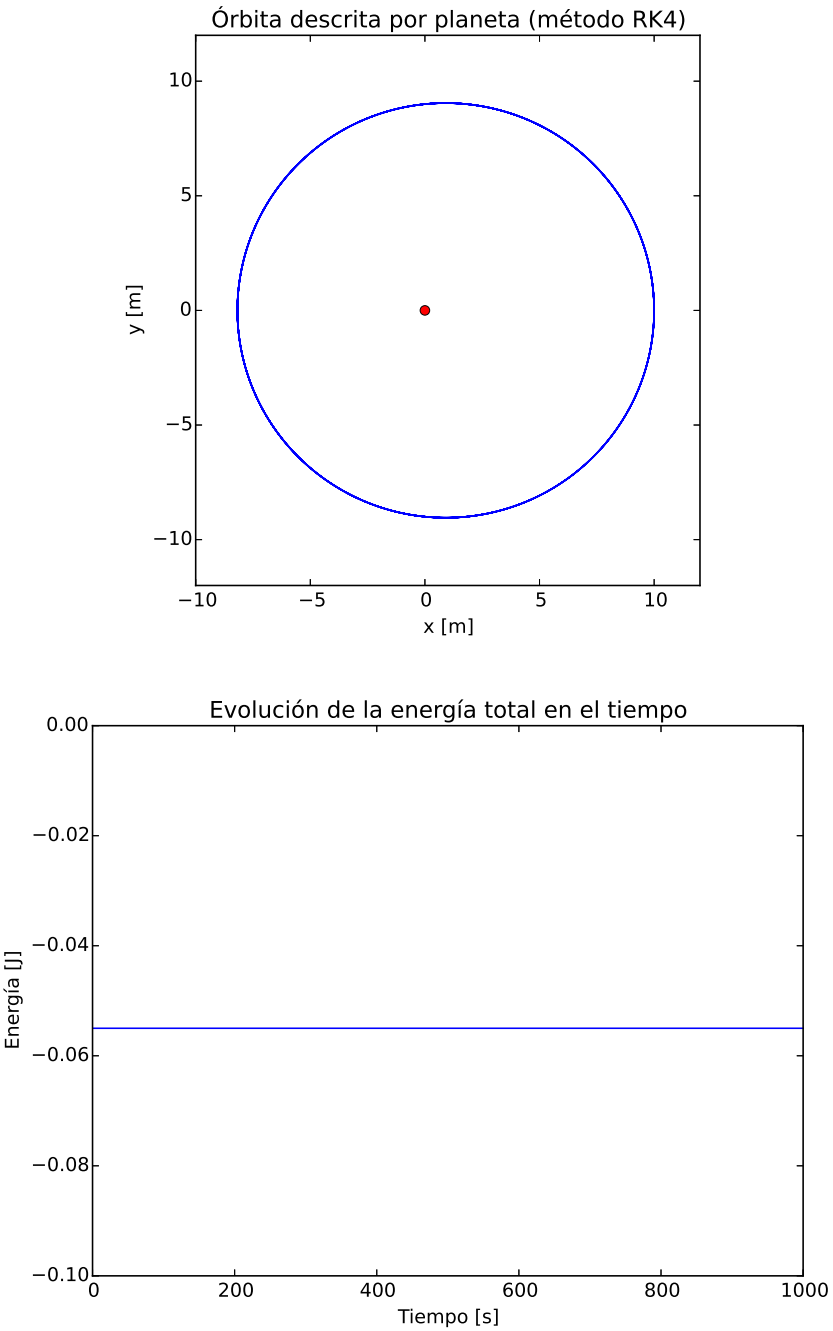

Figura 2: Solución a ley de gravitación universal utilizando un método RK4. Masa del cuerpo central = $1{,}498 \cdot 10^{10}$ kg, masa del planeta = 1 kg. El punto rojo muestra la posición del "Sol".

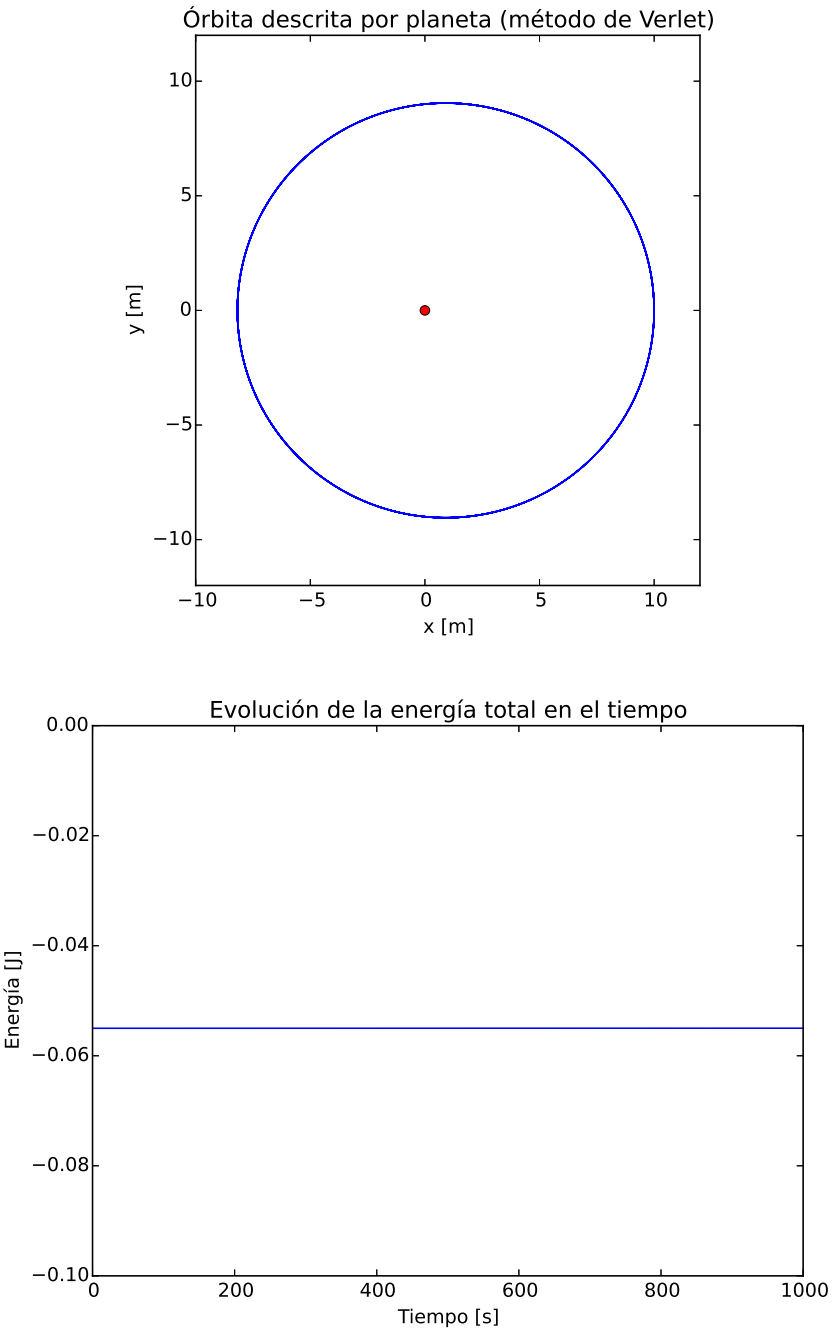

Figura 3: Solución a ley de gravitación universal utilizando el método de Verlet. Masa del cuerpo central = $1{,}498 \cdot 10^{10}$ kg, masa del planeta = 1 kg. El punto rojo muestra la posición del "Sol".

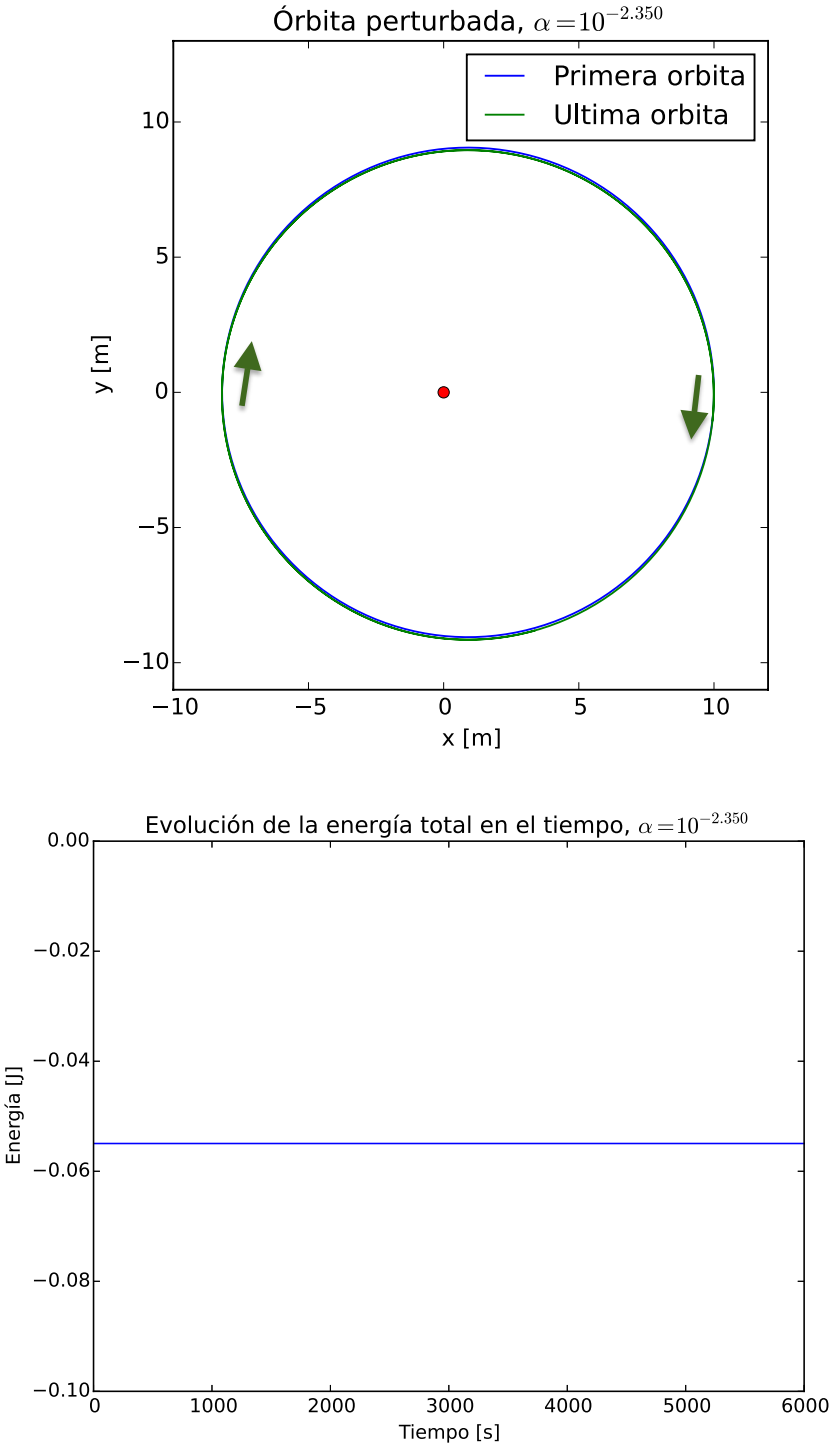

Figura 4: Solución a ley de gravitación universal con corrección relativista, utilizando el método de Verlet. Masa del cuerpo central = $1{,}498 \cdot 10^{10}$ kg, masa del planeta = 1 kg. El punto rojo muestra la posición del "Sol".

4. Conclusiones

Las preguntas planteadas en torno al proceso de diseño de software generaron una interesante reflexión acerca de cómo se debe programar para que el resultado sea legible y bien estructurado. Generar tests para los programas desarrollados puede ser muy útil al momento de seguir extendiéndolos, pues permite verificar constantemente si las nuevas modificaciones inducen comportamientos indeseados o no.

Las figuras presentadas para $\alpha = 0$ permiten concluir que el método de Euler explícito es altamente inestable, y de acuerdo a lo visto en clases, el error con respecto a la función solución esperada va creciendo indefinidamente en el tiempo. Este error se ve plasmado en el crecimiento de la energía con el tiempo, y también (de forma correlacionada) en el aumento del radio de la órbita.

Por otra parte, los algoritmos RK4 y de Verlet generaron soluciones altamente estables, donde el gráfico de la energía para la escala relevante del problema es una línea horizontal (i.e. es constante y por lo tanto se conserva). Las órbitas obtenidas son prácticamente indistinguibles entre estos dos algoritmos, apoyando su convergencia a la solución real.

La determinación de la precesión de la órbita en el caso $\alpha = 10^{-2,350}$ fue exitosa, pues se obtuvo un valor promedio para el ángulo de precesión que se corresponde muy bien con el que se predice algebraicamente. Sin embargo, la mayor precisión requerida para estabilizar la precesión angular en cada órbita hace que el programa tenga un tiempo de ejecución del orden de los 15 segundos, por lo que es un posible lugar donde llevar a cabo optimizaciones, en alguna posterior revisión del código.