Métodos numéricos para la ciencia y la ingeniería
Tarea 4

Jean Paul Martel 19055257-8

20 de Octubre 2015

1 Parte 1 Diseño de Programa

1.1 Introducción

Se quiere estudiar el tutorial sobre diseño de programas de la página
$http://swcarpentry.github.io/v4/invperc/index.html$

El objetivo es mejorar el diseño del software para desarrollar las tareas. Luego se pretende responder preguntas relacionadas a este tutorial:

1.2 Respuestas

Luego de estudiar el tutorial mencionado estás son las preguntas y sus respectivas respuestas desarrolladas:

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

La idea es escribir primero la estructura del programa, es decir, la secuencia de operaciones que realizará el programa, para luego definir las funciones que se utilizaran. Es buena idea porque permite tener un código ordenado que muestre exactamente lo que se está haciendo.

2- ¿Cual 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 es que asegura que se le han entregado valores adecuados (en el rango correcto) y reporta cuando no ha sido así. Es buena idea crearla pues dice más claro cuál es el problema que un error genérico de Python; además de aclarar cuál es el rango de valores esperados que la función recibe. También detecta como valores incorrectos algunos que Python los consideraría "legales". Es decir la función tiene como fin que sea más fácil de revisar posibles errores.

3- ¿Qué es refactoring?

Es un proceso en donde se reestructura (reordena) el código del programa, pero sin cambiar su funcionalidad o comportamiento, ni arreglar errores; si no para mejorar la comprensión del código con el fin de facilitar luego la resolución de errores o el añadir nuevas funcionalidades al programa.

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

El problema de crear un test escribiendo códigos muy complejos, es que puede provocar confusiones que obligan a revisar constantemente si el test mismo contiene algún error. Luego por cada test que se quiera probar debe escribirse otro código. Se utiliza por tanto mucho tiempo, existe gran probabilidad de errores y cuando el programa es cambiado, muchas veces es necesario cambiar también los test. La estrategia del tutorial es crear una lista que contenga "graficamente" (con strings) todos los test que quieran realizarse y sus respectivos resultados (texture). Eso es sencillo de escribir y también es fácil añadir más tests. Además se crea una función que corra todos los test al mismo tiempo, usando un loop que itera para cada test usando una serie general de comandos que funcionan para cualquier test. A pesar de que esto requiere bastante trabajo, facilita corregir errores y el código no debería cambiar si el programa se modifica después, por lo que se utilizan los mismos tests.

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

La primera idea es lo que se llama "Análisis asintótico". Con asintótico se refiere a que se analiza las funciones o algoritmos para números muy grandes para estudiar la eficacia computacional. La idea es que puede guardarse información en cierto sentido redundante, con el fin de no recalcular cosas que que requieran mucho tiempo de computación. Es decir, puede sacrificarse consumo de memoria por tiempo de ejecución y viceversa.

La segunda idea es "Busqueda binaria", que se utiliza cuando quiere encontrarse un elemento en una lista o arreglo. Dado un arreglo ordenado se escoge la posición media, se compara y determina si el valor buscado se encuentra en esa posición, o esta antes o después. Luego, en la mitad donde debe ser encuentra el elemento se vuelve a subdividir el arreglo, y así en sucesivas mitades, hasta encontrar el elemento buscado. Esto es más eficiente que buscar elementos de forma secuencial desde el primer elemento, pues $log_2(N)$ intentos son suficientes para encontrar un elemento en una lista de N.

6- ¿Qué es lazy evaluation?

Lazy evaluation o evaluación perezosa, es otra estrategia para optimización de programas. Consiste en no calcular una expresión hasta que su valor sea necesario. Así se incrementa el rendimiento (disminuye el tiempo de ejecución) evitando cálculos innecesarios.

7- Describa la other moral del tutorial

Es de las más importantes consejos para escribir un buen código. Si se quiere escribir un programa con buen rendimiento (rápido) es recomendable escribir una versión simple del programa para luego probarlo y mejorarlo poco a poco, reutilizando las pruebas (tests) para comprobar el trabajo en cada etapa.

2 Parte 2: Modelo corregido para la órbita de una planeta

2.1 Introducción

Se pretende modelar la orbita de un planeta cercano al Sol, utilizando una aproximación derivada de la teoría de la relatividad general de Einstein:

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

con α un número pequeño. Esta fórmula corresponde a una más exacto que la que describe la Ley de Gravitación Universal de Newton. Bajo este potencial, las órbitas ya no son cerradas, sino que precesan, es decir, el afelio gira alrededor del Sol. Utilizando coordenadas cartesianas y el potencial queda de la forma

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

Se utilizaran unidades tales que $GMm = 1$. Si se elige arbitrariamente $m = 1$, entonces se tendrá también que $GM = 1$.

2.2 Modelos

2.2.1 Caso 1: $\alpha = 0$

Primero se considera el caso en que el potencial sigue la Ley de Gravitación Universal de Newton. En este caso particular las ecuaciones de movimiento quedan:

$$\ddot{x} = -\frac{GMx}{(x^2+y^2)^{3/2}}$$

$$\ddot{y} = -\frac{GMy}{(x^2+y^2)^{3/2}}$$

Se desea integrar para 5 órbitas (aproximadamente) esta ecuación y obtener las soluciones (coordenadas) utilizando los métodos numéricos Euler explícito, RK4 y Verlet. Además quiere graficarse las órbitas y la energía total del sistema como función del tiempo para los 3 métodos usados. Las condiciones iniciales usadas en este caso y el siguiente son:

$$x_0 = 10$$

$$y_0 = 0$$
$$v_x = 0$$
$$v_y = 0$$

2.2.2 Caso 2: $\alpha = 10^{-2.257}$

Usando este α pequeño distinto de 0 las ecuaciones son las mostradas en *Introduccion*. Esta vez sólo quiere integrarse con el método de Verlet para 30 órbitas. Al igual que antes se quieren graficar los resultados para la órbita y la energía total en función del tiempo. Adicionalmente se busca determinar la velocidad angular de precesión del afelio en torno al foco

3 Procedimiento

3.1 Ecuaciones de movimiento

Utilizando la segunda ley de Newton ($F = ma$) y recordando que para una fuerza conservativa, esta corresponde a menos el gradiente del potencial; pueden encontrarse dos ecuaciones de movimiento, una para cada coordenada (considerando que el movimiento se desarrolla en un plano). Estas ecuaciones son:

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

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

Además la energía total de sistema corresponde a la energía cinética más la potencial:

$$E_{total} = \frac{1}{2}(\dot{x}^2 + \dot{y}^2) - \frac{GMm}{\sqrt{x^2+y^2}} + \alpha\frac{GMm}{x^2+y^2}$$

3.2 Creación de Clase Planeta

Primero se propone crear la clase *Planeta* en el archivo *planeta.py*. Esta clase contiene la función $ecuacion\_de\_movimiento(self)$ que implementa la ecuación de movimiento, 3 métodos con los que se integrará la ecuación: Euler explícito, RK4 y Verlet. Y Además contiene una función para calcular la energía total del sistema en un punto dado.

Considerando que los métodos deben aplicarse para dos ecuaciones (en x e y), los algoritmos quedan para cada caso: Las funciones $avanza\_euler(self, dt)$, $avanza\_rk4(self, dt)$ y $avanza\_verlet(self, dt)$ realizan una iteración de estos procedimientos para un avance de un intervalo de tiempo dt, que depende de la discretización utilizada

Para resolver numéricamente las ecuaciones de segundo orden, se considera los cambios de variables: $v_x = \dot{x}$ y $v_y = \dot{y}$, tal que:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ f(x) \\ f(y) \end{pmatrix}$$

Con $f(x) = -\frac{GMx}{(x^2+y^2)^{3/2}} + \alpha\frac{2GMx}{x^2+y^2}$ y $f(y) = -\frac{GMy}{(x^2+y^2)^{3/2}} + \alpha\frac{2GMy}{x^2+y^2}$

Así el sistema de 2 ecuaciones de segundo orden se transforma en dos sistemas de primer orden, en los cuales se aplican los métodos numéricos mencionados.

3.3 Desarrollo caso $\alpha = 0$

Para integrar la ecuación primero se importan los módulos de la case *Planeta*, se establece la condición inicial y se inicializa la clase con esta condición (al no especificar α se asume igual a 0). Se elige una discretización del tiempo tal que se integra hasta 5 órbitas con un paso $dt = 1$.

Por cada "método" numérico Se inician arreglos vacíos para x, y, *Energia*. Hasta 5 periodos se realizan iteraciones que implementan las funciones correspondientes (importadas de la clase *Planeta*) y van actualizando los valores de posiciones y energía, además de añadirlos al arreglo correspondiente que permitirá graficar. También se obtienen los valores de la energía en cada tiempo y así se tienen los datos necesarios para graficar

1) y vs.x

2)Energía vs tiempo

3.3.1 Resultados caso $\alpha = 0$

Los gráficos obtenidos se encuentran a continuación

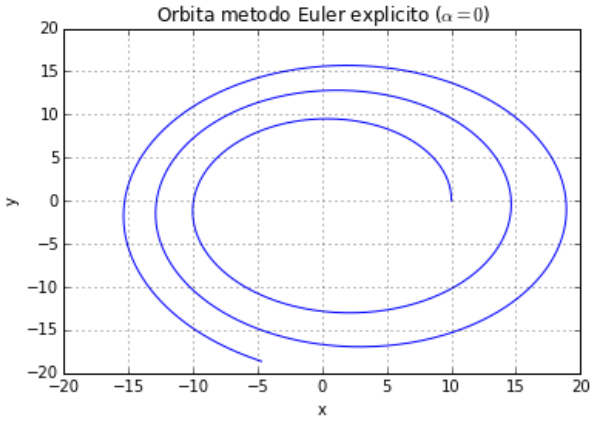

Figure 1: Grafico orbita y vs x Corresponde a la solución calculada con Euler explícito para $\alpha = 0$ y condiciones iniciales $x_0 = 10$, $y_0 = 0$, $v_x = 0$, $v_y = 0.3$.

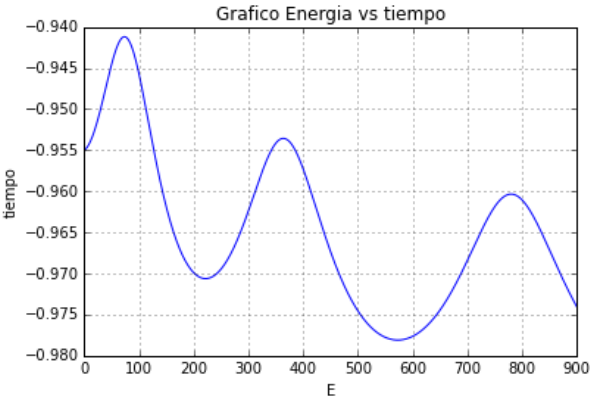

Figure 2: Grafico Energia total vs tiempo. Energía según los valores de la solución calculada con Euler explícito para $\alpha = 0$ y condiciones iniciales $x_0 = 10$, $y_0 = 0$, $v_x = 0$, $v_y = 0.3$.

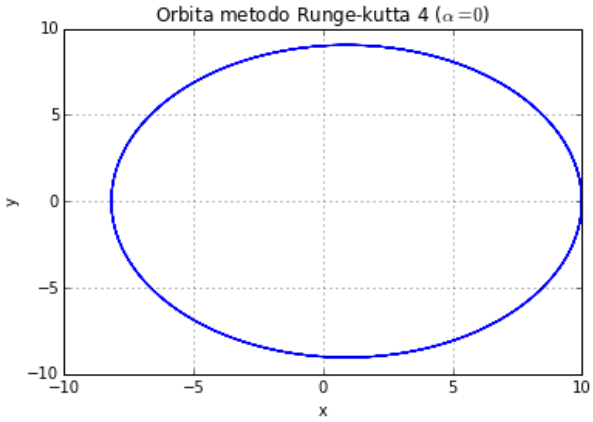

Figure 3: Gráfico órbita y vs.x. Corresponde a la solución calculada con Runge-kutta 4 para $\alpha = 0$ y condiciones iniciales $x_0 = 10$, $y_0 = 0$, $v_x = 0$, $v_y = 0.3$.

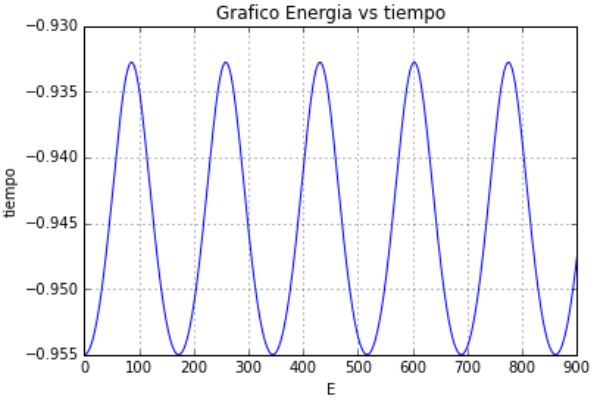

Figure 4: Grafico Energia total vs tiempo. Energía según los valores de la solución calculada con Euler explícito para $\alpha = 0$ y condiciones iniciales $x_0 = 10$, $y_0 = 0$, $v_x = 0$, $v_y = 0.3$.

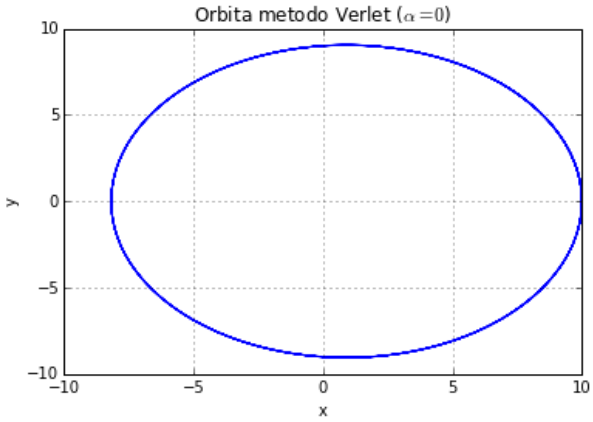

Figure 5: Grafico órbita y vs x. Corresponde a la solución calculada con Verlet para $\alpha = 0$ y condiciones iniciales $x_0 = 10$, $y_0 = 0$, $v_x = 0$, $v_y = 0.3$.

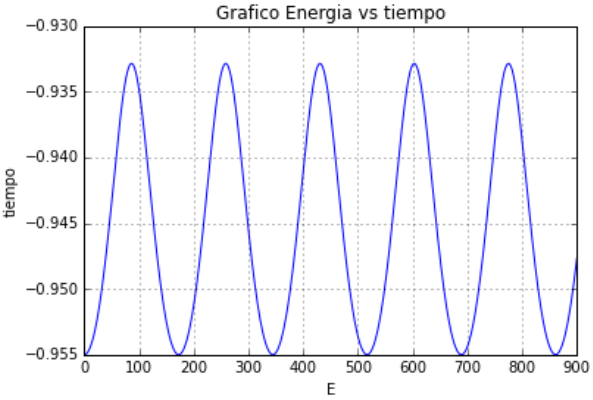

Figure 6: Grafico Energia total vs tiempo. Energía según los valores de la solución calculada con Euler explícito para $\alpha = 0$ y condiciones iniciales $x_0 = 10$, $y_0 = 0$, $v_x = 0$, $v_y = 0.3$.

3.4 Desarrollo caso $\alpha = 10^{-2.257}$

El procedimiento es análogo que para el caso $\alpha = 0$ para obtener los gráficos de posición y de energía; pero integrando hasta aproximadamente 30 órbitas. También se calcula la velocidad angular de precesión del afelio (punto más lejano de la órbita) en torno al foco. Conociendo el periodo se puede posicionar en el tiempo en que la última órbita ya ha comenzado. Desde ese tiempo se obtienen dos arreglos para las coordenadas (x,y), estos corresponden a los puntos de la última órbita aproximadamente. Usando la fórmula usual de distancia, se

calcula esta ente cada punto y la posición del foco. Luego se obtienen las coordenadas tal que esta distancia es máxima, es decir la coordenada del afelio. Al hacerlo de esta forma se asegura que la posición al afelio se tenga de forma bastante exacta. Es sencillo ahora tener el ángulo entre la primera y la última posición del afelio: $\phi = arctan\frac{y}{x}$. La velocidad angular corresponderá a:

$$\omega = \frac{\phi}{tiempo - total}$$

3.4.1 Resultados caso $\alpha = 10^{-2.257}$

La velocidad angular calculada es

$$\omega = 3.2908e - 05$$

Los gráficos obtenidos se encuentran a continuación:

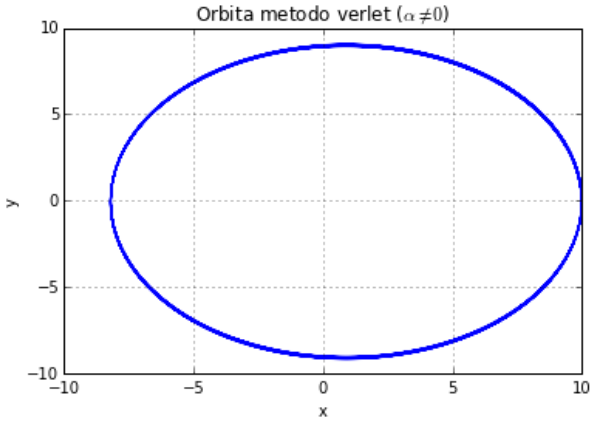

Figure 7: Grafico órbita y vsx. Corresponde a la solución calculada con Verlet para $\alpha = 10^{-2.257}$ y condiciones iniciales $x_0 = 10$, $y_0 = 0$, $v_x = 0$, $v_y = 0.3$.

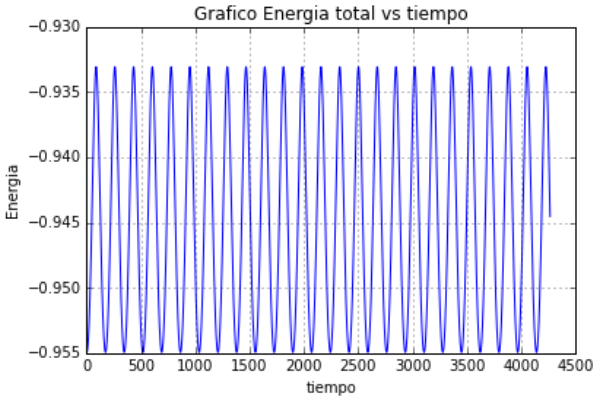

Figure 8: Grafico Energia total vs tiempo. Energía según los valores de la solución calculada con Euler explícito para $\alpha = 10^{-2.257}$ y condiciones iniciales $x_0 = 10$, $y_0 = 0$, $v_x = 0$, $v_y = 0.3$.

3.5 Conclusiones

En cuanto al cálculo para 5 órbitas para el caso $\alpha = 0$, se observa que el método de Euler no modela correctamente la órbita pues no mantiene una solución estable como la que se espera. En cambio los métodos de RK4 y Verlet sí producen el comportamiento esperado para la solución, manteniéndose estable para todas las órbitas. Además no se aprecian diferencias significativas entre estos dos últimos métodos. En cuanto al comportamiento de la energía en función del tiempo, para los métodos de RK4 y Verlet, aunque oscila, es estable en torno a un punto de equilibrio y mantiene un valor cercano a -1, obteniendo así una energía negativa con poca variación como se espera en este sistema. El método de Euler nuevamente no es representativo para los valores de la energía.

Con respecto al cálculo de 30 órbitas con el método de Verlet para el caso $\alpha = 10^{-2.257}$, con las condiciones iniciales escogidas, la órbita tiene una pequeña precesión en torno al foco, lo que era esperable con la corrección impuesta al potencial de la ley de gravitación universal. La energía no cambia significativamente de valor ni de comportamiento respecto al primer caso, ya que α es un valor muy pequeño. Por lo tanto Verlet es un buen método para modelar este tipo de problemas.