Tarea 04 - Métodos Numéricos

Luz Agüero Contreras 18.355.502-2
Profesor: Valentino Gonzales
Auxiliar: Felipe Pesce

20 Octubre 2015

1 Introducción

Parte 1
Se pide ver el tutorial *Program Design* de *Software Carpentry*, el cual enseña, por medio de un ejemplo, a optimizar el proceso de programación y a crear códigos más entendibles para cualquier usuario, lo cual ahorra tanto tiempo como recursos.

Parte 2
Esta parte consistió en completar el archivo `platena.py`, en donde se encontraba la clase `planeta`, la cual simula el movimiento de un planeta alrededor del sol sometido a un potencial $U(r)$ dado.

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

Se pedía que la simulación de este movimiento fuese integrado por medio de 3 métodos distintos, Euler, Runge Kutta de orden 4 y Verlet, para su posterior estudio.

2 Procedimiento

Parte 1
Las preguntas se encuentran respondidas en la sección de resultados.

Parte 2
Se ecribieron las funciones pre-definidas en el código de la clase `planeta`, partiendo por `ecuación_de_movimiento`, en donde se plantearon como sub-funciones las ecuaciones de movimiento calculadas desde el potencial de la siguiente forma:

$$m\vec{a} = -\nabla \cdot U(r)$$

Lo que se traduce a:

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

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

Luego, tanto `avanza_euler`, `avanza_rk4` y `avanza_verlet` se escribieron como los métodos correspondientes vistos en clases. Después, `energia_total` ocupa el potencial más la energía cinética calculada desde las velocidades para retornar la energía total.
Posteriormente, se modifica el archivo `solucion_usando_euler.py` (nombre cambiado a `soluciones.py`), en donde se crearon los códigos para usar las funciones de la clase `planeta`, partiendo por imponer las condiciones iniciales y luego definir arreglos vacíos en donde se guardarían los valores necesarios. Notar que para `avanza_euler` y `avanza_rg4` se ocuparon $Tiempo_{total} = 1000$ y $N_{pasos} = 1000$, en cambio para `avanza_verlet`, se usaron $Tiempo_{total} = 5000$ y $N_{pasos} = 1000$. Por último, se iteraron las funciones para conseguir los valores a gráficar, los que se presentan en la sección de resultados.

3 Resultados

Parte 1

- Describa la idea de escribir el *main driver* primero y llenar los huecos luego. ¿Por qué es buena idea?
  Hacer un esqueleto del programa que se desea ejecutar, permite tener una claridad del objetivo y visualizar cómo es que se resolverá el problema a grandes rasgos.

- ¿Cuál es la idea detrás de la función `mark_filled`? ¿Por qué es buena idea crearla en vez del código original al que reemplaza?
  Ésta función se asegura de controlar que el programa funcione bajo parametros coherentes, lo cual ayuda a encontrar errores más rápido y a corregirlos de forma directa.

- ¿Qué es *refactoring*?
  Consiste en revisar el programa para re-organizar su estructura, agregar comentarios que faciliten su comprensión, etc. Es buscar maneras de optimizar el código sin hacer grandes cambios en el proceso de él.

- ¿Por qué es importante implementar tests que sean sencillos de escribir? ¿Cuál es la estrategia usada en el tutorial?
  Los tests buscan comprobar el buen funcionamiento del programa y evitar *bugs*, por lo que lógicamente debiesen ser sencillos para evitar que en ellos se produzcan errores. En el tutorial, con los *fixtures* se generan resultados

esperados correctos, los cuales son comparados con los resultados de los tests y así verificar el funcionamiento del programa.

- El tutorial habla de dos grandes ideas para optimizar programas, ¿cuáles son esas ideas? Descríbalas.
  Las ideas son *Memoria a cambio de tiempo* y *Dedicarle más tiempo a programar a cambio de optimización*. La primera habla de usar recursos calculados por nuestro código, para ahorrar tiempo en el proceso de éste. Y la segunda, de ocupar más tiempo en programar el código, para obtener resultados más limpios y de sencilla correción y comprensión.

- ¿Qué es *lazy evaluation*?
  Se trata de definir variables solo cuando sean necesarias, no antes, ya que así se evita gastar memoria antes de tiempo y tener variables que no se usaran finalmente, lo cual solo demora más el programa.

- Describa la *other moral* del tutorial (es una de las más importantes a la hora de escribir buen código).
  La enseñanza final del tutorial dice que para obtener un programa eficiente se debe comenzar por uno simple, el cual se debe ir modificando por parte, agregando comentarios y siempre testeando.

Parte 2
Los gráficos presentados a continuación fueron hechos con las condiciones iniciales $x_0, y_0, v_{x0}, v_{y0} = (10, 0, 0, 0.35)$ y $\alpha = 0$

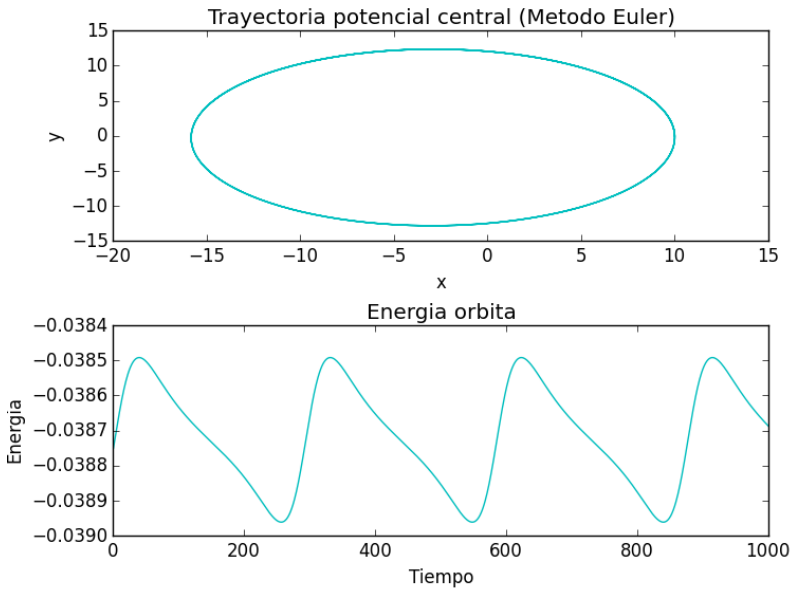

Figure 1: Gráfico método de Euler explicito con un tiempo final de 1000

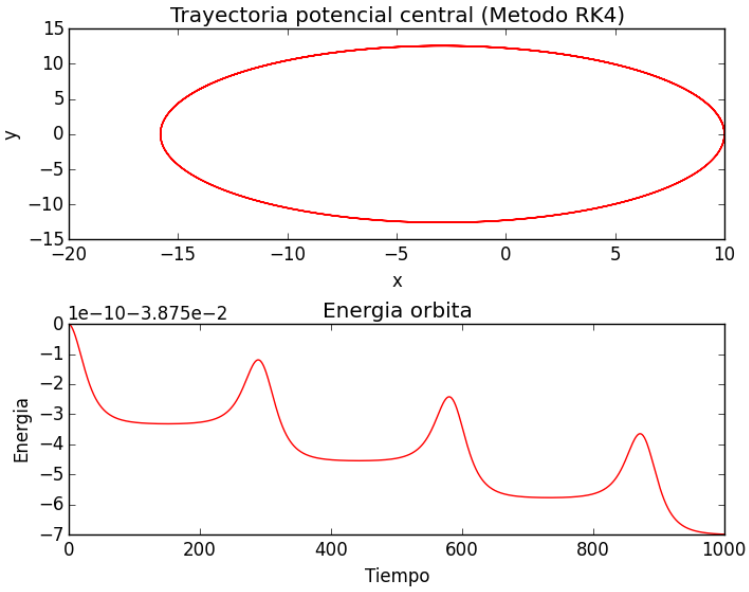

Figure 2: Gráfico método de Runge kutta de orden 4 con un tiempo final de 1000

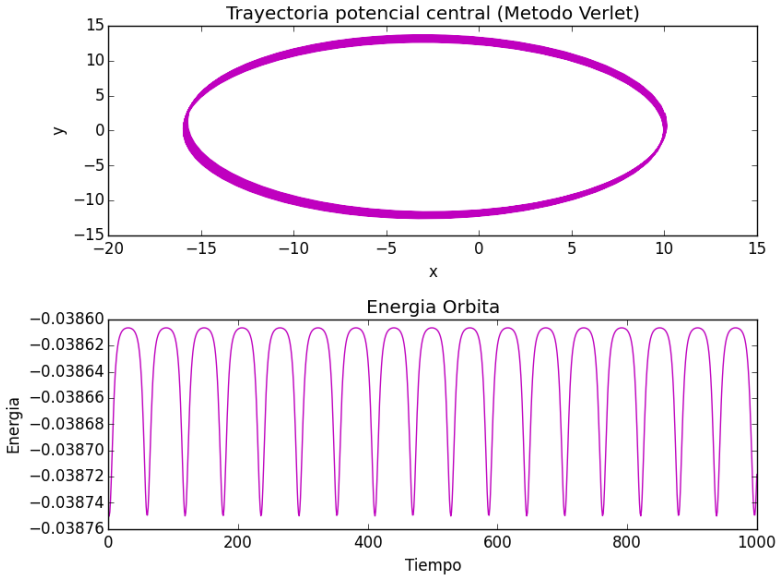

Figure 3: Gráfico método de Verlet con un tiempo final de 5000

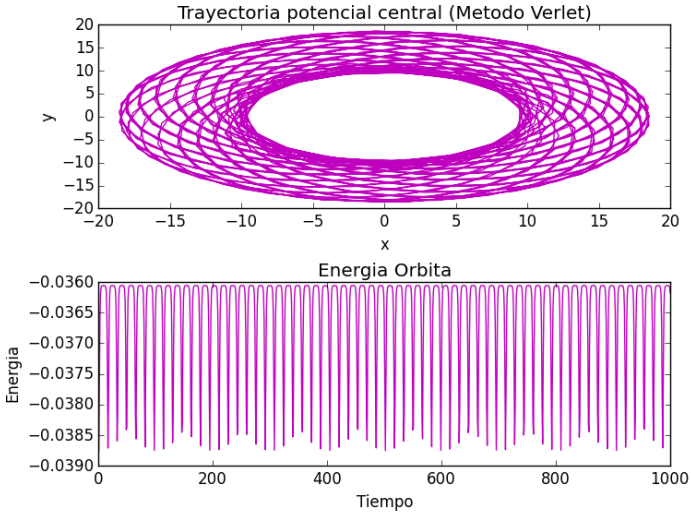

Figure 4: Gráfico hecho con un *tiempo final* = 20000

4 Conclusiones

Se puede concluir que el mejor método para la simulación del movimiento orbital de planetas alrededor del sol, dado un potencial $U(r)$, es el método de Verlet, lo cual se ve reflejado en la figura 3, ya que se puede apreciar la precesión de la orbita. Esto último se ve claramente reflejado en la Figura 4, gráfico que fue hecho con un *tiempo final* de 20.000. Se debe notar que se usaron *tiempos finales* menores en los otros 2 métodos porque a mayores tiempos, la simulación se volvía inestable, lo cual se presentaba de forma de espiral o rectas.
Además, en cuanto a lo aprendido en el tutorial *Program Design*, las ideas planteadas son de alta utilidad al momento de programar, se deben intentar implementar para tener códigos más limpios y eficientes.