TAREA 5

Fecha de entrega: 26/09/2019 23:59 hrs

Problema 1

El péndulo de Kapitza. Considere una vara ideal sin masa de largo $L = 1m$, conectada a una masa m en un extremo, y en el otro extremo a un punto que oscila sinusoidalmente en la dirección vertical con frecuencia w_1 y amplitud $A = L/2$. Se puede demostrar que la ecuación de movimiento para este péndulo es:

$$\frac{d^2\phi(t)}{dt^2} = -w_0^2 sin(\phi) - \frac{A}{L} w_1^2 cos(w_1 t) sin(\phi)$$

donde ϕ es el ángulo que forma la vara contra la vertical y $w_0 = \sqrt{g/L}$. Definiremos $\lambda = \frac{Aw_1}{2w_0 L}$ y estudiaremos el comportamiento del péndulo para $\lambda = 1, 2$ y 3. Considere además las siguientes condiciones iniciales:

1. $\phi(t = 0) = 0.00RRR$ y velocidad angular 0.
2. $\phi(t = 0) = \pi - 0.00RRR$ y velocidad angular 0.

donde RRR son los últimos 3 dígitos de su RUT, antes del dígito verificador.

Integre la ecuación del péndulo de Kapitza utilizando su propia implementación del algoritmo de Runge-Kutta de orden 4. Note que debe realizar la integración para 3 valores distintos de λ y 2 sets de condiciones iniciales, es decir, debe hacerlo 6 veces.

Ahora elija uno de los 6 resultados anteriores –uno que le haya parecido interesante– y realice la integración una vez más utilizando alguna implementación de Runge-Kutta de libre disposición (lo más sencillo es buscar en `scipy.integrate`). Compare este nuevo resultado con el obtenido anteriormente en términos de la precisión de la solución, y del tiempo de ejecución.

Instrucciones importantes.

- Utilice `git` durante el desarrollo de la tarea para mantener un historial de los cambios realizados. La siguiente cheat sheet le puede ser útil. **Esta vez revisaremos el uso apropiado de la herramienta y asignaremos una fracción del puntaje a este ítem.** Realice cambios pequeños y guarde su progreso (a través de *commits*) regularmente. No guarde código que no corre o compila (si lo hace por algún motivo deje un mensaje claro que lo indique). Escriba mensajes claros que permitan hacerse una idea de lo que se agregó y/o cambió de un `commit` al siguiente.

- También revisaremos su uso correcto de `python`. Si define una función relativametne larga o con muchos parámetros, recuerde escribir el *docstring* que describa los parámetros que recibe la función, el output, y el detalle de qué es lo que hace la función. Recuerde que generalmente es mejor usar varias funciones cortas (que hagan una sola cosa bien) que una muy larga (que lo haga todo). Utilice nombres explicativos tanto para las funciones como para las variables de su código. El mejor nombre es aquel que permite entender qué hace la función sin tener que leer su implementación.

- Para `python` existe una guía de estilo sintáctico (PEP8) que entrega un set de reglas simples para crear código ordenado y fácilmente legible por otras personas. Por ejemplo, se recomienda no usar lineas más largas que 79 caracteres. Para esta tarea, su código debe aprobar `pep8`. En esta página puede chequear si su código aprueba `PEP8`. También hay utilidades en la línea de comando que permiten hacer la prueba directamente en sus computadores:

  ```
  > pip install pycodestyle
  > pycodestyle <filename>
  ```

- La tarea se entrega subiendo su trabajo a github. Cuando termine asegúrese de hacer un último `commit` y luego un `push` para subir todo su trabajo a github.

- El informe debe ser entregado en formato `pdf`, este debe ser claro sin información de más ni de menos. **Esto es muy importante, no escriba de más, esto no mejorará su nota sino que al contrario**. Por ejemplo, la presente tarea probablemente no requiere informes de más de 4 páginas en total (esto no es una regla estricta, sólo una referencia útil). Asegúrese de utilizar figuras efectivas y tablas para resumir sus resultados. Revise su ortografía.

- Repartición de puntaje: 45 % implementación y resolución del problema (independiente de la calidad de su código); 40 % calidad del reporte entregado: demuestra comprensión del problema y su solución, claridad del lenguaje, calidad de las figuras utilizadas; 5 % aprueba o no `PEP8` (acá es todo o nada); 5 % uso apropopiado de `git`; 5 % diseño del código: modularidad, uso efectivo de nombres de variables y funciones, docstrings, etc.