INFORME TAREA 3

Benjamín Briceño
RUT:20.653.153-3
Github: @Benbriel

1. Introducción

Esta tarea tiene como objetivo encontrar una solución numérica para la ecuación de movimiento del péndulo de Kapitza; una vara ideal rígida de largo $L = 1$ m conectada a una masa m en un extremo, y en el otro a un oscilador armónico en dirección vertical de frecuencia ω_1 y amplitud $A = L/2$. Además, ϕ es el ángulo formado con la vertical y $\omega_0 = \sqrt{g/L}$ Luego, la ecuación de movimiento del péndulo es

$$\frac{d^2\phi(t)}{dt^2} = -\left(\omega_0^2 + \frac{A}{L}\omega_1^2 \cos(\omega_1 t)\right) \sin\phi \tag{1}$$

Para esto se estudiarán los ángulos iniciales $\phi_0 = \phi(t=0) = 0.00153$ y $\phi_0 = \pi - 0.00153$, ambos casos con velocidad angular $\dot{\phi}(t=0) = 0$. También se analizará el comportamiento de $\phi(t)$ para $\lambda = 1, 2, 3$, donde $\omega_1 = 2\omega_0\lambda L/A$.

Se implementará un algoritmo de integración utilizando el método de Runge-Kutta de orden 4, que ayudará a resolver la ecuación (1) de forma numérica. También se discutirán los resultados obtenidos y las diferencias entre los valores de ϕ para cada valor de λ y ϕ_0. Por último, se utilizará la función `scipy.integrate.solve_ivp` y se comparará con el método implementado en velocidad de ejecución y precisión numérica.

2. Desarrollo

Primero, se expresará la ecuación (1) de forma vectorial, considerando $(\phi(t), \frac{d\phi}{dt})^T$ tal que

$$\begin{pmatrix} \phi \\ \frac{d\phi}{dt} \end{pmatrix}' = \begin{pmatrix} \frac{d\phi}{dt} \\ -\left(\omega_0^2 + \frac{A}{L}\omega_1^2 \cos(\omega_1 t)\right) \sin\phi \end{pmatrix} \tag{2}$$

Donde se ha convertido una ecuación diferencial de segundo orden en dos de primer orden. Este sistema puede ser integrado con un método tradicional sobre cada término del vector por separado. Para esto, se crea la clase de `python RK4Integrate`, que calcula la integral del vector de la ecuación 2 dadas las condiciones de borde, utilizando el método de Runge-Kutta de orden 4. Esto es aplicado para 3 valores distintos de ω_1 y 2 condiciones de borde, obteniendo 6 resultados distintos. Cabe destacar que en este caso como $\omega_1 = 2\omega_0\lambda L/A$ y $A = L/2$, resulta $\omega_1 = 4\omega_0\lambda$, con $\lambda = 1, 2, 3$. Para este método se utilizó un paso adaptativo con `dt_min=1e-5` y `dt_max=1e-2`. Luego, se obtuvo

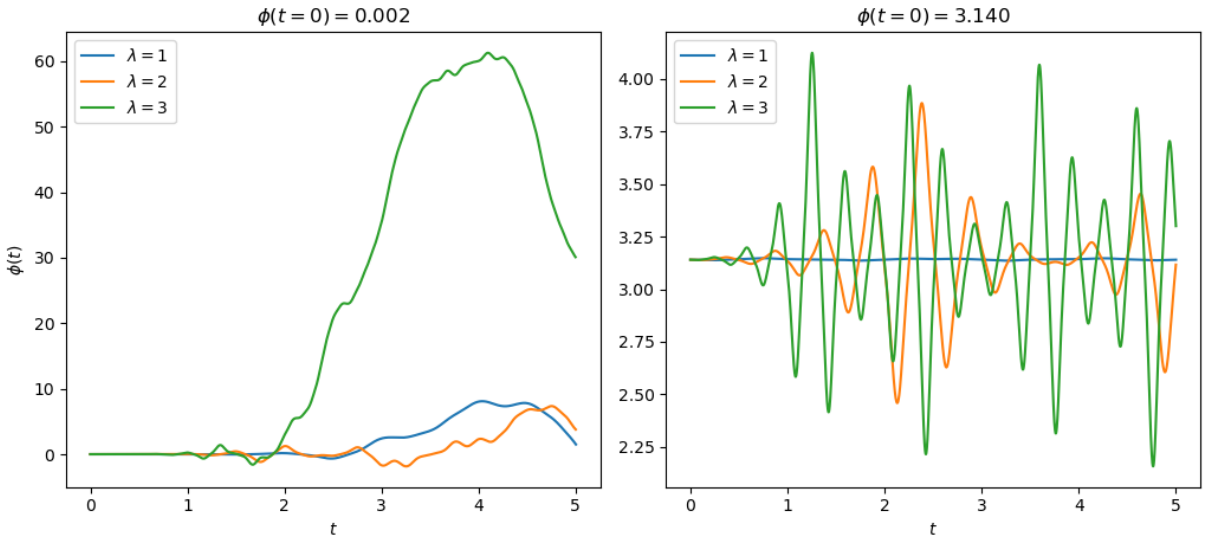

Figura 1: Valores de la función $\phi(t)$ integrados de forma numérica para los distintos valores de λ. A la izquierda, la condición inicial $\phi_0 = 0.002$ se hace más inestable conforme aumenta λ. A la derecha, la condición $\phi_0 \approx 3.14$ oscila en torno al ángulo $\phi = \pi$, aumentando su amplitud y frecuencia si aumenta λ.

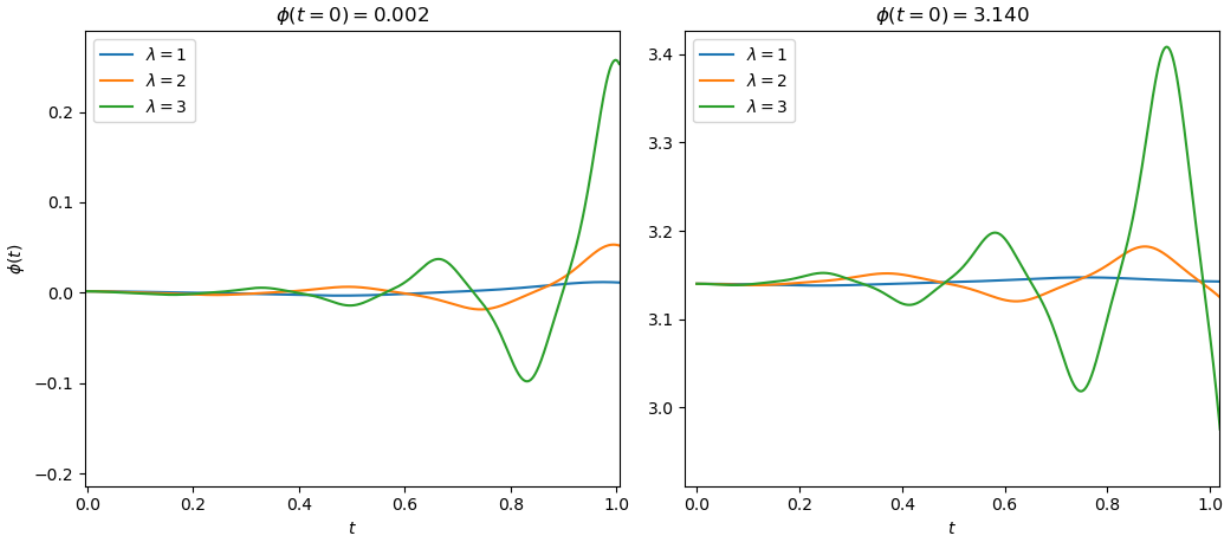

Figura 2: Figura 1 acercada para valores de t entre 0 y 1, previo a las inestabilidades de la condición $\phi_0 = 0.002$.

Los resultados de la figura 1 indican el comportamiento caótico del péndulo según sus condiciones iniciales y valores de w_1. Sin embargo, para valores bajos de w_1, la solución se mantiene en un principio en bajas amplitudes y frecuencias, hasta que para el caso $\phi_0 = 0.002$ el péndulo de desestabiliza. Además, en el lado izquierdo de la figura se pueden ver valores de ϕ mayores que 2π.

Esto se interpreta como el péndulo dando una vuelta en sentido antihorario sobre su eje, causada por la oscilación vertical del punto sobre el que este péndulo oscila, de frecuencia w_1. Si se prueban valores mayores para w_1, pareciera que el caso donde el péndulo gira sobre su eje es una solución estable para el sistema.

En el caso derecho se pueden ver oscilaciones del péndulo en torno a $\phi = \pi$, lo que significaría que es un mínimo en el potencial del péndulo. Aparentemente para $\lambda = 1,2$ las oscilaciones son periódicas, sin embargo, no es posible determinarlo por la figura. Esto pues se desconoce hasta qué tiempo la solución es numéricamente precisa. Por otro lado, se observa que para mayor w_1, la solución oscila a mayores amplitudes y frecuencia, consistente con una mayor energía entregada por el punto oscilatorio, forzando aún más el péndulo.

Por otra parte, el comportamiento del péndulo para un tiempo pequeño, mostrado en la Figura 2, es interesante, pues no presenta un comportamiento caótico. Además, llama la atención la similitud entre ambos lados de la figura, donde difieren únicamente en la amplitud de oscilación. De esta forma es posible observar que, para el caso izquierdo y t pequeño, la solución tiene un comportamiento oscilatorio.

Posteriormente, se importó la función `scipy.integrate.solve_ivp`, que toma una función vectorial de argumentos `(t, y, *args)`, el límite temporal inferior y superior y las condiciones de borde en el t inicial, y entrega un objeto de clase `scipy.integrate._ivp.ivp.OdeResult` con la solución numérica de la integral. Este objeto tiene atributos `self.t` y `self.y`, los cuales contienen arrays con los tiempos donde se evaluó la función y la solución en forma vectorial, respectivamente. Este resolvedor ocupa por defecto el método de Runge-Kutta de orden 5 para calcular un paso, pero asume una precisión de orden 4. Además, se estableció un paso máximo `max_step=1e-2` equivalente al del método implementado, para aumentar la precisión de la solución y poder comparar ambos métodos. Se escogió el caso $\lambda = 3$ y la condición inicial $\phi_0 \approx 3.14$, pues presenta las oscilaciones más caóticas, y es el valor de λ para el que antes ocurre la imprecisión entre ambos métodos sin divergir necesariamente.

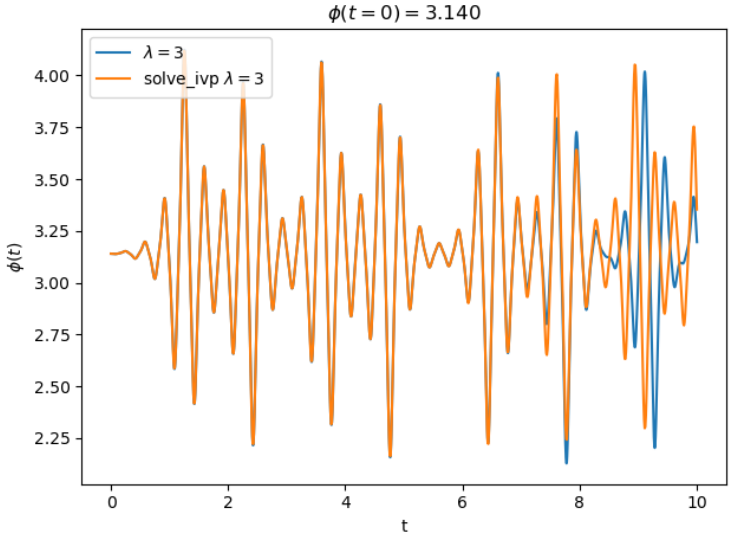

Figura 3: Comparación entre el método `RK4Integrate` y `solve_ivp` para $\phi_0 \approx 3.14$ y $\lambda = 3$. Ambos métodos tienen un error pequeño hasta $t \approx 7.5$ segundos.

Cabe destacar que la Figura 3 calcula las soluciones numéricas hasta $t = 10$, valor que es bastante elevado, y que requiere de un código más preciso o un menor `dt` para que algun método de integración pueda llegar a él con un error pequeño asociado. Por esto, ambas soluciones comienzan a diferenciarse entre sí en un menor t. Sin embargo, para $\lambda = 1, 2$ ambas soluciones son casi idénticas dentro del intervalo, lo que implica un error relativo pequeño. No es posible calcular de forma numérica el error entre el método `RK4Integrate` y `solve_ivp`, pues ambos contienen distintos valores para t en su dominio, y no pueden ser comparados. Sin embargo, se obtiene que el arreglo de t para `solve_ivp` contiene 1002 elementos, mientras que el de `RK4Integrate` contiene 55210. Por otro lado, se compararon los métodos según su tiempo de ejecución para la condición $\phi_0 \approx 3.14$ y $\lambda = 3$, en el intervalo temporal entre 0 y 1.

Método Int.	**Tiempo de ejecución**
RK4Integrate	860 ms ± 10.1 ms
solve_ivp	17.6 ms ± 1.54 ms

Cuadro 1: Tiempos de ejecución del cálculo numérico de ϕ para $t \in (0, 1)$ con las condiciones $\lambda = 3$ y $\phi_0 \approx 3.14$ para ambos métodos. Resultado como media ± desviación estándar de 7 runs del código.

La tabla 1 muestra que el tiempo medio de ejecución de `RK4Integrate` es casi 50 veces mayor que el de `solve_ivp`, a pesar de otorgar un resultado de casi igual precisión para $t < 8$. Esto es de esperar, puesto que `RK4Integrate` obtiene más de 50 veces la cantidad de puntos en t que `solve_ivp`.

3. Discusión y Conclusiones

La resolución numérica de ecuaciones diferenciales por medio de la vectorización y utilización de un algoritmo de integración como Runge-Kutta puede ser muy útil a la hora de visualizar funciones que no presentan una función analítica. Sin embargo, hay que tener cuidado a la hora de escoger la precisión del cálculo y las condiciones iniciales, pues comportamientos caóticos pueden llevar a soluciones totalmente distintas del resultado real.

El péndulo de Kapitza no queda exento de lo anterior, pues presenta comportamientos muy distintos según sus condiciones iniciales y valores de λ. Cerca de $\phi = \pi$ se encuentra un mínimo en el potencial, lo que produce oscilaciones en torno a este valor, resultando en un péndulo invertido, al menos para $\lambda = 1, 2, 3$. Además, cerca de $\phi = 0$ se generan configuraciones caóticas, que dependen fuertemente de λ y ϕ_0.

El método `RK4Integrate` implementado encuentra una solución con 50 veces más puntos que el método `solve_ivp`, lo que produce un tiempo de ejecución casi 50 veces mayor. Esto podría significar una mayor precisión numérica para valores grandes de t, sin embargo, para valores de t más pequeños, podría ser más eficaz utilizar `solve_ivp` u otro método más rápido. También se podría modificar el valor de `max_step`, para obtener una solución de menor error