Tarea 3: Métodos Numéricos para la Ciencia e Ingeniería

Constanza Urzúa Cisterna ; rut 18837487-5

8 de Octubre del 2015

1 Introducción

El oscilador de Van der Pool se define por la siguiente ecuación

$$\frac{d^2x}{dt^2} = -kx - \mu(x^2 - a^2)\frac{dx}{dt}$$

Esta es una ecuación no Lineal de orden 2 que fue usada durante los años 20, se requiere realizar un cambio de variable tal que se llegue a:

$$\frac{d^2y}{ds^2} = -y - \mu^*(y^2 - 1)\frac{dy}{ds}$$

con el fin de resolver la EDO, para luego graficar $y(s)$ y la trayectoria en el espacio y, $\frac{dy}{ds}$.

Luego se debe resolver el sistema de Lorenz el cual es un set de ecuaciones diferenciales ordinarias que se caracteriza por tener resultados caóticos. El sistema de ecuaciones por el cual está conformado es:

- $\frac{dx}{ds} = \sigma(y - x)$
- $\frac{dy}{ds} = x(\rho - z) - y$
- $\frac{dz}{ds} = xy - \beta z$

Resolviendo este sistema de EDOS se debe graficar la solución obteniendo un gráfico en 3D $X(t), Y(t), Z(t)$

2 Procedimiento

2.1 Parte 1

Se debe realizar un cambio de variable a la ecuación del oscilador de Van der pool de tal forma la ecuación dependa sólo de $\mu*$. El cambio de variable elegido es $y(s) = \frac{x}{a}$ y $s = t\sqrt[2]{k}$

$$\frac{dx}{dt} = a\frac{dy}{ds}\frac{ds}{dt} = a\frac{dy}{ds}\sqrt[2]{k}$$
$$\frac{d^2x}{dt^2} = a\sqrt[2]{k}\frac{d^2y}{ds^2}\frac{ds}{dt} = ak\frac{d^2y}{ds^2}$$
$$ak\frac{d^2y}{ds^2} = -kya - \mu(y^2a^2 - a^2)a\frac{dy}{ds}\sqrt[2]{k}$$
$$\frac{d^2y}{ds^2} = -y - \mu^*(y^2 - 1)\frac{dy}{ds}$$
$$\mu* = a^2\frac{\sqrt[2]{k}}{k}$$

El problema del oscilador de Van der pool se soluciona a través de Runge-Kutta de orden 3 en donde se define el problema como:

$$y' = f(x, y)$$
$$\frac{d}{ds}\begin{pmatrix} y \\ \frac{dy}{ds} \end{pmatrix} = \begin{pmatrix} \frac{dy}{ds} \\ -y - \mu^*(y^2 - 1)\frac{dy}{ds} \end{pmatrix} \tag{1}$$

En donde Runge-Kutta de orden 3 se describe a través del siguiente algoritmo:

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 4k_2 + k_3)$$

Para finalmente obtener los gráficos de solución de este péndulo.

2.2 Parte 2

Se debe resolver el sistema de Lorenz a través de Runge-Kutta de orden 4, pero con algoritmo ya implementado, en este caso se utiliza el módulo Scipy.Integrate.

3 Resultados

Los resultados de la EDO fueron calculados para 2 condiciones iniciales diferentes; primero con $y(s) = 0.1$ y $\frac{dy}{ds} = 0$ y luego la segunda condición inicial es $y(s) = 4$ y $\frac{dy}{ds} = 0$, en donde se puede ver claramente el plano de fase del oscilador en las figura 1 y figura 2.

En la figura 3 se puede apreciar claramente el comportamiento caótico del oscilador con diferentes condiciones iniciales en donde $\mu* = 0.$RRR (RRR son los últimos 3 digitos del rut (18.837.487-5))

Los gráficos obtenidos al solucionar la EDO fueron los siguientes:

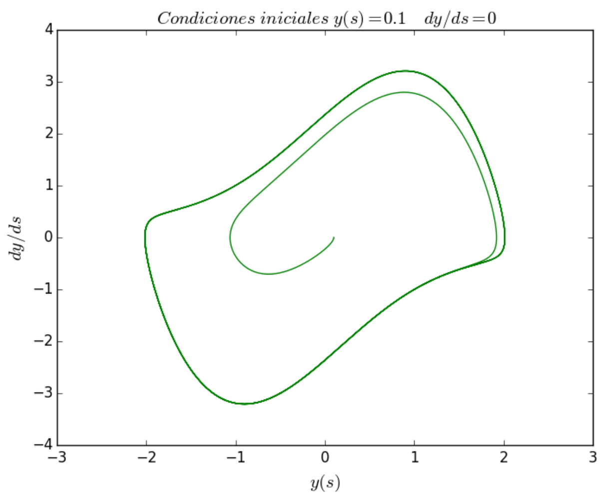

Figure 1: Condiciones iniciales : y(s)=0.1 y $\frac{dy}{ds} = 0$

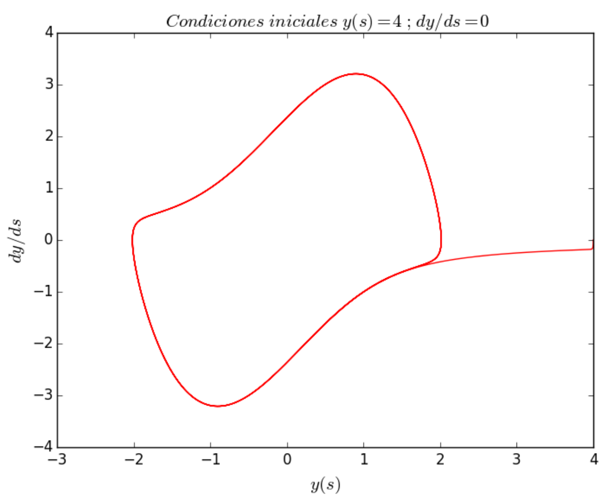

Figure 2: Condiciones iniciales : y(s)=4 y $\frac{dy}{ds} = 0$

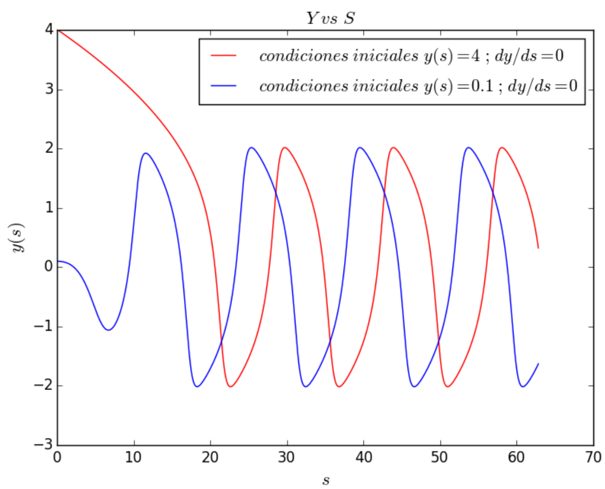

Figure 3: Gráficos y(s) vs s para ambas condiciones iniciales

3.1 Parte 2

EL sistema de ecuaciones de Lorenz al ser resuelto por Runge-Kutta de orden 4 con condiciones iniciales x=1 , y =1, z= 1 y las constantes $\sigma = 10$ $\rho = 28$ $\beta = 8/3$ da la siguiente solución

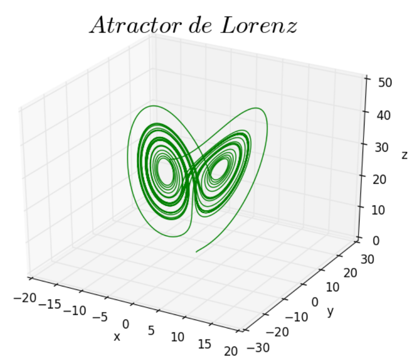

Figure 4: Gráficos y(s) vs s para ambas condiciones iniciales)

4 Conclusiones

Se comprueba el comportamiento caótico del péndulo de Van der Pool, y al mismo tiempo el comportamiento caótico del Sistema de Lorenz que se traduce en el atractor de Lorenz como el comportamiento del clima, cambiar sus condiciones iniciales produce un cambio en la solución del sistema, teniendo como consecuencia una nueva solución totalmente distinta.

En las figuras 1 y 2 se aprecia la convergencia del péndulo a su estado estacionario, es más en y(s)= 2 se comienza el ciclo y en la figura 3 y(s)=2 se inicia la "peridiocidad", lo cual se comprueba con lo esperado.

El método Runge-Kutta es un buen método para el cálculo de EDO, debido a su convergencia, y además por ser un método ya implementado a en la librerías de python