Elementos de Cálculo Numérico / Cálculo Numérico

Primer Cuatrimestre 2021

Cuarto ejercicio computacional
19/04/21 al Lunes 26/04/21
Recuerde subir el archivo en formato ejercicioX_NOMBREAPELLIDO.py
Recuerde al hacer consultas enviar su código

Podrá encontrar un ejemplo de resolución de un sistema de ecuaciones ordinarias en varias dimensiones en los videos de la parte computacional. Para resolver este ejercicio, también puede apoyarse en el ejercicio 4 del cuatrimestre anterior de ser necesario.

Esta semana simularemos el sistema de Lorentz. Dados parámetros $a, b, c > 0$ consideramos el sistema:

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x)\\ \frac{dy}{dt} &= x(\rho - z) - y\\ \frac{dz}{dt} &= xy - \beta z\end{aligned}$$

donde σ es llamado el número de Prandtl y ρ se llama el número de Rayleigh.

Consideraremos el caso $\sigma = 10, \beta = \frac{8}{3}, \rho = 28$, con condiciones iniciales dadas por $(x, y, z) = (1, 1, 1)$, y nos interesa calcular la solución hasta $t = 40$.

Para simular este modelo, emplearemos Runge-Kutta de orden 4, y consideraremos un vector $V = (x, y)$ de forma que

$$V' = F(V) \tag{1}$$

Considere los siguientes elementos para construir el modelo:

A Inicialización de valores iniciales:

```
rho = 28.0
sigma = 10.0
beta = 8.0 / 3.0
estado_0 = [1.0, 1.0, 1.0]
t = np.arange(0.0, 40.0, 0.01)
```

B Una función F que implementa el lado derecho del sistema

```
def F(estado, t):
    x, y, z = estado
    rhs = sigma * (y - x), x * (rho - z) - y, x * y - beta * z
    return np.array(rhs)
```

B Considere la siguiente función que implementa la integración de Euler

```
def integra_euler(f, estado_0, t0, tf, dt):
    t = np.arange(t0, tf, dt)
    estados = [estado_0]
    for tn in t:
        vn = np.array(estados[-1])
        paso = vn + dt * F(vn, tn)
        estados.append( paso )
    return np.array(estados)
```

C Modifique la función anterior para introducir un paso del método de Runge-Kutta definido mediante una función

```
def paso_runge_kutta(##):
    k1 = # COMPLETAR
    k2 = # COMPLETAR
    k3 = # COMPLETAR
    k4 = # COMPLETAR
    ### COMPLETAR
    return(##)
```

Realice la simulación considerando los valores propuestos. Genere un gráfico de la trayectoria $x(t), y(t), z(t)$ en tres dimensiones. Para esto último, puede utilizar la siguiente ayuda:

```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(estados[:, 0], estados[:, 1], estados[:, 2])
plt.show()
```

Verifique que los errores numéricos son pequeños, tomando varios pasos temporales suficientemente pequeños (encuentre valores razonables) de modo que el gráfico de la solución numérica no se altere perceptiblemente. Para corroborar esto, genere dos gráficos de trayectorias superpuestas, una en color azul y la otra naranja.