tarea11

November 22, 2020

1 Tarea 11

Tarea 11 de *Benjamín Rivera* para el curso de **Métodos Numéricos** impartido por *Joaquín Peña Acevedo*.

Fecha limite de entrega **…**.

[1]:
```python
import sys
import seaborn as sns
import scipy

import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import solve_triangular # Para backward y forward substitution

NOTEBOOK = True
```

1.1 Ejercicio 1

Programar el **metodo de Runge-Kutta** de orden 2 para resolver un problemade valor inicial: y(a) =y0;

$$(PVI)\begin{cases} y' = f(x,y) & x \in (a,b] \\ y(a) = y_0 \end{cases}$$

1.1.1 Solucion numerica del PVI

Escribir el codigo de la funcion que calcula la solucion numerica del PVI con **algoritmo de RungeKutta** de segundo orden. La funcion debe recibir como argumentos: - un arreglo con los puntos de la particion uniforme $a = x_0 < x_1 < \cdots < x_n = n$, - un valor inicial $y0$, - el numero de subdivisiones n del intervalo $[a, b]$, y - el apuntador la funcion $f(x, y)$.

Crear el arreglo para almacenar los valores $y_0, y_1, \ldots, y_n$. Hacer $y_0 = y0$, $h = x_1 - x_0$ y para cada $i = 0, 1, \ldots, n-1$ calcular

$$\begin{aligned} K_1 &= f(x_i, y_i), \\ K_2 &= f(x_i + h, yi_+ hK_1), \\ y_{i+1} &= y_i + 0.5h(K_1 + K_2) \end{aligned}$$

La funci´on debe devolver el arreglo con los valores y_i.

[2]:
```
def runge_kutta(xs, y0, n, f):
    """ Funcion que implementa el algoritmo deRunge-Kutta. """

    ys = np.zeros(n+1)
    ys[0] = y0
    h = xs[1] - xs[0]

    for i in range(n):
        K1 = f(xs[i], ys[i])
        K2 = f(xs[i] + h, ys[i] + h*K1)
        ys[i+1] = ys[i] + 0.5*h*(K1 + K2)

    return ys
```

1.1.2 Linea de comandos

Escriba el programa que reciba desde la lınea de comandos el valor n que define el numero de divisiones del intervalo de solucion $[a, b]$. Programe la funcion f que corresponde al problema de valor inicial:

$$(PVS) \begin{cases} y' = 4x^2 - 6x + \frac{y}{x} & x \in (1, 6] \\ y(1) = 4 \end{cases}$$

La solucion analıtica de este problema es $y(x) = -6x2 + 8x + 2x3$. Genere una particion del intervalo $[1, 6]$ con $x_k = 1 + hk$, para $k = 0, 1, \ldots, n$, con $h = (6–1)/n$, y calcule los valores de la solucion numerica $y_0, y_1, y_2, \ldots, y_n$ del PVI usando la funcion del inciso anterior.

Programe la funcion que evalua la solucion analıtica $y(x)$ y haga que el programa calcule el maximo del error relativo:

$$E_{max} = \max_{k=1,\ldots,n} \frac{|y_k - y(x_k)|}{|y(x_k)|}$$

Haga que el programa imprima los valores E_{max}

[3]:
```
# definimos f
def f(x, y):
    return 4*x**2-6*x+(y/x)
def y(x):
    return -6*x**2 + 8*x + 2*x**3
```

[4]:
```
def gui(n, f=f, y=y, a=1, b=6, y0=4):
    """ Funcion para llamar a la funcion del ejercicio anterior con un
    caso especifico
    """

    part = np.linspace(a, b, n+1, True)
```

```
    ys = runge_kutta(part, y0, n, f)

    xs = part
    e_max = max([abs(ys[k] - y(xs[k]))
                 / abs(y(xs[k]))
                 for k in range(1, n+1)])

    print(e_max)
```

1.1.3 Prueba

Prueba el programa con $n = 10$ y $n = 100$, y escriba un comentario sobre los resultados obtenidos.

[5]:
```
gui(10)
gui(50)
gui(100)
```

```
0.015873015873015987
0.0005800005553969491
0.0001404937293847703
```

Además de los solicitados, también probe el metodo para $n = 50$. Me sorprende lo rapido que decience el error con *relativamente* pocas iteraciones

[ ]:
```
```