– Práctica 2 –
Métodos RK encajados

1. La función Python `rk23` implementa el método encajado RK2(3) dado por el tablero para resolver

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & 0 & 1 & 0 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

el problema :

$$(\mathrm{P}) \qquad \begin{cases} y' = -y + e^{-t}\cos(t), & t \in [0,30], \\ y(0) = 0, \end{cases}$$

cuya solución exacta es

$$y(t) = e^{-t}\sin(t).$$

```
# Resolucion del problema de valor inicial
# y'=f(t,y), y(t0)=y0,
# mediante el metodo RK2(3).

from pylab import *
from time import perf_counter

def f(t, y):
    """Funcion que define la ecuacion diferencial"""
    return -y+exp(-t)*cos(t)

def exacta(t):
    """Solucion exacta del problema de valor inicial"""
    return exp(-t)*sin(t)

def rk23(a, b, fun, y0, h0, tol):
    """Implementacion del metodo encajado RK2(3)
    en el intervalo [a, b] con condicion inicial y0,
    paso inicial h0 y tolerancia tol"""


    hmin = (b-a)*1.e-5 # paso de malla minimo
    hmax = (b-a)/10. # paso de malla maximo


    # coeficientes RK
    q = 3 # numero de etapas
    p = 2 # orden del metodo menos preciso
    A = zeros([q, q])
    A[1, 0] = 1./2.
    A[2, 0] = -1.
    A[2, 1] = 2.

    B = zeros(q)
    B[1] = 1.
```

```
    BB = zeros(q)
    BB[0] = 1./6.
    BB[1] = 2./3.
    BB[2] = 1./6.

    C = zeros(q)
    for i in range(q):
        C[i] = sum(A[i,:])

    # inicializacion de variables
    t = array([a]) # nodos
    y = array([y0]) # soluciones
    h = array([h0]) # pasos de malla
    K = zeros(3)
    k = 0 # contador de iteraciones

    while (t[k] < b):
        h[k] = min(h[k], b-t[k]) # ajuste del ultimo paso de malla
        for i in range(q):
            K[i] = fun(t[k]+C[i]*h[k], y[k]+h[k]*sum(A[i,:]*K))
        incrlow = sum(B*K) # metodo de orden 2
        incrhigh = sum(BB*K) # metodo de orden 3
        error = h[k]*(incrhigh-incrlow) # estimacion del error
        y = append(y, y[k]+h[k]*incrlow) # y_(k+1)
        t = append(t, t[k]+h[k]); # t_(k+1)
        hnew = 0.9*h[k]*abs(tol/error)**(1./(p+1)) # h_(k+1)
        hnew = min(max(hnew,hmin),hmax) # hmin <= h_(k+1) <= hmax
        h = append(h, hnew)
        k += 1


    return (t, y, h)



# Datos del problema
a = 0. # extremo inferior del intervalo
b = 30. # extremo superior del intervalo
y0 = 0. # condicion inicial
h0 = 0.1 #paso inicial
tol = 1.e-6 #tolerancia


tini = perf_counter()
(t, y, h) = rk23(a, b, f, y0, h0, tol) # llamada al metodo RK2(3)
tfin = perf_counter()


# calculo de la solucion exacta
te = linspace(a,b,200)
ye = exacta(te)

# Dibujamos las soluciones
subplot(211)
plot(t, y, 'bo-') # dibuja la solucion aproximada
plot(te, ye, 'r') # dibuja la solucion exacta
xlabel('t')
ylabel('y')
legend(['RK2(3)', 'exacta'])
grid(True)
subplot(212)
plot(t[:-1],h[:-1],'*-')
# se excluye el ultimo valor de h porque no se usa para avanzar
xlabel('t')
```

```
ylabel('h')
legend(['pasos usados'])
# Calculo del error cometido
error = max(abs(y-exacta(t)))
hn = min(h[:-2]) # minimo valor utilizado del paso de malla
hm = max(h[:-2]) # maximo valor utilizado del paso de malla
# se elimina el ultimo h calculado porque no se usa y
# el penultimo porque se ha ajustado para terminar en b

# Resultados
print('-----')
print("Error: " + str(error))
print("No. de iteraciones: " + str(len(y)))
print('Tiempo CPU: ' + str(tfin-tini))
print("Paso de malla minimo: " + str(hn))
print("Paso de malla maximo: " + str(hm))
print(sum(h))
print('-----')
```

Use el programa para resolver el problema (P) con diferentes valores de la tolerancia `tol`. Observe en cada caso el número de iteraciones, el máximo y el mínimo paso h_k usado, así como la evolución del paso de malla.

2. Modifique la función `rk23` para implementar el método RK4(5) de Fehlberg dado por el Cuadro 1. Repita el Ejercicio 1 usando RK4(5) y compare los resultados obtenidos con uno y otro método para una misma tolerancia.

0	0	0	0	0	0	0
$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$	0	0	0	0
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$	0	0	0
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$	0	0
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	0
orden 4	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
orden 5	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Cuadro 1: Tablero del método encajada RK4(5)

3. Modifique el programa correspondiente el método RK2(3) a fin de poderlo aplicar a la resolución de sistemas. (**Observación:** en el caso de un sistema tanto el error de discretización local ε_k como su estimador $\widetilde{\varepsilon}_k$ son vectores: sustituya $|\widetilde{\varepsilon}_k|$ por $||\widetilde{\varepsilon}_k||_\infty$ en la expresión del nuevo paso h_{k+1}.) Aplique el programa a la resolución de los problemas 4, 5, 6(a), 7(b) y 7(c) de la Práctica 1 con tolerancia 10^{-4} y 10^{-6}. Además de las gráficas de las soluciones, dibuje en cada caso otra que muestre la evolución del paso de malla.

4. Idem para RK4(5).