Códigos grupo 1

Pedro Bonilla Nadal   Johanna Capote Robayna
Guillermo Galindo Ortuño

Programa 1

```
"""
Datos de entrada
"""

# Intervalo [a,b]
a = 0.0
b = 1.0

# Número de divisiones del intervalo
n = 10

# Número de pasos
k = 3

# Valor inicial y0 = y(a)
y0 = 1

# Función f de dos variables
f = lambda t,y: -y+t+1


h = (b-a) / n
t = [a + j*h for j in range(n+1)]

'''
Método de Euler
'''
def euler():
    u = []
    for j in range(n+1):
        if j == 0:
            u.append(y0)
```

```
        else:
            u.append(u[j-1]+h*f(t[j-1],u[j-1]))
    return u

result = euler()

print("\nEl valor aproximado es:",result)
```

Programa 3

```
"""
Datos de entrada
"""

# Intervalo [a,b]
a = 0.0
b = 1.0

# Número de divisiones del intervalo
n = 5

# Número de pasos
k = 3

# Valor inicial y0 = y(a)
y0 = 1

# Función f de dos variables
f = lambda t,y: -y+t+1

h = (b-a) / n
t = [a + j*h for j in range(n+1)]

'''
Método de Euler Mejorado
'''
def eulerMejorado():
    u = []
    for j in range(n+1):
        if j == 0:
            u.append(y0)
        else:
            u.append(
                u[j-1]+
```

```
                h*f( t[j-1] + h/2,
                     u[j-1] + h*f(t[j-1],u[j-1]) )
            )
    return u

result = eulerMejorado()

print("\nEl valor aproximado es:",result)
```

Programa 5

```
"""
Datos de entrada
"""

# Intervalo [a,b]
a = 0.0
b = 1.0

# Número de divisiones del intervalo
n = 5

# Número de pasos
k = 3

# Valor inicial y0 = y(a)
y0 = 1

# Función f de dos variables
f = lambda t,y: -y+t+1

h = (b-a) / n
t = [a + j*h for j in range(n+1)]

'''
Método de Runge Kutta
'''
def rungeKutta():
    u = []
    for j in range(n+1):
        if j == 0:
            u.append(y0)
        else:
            K1 = f(t[j-1]      , u[j-1]         )
```

```
            K2 = f(t[j-1] + h/2, u[j-1] + h/2*K1)
            K3 = f(t[j-1] + h/2, u[j-1] + h/2*K2)
            K4 = f(t[j-1] + h  , u[j-1] + h  *K3)

            u.append( u[j-1] + h/6*( K1 + 2*K2 + 2*K3 + K4 )             )
    return u

result = rungeKutta()

print("\nEl valor aproximado es:",result)
```

Programa 9

```
from scipy.integrate import quad
import numpy as np
import math as mth

"""
Datos de entrada
"""

# Intervalo [a,b]
a = 0.0
b = 1.0

# Número de divisiones del intervalo
n = 10

# Número de pasos
k = 3

# Valor inicial y0 = y(a)
y0 = 1

# Función f de dos variables
def f(t, y):
    return -y + t + 1.0

# Función y solución exacta (si existe)
def y(t):
    return exp(-t) + t

'''
Método de Euler
```

```
'''
# Intervalo (a,b)
# n número de nodos
# f f(t, y(t)) = y'
# y0 valor de y en t_0

def eulerMejorado(a, b, n, f, y0):
    h = (b-a) / n
    t = [a + j*h for j in range(n+1)]

    u = []
    for j in range(n+1):
        if j == 0:
            u.append(y0)
        else:
            u.append(
                u[j-1]+
                h*f( t[j-1] + h/2,
                     u[j-1] + h*f(t[j-1],u[j-1]) )
            )
    return u

'''
Método de Adams Moulton
'''

def func(x, i):
    arr = []
    for j in range(i+1):
        arr.append(x+j-1)
    for j in range(i+2, k+1):
        arr.append(x+j-1)

    return np.prod(np.array(arr))

# (a,b) intervalo
# n número de nodos
# k número de pasos
# f f(t, y(t)) = y'
# u = vector con las primeras j aproximaciones u_0, ..., u_j
def adamsMoulton(a, b, k, j, f, u):
    h = (b-a) / n
    t = [a + j*h for j in range(n+1)]

    # Coeficientes b_j
    b = []
```

```
    for i in range(-1,k):
        b[i] = (-1)**(i+1)/(mth.factorial(i+1) * mth.factorial(k-i-1))*
               quad(func, 0, 1, args=(i,) )[0]

    sumatory = sum ( b[i]*f(t[j-i],u[j-i]) for i in range(0, k) )

    # Función que define a la ecuación implícita (si g(x) = 0 -> x = u_j+1)
    g = lambda x: x - u[j] - h*(sumatory + b[-1]*f(t[j+1], x))

    # Derivada de g aproximada
    dg = lambda x: (g(x+h) - g(x))/(2.0*1e-7)

    u_k = optimize.newton(g, u[k-1], dg)
    return u_k

'''
Programa principal
'''

u = eulerMejorado(a, b, k, f, y0)

for j in range(k, n+1) :
    u.append(adamsMoulton(a, b, k, j, f, u))

print("\El resultado es: ", u)
```