Códigos grupo 1

Pedro Bonilla Johanna Capote Guillermo Galindo

Programa 2

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

def l(x, i, n, nodos):
    pol = []

    for j in range(i):
        pol.append((x - nodos[j])/(nodos[i]-nodos[j]))

    for j in range(i+1, n+1):
        pol.append((x - nodos[j])/(nodos[i]-nodos[j]))

    return np.prod(np.array(pol))


def newton_cotes( f, a, b, n ):
    nodos = []
    h = (b-a)/n
    for i in range(n+1):
        nodos.append(a + i*h)

    coef = []
    for i in range(n+1):
        coef.append(quad(l, a, b, args=(i, n, nodos))[0])

    valores = []
    for i in range(n+1):
        valores.append(coef[i]*f(nodos[i]))

    return  sum(valores)

#Datos
f = lambda x: np.log(x)
```

```
a = 1
b = 2
n = 5

result = newton_cotes(f, a,b,n)

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

Programa 3

```
import scipy as sp
import numpy as np
from numpy.polynomial import polynomial as P


def intTrapecio( f, a, b, n ):
     h = (b-a)/n
     x=[ a+h*i for i in range(n+1)]

     if(n<1):
          resultado=0

     elif(n==1):
          resultado = (f(b) + f(a))*(b-a)/2

     else:
          resultado = ( h *( ( f(x[0]) + f(x[n]) )/2
            + sum([f(x[i+1]) for i in range(n-1)]) ) )

     return resultado

f= lambda x: np.log(x)
a=1
b=2
n=200

resultado = intTrapecio( f, a, b, n )
print("\nEl valor aproximado es:",resultado)
```

Programa 5

```
import scipy as sp
import numpy as np
from numpy.polynomial import polynomial as P

def intTrapecio( f, a, b, n ):
     if(n==1):
          resultado = (f(b) + f(a))*(b-a)/2
     else:
          h = (b-a)/n
          x=[ a+h*i for i in range(n+1)]
          resultado = ( h *( ( f(x[0]) + f(x[n]) )/2
            + sum([f(x[i+1]) for i in range(n-1)]) ) )

     return resultado

#Rk = R_{k,j-1} RK1 = R_{k-1,j-1}
def rombergParcial( Rk, Rk1, j ):
     return Rk + (1/(4**j-1))*(Rk - Rk1)

def romberg(f, a, b, k):
     R=[]
     R.append( intTrapecio( f, a, b, 1 ) )
     for i in range(k):
          R.append( intTrapecio( f, a, b, 2**(i+1) ) )

     for i in range(k):
          for j in range(k-i):
               R[j]=rombergParcial( R[j+1], R[j], j+1 )

     return R[0]

f = lambda x: np.log(x)
a=1
b=2
n=10

resultado = romberg( f, a, b, n )
print("\nEl valor aproximado es:",resultado)
```