Código en python de los algoritmos

Javier Sáez, Laura Gómez, Daniel Pozo, Luis Ortega

1. Programa 1- Newton Cotes

```
from sympy import Symbol
from sympy import integrate
from decimal import *
"""
    Estos seran los datos de entrada de nuestro programa.
        f(x) representa la funcion que queremos integrar
        [a,b] es el intervalo donde queremos obtener el valor de la
            integral
        n+1 son los nodos que utilizaremos.
"""
x=Symbol('x')
f = 1/(1+x**2)

a=-4
b=4
n=10

"""
    A partir de aqui, comienza el codigo de nuestro programa.
    Primero de todo, calcularemos el valor de h, los nodos que
        utilizaremos,
    asi como la base de Lagrange o pesos que utilizaremos para
        nuestro calculo
    de la integral.
"""

h=Decimal((b-a))/Decimal(n)

nodos=[]
for i in range(n+1):
    nodos.append(a+i*h)
```

```
pesos=[]
for i in range(n+1):
    aux=1
    for j in range(n+1):
        if(j != i):
            aux=aux*(x-nodos[j])/(nodos[i]-nodos[j])
            """print(aux)"""
    pesos.append(aux)

"""print(pesos[0])
print(integrate(pesos[0],(x,a,b)))"""

"""
    Una vez que tenemos calculados los pesos, en funcion de x,
        calcularemos el
    valor de una aproximacion de la integral.
"""
integral=0
for i in range(n+1):
    integral=integral+integrate(pesos[i],(x,a,b))*f.subs(x,nodos[i])

print(integral)
```

2. Programa 2- Newton compuesto

```
from sympy import Symbol
from sympy import integrate
from decimal import *
"""
    Estos seran los datos de entrada de nuestro programa.
        f(x) representa la funcion que queremos integrar
        [a,b] es el intervalo donde queremos obtener el valor de la
            integral
        m son los intervalos que utilizaremos.
"""
x=Symbol('x')
f = 1/(1+x**2)

a=-4
b=4
m=10
```

```
"""
    A partir de aqui, comienza el codigo de nuestro programa.
    Primero de todo, calcularemos el valor de h y los nodos que
        utilizaremos.
"""
n=2*m

h=Decimal((b-a))/Decimal(n)

nodos=[]
for i in range(n+1):
    nodos.append(a+i*h)
    """print(nodos[i])"""

"""
    A continuacion, calcularemos el valor de la integral.
"""
suma1=0
suma2=0
for i in range(1,m+1):
    suma1=suma1+4*f.subs(x,nodos[2*i-1])
    if(i>=2):
        suma2=suma2+2*f.subs(x,nodos[2*i-2])
"""print("\n")
print(f.subs(x,nodos[0]))
print(suma1)
print(suma2)
print(f.subs(x,nodos[n]))

print("\n")
print(h)
print(h/3)
print(f.subs(x,nodos[0])+suma1+suma2+f.subs(x,nodos[n]))"""
integral=(h/3)*(f.subs(x,nodos[0])+suma1+suma2+f.subs(x,nodos[n]))

print(integral)
```

3. Programa 3- Aproximación de Romberg

```
   #Librerias
import numpy as num
import scipy as sci
```

```
from numpy.polynomial import polynomial as pol

def rkj(f,a,b,k,j):
    if(j == 0):
        h = (b-a)/(2**k)
        parcial = 0
        for i in range (2**k - 1):
            parcial = parcial + f(a+i*h)

        res = (h/2)*(f(a) + 2*parcial +f(b))

    else:
        res = rkj(f,a,b,k,j-1) + (1/(4**j-1))*(rkj(f,a,b,k,j-1) -
            rkj(f,a,b,k-1,j-1))

    return res;



#k = n
def romberg(f,a,b,n):
    aprox = rkj(f,a,b,n,n)
    return aprox



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

res = romberg(f,a,b,n)
print("\n El valor de la aproximacion por el metodo de Romberg
    es:", res)
```