UNIVERSIDAD DON BOSCO

Aplicación de Métodos Numéricos

CICLO 01-2024

GUIA DE EJERCICIOS 3 – PARTE 1

INTEGRANTES:

MIRANDA RAMIREZ, RODRIGO ALEXANDER MR181415

VILLALTA, RIGOBERTO ALCIDES VV00329

Emplee la extrapolación de Richardson para aproximar *g'(1.65)* mediante $N_7(h)$ si se sabe que: $g(x) = 7ln^2(x) + 3tan(5x)$. Utilice $h=1/80$. Además, obtenga el valor exacto y el error en la aproximación

Solución

Para encontrar la solución necesitamos la siguiente matriz:

$N_1(h)$	$N_2(h)$	$N_3(h)$	$N_4(h)$	$N_5(h)$	N6(h)	N7(h)
N_1h/2)	N_2(h/2)	N_3(h/2)	N_4(h/2)	N_5(h/2)	N6(h/2)	
N_1h/4)	N_2(h/4)	N_3(h/4)	N_4(h/4)	N_5(h/4)		
N_1h/8)	N_2(h/8)	N_3(h/8)	N_4(h/8)			
N_1h/16)	N_2(h/16)	N_3(h/16)				
N_1h/32)	N_2(h/32)					
N_1h/64)						

En donde encontramos nueestros valores iniciales (primera columna con):

$$N_1(h) = \frac{g(c+h) - g(c-h)}{2h}$$

$$\dots$$

$$N_7(h) = \frac{g(c+h/64) - g(c-h/64)}{2(h/64)}$$

In [1]:
```
from sympy import *
from numpy import zeros, float64

x = symbols("x")
g = 7* (ln(x))**2 + 3 * tan(5*x)
c = 1.65
h = 1/80
```

```
# Aquí declaro mi n para ocuparla en el resto del ejercicio
n=7

N = zeros((n, n), dtype=float64)

#Imprimo la columna y verifico mi matriz
print(N)
```

```
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]
```

In [2]:

```
# Ahora a tráves de un bucle for lleno mi primera columna

for i in range(n):
    N[i][0] = (g.subs(x, c+h) - g.subs(x, c-h)) / (2*h)
    h = h/2

# Reinicio H
h= 1/80
# Verifico N
print(N)
```

```
[[107.4989198    0.           0.           0.           0.
    0.           0.        ]
 [105.65409163   0.           0.           0.           0.
    0.           0.        ]
 [105.20366512   0.           0.           0.           0.
    0.           0.        ]
 [105.09171575   0.           0.           0.           0.
    0.           0.        ]
 [105.06376923   0.           0.           0.           0.
    0.           0.        ]
 [105.05678515   0.           0.           0.           0.
    0.           0.        ]
 [105.05503929   0.           0.           0.           0.
    0.           0.        ]]
```

In [3]:
```
# Verifico si los valores son aproximados al "valor exacto"
# Ya que de no ser así hay algún error en las fórmulas

valor_exacto = diff(g).subs(x, 1.65)
print(valor_exacto)
```

```
105.054457349555
```

In [19]:
```
# Para llenar el resto de la matriz, realizo un bucle for anidado
# para todo el resto de elementos

for j in range(1, n):
    for i in range(n-j):
        # La j es diferente a la fórmual por la forma de indexar de Python
        N[i][j] = (4**(j) * N[i+1][j-1] - N[i][j-1]) / (4**(j) - 1)

#imprimo de forma que se vea bien la matriz
for i in N:
    for j in i:
        print(str(j).ljust(18), end="  ")
    print("\n")
```

```
107.49891980447075  105.03914891065487  105.05448121735434  105.0544573402626   105.05445734955771  105.05445734955808  105.05445734955472

105.65409163410884  105.05352294818562  105.05445771334216  105.0544573495214   105.05445734955808  105.05445734955474  0.0

105.20366511966643  105.05439929051988  105.0544573552061   105.05445734955794  105.05445734955474  0.0                 0.0

105.09171574780652  105.05445372616322  105.05445734964619  105.05445734955475  0.0                 0.0                 0.0

105.06376923157404  105.05445712317851  105.05445734955619  0.0                 0.0                 0.0                 0.0

105.05678515027739  105.05445733540758  0.0                 0.0                 0.0                 0.0                 0.0

105.05503928912503  0.0                 0.0                 0.0                 0.0                 0.0                 0.0
```

In [11]:
```
# Valores aproximado, exacto y error

valor_aproximado = N[0][n-1]
print("El valor aproximado es: ", valor_aproximado)
print("El valor exacto es:", valor_exacto)
```

```
error = abs(valor_aproximado - valor_exacto)
print("El error es de: ", error)
```

```
El valor aproximado es:  105.05445734955472
El valor exacto es: 105.054457349555
El error es de:  6.82121026329696e-13
```

4. En un circuito con un voltaje $V(t)$ y una inductancia L, la primera ley de Kirchhoff nos da la siguiente relación:

$$V(t) = l\frac{di}{dt} + Ri$$

Donde *R* es la resistencia del circuito e *i* es la corriente. Suponga que medimos la corriente con varios valores de *t* y obtenemos:

t	1.00	1.01	1.02	1.03	1.04
i	3.10	3.12	3.14	3.18	3.34

Donde *t* se mide en segundos, *i* se d en amperios, la inductancia *L* es una constante de 0.98 henrios y la resistencia es de 0.142 ohmios. Aproxime el voltaje *V(t)* cuando t= 1.00 y t=1.02

NOTA: Debe ocupar todos los datos de la tabla.

Solución

Para este caso ocuparemos las fórmulas progresiva y centrada de 5 puntos para el cálculo de la derivada numérica evaluada en los puntos dados.

El dato buscado es el voltaje en función del tiempo, pero la incognita es la derivada de la corriente respecto al tiempo, para ello ocupamos de la siguiente forma los puntos:

Para V(t) = 1.00

$$\frac{di}{dt}(1.00) = \frac{-25f(c) + 48f(c+h) - 36f(c+2h) + 16f(c+3h) - 3f(c+4h)}{12h}$$

En donde:

f(c) = 3.10

f(c + h) = 3.12

f(c + 2h) = 3.14

f(c + 3h)= 3.18

f(c + 4h) = 3.34

Ahora podemos sustituir en el programa

In [1]:
```
# Para t=1.00

# 1.01 - 1.00
h = 0.01
```

```
l = 0.98
R = 0.142

f_prima_de_uno = (-25*3.10 + 48*3.12 - 36*3.14 + 16*3.18 - 3*3.34) / (12*h)
print("El valor aproximado de f'(1.00) es: ", f_prima_de_uno)
```

```
El valor aproximado de f'(1.00) es:  0.6666666666665637
```

In [2]:
```
#Calculamos el valor del voltaje aproximado

v_de_uno = l * 0.6666666666665637 + R*3.10
print("V(1.00) es aproximadamente:", v_de_uno)
```

```
V(1.00) es aproximadamente: 1.0935333333332324
```

Ahora calculamos el f'(1.02) con la fórmula centrada de 5 puntos

$$f'(1.02) = \frac{f(c-2h) - 8f(c-h) + 8f(c+h) - f(c+2h)}{12h}$$

En donde:

f(c-2h) = 3.10

f(c-h) = 3.12

f(c+h) = 3.18

f(c+2h) = 3.34

sustituimos en el programa

In [3]:
```
f_prima_de_1_0_2 = (3.10 - 8*3.12 + 8*3.18 - 3.34) / (12*h)
print("V(1.02) es aproximadamente:",f_prima_de_1_0_2)
```

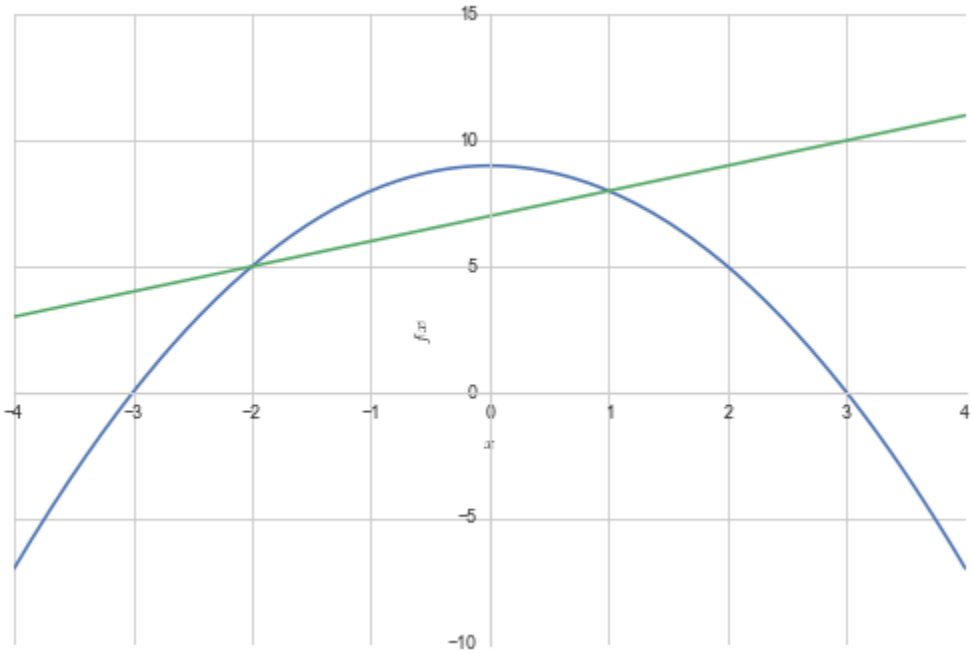
```
V(1.02) es aproximadamente: 2.000000000000017
```

In [14]:
```
# calculamos el voltaje en base al tiempo

v_de_1_0_2 = l * 2 + R*3.16
print("V(1.02) es aproximadamente:", v_de_1_0_2)
```

```
V(1.02) es aproximadamente: 2.4087199999999998
```

5. Determine el volumen del sólido de revolución que se obtiene al girar alrededor del eje *x* la región limitada por $y = 9 - x^2$, $y = 7 + x$. Emplee la fórmula de Newton-Cotes con n=5. Calcule el valor exacto y el error en la aproximación. Use 8 decimales.

In [27]:
```
# Graficamos la función para reconocer mejor los límites
from sympy import *
from matplotlib import style
style.use('seaborn-v0_8-whitegrid')

f = 9-x**2
g = 7+x

x = symbols("x")
plot(f, g, (x, -4, 4), show=True,)
```



Out[27]: `<sympy.plotting.plot.Plot at 0x7effdc68f280>`

Dado que el eje de rotación es el eje *x* nuestro diferencial es vertical y los limites de integración se muestrán claramente en la cuadrícula desde -2 hasta 1. Por la forma de la rotación ocupamos el método de arandelas en donde, para nuestro problema:

$$R = 9 - x^2 - 0$$

Usamos el cero para denotar el eje de rotación, pero desaparecerá en la integral. Igual para el caso de *r*:

$$r = 7 + x - 0$$

Por lo tanto nuestra integral es:

$$\int_a^v \pi \left(R(x)^2 - r(x)^2\right) dx$$

Sustituyendo

$$\int_{-2}^{1} \pi \left((9 - x^2)^2 - (7 + x)^2\right) dx$$

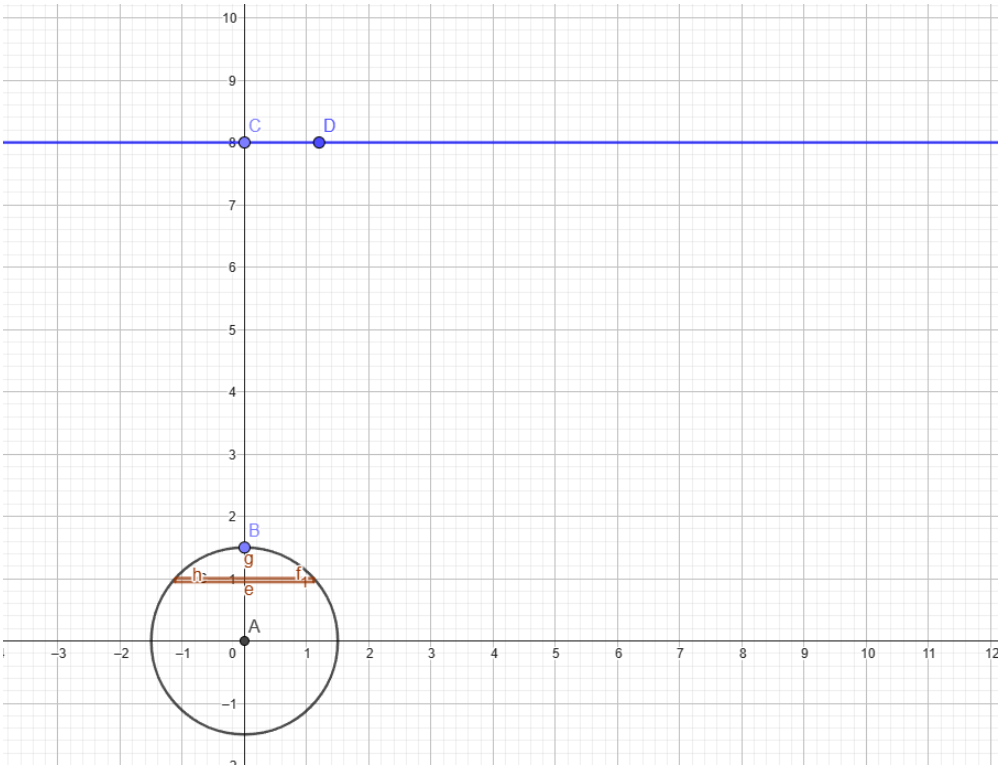
Ahora ocupamos la fórmula de 5 puntos de Newton-Cotes para calcular la resolución de la integral. Paso a Python a declara los datos y resolver.

In [44]:
```
a = -2
b= 1
x_0 = a
x_5 = b
h = (b-a)/ 5
x_1 = a + h
x_2 = a + 2*h
x_3 = a + 3*h
x_4 = a + 4*h

f_de_x = pi * ((9 - x**2)**2 - (7 + x)**2)

# Pruebo la función

valor_aproximado = ((5 * h) / 288) * (
    (19 * f_de_x.subs(x, x_0)) +  (75 * f_de_x.subs(x, x_1)) + (50 * f_de_x.subs(x,
    (50 * f_de_x.subs(x, x_3)) + (75 * f_de_x.subs(x, x_4)) + (19 * f_de_x.subs(x,

print("El valor aproximado es:", round(valor_aproximado, 8))

valor_exacto = float(integrate(f_de_x, (x, a, b)))
print("El valor exacto es:", round(valor_exacto, 8))

error = abs(valor_exacto - valor_aproximado)
print("El error es de:", float(error))
```

```
El valor aproximado es: 209.23007073
El valor exacto es: 209.23007073
El error es de: 7.211002419599581e-14
```

Ejercicio 10

10. Una ventana circular de observación en un buque para la investigación marina tiene un radio de 1.5 pie, y el centro de la ventana se encuentra a 20 pies de distancia del nivel del agua. Determine la fuerza del fluido sobre la ventana, considere el peso específico del agua de mar igual a 64 lb/pie3, empleando la regla compuesta de Newton-Cotes con n=6. Además, obtenga el valor exacto y el error de aproximación.



Datos:

Profundidad: $h(y) = 20 - y$

Longitud horizontal de la ventana: $L(y) = 3x = 3\sqrt[2]{1.5 - y}$ $\quad y\epsilon[-1.5, 1.5]$

Densidad del agua de mar: $64lb/pie^3$

Formula cerrada de Newton-Cotes para n=6 $\int_a^b f(x)\,dx \cong \frac{h}{140}[41f(x0) + 216f(x1) + 27f(x2) + 272f(x3) + 27f(x4) + 216f(x5) + 41f(x6)]$

La integral que buscamos quedaria de la siguiente manera: $F = w\int_a^b h(y)L(y)\,dy$

$F = 64\int_{-1.5}^{1.5}(20 - y)3(1.5 - y^2)\,dy$

$F = 192\int_{-1.5}^{1.5}(20 - y)(1.5 - y^2)\,dy$

Ingresamos a matlab:

a=-1.5;

b=1.5;

$h = (b - a)/6$;

syms x;

$f = 192 * ((20 - x) * (1.5 - x^2));$
$aprox6 = double((h/140) * (41 * subs(f, a) + 216 * subs(f, a + h) + 27 * subs(f, a + 2 * h) + 272 * subs(f, a + 3 * h) + 27 * subs(f, a + 4 * h) + 216 * subs(f, a + 5 * h) + 41 * subs(f, b)))$

aprox6 =

8640

```
>> a=-1.5;
>> b=1.5;
>> h=(b-a)/6;
>> syms x;
>> f=192*((20-x)*(1.5-x^2));
>> aprox6=double((h/140)*(41*subs(f,a)+216*subs(f,a+h)+27*subs(f,a+2*h)+272*subs(f,a+3*h)+27*subs(f,a+4*h)+216*subs(f,a+5*h)+41*subs(f,b)))

aprox6 =

    8640
```

```
>> exacto=double(int(f,a,b))

exacto =

     8640

>> error6=abs(aprox6-exacto)

error6 =

   0
```

Ejercicio 13

13. Un tanque abierto tiene la forma de un cono circular recto. El tanque tiene 20 pies de diámetro en la parte superior y 15 pies de de altura. Si la altura de agua dentro del tanque es tres cuartos de su altura, ¿Cuánto trabajo se realiza bombeando el agua hasta la parte superior del tanque?

Solucion:

Iniciamos determinando los datos inciales:

$h = 15$

$D = 20$

$hAgua = h * 3/4$

$\rho = 62.428 lb/pie^3$

$g = 32.17 pie/s^2$

Debemos imaginar nuestro cono partido a la mitad, donde podemo decir que nuestro radio r=10pies en x, y de altura h=15 pies en y. Por lo que podemos obtener la relacion $\frac{x}{y} = \frac{10}{15}$

Nuestro dy se desplazara en un factor de $15 - y$

Del dato inicial podemo decir que $\frac{x}{y} = \frac{10}{15} = \frac{2}{3}$

$x = \frac{2}{3}y \quad x^2 = \frac{4}{9}y^2$

$dW = \rho g \pi x^2 y dy$

$dW = \frac{4\rho g \pi}{9} y^3 dy$

$\mathcal{W} = \int_{11.25}^{15} \frac{4\rho g \pi}{9} y^3 \, dx$

```
>> syms x
>> h=15;
>> D=30;
>> r=D/2;
>> hAgua=h*3/4;
>> densidad=62.428;
>> g=32.17;
```

```
>> f=((4*densidad*g*pi)/(9))*(15-x)^3
```

Ingresando los datos a matlab, obtenemos:

```
>> a=hAgua;
>> b=h;
>> h1=(b-a)/4;
>> aprox6=double((2*h1/45)*(7*subs(f,a)+32*subs(f,a+h)+12*subs(f,a+2*h)+32*subs(f,a+3*h)+7*subs(f,b)))

aprox6 =

   -2.930676620453491e+08
```