Universidad Central de Venezuela

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

Laboratorio N°4

Periodo 2023-3

12/02/2023
CI 29571461 Ricardo Santana

1. Introducción

Tanto en ciencias como en ingeniería, e incluso en otras áreas como la economía, dinámica social, etc; se estudian sistemas que cambian y por tanto existe el interés de entender la manera en que ocurren esos cambios. En la formación básica de ingeniería y ciencias, la matemática del cambio se aborda formalmente en los cursos de cálculo diferencial e integral.

Al modelar sistemas a través del cálculo diferencial e integral se presentan ecuaciones que nos permiten resolver una problematica, estas ecuaciones pueden presentar derivadas e integrales de distinto orden y complejidad que a veces suelen complicarse al tratar de resolverlas manualmente, tanto es así que es necesario recurrir a herramientas computacionales o métodos que nos permitan calcular tales expresiones de una forma sencilla y practica.

Entre los diversos métodos numericos que nos facilita derivar e integrar una función cabe mencionar el método de diferencias finitas, el cual se basa en la definición de la derivada; por otro lado, para integrar se cuenta con el método del trapecio o de Simpson; entre muchos otros. Además, las derivadas y las integrales de los polinomios se obtienen y evalúan con facilidad. No debería sorprender, entonces, que muchos procedimientos para aproximar derivadas e integrales utilicen los polinomios que aproximan la función. Todo lo anterior considerando un error estimado pequeño que puede despreciarse según la aplicación del método numerico utilizado.

2. Marco Teórico

2.1. Diferenciación [1]

2.1.1. Teorema del valor medio

La aparición de ξ, un punto entre x_i y x_{i+1}, sugiere una conexión entre este resultado y el teorema del valor medio, el cual establece que un arco entre dos puntos, existe al menos un punto en el cual la tangente del arco es paralelo a la secante entre ambos puntos.

Si la función f es continua en el intervalo $[x_i; x_{i+1}]$ y diferenciable en $(x_i; x_{i+1})$, entonces existe un punto ξ tal que

$$f'(\xi) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

La serie de Taylor de orden cero es

$$f(x_{i+1}) \cong f(x_i) + R_0$$

Donde $R_0 = f'(\xi) h y h = x_{i+1} - x_i$ entonces

$$f(x_{i+1}) \cong f(x_i) + f'(\xi)(x_{i+1} - x_i)$$

Donde ξ está entre x_i y x_{i+1}. Éste es el teorema del punto medio, el cual se usa para probar el teorema de Taylor.

2.1.2. Diferencias hacia adelante

Si truncamos la serie de Taylor de primer orden tenemos

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i) + R_1$$

donde

$$R_1 = \frac{f'(\xi)}{2} h^2$$

$$\frac{R_1}{h} = \frac{f'(\xi)}{2} h$$

Rearreglando

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{R_1}{h}$$

$$\boxed{f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)} \quad (1)$$

2.1.3. Diferencias hacia atrás

La serie de Taylor se puede expresar hacia atrás para calcular el valor previo basado en el valor actual.

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(xi)}{3!}h^3 + \dots$$

Truncando la serie de primer orden

$$f(x_{i-1}) \cong f(x_i) - f'(x_i)h + R_1$$

Rearreglando

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} - \frac{R_1}{h}$$

$$\boxed{f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + O(h)} \quad (2)$$

2.1.4. Diferencias centradas

La tercera forma para aproximar la primera derivada es restando la diferencia hacia adelante y la diferencia hacia atrás.

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(xi)}{3!}h^3 + \dots$$
$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(xi)}{3!}h^3 + \dots$$
$$f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)h + \frac{2f'''(xi)}{3!}h^3 + \dots$$

Rearreglando

$$\boxed{f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2)} \quad (3)$$

2.1.4. Diferencias de 2° orden

Para aproximar la derivada de segundo orden, escribiremos la expansión de la serie de Taylor hacia adelante para $f(x_{i+2})$ en términos de $f(x_i)$

$$f(x_{i+2}) = f(x_i) + f'(x_i)(2h) + \frac{f''(x_i)}{2!}(2h)^2 + \frac{f'''(xi)}{3!}(2h)^3 + \dots$$

Truncamos la serie de segundo orden

$$f(x_{i+2}) \approx f(x_i) + 2f'(x_i)h + 2f''(x_i)h^2 + R_2$$

Truncando la diferencia hacia adelante de segundo orden y multiplicando por 2

$$2f(x_{i+1}) \approx 2f(x_i) + 2f'(x_i)h + f''(x_i)h^2 + 2R_2$$

Restando de la ecuación anterior tenemos

$$f(x_{i+2}) - 2f(x_{i+1}) = -f(x_i) + f''(x_i)h^2 - R_2$$

Rearreglando

$$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} + \frac{R_2}{h^2}$$

El residuo se puede escribir como

$$R_2 = \frac{f'''(\xi)}{3!}h^3$$
$$\frac{R_2}{h^2} = \frac{f'''(\xi)}{3!}h = O(h)$$

Por lo tanto

$$\boxed{f'(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} + O(h)} \quad (4)$$

Seguimos el mismo procedimiento para la derivada de 2° orden para la diferencia hacia atrás

$$\boxed{f'(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2} + O(h)} \quad (5)$$

y centrada

$$\boxed{f'(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h)} \quad (6)$$

2.2. Integración [2]

El método básico asociado con la aproximación de $\int_a^b f(x)dx$ recibe el nombre de cuadratura numérica. Éste utiliza unasuma $\sum_{i=0}^n a_i f(x_i)$ para aproximar $\int_a^b f(x)dx$.

La idea básica es seleccionar un conjunto de nodos distintos $\{x_0, \ldots, x_n\}$ del intervalo $[a, b]$. Entonces integramos el polinomio interpolante de Lagrange

$$P_n(x) = \sum_{i=0}^{n} f(x_i)L_i(x)$$

y su término de error de truncamiento sobre $[a, b]$ para obtener

$$\begin{aligned}\int_a^b f(x)dx &= \int_a^b \sum_{i=0}^{n} f(x_i)L_i(x)dx + \int_a^b \prod_{i=0}^{n}(x - x_i)\frac{f^{(n+1)}(\xi(x))}{(n+1)!}dx \\ &= \sum_{i=0}^{n} a_i f(x_i) + \frac{1}{(n+1)!}\int_a^b \prod_{i=0}^{n}(x - x_i)f^{(n+1)}(\xi(x))dx\end{aligned}$$

donde $\xi(x)$ se encuentra en $[a, b]$ para cada x y

$$a_i = \int_a^b L_i(x)dx, \ para\ cada\ i = 0, 1, \ldots, n.$$

La fórmula de cuadratura es, por lo tanto,

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i)$$

con un error dado por

$$E(f) = \frac{1}{(n+1)!}\int_a^b \prod_{i=0}^{n}(x - x_i)f^{(n+1)}(\xi(x))dx$$

Consideremos las fórmulas producidas mediante el uso del primer y del segundo polinomios de Lagrange con nodos igualmente espaciados. Esto da la regla trapezoidal y la regla de Simpson, las cuales se presentan generalmente en cursos de cálculo.

2.2.1. La regla trapezoidal

Para derivar la regla trapezoidal (o regla del trapecio) para aproximar $\int_a^b f(x)dx$, sean $x_0 = a$, $x_1 = b$, $h = b - a$ y utilice el polinomio de Lagrange

$$P_1(x) = \frac{(x - x_1)}{(x_0 - x_1)}f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)}f(x_1)$$

Entonces

$$\begin{aligned}\int_a^b f(x)dx =& \int_{x_0}^{x_1} \left[\frac{(x - x_1)}{(x_0 - x_1)}f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)}f(x_1)\right]dx \\ &+ \frac{1}{2}\int_{x_0}^{x_1} f''(\xi(x))(x - x_0)(x - x_1)dx\end{aligned}$$

El producto $(x - x_0)(x - x_1)$ no cambia de signo en $[x_0, x_1]$, por lo que el teorema del valor promedio ponderado para integrales se puede aplicar al término de error para obtener, para algunos ξ en (x_0, x_1),

$$\begin{aligned}\int_{x_0}^{x_1} f''(\xi(x))(x - x_0)(x - x_1)dx &= f''(\xi)\int_{x_0}^{x_1}(x - x_0)(x - x_1)dx \\ &= f''(\xi)\left[\frac{x^3}{3} - \frac{(x_1 + x_0)}{2}x^2 + x_0x_1x\right]_{x_0}^{x_1} \\ &= -\frac{h^3}{6}f''(\xi)\end{aligned}$$

Por consiguiente, implica que

$$\int_a^b f(x)dx = \left[\frac{(x-x_1)^2}{2(x_0-x_1)}f(x_0) + \frac{(x-x_0)^2}{2(x_1-x_0)}f(x_1)\right]_{x_0}^{x_1} - \frac{h^3}{12}f''(\xi)$$
$$= \frac{(x_1-x_0)}{2}[f(x_0)+f(x_1)] - \frac{h^3}{12}f''(\xi)$$

Por medio de la notación $h = x_1 - x_0$ obtenemos la siguiente regla:

Regla trapezoidal:

$$\boxed{\int_a^b f(x)dx = \frac{h}{2}[f(x_0)+f(x_1)] - \frac{h^3}{12}f''(\xi)} \qquad (7)$$

Esto recibe el nombre de regla trapezoidal porque cuando f es una función con valores positivos, $\int_a^b f(x)dx$ se aproxima mediante el área de un trapecio.

El término de error para la regla trapezoidal implica f'', por lo que la regla da el resultado exacto cuando se aplica a cualquier función cuya segunda derivada es idénticamente cero, es decir, cualquier polinomio de grado uno o menos.

2.2.2. Regla de Simpson

La regla de Simpson resulta de la integración sobre $[a, b]$ del segundo polinomio de Lagrange con nodos igualmente espaciados $x_0 = a$, $x_2 = b$, y $x_1 = a + h$, en donde $h = (b - a)/2$.

Por lo tanto,

$$\int_a^b f(x)dx = \int_{x_0}^{x_2}\left[\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f(x_1)\right.$$
$$\left. + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}f(x_2)\right]dx$$
$$+ \int_{x_0}^{x_2}\frac{(x-x_0)(x-x_1)(x-x_2)}{6}f^{(3)}(\xi(x))dx$$

Al deducir la regla de Simpson de esta forma, sin embargo, da un solo término de error $O(h^4)$ relacionado con $f^(3)$. Al aproximar el problema de otra forma, se puede derivar otro término de orden superior relacionado con $f^(4)$.

Para ilustrar este método alternativo, suponga que f se expande en el tercer polinomio de Taylor alrededor de x_1. Entonces, para cada x en $[x_0, x_2]$, existe un número $\xi(x)$ en (x_0, x_2) con

$$f(x) = f(x_1) + f'(x_1)(x-x_1) + \frac{f''(x_1)}{2}(x-x_1)^2 + \frac{f'''(x_1)}{6}(x-x_1)^3$$
$$+ \frac{f^{(4)}(\xi(x))}{24}(x-x_1)^4$$

y

$$\int_{x_0}^{x_2} f(x)dx = \left[f(x_1)(x-x_1) + \frac{f'(x_1)}{2}(x-x_1)^2 + \frac{f''(x_1)}{6}(x-x_1)^3\right.$$
$$\left. + \frac{f'''(x_1)}{24}(x-x_1)^4\right]_{x_0}^{x_2} + \frac{1}{24}\int_{x_0}^{x_2} f^{(4)}(\xi(x))(x-x_1)^4dx \qquad (i)$$

Puesto que $(x - x_1)^4$ nunca es negativo en $[x_0, x_2]$, el teorema de valor promedio ponderado para las integrales implica que

$$\frac{1}{24}\int_{x_0}^{x_2} f^{(4)}(\xi(x))(x-x_1)^4dx = \frac{f^{(4)}(\xi_1)}{24}\int_{x_0}^{x_2}(x-x_1)^4dx = \frac{f^{(4)}(\xi_1)}{120}(x-x_1)^5\Big]_{x_0}^{x_2}$$

para algún número ξ_1 en (x_0, x_2).

Sin embargo, $h = x_2 - x_1 = x_1 - x_0$, por lo que

$$(x_2-x_1)^2 - (x_1-x_0)^2 = (x_2-x_1)^4 - (x_1-x_0)^4 = 0$$

mientras

$$(x_2-x_1)^3 - (x_1-x_0)^3 = 2h^3 \ y \ (x_2-x_1)^5 - (x_1-x_0)^5 = 2h^5$$

Por consiguiente, la ecuación (i) se puede reescribir como

$$\int_{x_0}^{x_2} f(x)dx = 2hf(x_1) + \frac{h^3}{3}f''(x_1) + \frac{f^{(4)}(\xi_1)}{60}h^5$$

Ahora, si reemplazamos f"(x_1) por medio de la aproximación determinada en la ecuación de diferencias finitas de la sección 2.1, tenemos

$$\int_{x_0}^{x_2} f(x)dx = 2hf(x_1) + \frac{h^3}{3}\left\{\frac{1}{h^2}[f(x_0) - 2f(x_1) + f(x_2)] - \frac{h^2}{12}f^{(4)}(\xi_2)\right\} + \frac{f^{(4)}(\xi_1)}{60}h^5$$
$$= \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{12}\left[\frac{1}{3}f^{(4)}(\xi_2) - \frac{1}{5}f^{(4)}(\xi_1)\right]$$

Regla de Simpson:

$$\boxed{\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{90}f^{(4)}(\xi)} \quad (1)$$

El término de error en la regla de Simpson implica la cuarta derivada de f, por lo que da resultados exactos cuando se aplica a cualquier polinomio de grado tres o menos.

2.3. Estimación de errores

El error, según el método aplicado, se estimará a través del término $O(h^n)$ relacionado a los modelos del (1) al (8), tal que $\xi \to 0$ para la diferenciación y $\xi \in [a, b]$ para la integración. Por otro lado el error se evaluará en la comparación del cálculo de la derivada o integral realizado con la libreria sympy de python, a través de las siguientes ecuaciones

$$E(x_0) = |f'_{sympy}(x)]_{x=x_0} - f'_{calculado}(x_0)]_{x=x_0}|$$
$$E(x_0) = \left|\left[\int_a^b f_{sympy}(x)dx\right]_{x=x_0} - \left[\int_a^b f_{calculada}(x)dx\right]_{x=x_0}\right|$$

3. Práctica

3.1. Programas a utilizar

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#librerias necesarias
import math as mt
import random as rd
import sympy as sp
from sympy.abc import x,y,z
import numpy as np
import pandas as pd
import sys
x = sp.symbols('x')

#programas diseñados
ruta = '/content/drive/MyDrive/Colab-Notebooks-def/29571461-CN/29571461-Proy2/29571461-Lab4'
ruta_interpolar = '/content/drive/MyDrive/Colab-Notebooks-def/29571461-CN/29571461-Proy1/29571461-Lab2'
#print(sys.path)
sys.path.append(ruta)
sys.path.append(ruta_interpolar)
from intg import *
from difFin import *
from interpolante import *
```

3.2. Problema por tabla de datos

```
df = pd.read_csv('/content/drive/MyDrive/Colab-Notebooks-def/29571461-CN/29571461-Proy2/29571461-Lab4/datos.csv')
x0 = df.iloc[:,0].to_numpy()
fx0 = df.iloc[:,1].to_numpy()
df
```

	x	f(x)
0	1	0.500
1	2	0.143
2	3	0.071
3	4	0.044
4	5	0.029
5	6	0.021
6	7	0.016
7	8	0.013
8	9	0.010
9	10	0.008
10	11	0.007
11	12	0.006

3.2.1. Primera derivada por diferencias finitas

Tabla 1. Derivada por diferencias finitas

```
f_prima_cen = difer(x0, fx0, 0, 1)
f_prima_ade = difer(x0, fx0, 1, 1)
f_prima_atr = difer(x0, fx0, -1, 1)
df_cen = pd.DataFrame(f_prima_cen, columns=["f'(x)cen"])
df_ade = pd.DataFrame(f_prima_ade, columns=["f'(x)ade"])
df_atr = pd.DataFrame(f_prima_atr, columns=["f'(x)atr"])
df_tabla1 = pd.concat([pd.DataFrame(x0,columns=['x']),pd.DataFrame(fx0,columns=['f(x)']),df_cen,df_ade,df_atr], axis=1)
df_tabla1.to_csv('diferencias.csv', index=False)
df_tabla1
```

	x	f(x)	f'(x)cen	f'(x)ade	f'(x)atr
0	1	0.500	-0.2145	-0.357	-0.357
1	2	0.143	-0.0495	-0.072	-0.072
2	3	0.071	-0.0210	-0.027	-0.027
3	4	0.044	-0.0115	-0.015	-0.015
4	5	0.029	-0.0065	-0.008	-0.008
5	6	0.021	-0.0040	-0.005	-0.005
6	7	0.016	-0.0030	-0.003	-0.003
7	8	0.013	-0.0025	-0.003	-0.003
8	9	0.010	-0.0015	-0.002	-0.002
9	10	0.008	-0.0010	-0.001	-0.001
10	11	0.007	NaN	-0.001	-0.001
11	12	0.006	NaN	NaN	NaN

3.2.2. Polinomio interpolante de Lagrange

```
#basandose en el laboratorio 2
inst = data(x0, fx0)
px = inst.polinomio_lagrange()
print('P(x) = ')
px
```

P(x) =
$$4.43422318422323 \cdot 10^{-9}x^{11} - 3.03681657848324 \cdot 10^{-7}x^{10} + 9.03466710758377 \cdot 10^{-6}x^{9} - 0.000152397486772488x^{8} + 0.00158848131613758x^{7} - 0.0103329050925927x^{6} + 0.0386102554563496x^{5} - 0.0479009810405657x^{4} - 0.238595510361555x^{3} + 1.25738658730157x^{2} - 2.45361226551226x + 1.953$$

```
inst.graficar_pol_sp(px,'Polinomio interpolante de Lagrange')
```

3.2.3. Modelos de interpolación

3.2.3.1. Modelos 1

$$y = \frac{1}{ax^2 + bx + c}$$

Linealizando

$$\frac{1}{y} = ax^2 + bx + c$$
$$y_1 = ax^2 + bx + c$$

Entonces al linealizar

```
y1 = 1/fx0
dfx1 = pd.DataFrame(x0, columns=['x'])
dfy1 = pd.DataFrame(y1)
dfx1['y1'] = dfy1
print('datos linealizados para modelo 01')
dfx1
```

```
datos linealizados para modelo 01
```

	x	y1
0	1	2.000000
1	2	6.993007
2	3	14.084507
3	4	22.727273
4	5	34.482759
5	6	47.619048
6	7	62.500000
7	8	76.923077
8	9	100.000000
9	10	125.000000
10	11	142.857143
11	12	166.666667

```
mod1 = data(x0,y1)
coe1, pol1 = mod1.polinomio_min_cuad(2)
print(pol1)
```

```
          2
0.9967 x + 2.215 x - 1.562
```

a = 0.9967, b = 2.215, c = - 1.562

Entonces:

$$g_1(x) = \frac{1}{0.9967x^2 + 2.215x + -1.562}$$

```
g1 = 1 / (coe1[2]*(x**2) + coe1[1]*x + coe1[0])
error1 = mod1.error(fx0, (g1.subs(x,i) for i in x0))
print('error del modelo 01')
error1
```

```
error del modelo 01
```
0.0112858272437337

3.2.3.2. Modelos 2

$$y = ae^{bx}$$

Linealizando

$$Ln(y) = Ln(a) + bx$$
$$y_2 = A + bx$$

Entonces

```
y2 = np.log(fx0)
dfx2 = pd.DataFrame(x0, columns=['x'])
dfy2 = pd.DataFrame(y2)
dfx2['y2'] = dfy2
print('datos linealizados para modelo 02')
dfx2
```

```
datos linealizados para modelo 02
```

	x	y2
0	1	-0.693147
1	2	-1.944911
2	3	-2.645075
3	4	-3.123566
4	5	-3.540459
5	6	-3.863233
6	7	-4.135167
7	8	-4.342806
8	9	-4.605170
9	10	-4.828314
10	11	-4.961845
11	12	-5.115996

```
mod2 = data(x0, y2)
coe2, pol2 = mod2.polinomio_min_cuad(1)
print(pol2)
```

```
-0.3538 x - 1.351
```

A = -1.351, b = -0.3538

$$A = Ln(a) \rightarrow a = e^A$$

a = 0.2590

Entonces:

$$g_2(x) = 0.259e^{-0.3538x}$$

```
g2 = sp.exp(coe2[0])*sp.exp(coe2[1]*x)
error2 = mod2.error(fx0, (g2.subs(x,i) for i in x0))
print('error del modelo 02')
error2
```

```
error del modelo 02
```
$$0.102510694288111$$

3.2.3.3. Modelos 3

$$y = ax^{-2b}$$

Linealizando

$$Ln(y) = Ln(a) - 2bLn(x)$$
$$y_3 = A + Bx_3$$

Entonces

```
y3 = np.log(fx0)
x3 = np.log(x0)
dfx3 = pd.DataFrame(x3, columns=['x3'])
dfy3 = pd.DataFrame(y3)
dfx3['y3'] = dfy3
print('datos linealizados para modelo 03')
dfx3
```

```
datos linealizados para modelo 03
```

	x3	y3
0	0.000000	-0.693147
1	0.693147	-1.944911
2	1.098612	-2.645075
3	1.386294	-3.123566
4	1.609438	-3.540459
5	1.791759	-3.863233
6	1.945910	-4.135167
7	2.079442	-4.342806
8	2.197225	-4.605170
9	2.302585	-4.828314
10	2.397895	-4.961845
11	2.484907	-5.115996

```
mod3 = data(x3,y3)
coe3, pol3 = mod3.polinomio_min_cuad(1)
print(pol3)
```

```
-1.778 x - 0.6879
```

A = -0.6879, B = -1.778

$$A = Ln(a) \rightarrow a = e^{A} = e^{-0.6879} = 0.503$$
$$B = -2b \rightarrow b = -\frac{B}{2} = -\frac{-1.778}{2} = 0.889$$

a = 0.503, b = 0.889

Entonces:

$$g_3(x) = 0.503x^{-2(0.889)}$$

```
g3 = sp.exp(coe3[0])*(x**coe3[1])
error3 = mod3.error(fx0, (g3.subs(x,i) for i in x0))
print('error del modelo 03')
error3
```

```
error del modelo 03
```

$2.17463577041104 \cdot 10^{-5}$

Tabla errores de modelos

modelo	1	2	3
error	0.0112858272437337	0.102510694288111	$2.17463577041104 \cdot 10^{-5}$

Debido a que el error del modelo 3 es menor respecto al de los otros modelos, el modelo 3 es el que mejor se ajusta a la data dada.

$$g_3(x) = 0.503x^{-2(0.889)}$$

3.3. Valores de la derivada según modelos preestablecidos

```
#calculado
px0 = sp.lambdify(x, px)
y_px = px0(x0)
```

```
px_prima = difer(x0,y_px,1,1)

#verificación por sympy
px_prima_sp = sp.diff(px, x)
px0_prima_sp = sp.lambdify(x, px_prima_sp)

#construyendo tabla
df1 = pd.DataFrame(x0,columns=['x'])
df2 = pd.DataFrame(y_px,columns=["p(x)"])
df3 = pd.DataFrame(px_prima,columns=["p'(x) por dif"])
df4 = pd.DataFrame(px0_prima_sp(x0), columns=["p'(x) por sympy"])

df_px = pd.concat([df1,df2,df3,df4], axis=1)
df_px.to_csv('derivada_pol_inter.csv', index=False)

print('Tabla 2. diferencias finitas para polinomio interpolante P(x)'
```

Tabla 2. diferencias finitas para polinomio interpolante P(x)

	x	p(x)	p'(x) por dif	p'(x) por sympy
0	1	0.500	0.357	0.705107
1	2	0.143	-0.072	-0.140245
2	3	0.071	-0.027	-0.036631
3	4	0.044	-0.015	-0.020178
4	5	0.029	-0.008	-0.010556
5	6	0.021	-0.005	-0.006199
6	7	0.016	-0.003	-0.003823
7	8	0.013	-0.003	-0.002625
8	9	0.010	-0.002	-0.003248
9	10	0.008	-0.001	-0.000101
10	11	0.007	-0.001	-0.005606
11	12	0.006	NaN	0.040533

```
#calculado
g3x0 = sp.lambdify(x, g3)
y_g3 = g3x0(x0)
g3_prima = difer(x0,y_g3,1,1)

#verificación por sympy
g3_prima_sp = sp.diff(g3, x)
g3x0_prima_sp = sp.lambdify(x, g3_prima_sp)

#construyendo tabla
df1 = pd.DataFrame(x0,columns=['x'])
df2 = pd.DataFrame(y_g3,columns=["g3(x)"])
df3 = pd.DataFrame(g3_prima,columns=["g3'(x) por dif"])
df4 = pd.DataFrame(g3x0_prima_sp(x0), columns=["g3'(x) por sympy"])

df_g3 = pd.concat([df1,df2,df3,df4], axis=1)
df_g3.to_csv('derivada_ajus_min.csv', index=False)

print('Tabla 3. diferencias finitas para ajuste por mínimos cuadrados g3(x)')
df_g3
```

Tabla 3. diferencias finitas para ajuste por mínimos cuadrados g3(x)

	x	g3(x)	g3'(x) por dif	g3'(x) por sympy
0	1	0.502642	-0.356118	-0.893896
1	2	0.146524	-0.075280	-0.130289
2	3	0.071244	-0.028531	-0.042233
3	4	0.042713	-0.013991	-0.018990
4	5	0.028722	-0.007954	-0.010216
5	6	0.020768	-0.004980	-0.006156
6	7	0.015788	-0.003337	-0.004011
7	8	0.012451	-0.002353	-0.002768
8	9	0.010098	-0.001725	-0.001995
9	10	0.008373	-0.001305	-0.001489
10	11	0.007067	-0.001013	-0.001143
11	12	0.006054	NaN	-0.000897

3.4. Estimación de la integral

$$I = \int_{1}^{12} f(x)dx$$

3.4.1. Resolución por tabla de datos

```
#funcion que a cada valor x de la data, corresponde el f(x)
def fx0_eval(x):
    k = np.where(x0 == x)[0][0]
    return fx0[k]

#límites de la integral
a,b = 1,12

#instancia a utilizar
inst1 = integrar(fx0_eval, a, b, len(x0)-1)
```

3.4.1.1. Método del trapecio

```
print('I=')
inst1.trapecio()
```

```
I=
0.6150000000000001
```

3.4.1.2. Método de Simpson

```
print('I=')
inst1.simpson()
```

```
I=
0.5626666666666666
```

```
df01 = pd.DataFrame({'integral(f(x))':[inst1.trapecio(),inst1.simpson()]})
df01er = pd.DataFrame({'error f(x)':[None, None]})
```

3.4.2. Resolución por $f(x) = P(x)$

Al tener una función se puede ajustar el número de pasos que se quieran dar, de manera que a más pasos mayor deberia ser la presición de la integral, entonces se tomará $n = 100$ para los proximos ajustes por funciones.

```
#funcion que a cada valor x de la data, corresponde el f(x)
px, px0
n = 100

#instancia a utilizar
inst2 = integrar(px0, a, b, n)

px
```

$$4.43422318422323 \cdot 10^{-9}x^{11} - 3.03681657848324 \cdot 10^{-7}x^{10} + 9.03466710758377 \cdot 10^{-6}x^{9} - 0.000152397486772488x^{8} + 0.00158848131613758x^{7} - 0.0103329050925927x^{6} + 0.0386102554563496x^{5} - 0.0479009810405657x^{4} - 0.238595510361555x^{3} + 1.25738658730157x^{2} - 2.45361226551226x + 1.953$$

3.4.2.1. Método del trapecio

```
print('I=')
inst2.trapecio()
```

```
I=
0.5563013964225865
```

```
print('error por el método del trapecio:')
inst2.err_trapecio(px)
```

```
error por el método del trapecio:
```

$$2.78458099974402 \cdot 10^{-6}$$

3.4.2.2. Método de Simpson

```
print('I=')
inst2.simpson()
```

```
I=
0.555551500433523
```

```
print('error por el método del simpson:')
inst2.err_simpson(px)
```

```
error por el método del simpson:
```

$$1.36435074624027 \cdot 10^{-7}$$

```
df02 = pd.DataFrame({'integral(P(x))':[inst2.trapecio(),inst2.simpson()]})
df02er = pd.DataFrame({'error P(x)':[inst2.err_trapecio(px), inst2.err_simpson(px)]})
```

3.4.3. Resolución por $f(x) = g_3(x)$

```
#funcion que a cada valor x de la data, corresponde el f(x)
g3, g3x0

#instancia a utilizar
inst3 = integrar(g3x0, a, b, n)#, len(x0))

g3
```

$$\frac{0.502641813490475}{x^{1.7783948313526}}$$

3.4.3.1. Método del trapecio

```
print('I=')
inst3.trapecio()
```

```
I=
0.5533082212909332
```

```
print('error por el método del trapecio:')
inst3.err_trapecio(g3)
```

error por el método del trapecio:
$1.80497231128644 \cdot 10^{-7}$

3.4.3.2. Método de Simpson

```
print('I=')
inst3.simpson()
```

```
I=
0.5524170221618813
```

```
print('error por el método del simpson:')
inst3.err_simpson(g3)
```

error por el método del simpson:
$1.08547607858554 \cdot 10^{-10}$

```
df03 = pd.DataFrame({'integral(g3(x))':[inst3.trapecio(),inst3.simpson()]})
df03er = pd.DataFrame({'error P(x)':[inst2.err_trapecio(g3), inst2.err_simpson(g3)]})
```

```
dfi = pd.DataFrame({'metodo':['Trapecio','Simpson']})
df_tabla2 = pd.concat([dfi,df01,df02,df03], axis=1)
df_tabla2.to_csv('resultados_integral.csv', index=False)
print('Tabla 5. Resultados de integral')
df_tabla2
```

Tabla 5. Resultados de integral

	metodo	integral(f(x))	integral(P(x))	integral(g3(x))
0	Trapecio	0.615000	0.556301	0.553308
1	Simpson	0.562667	0.555552	0.552417

```
df_tabla3 = pd.concat([dfi,df01er,df02er,df03er], axis=1)
df_tabla3.to_csv('errores_integral.csv', index=False)
print('Tabla 3. Errores de integración')
df_tabla3
```

Tabla 3. Errores de integración

	metodo	error f(x)	error P(x)	error P(x)
0	Trapecio	None	2.78458099974402e-6	2.66570872038688e-8
1	Simpson	None	1.36435074624027e-7	5.82470834605626e-12

4. Análisis de resultados

Al analizar las diferencias finitas calculadas en la tabla 1 se observa cierta discrepancia con las diferencias centrada respecto a las otras, además, al tener que evaluar un tanto hacia adelante como hacia atrás limita el número de resultados respecto a los extremos del intervalo de estudio. No obstante, las diferencias hacia adelante y hacia atrás dan una aproximación razonable a la derivada, hecho que se comprueba en los resultados de las tablas 2 y 3 que implementan una función en conjunto con el método de las diferencias finitas.

En la tabla 2 se puede observa que el polinomio $P(x)$ ajusta la data de forma casi exacta, y que sus derivadas se van aproximando cada vez más al valor estimado en la librería sympy de python a medida que se va recorriendo el intervalo, sin embargo existe cierta imprecisión en las primera aproximaciones utilizando diferencias finitas. Así mismo, este comportamiento se repite en la tabla 3 para el modelo de ajuste por mínimos cuadrados $g_3(x)$, el cual a diferencia del otro ajusta la data con discrepancias en el orden decimal que uno se encuentre en el punto; sin embargo el trazo más suave de la curva permite una mejor precisión a la hora de calcular la derivada por diferencias.

Para los resultados obtenidos a través de métodos numéricos de integración, detallados en las tablas 4 y 5, se puede apreciar resultados bien aproximados debido al error en el orden de 10^{-7} para el método de los trapecios y de 10^{-10} para el método de Simpson, sin embargo al tratar la data de la tabla de datos directamente, no fue posible estimar un error, a pesar de que en la tabla 4 se observe que los resultados coinciden parcialmente con la estimación a través de los modelos.

5. Conclusión

Para una data dada por tablas el método de las diferencias finitas para calcular la derivada de una función no es el indicado, a menos que se requiera una aproximación razonable de la derivada que coincide en el orden decimal correspondiente, que para valores de mínimos se puede

realizar en base a los experimentos realizados; sin embargo se recomendaría hacer prueba para valores altos de la derivada en el estudio de máximos.

Al integrar por métodos numéricos es notable que si se quiere obtener una mayor precisión, lo ideal es utilizar el método de Simpson, que ante el método de los trapecios, ya tiene un menor error. También hay que considerar la ventaja de poseer un modelo matemático que se ajuste a una data dada, debido a que se puede muestrear en intervalos más pequeños el área que representa la integral a calcular y por tanto obtener resultados más precisos.

Cabe mencionar que existen muchos más métodos numéricos aplicables para el cálculo diferencial o integral y qué, como se pudo experimentar, facilita y agiliza el estudio de un sistema en diversas áreas que registran datos, además los métodos estudiados debido a su simplicidad, son potenciales comprobantes y procesadores de datos.