

MATERIA
MÉTODOS NUMÉRICOS

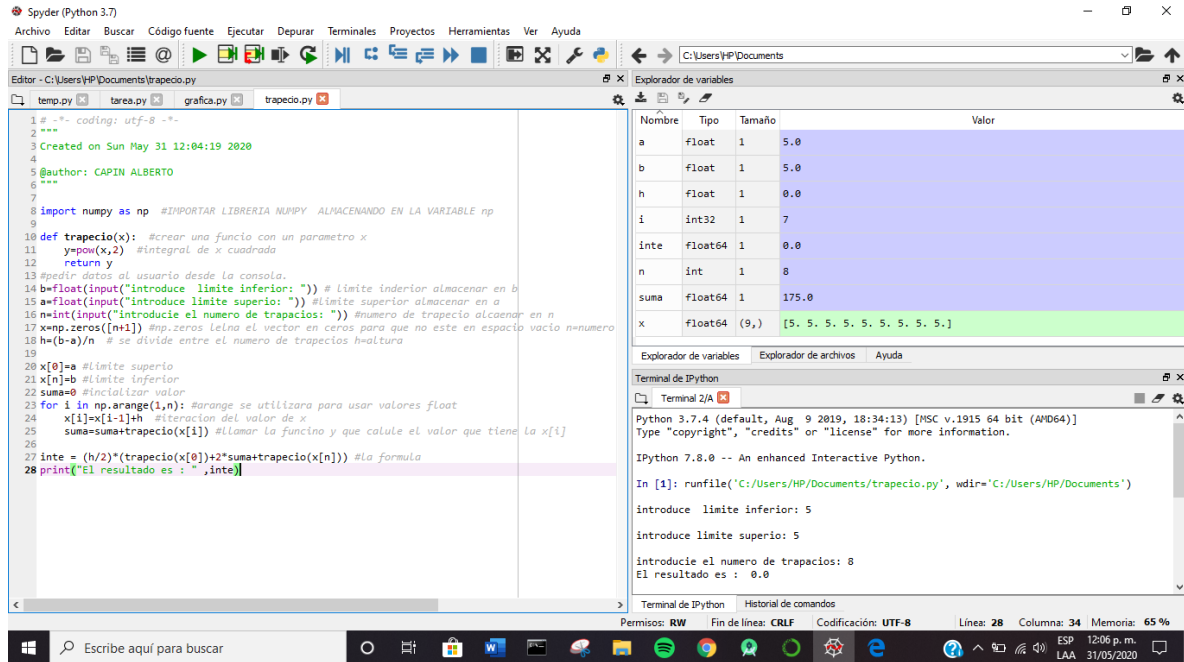
TEMA
REPORTE DE ACTIVIDADES

ESTUDIANTE
CAPIN ALBERTO HERNÁNDEZ PÉREZ

DOCENTE
ING. EFREN FLORES CRUZ

FECHA DE ENTREGA
29 DE MAYO DEL 2020

1.-



The screenshot shows the Spyder Python IDE with a file named `trapecio.py` open. The code defines a function `trapecio(x)` that calculates the area of a trapezoid. The user has input a lower limit of 5 and an upper limit of 5, resulting in an area of 0.0.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun May 31 12:04:19 2020
4
5 @author: CAPIN ALBERTO
6 """
7
8 import numpy as np #IMPORTAR LIBRERIA NUMPY ALMACENANDO EN LA VARIABLE np
9
10 def trapecio(x): #crear una función con un parametro x
11     y=pow(x,2) #integral de x cuadrada
12     return y
13 #pedir datos al usuario desde la consola.
14 b=float(input("introduce limite inferior: ")) # limite inferior almacenar en b
15 a=float(input("introduce limite superior: ")) #limite superior almacenar en a
16 n=int(input("introduce el numero de trapacios: ")) #numero de trapecio almacenar en n
17 x=np.zeros([n+1]) #np.zeros le da el vector en zeros para que no este en espacio vacío n=numero
18 h=(b-a)/n # se divide entre el numero de trapecios h=altura
19
20 x[0]=a #limite superior
21 x[n]=b #limite inferior
22 suma=0 #inicializar valor
23 for i in np.arange(1,n): #arange se utilizara para usar valores float
24     x[i]=x[i-1]+h #iteracion del valor de x
25     suma=suma+trapecio(x[i]) #llamar la función y que calcule el valor que tiene la x[i]
26
27 inte = (h/2)*(trapecio(x[0])+2*suma+trapecio(x[n])) #la formula
28 print("El resultado es : ", inte)

```

The variable explorer shows the following variables:

Nombre	Tipo	Tamaño	Valor
a	float	1	5.0
b	float	1	5.0
h	float	1	0.0
i	int32	1	7
inte	float64	1	0.0
n	int	1	8
suma	float64	1	175.0
x	float64	(9,)	[5. 5. 5. 5. 5. 5. 5. 5.]

The terminal shows the execution of the program:

```

Python 3.7.4 (default, Aug 9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/HP/Documents/trapecio.py', wdir='C:/Users/HP/Documents')

introduce limite inferior: 5

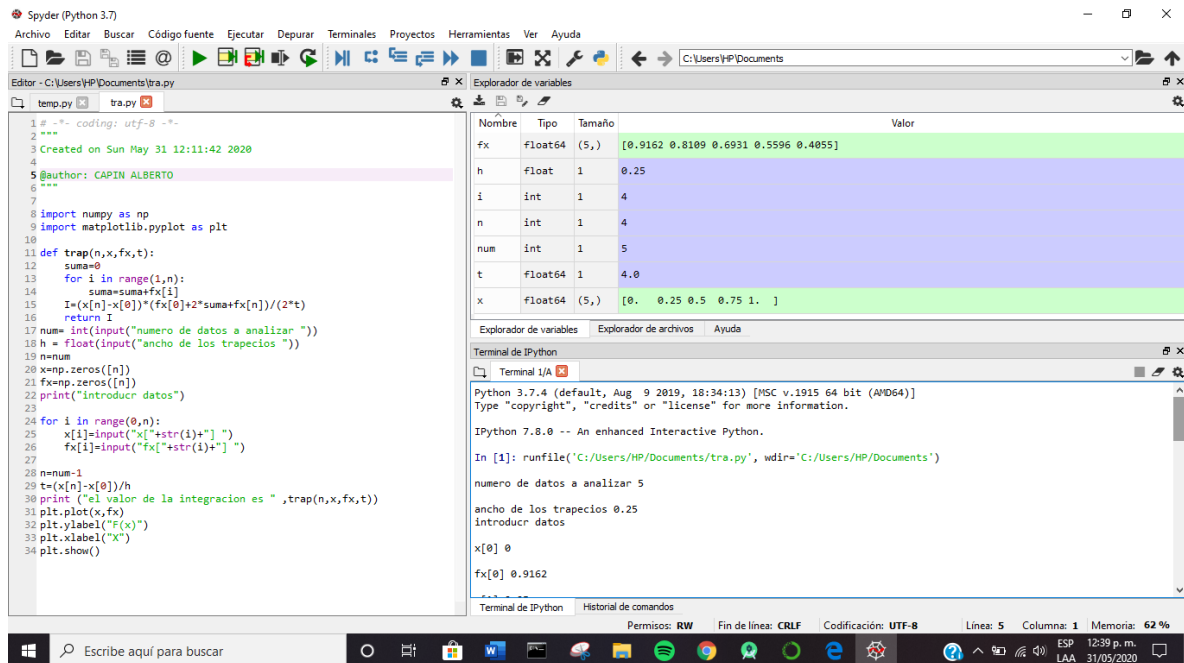
introduce limite superior: 5

introduce el numero de trapacios: 8

El resultado es : 0.0

```

2.-



The screenshot shows the Spyder Python IDE with a file named `tra.py` open. The code defines a function `trap(n, x, fx, t)` that calculates the area of a trapezoid and plots the function. The user has input a lower limit of 0.25 and an upper limit of 0.25, resulting in an area of 0.0.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun May 31 12:11:42 2020
4
5 @author: CAPIN ALBERTO
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10
11 def trap(n, x, fx, t):
12     suma=0
13     for i in range(1,n):
14         suma=suma+fx[i]
15     I=(x[n]-x[0])*(fx[0]+2*suma+fx[n])/(2*t)
16     return I
17 num= int(input("numero de datos a analizar: "))
18 h = float(input("ancho de los trapecios: "))
19 n=num
20 x=np.zeros([n])
21 fx=np.zeros([n])
22 print("introducir datos")
23
24 for i in range(0,n):
25     x[i]=input("x["+str(i)+"] ")
26     fx[i]=input("fx["+str(i)+"] ")
27
28 n=num-1
29 t=(x[n]-x[0])/h
30 print("el valor de la integracion es ", trap(n, x, fx, t))
31 plt.plot(x, fx)
32 plt.ylabel("f(x)")
33 plt.xlabel("x")
34 plt.show()

```

The variable explorer shows the following variables:

Nombre	Tipo	Tamaño	Valor
fx	float64	(5,)	[0.9162 0.8109 0.6931 0.5596 0.4055]
h	float	1	0.25
i	int	1	4
n	int	1	4
num	int	1	5
t	float64	1	4.0
x	float64	(5,)	[0. 0.25 0.5 0.75 1.]

The terminal shows the execution of the program:

```

Python 3.7.4 (default, Aug 9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/HP/Documents/tra.py', wdir='C:/Users/HP/Documents')

numero de datos a analizar 5

ancho de los trapecios 0.25

introducir datos

x[0] 0

fx[0] 0.9162

```



Spyder (Python 3.7)

Archivo Editor Buscar Código fuente Ejecutar Depurar Terminales Proyectos Herramientas Ver Ayuda

Editor - C:\Users\YIP\Documents\tra.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun May 31 12:11:42 2020
4
5 @author: CAPIN ALBERTO
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10
11 def trap(n,x,fx,t):
12     suma=0
13     for i in range(1,n):
14         suma=suma+fx[i]
15     I=(x[n]-x[0])*(fx[0]+2*suma+fx[n])/(2*t)
16     return I
17 num= int(input("numero de datos a analizar "))
18 h = float(input("ancho de los trapezios "))
19 n=num
20 x=np.zeros([n])
21 fx=np.zeros([n])
22 print("Introducir datos")
23
24 for i in range(0,n):
25     x[i]=input("x["+str(i)+"] ")
26     fx[i]=input("fx["+str(i)+"] ")
27
28 n=num-1
29 t=(x[n]-x[0])/h
30 print ("el valor de la integracion es ",trap(n,x,fx,t))
31 plt.plot(x,fx)
32 plt.ylabel("f(x)")
33 plt.xlabel("x")
34 plt.show()
```

Explorador de variables

Nombre	Tipo	Tamaño	Valor
fx	float64	(5,)	[0.9162 0.8109 0.6931 0.5596 0.4055]
h	float	1	0.25
i	int	1	4
n	int	1	4
num	int	1	5
t	float64	1	4.0
x	float64	(5,)	[0. 0.25 0.5 0.75 1.]

Explorador de variables Explorador de archivos Ayuda

Terminal de IPython

Terminal 1/A

```
x[1] 0.25
fx[1] 0.8109
x[2] 0.5
fx[2] 0.6931
x[3] 0.75
fx[3] 0.5596
x[4] 1
fx[4] 0.4055
el valor de la integracion es 0.6811125
```

Terminal de IPython Historial de comandos

Permisos: RW Fin de línea: CRLF Codificación: UTF-8 Línea: 5 Columna: 1 Memoria: 63 %

Spyder (Python 3.7)

Archivo Editor Buscar Código fuente Ejecutar Depurar Terminales Proyectos Herramientas Ver Ayuda

Editor - C:\Users\YIP\Documents\tra.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun May 31 12:11:42 2020
4
5 @author: CAPIN ALBERTO
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10
11 def trap(n,x,fx,t):
12     suma=0
13     for i in range(1,n):
14         suma=suma+fx[i]
15     I=(x[n]-x[0])*(fx[0]+2*suma+fx[n])/(2*t)
16     return I
17 num= int(input("numero de datos a analizar "))
18 h = float(input("ancho de los trapezios "))
19 n=num
20 x=np.zeros([n])
21 fx=np.zeros([n])
22 print("Introducir datos")
23
24 for i in range(0,n):
25     x[i]=input("x["+str(i)+"] ")
26     fx[i]=input("fx["+str(i)+"] ")
27
28 n=num-1
29 t=(x[n]-x[0])/h
30 print ("el valor de la integracion es ",trap(n,x,fx,t))
31 plt.plot(x,fx)
32 plt.ylabel("f(x)")
33 plt.xlabel("x")
34 plt.show()
```

Explorador de variables

Nombre	Tipo	Tamaño	Valor
fx	float64	(5,)	[0.9162 0.8109 0.6931 0.5596 0.4055]
h	float	1	0.25
i	int	1	4
n	int	1	4
num	int	1	5
t	float64	1	4.0
x	float64	(5,)	[0. 0.25 0.5 0.75 1.]

Explorador de variables Explorador de archivos Ayuda

Terminal de IPython

Terminal 1/A

Terminal de IPython Historial de comandos

Permisos: RW Fin de línea: CRLF Codificación: UTF-8 Línea: 5 Columna: 1 Memoria: 63 %



Unidad 5

Interpolación y ajuste de funciones

5.1 Polinomio de interpolación de Newton

Utilizar la matriz de Vandermonde para muchos no es muy buena idea ya que el tiempo de cálculo para matrices grandes es excesivo. Es mucho más sencillo utilizar el método clásico de las diferencias divididas de Newton.

Recordemos su definición, para dos nodos, se llama diferencia dividida de orden uno a:

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

Mientras que la diferencia dividida de orden n se obtiene por recurrencia a partir de las anteriores como

$$f[x_0, \dots, x_n] = \frac{f[x_1, \dots, x_n] - f[x_0, \dots, x_{n-1}]}{x_n - x_0}$$

El polinomio de interpolación con diferencias divididas de Newton, entre otros es la forma más popular además de la más útil en interpolación lineal.

La forma más simple de interpolar es la de conectar dos puntos con una línea recta. Este método, llamado interpolación lineal,

5.2 Polinomio de interpolación de Lagrange

Se trata de encontrar un polinomio de grado n que pase por los puntos $(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_n, f(x_n))$, se construye un cociente $L_n, K(x)$ con la propiedad de que:

$$L_n, K(x_i) = 0 \text{ cuando } i \neq j, L_n, K(x_j) = 1$$

Se requiere entonces que el numerador contenga

$$(x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)$$



El denominador debe coincidir con el numerador cuando $x = x_k$

$$\frac{\dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_0)}{\dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_0)} = \prod_{\substack{i=0 \\ i \neq k}}^n \frac{(x - x_i)}{(x_k - x_i)}$$

Teorema

Si $x_0, x_1, x_2, \dots, x_n$ son $n+1$ números distintos y si f es una función cuyos valores están dados en esos números, entonces existe un polinomio de grado a lo más n , con la propiedad de que

$$f(x_k) = p(x_k) \text{ para cada } k = 0, 1, 2, \dots, n$$

5.3 Interpolación segmentada

Esta interpolación se llama interpolación segmentaria o interpolación por splines, la idea central es que en vez de usar un solo polinomio para interpolar los datos, podemos usar segmentos de polinomios y unirlos adecuadamente para formar nuestra interpolación.

Cabe mencionar que entre todas, las splines cúbicas han resultado ser la más adecuadas para aplicaciones como la mencionada.

5.4 Regresión y correlación

Para estudiar la relación lineal existente entre dos variables continuas es necesario disponer de parámetros que permitan cuantificar dicha relación, uno de estos parámetros es la covarianza que indica el grado de variación conjunta de dos variables aleatorias siendo \bar{x} e \bar{y} la media de cada variable y x_i e y_i el valor de las variables para la observación de i , la covarianza depende de las escalas en que se miden las variables estudiadas, por lo tanto, no es comparable.



5.5 Mínimos cuadrados

Es una técnica de análisis numérico en cuadrado dentro de la optimización matemática, en la que, dados un conjunto de pares ordenados (variable independiente, variable dependiente) y una familia de funciones, se intenta encontrar la función, dentro de dicha familia, que mejor se aproxime a los datos (un mejor ajuste), de acuerdo con el criterio de mínimo error cuadrático.

Intento minimizar la suma de cuadrados de las diferencias ordenadas (llamados residuos) entre los puntos generados por la función y los correspondientes en los datos.

5.6 Problemas de aplicación

En el cálculo de estructuras, instalaciones eléctricas, hidráulicas y sanitarias, en cálculos de carreteras, topografía y hasta en diseño de las estructuras, no en todos los casos pero principalmente cuando hay mala toma de datos o haya datos faltantes.



Unidad 6

Solución de ecuaciones diferenciales

6.1 Métodos de un paso

Método de Euler

Este método se aplica para encontrar la solución a ecuaciones diferenciales ordinarias (EDO), esto es, cuando la función involucra solo una variable independiente

$$\frac{dy}{dx} = f(x, y)$$

El método se basa de forma general en la pendiente estimado de la función para extrapolar desde un valor anterior a un nuevo valor:

Nuevo valor = valor anterior + pendiente \times tamaño de paso

$$y_{i+1} = y_i + \Delta h$$

6.2 Método de pasos múltiples

Los métodos de un paso descritos en las secciones anteriores utilizan información en un solo punto x_i para predecir un valor de la variable dependiente y_{i+1} en un punto futuro x_{i+1} . Procedimientos alternativos, llamados métodos multi paso, se basan en el conocimiento de que una vez empezado el cálculo, se tiene información valiosa de los puntos anteriores y esto a nuestra disposición. La curvatura de las líneas que conectan esos valores previos proporciona información con respecto a la trayectoria de la solución, los métodos multipaso que exploraremos aprovechan esta información para resolver las EDO.



6.3 Sistema de ecuaciones diferenciales ordinarias.

Se llama ecuación diferencial a aquella ecuación que contiene derivadas.

Si la ecuación solo tiene una sola variable independiente recibe el nombre de ecuaciones diferenciales ordinarias

EDO. Si la ecuación contiene mas de una variable independiente, apareciendo así sus derivadas parciales, recibe el nombre de ecuaciones diferenciales en derivadas parciales.

6.4 Aplicaciones

Esto es un ejemplo de la aplicación en Ing. mecánica. Consiste de un resorte o muelle en espiral suspendido de un soporte rígido con una masa suelta al extremo para analizar este fenómeno usamos la ley de Hooke y la segunda ley de Newton. La ley de Hooke establece que el resorte ejerce una fuerza de restitución opuesta a la dirección.



Ejercicio con Pasos múltiples

Resolver $\frac{3x+4}{3} - 5x = 6$

Este problema involucra una fracción, antes que podamos combinar los términos de la variable, necesitamos tratar con ellos. Vamos a colocar todos los términos en la izquierda sobre un denominador común, que es tres.

$$\frac{3x+4}{3} - \frac{15x}{3} = 6$$

Luego combinamos las fracciones

$$\frac{3x+4-15x}{3} = 6$$

combinar y eliminar semejantes

$$\frac{4-12x}{3} = 6$$

Multiplicar ambas lados por 3

$$4-12x = 18$$

restar 4 de ambos lados

$$-12x = 14$$

Dividir ambos lados por -12.

$$\frac{-12x}{-12} = \frac{14}{-12}$$

Solución

$$x = -\frac{7}{6}$$