Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería



**METODOS NUMERICOS**

**Método del Trapecio**

**Docente:** Itzel Barriba Cazares

**Alumno:** Gómez Cárdenas Emmanuel Alberto

Matrícula: 1261509

# Descripción de la practica

Aplicar la regla del trapecio para la Integración numérica, mediante los recursos tecnológicos, identificando los elementos y criterios y ventajas de este, para resolver situaciones problemáticas de ingeniería en donde se requiera la determinación del área bajo la curva, en forma creativa y responsable.

# Código

#GOMEZ CARDENS EMMANUEL ALBERTO     01261509

#METODOS NUMERICOS                30/ABRIL/2020

# METODO DEL TRAPECIO SIMPLE Y COMPUESTO

# Aplicacion de la regla del trapecio para la integracion

# numerica con dos funciones: Trapecio Simple (SimpleTrap)

# y Trapecio Mutiple (MultiTrap), el programa cuenta con

# una funcion para comprobar que aplicar la regla del trapecio

# sea posible

# Capturar entero

def getInt(str):

    return int(input(str + ": "))

# Capturar flotante

def getFloat(str):

    return float(input(str + ": "))

# Capturar string

def getStr(str):

    return str(input(str + ": "))

# Evalua la funcion con el numero dado

def EvalFunction(eq, number):

  res = 0

  for i in range(len(eq)):

    res += (eq[i] \* (number \*\* (i)))

  return res

# Crea la ecuacion en forma de lista guardando el coeficiente

def CreateEq(size):

    eq = []

    for i in range(size):

        eq.append(getFloat("Introduzca el coeficiente de x^" + str(i)))

    return eq

# Comprueba que b sea mayor que a y que n sea mayor a 0

def comprobations(a, b, n):

    if ( (a < b) & (n > 0) ):

        return True

    else:

        return False

# Metodo del trapecio simple (n=1)

def SingleTrap(eq, a, b):

    fa = EvalFunction(eq, a)

    fb = EvalFunction(eq, b)

    ans = (b-a)\*((fa + fb)/2)

    return ans

#N Metodo del trapecio mutiple o compuesto (n>1)

def MultiTrap(eq, a, b, n):

    h = (b-a)/n

    xi = a

    riemmanSum = 0

    fa = EvalFunction(eq, a)

    fb = EvalFunction(eq, b)

    for i in range(1, n):

        xi = a + i\*h

        riemmanSum += EvalFunction(eq, xi)

    ans = ( ( (b-a)/(2\*n) ) \* ( fa + (2 \* riemmanSum) + fb ) )

    return ans

size = 1 + getInt("Introduzca el grado de la ecuacion")

eq = CreateEq(size)

a = getFloat("Valor inicial (a)")

b = getFloat("Valor Final (b)")

n = getInt("Trapecios a realizar (n)")

aprox = 0

if ( (comprobations(a, b, n) == True) & (n > 0)  ):

    if ( n == 1 ):

        aprox = SingleTrap(eq, a, b)

    else:

        aprox = MultiTrap(eq, a, b, n)

else:

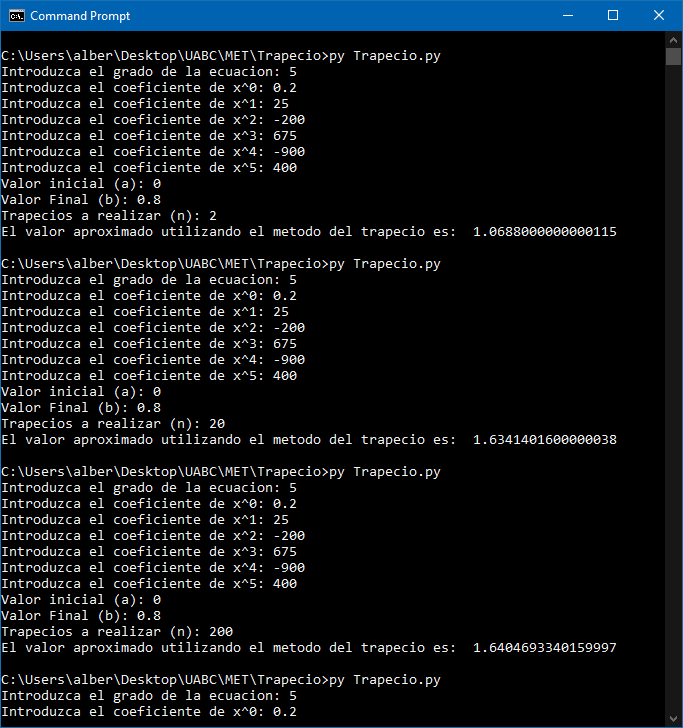
    print("El valor final debe ser mas grande al valor de inicio y el numero de trapecios debe ser mayor a 0")

    exit()

print("El valor aproximado utilizando el metodo del trapecio es: ",aprox)

# Pruebas de funcionamiento

Función utilizada

El valor exacto de la integral es 1.640533

