POLI– Metodos numericos – Trabajo colaborativo

Por.

CARLOS FELIPE CORTÉS CATAÑO.

ERVIN ARLEY GARCIA HERNANDEZ.

ERIC ARTURO MARTÍNEZ ACELAS.

YEINER ENRIQUE FERNANDEZ BUSTOS.

In [1]:
```
"""Bloque de codigo definiendo la función:
algoritmo de trazadores cubicos
e importando librerías a utilizar"""
import numpy as np
import sympy as sym
import matplotlib.pyplot as plt
def traza3natural(xi,yi):
    n = len(xi)
    # Valores h
    h = np.zeros(n-1, dtype = float)
    for j in range(0,n-1,1):
        h[j] = xi[j+1] - xi[j]

    # Sistema de ecuaciones
    A = np.zeros(shape=(n-2,n-2), dtype = float)
    B = np.zeros(n-2, dtype = float)
    S = np.zeros(n, dtype = float)

    A[0,0] = 2*(h[0]+h[1])
    A[0,1] = h[1]
    B[0] = 6*((yi[2]-yi[1])/h[1] - (yi[1]-yi[0])/h[0])

    for i in range(1,n-3,1):
        A[i,i-1] = h[i]
        A[i,i] = 2*(h[i]+h[i+1])
        A[i,i+1] = h[i+1]
        factor21 = (yi[i+2]-yi[i+1])/h[i+1]
        factor10 = (yi[i+1]-yi[i])/h[i]
        B[i] = 6*(factor21 - factor10)

    A[n-3,n-4] = h[n-3]
    A[n-3,n-3] = 2*(h[n-3]+h[n-2])
    factor12 = (yi[n-1]-yi[n-2])/h[n-2]
    factor23 = (yi[n-2]-yi[n-3])/h[n-3]
    B[n-3] = 6*(factor12 - factor23)

    # Resolver sistema de ecuaciones S
    r = np.linalg.solve(A,B)
```

```
    for j in range(1,n-1,1):
        S[j] = r[j-1]
    S[0] = 0
    S[n-1] = 0

    # Coeficientes
    a = np.zeros(n-1, dtype = float)
    b = np.zeros(n-1, dtype = float)
    c = np.zeros(n-1, dtype = float)
    d = np.zeros(n-1, dtype = float)
    for j in range(0,n-1,1):
        a[j] = (S[j+1]-S[j])/(6*h[j])
        b[j] = S[j]/2
        factor10 = (yi[j+1]-yi[j])/h[j]
        c[j] = factor10 - (2*h[j]*S[j]+h[j]*S[j+1])/6
        d[j] = yi[j]

    # Polinomio trazador
    x = sym.Symbol('x')
    px_tabla = []
    for j in range(0,n-1,1):

        pxtramo = a[j]*(x-xi[j])**3 + b[j]*(x-xi[j])**2
        pxtramo = pxtramo + c[j]*(x-xi[j])+ d[j]

        pxtramo = pxtramo.expand()
        px_tabla.append(pxtramo)

    return(px_tabla)
```

In [2]:

```
# Ingresando datos de prueba reportados en el instrucctivo "Enunciado_TC"
xi = np.array([60 , 120, 180, 240])
fi = np.array([5.5, 7.4, 15.4, 20.1])
import sympy as sp
x = sp.Symbol('x')
# Aplicando función trazadores cubicos para crear la tabla de polinomios
n = len(xi)
px_tabla = traza3natural(xi,fi)
print('Polinomios por trozos: ')
for tramo in range(1,n,1):
    print(' x = ['+str(xi[tramo-1])
          +','+str(xi[tramo])+']')
    print(str(px_tabla[tramo-1]))
# Una vez creada la tabla se integra los polinomios por medio del siguien
h = 0
for tramo in range(1,n,1):
    t = sp.integrate((px_tabla[tramo-1]), (x,(xi[tramo-1]), (xi[tramo])))
    #Sumamos los valores integrados para hallar el valor de h total o exp
    h = h + t
    print(h)
print("El valor de H es igual a: " + str(round(h, 1)))
```

```
Polinomios por trozos:
 x = [60,120]
8.54938271604938e-6*x**3 - 0.00153888888888889*x**2 + 0.09322222222222
22*x + 3.6
 x = [120,180]
-1.45061728395062e-5*x**3 + 0.00676111111111111*x**2 - 0.9027777777777
77*x + 43.44
 x = [180,240]
```

Comprobado los datos de prueba iniciamos con el ejercicio

In [3]:
```
#Importamos la tabla csv
import csv
from datetime import datetime
rad_arr = []
with open('DatosEMA-2020-03-23.csv') as File:
    reader = csv.reader(File)
    for row in reader:
        if row[0] == 'time':
            continue
        #Usamos solo los datos de 6 a.m a 6 p.m
        fecha_string = row[0].replace('2020-03-23T', '')
        fecha_string = fecha_string.replace('-05:00', '')
        hora_minutos = datetime.strptime(fecha_string, '%H:%M:%S')
        #Usamos solo datos de la columna 4 que es radiación
        if 6 <= hora_minutos.hour <= 17:
            rad_arr.append(
                float(row[4])
            )
```

In [4]:
```
# De acuerdo a la lista creada con la radiación encontramos su longitud
#Para saber el número de iteraciones
s = len(rad_arr)
cantidad = list(range(s))
import sympy as sp
# Con estos datos podemos definir fi cómo la radiación y xi como la difer
#n como la longitu de la tabla para hallar finalmente la tabla de polinom
fi = np.array(rad_arr)
xi = np.array((cantidad))
x = sp.Symbol('x')
n = len(xi)
px_tabla = traza3natural(xi, fi)
```

In [5]:
```
# Creando la tabla de polinomios
print('Tabla de polinomios: ')
for tramo in range(1,n,1):
    print(' x = ['+str(xi[tramo-1])
          +','+str(xi[tramo])+']')
    print(str(px_tabla[tramo-1]))
```

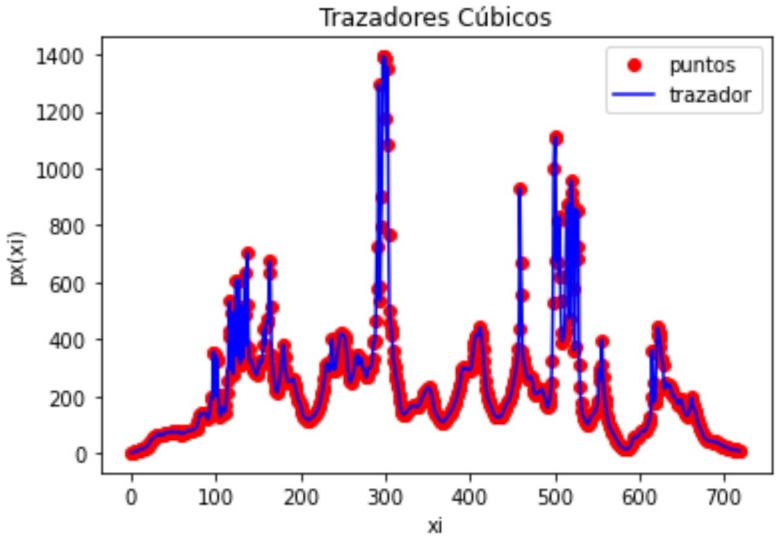
```
Tabla de polinomios:
 x = [0,1]
0.13689181681253*x**3 + 0.45310818318747*x + 1.86
 x = [1,2]
-0.984459084062649*x**3 + 3.36405270262554*x**2 - 2.91094451943807*x +
2.98135090087518
 x = [2,3]
2.63094451943807*x**3 - 18.3283689183788*x**2 + 40.4738987225705*x - 2
5.9418779271306
 x = [3,4]
-2.91931899368962*x**3 + 31.6240026997704*x**2 - 109.383216131877*x +
123.915236927317
 x = [4,5]
0.956331455320406*x**3 - 14.8838026883499*x**2 + 76.6480054206042*x -
```

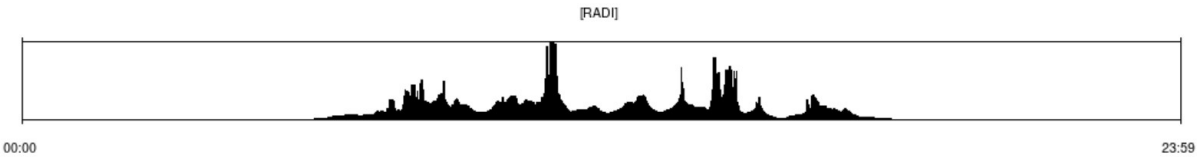
In [6]:
```
#Integramos los polinomios
h = 0
for tramo in range(1,n,1):
    t = sp.integrate((px_tabla[tramo-1]), (x,(xi[tramo-1]), (xi[tramo])))
    h = h + t
print("El valor de H es igual a: " + str(round(h, 1)))
```

```
El valor de H es igual a: 168877.6
```

In [7]:
```
# Creamos la grafica para verificar que si cohincida con la reportada por
import numpy as np
xtraza = np.array([])
ytraza = np.array([])
tramo = 1
muestras = 1 # entre cada par de puntos
while not(tramo>=n):
    a = xi[tramo-1]
    b = xi[tramo]
    xtramo = np.linspace(a,b,muestras)
    # evalua polinomio del tramo
    pxtramo = px_tabla[tramo-1]
    pxt = sym.lambdify('x',pxtramo)
    ytramo = pxt(xtramo)
    # vectores de trazador en x,y
    xtraza = np.concatenate((xtraza,xtramo))
    ytraza = np.concatenate((ytraza,ytramo))
    tramo = tramo + 1

# Gráfica
plt.plot(xi,fi,'ro', label='puntos')
plt.plot(xtraza,ytraza, label='trazador'
         , color='blue')
plt.title('Trazadores Cúbicos')
plt.xlabel('xi')
plt.ylabel('px(xi)')
plt.legend()
plt.show()
```





In [ ]: