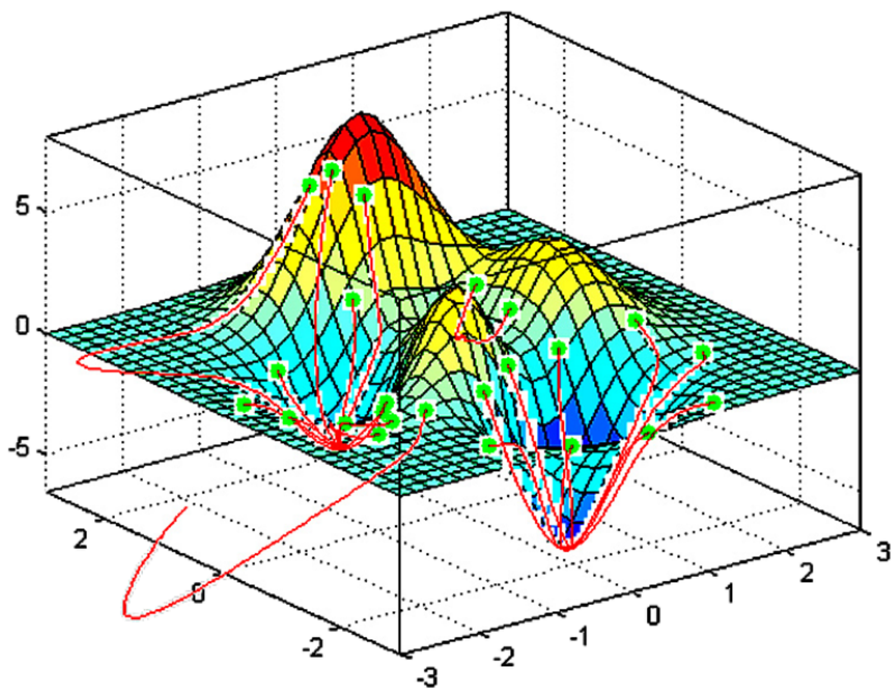
Algoritmos de Optimización

Maximiliano Cartes,Jan Seiffert, Diego Zapata
mcartes2017@alu.uct.cl, jseiffert2016@alu.uct.cl, dzapata2017@alu.uct.cl
Calculo Avanzado
MAT1189
Semestre I-2021

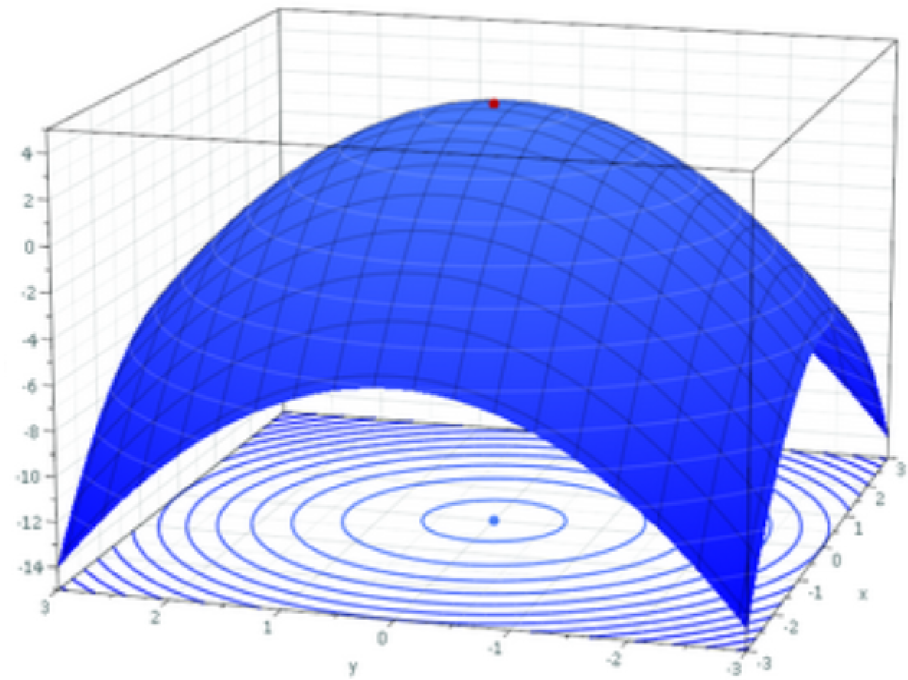
Abril 2021



Índice

1. Introducción

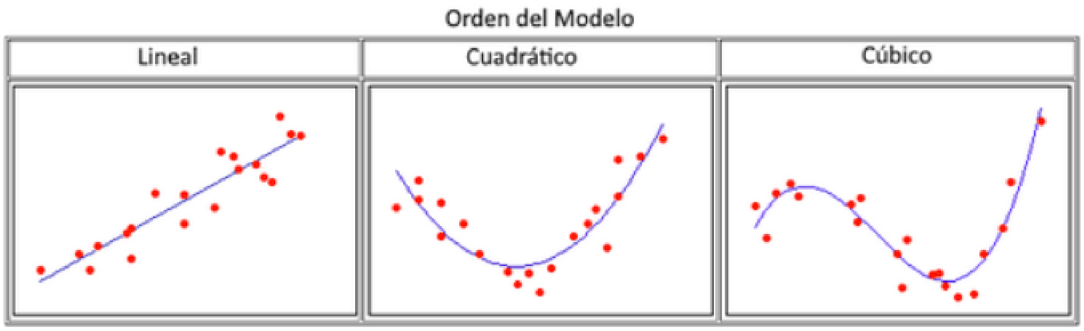
Un modelo es la abstracción de un problema real, en el cual se aplican ciertas consideraciones matemáticas permitiendo obtener resultados óptimos. Estos modelos son muy importantes para las tomas de decisiones dentro de empresas y organizaciones. Cuando ya se establecieron las mejores a partir del modelo realizado, los valores que son óptimos pueden diferir de la realidad debido a factores externos.



En otras palabras, Un modelo de optimización es la representación matemática a un problema real, en el cual tenemos conocimiento del impacto de cada una de las existentes variables, de las cuales intentamos encontrar el mínimo o máximo valor posible dentro de una función objetivo.

2. Optimización en ajuste de curvas

Para poder hablar de la optimización de ajuste de curvas, tenemos que comenzar con saber que es el ajuste de curvas, El ajuste de curvas implica encontrar una curva que sea contenida dentro de una serie de puntos los cuales satisfacen otras restricciones.



Por otro lado, la optimización de una función, implica encontrar sus valores máximos y mínimos. La optimización matemática es un concepto matemático esencial para abordar problemas de la vida cotidiana.

3. Métodos de optimización

En los métodos de optimización existen diversos modelos, todos estos para resolver diferentes problemas, ejemplo de ellos son:

- Reacción de orden cero: La velocidad de una reacción de orden cero es una constante, la cual no depende de la unión de los reactivos.

$$Q_1 = Q_0 + K_0 t$$

- Reacción de primer orden: Es una reacción donde la velocidad depende de la concentración de un reactivo elevado a la primera potencia.

$$ln(Q)_t = ln(Q)_0 + k_1 t$$

- Reacción de segundo orden: La velocidad con que ocurra una reacción depende que la concentración del reactivo elevado al cuadrado o de dos reactivos, de manera en que cada uno quede elevado a la uno.

$$Q_t / Q_\infty (Q_\infty - Q_t) K_2 t$$

- Modelo de Hixon-Crowel: Tambien conocido como el modelo de la raíz cúbica.

$$Q_0^{1/3} - Q_t^{1/3} = K_s t$$

- Modelo de Weibull: Este modelo describe la conducta de los sistemas o eventos que tienen algún grado de alteración.

$$log[-ln(1 - (Q_t/Q_\infty))] = b \times log(t) - log(a)$$

- Modelo de Higuchi:.

$$Q_t = K_H \sqrt{t}$$

- Modelo de Baker-Lonsdale:

$$(3/2)[1 - (-1(Q_t/Q_\infty))^2/3] - (Q_t/Q_\infty) = K_t$$

- Modelo de Korsmeyer-Peppas:

$$Q_t/Q_\infty = K_h t^n$$

- Modelo Cuadrático: Este modelo describe una función cuadrática para representar una situación u objeto real.

$$Q_t = 100(K_1 t^2 + K_2 t)$$

- Modelo Logistico: Este modelo constituye un perfeccionamiento del modelo exponencial para el aumento de una magnitud. Modela la función sigmoidea de crecimiento de un conjunto Q.

$$Q_t = A/[1 + e^{-k(t-y)}]$$

- Modelo de Gompertz: Este modelo describe el crecimiento más lento al comienzo y al final de un período de tiempo establecido.

$$Q_t = Ae^{-e-K(t-y)}$$

- Modelo de Hopfenberg: Este modelo describe la liberación por erosión de la superficie de la matriz.

$$Q_t/Q_\infty = 1 - [1 - K_0 t/C_o a0]^n$$

En este trabajo vamos a considerar dos modelos y su función costo: Con un parámetro (simplificación del modelo de Weibull). Es decir, Trabajaremos usando el modelo Weibull y sus resultados se reemplazaran en la funcion costo, los cuales graficaremos durante las siguientes paginas.

$$y(x) = 1 - e^{-ax}$$

Con dos parámetros (modelo de Korsmeyer-Peppas):

$$y(x) = ax^b$$

Función Costo: Es aquella que mide la diferencia entre lo pronosticado por el modelo y los datos experimentales.

$$\sum_{i=1}^{n} (y(p; x_i) - \widehat{y}_i)^2$$

La tabla con los datos a probar con los modelos que tenemos arriba es la siguiente:

t_i	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
u_i	0,2	0,3	0,45	0,55	0,6	0,7	0,75	0,8	0,8	0,8

4. Contexto de ámbito profesional en donde se deberá implementar distintos tipos de optimización

Durante la vida profesional un ingeniero en informática puede encontrarse con variadas situaciones de optimización dependiendo del campo al que se dedique, aunque existe un campo donde estos problemas son más notorios, este es el campo llamado “Machine Learning”, en donde se pueden encontrar diversos algoritmos de trabajo para calcular distintos tipos de información o datos que se tienen. El que más se relaciona con esta materia que estamos viendo es la interpolación, puesto que el ajuste de curvas consisten en encontrar una curva que contenga una serie de puntos y que posiblemente cumpla una serie de restricciones(previamente dada) adicionales a este. Un ejemplo de aquello sería realizar el ajuste de un conjunto de datos pertenecientes a una ecuación cuadrática.

En el campo de Machine Learning de la informática estos problemas son muy comunes, por lo cual en este contexto profesional del M.L se trabajará de manera continua con este y otros algoritmos existentes. Pero si abarcamos mucho más en el ámbito de la Ingeniería en sí, a modo de visión general, el ajuste de curvas también se puede buscar la forma de implementarlo en la programación lineal, en el cálculo de máximos y mínimos que tendrán una serie de restricciones específicas dependiendo la situación que se está trabajando, Ejemplo de aquello, sería minimizar el costo de alguna fábrica y buscar sobre este mismo, la implementación de un ajuste de curvas al método gráfico que se está trabajando. Esto siempre y cuando se pueda hacer uso de este con los datos y restricciones que esta clase de problemas plantea.

5. Otros ejemplos de CurveFitting, Interpolaciones y optimización en Python

Optimización

La función Minimize de la librería Scipy nos provee con algoritmos de optimización para funciones. En la siguiente imagen se puede apreciar un código que en pocas líneas, optimiza la función Rosenbrock.

```python
import numpy as np
from scipy.optimize import minimize
def rosen(x):
    return sum(100.0*(x[1:]-x[:-1]**2.0)**2.0 + (1-x[:-1])**2.0)
x0 = np.array([1.3, 0.7, 0.8, 1.9, 1.2])
res = minimize(rosen, x0, method='nelder-mead',
    options={'xatol': 1e-8, 'disp': True})
print(res.x)
```

```
Running: C:\Users\janse\Desktop\rosen.py (Thu Apr 29 21:00:45 2021)

Optimization terminated successfully.
         Current function value: 0.000000
         Iterations: 339
         Function evaluations: 571
[1. 1. 1. 1. 1.]


Execution Successful!
```

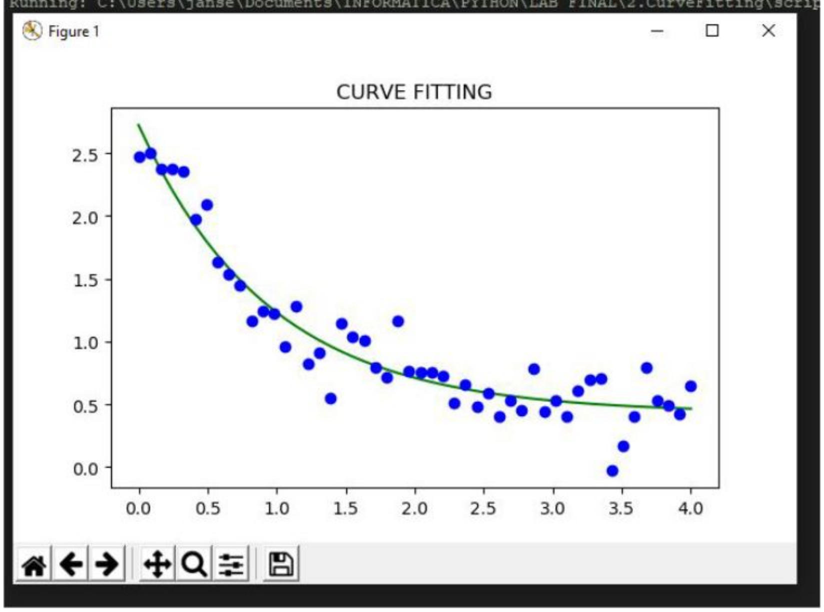
El codigo nos arroja en la terminal los resultados de la optimización, las evaluaciónes a la funcion e iteraciones.

CurveFitting

La función Curvefit de la librería Scipy.optimize.curvefit nos provee con algoritmos para lograr curve fitting.En el siguiente codigo implementamos el curve fitting con graficos de matplotlib.

```
from scipy.optimize import *
import matplotlib.pyplot as plt, numpy as np

def MyFx(x,a,b,c):
    return a*np.exp(-b*x) + c

x_0 = np.linspace(0,4,50)
y_0 = MyFx( x_0 , 2.5 , 1.3 , 0.5 ) + 0.2*np.random.normal(size=len(x_0))

popt, pcov = curve_fit(MyFx, x_0, y_0)

plt.title("CURVE FITTING")
plt.plot(x_0, MyFx(x_0, *popt), 'g-')
plt.plot(x_0, y_0,"bo")
plt.show()
```

```
Running: C:\Users\janse\Documents\INFORMATICA\PYTHON\LAB FINAL\2.CurveFitting\scrip
```



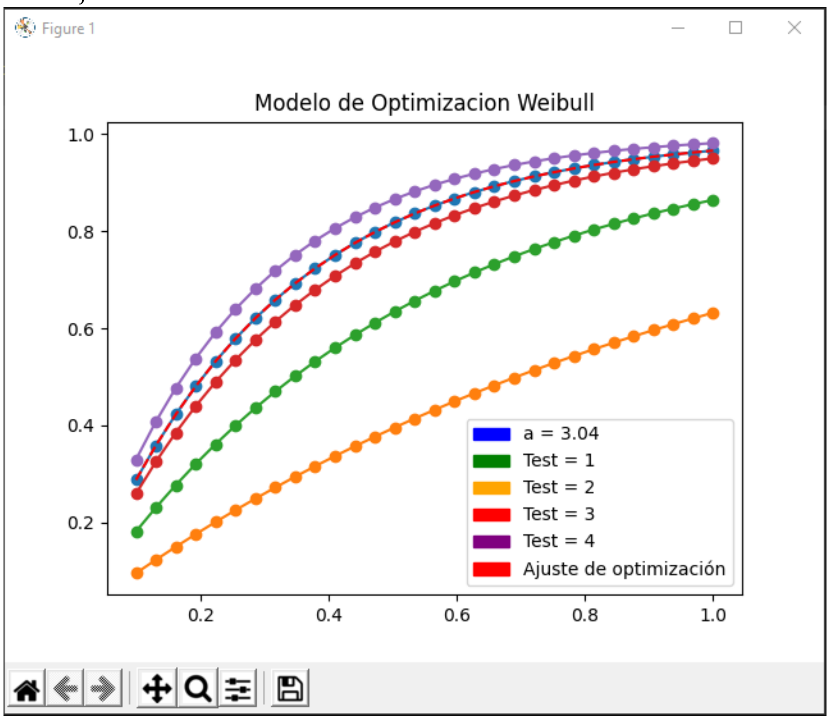
6. Algoritmo para calcular la optimización de curvas en diversos modelos

El siguiente script tiene como objetivo realizar la optimización de curvas utilizando los diversos modelos a matemáticos, en este caso se utilizarán los modelos de Weibull, Hixson-Crowell, Higuchi y el módelo Cuadrático.

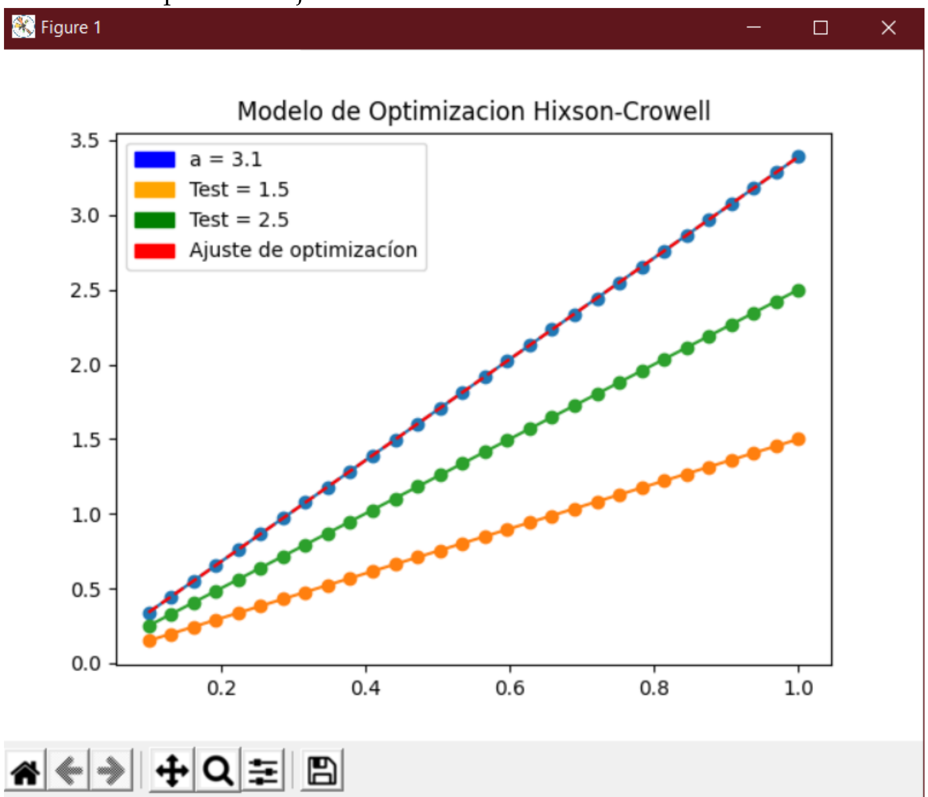
```
import numpy as np; from scipy.optimize import curve_fit; import matplotlib.pyplot as plt;impor
from matplotlib import cm; from matplotlib.ticker import LinearLocator, FormatStrFormatter

#modelos
def Weibull(t,a):
    modelo = 1-np.exp(-(a*t))
    return modelo

def HixsonCrowell(t,a):
    modelo= a*t
    return modelo

def Higuchi(t,a,b):
    modelo = a*t**(b)
    return modelo

def Quadratic(t,a):
    modelo = 100*(a*t**2 + a*t)
    return modelo
#Se crea el arreglo que se llenara con valores predefinidos por la funcion np.arange de numpy
a = []

t = np.linspace(0.1,1,30)
for i in np.arange(1, 5.01, 0.01):
    a.append(round(i,2))
```

```
#datos experimentales a probar en nuestra funcion
d_exp = [1.1,1.2,1.2,1.2,1.3,1.4,1.7,1.8,1.8,2.8,
        2.8,2.8,2.8,2.9,2.9,2.9,2.9,2.9,2.9,2.9,
        2.9,2.9,2.9,2.9,2.9,2.9,2.9,2.9,2.9,2.9]
#Aplicacion de la funcion costo
tamA = len(a)
tamT = len(t)
sumatoriaHig = [0.0]*tamA
for x in range(tamA):
    for y in range(tamT):
        sumatoriaHig[x] = sumatoriaHig[x] + (a[x]*(t[y]**0.5)-d_exp[y])**2
pOpt = sumatoriaHig.index(min(sumatoriaHig))

#Se define a nuevamente pero esta vez con el valor optimo de nuestro calculo previo
a = a[pOpt]
b = 0.5
print('a = ',a)
```

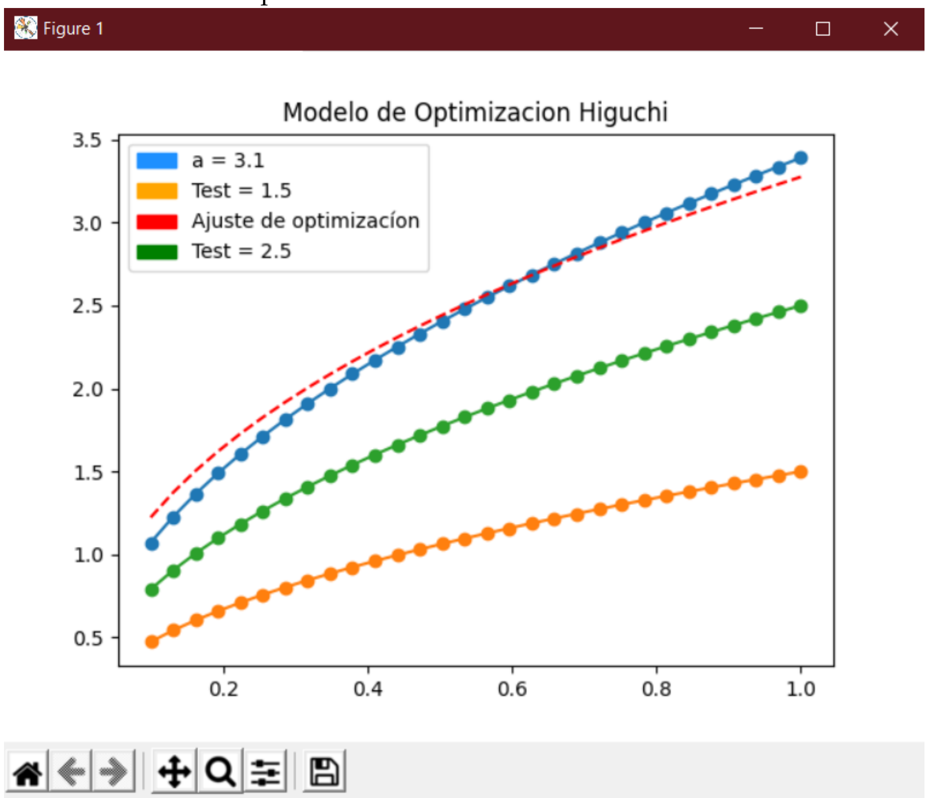
En la siguiente imagen podemos apreciar el ajuste de curvas realizado con los datos experimentales más el modelo weibull, la linea -”de color rojo corresponde al ajuste realizado:



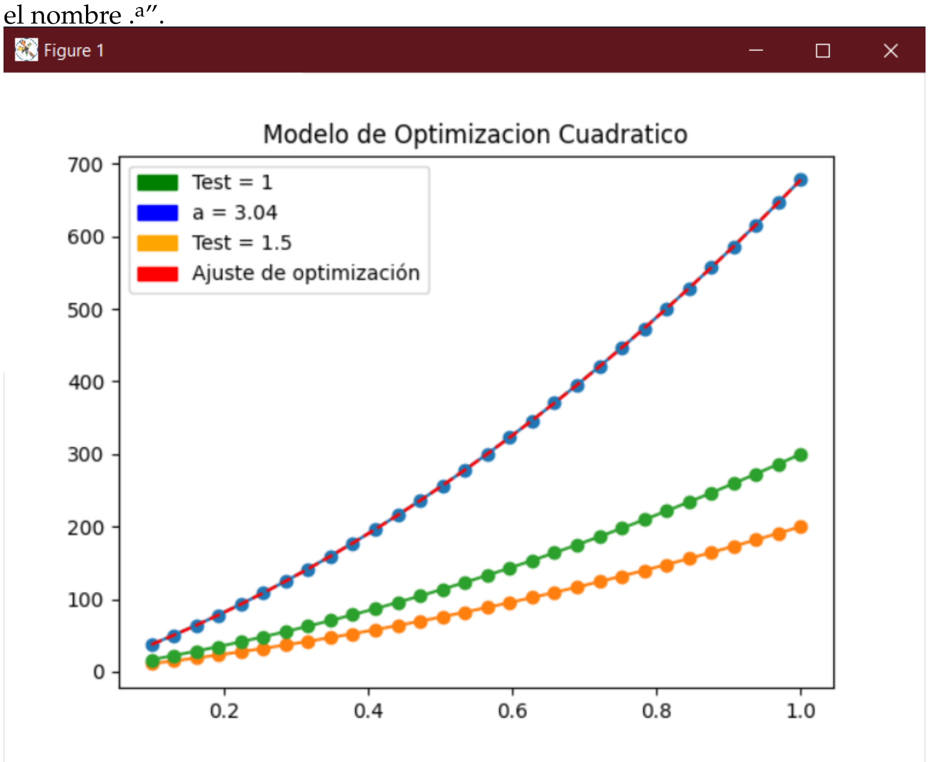
Luego se realizara el ajuste de curvas usando el modelo Hixson-Crowell, con los mismos valores experimentales, donde la linea '–' roja tambien corresponde al resultado optimo del ajuste:



Como tercer Grafico corresponde al modelo Higuchi, tambien comparado con lineas Test o lineas de prueba con valores tambien de prueba que se dejan ver en el cuadro de descripcion:



Y para terminar y el ultimo grafico de prueba de ajuste de optimizacion es el modelo Cuadratico. Debo añadir que el valor optimo corresponde al que tiene el nombre .ª".



Tras observar los distintos graficos de Optimizacion de curvas en distintos modelos,podemos observar que en 3 de ellos hubo un calculo correcto al realizar el ajuste, estos son el modelo Hixson-Crowell, Weibull y el modelo cuadratico, siendo el modelo de Higuchi el unico en el que no se obtuvo un ajuste totalmente perfecto, sino que tuvo una variacion de los valores muy pequeña, pero aun asi se logra un ajuste bastante apreciable

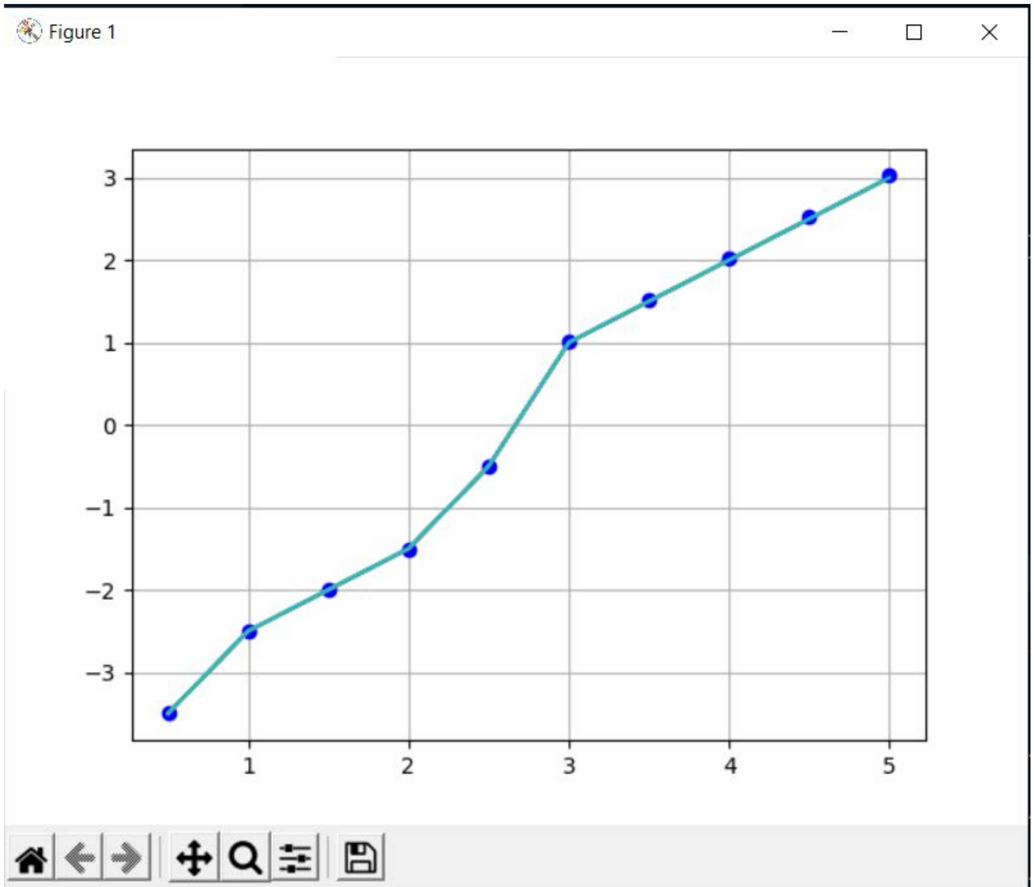
7. Código para graficar el modelo de higuchi con datos experimentales

El siguiente codigo logra graficar los datos experimentales de A y T junto con la función de Higuchi. Se logra apreciar que ambas graficas calzan entre si, debido al ajuste de datos experimentales para lograr este efecto.

```
import matplotlib.pyplot as plt
import numpy as np

def higuchi(a,t):
    return a*(t**0.005)

t = [0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0]
a = [-3.5,-2.5,-2,-1.5,-0.5,1,1.5,2,2.5,3.0]

for i in range(len(a)):
    num = higuchi(a[i],t[i])
    plt.plot(t[i],num,marker="o", color="b")
    plt.plot(t,a)

plt.grid(True)
plt.show()
```

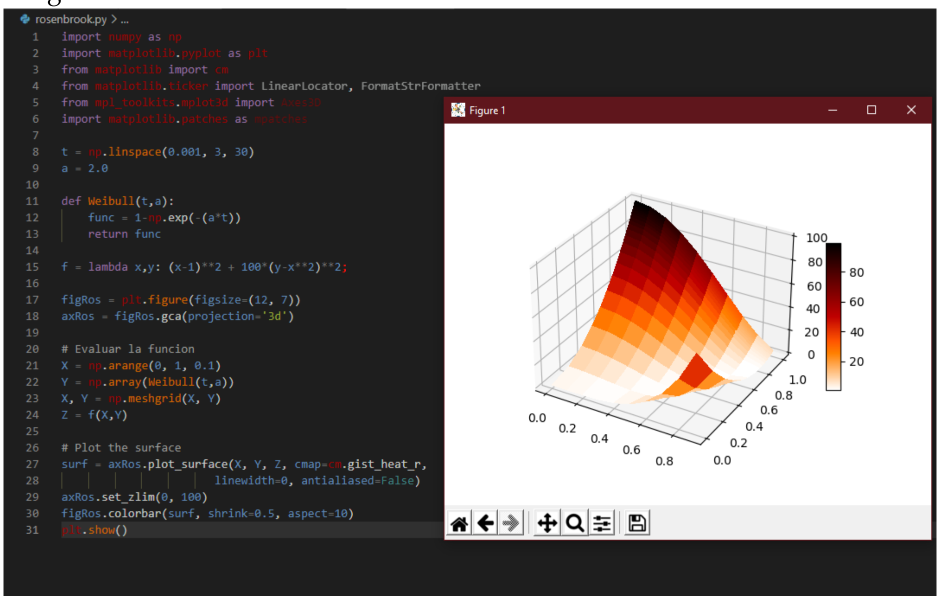
8. Grafica del modelo de higuchi con datos experimentales

Al ejecutar el codigo podemos observar el gráfico con los datos experimentales, donde se nos muestra en puntos azules los resultados arrojados por la función de Higuchi y con una linea celeste se muestran los datos de A y T.



9. Funcion Rosenbrook con su correspondiente grafico

En la imagen de abajo podemos observar el codigo y su correspondiente gráfico para la funcion Rosenbrook, en esta se realiza un plot 3D haciendo el uso del gradiente de la funcion anteriormente mencionada



10. Referencias

[Scipy, 2021] The SciPy community(2021, 26 abril). optimize.curve . Scipy. https://docs.scipy.org/doc/scipy/reference/interpolate.html

[Scipy, 2021] The SciPy community(2021, 26 abril). Interpolation _fit. Scipy. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

[Numpy, 2021] The Numpy community(2021). Numpy.arange, Numpy. https://numpy.org/doc/stable/reference/generated/numpy.arange.html

[UACH, 2013] YÁÑEZ D.(2013). EVALUACIÓN DE CINÉTICAS DE LIBERACIÓN DE ALIMENTOS MEDICADOS A BASE DE BENZOATO DE EMAMECTINA, UILIZANDO MEDIOS ALTERNATIVOS DE DISOLUCIÓN. cybertesis. http://cybertesis.uach.cl/tesis/uach/2013/fcy.22e/doc/fcy.22e.pdf

[Google Sites, 2021] ittgmetodosnumericos(2021). Unidad 4: Ajuste de curvas e interpolación. Google Sites. https://sites.google.com/site/ittgmetodosnumericos/home/unidad-4-ajuste-de-curvas-e-interpolacion