# Nonlinear Regression with Matlab

Instead of MatLab i choose to use Python, firstly I solved the problem using the GEKKO library

# Solving with Python and GEKKO

Asdf

## Python code

from gekko import GEKKO

import numpy as np

import matplotlib.pyplot as plt

*# measurements*

xm = np.array([0.0,0.0,1.0,1.0,2.0,2.0,3.0,3.0,4.0,4.0,5.0,5.0])

ym = np.array([2.86,2.64,1.57,1.24,0.45,1.02,0.65,0.18,0.15,0.01,0.04,0.36])

*# GEKKO model*

m = GEKKO(remote=False)

*# parameters*

x = m.Param(value=xm)

B1 = m.FV()

B1.STATUS=1

B2 = m.FV()

B2.STATUS=1

B3 = m.FV()

B3.STATUS=1

*# variables*

y = m.CV(value=ym)

y.FSTATUS=1

*# regression equation*

m.Equation(y==B1+B2\*m.exp(-B3\*x))

*# regression mode*

m.options.IMODE = 2

*# optimize*

m.solve(disp=True)

*# print parameters*

print(f'Optimized B1 = {B1[0]}')

print(f'Optimized B2 = {B2[0]}')

print(f'Optimized B3 = {B3[0]}')

plt.figure(1)

plt.plot(xm,ym,'bo')

plt.plot(xm,y.value,'r-')

plt.xlabel('x')

plt.ylabel('y')

plt.legend(['Measured','Model'],loc='best')

plt.show()

## Console result

Text

Description automatically generated

## Plot result

Chart, scatter chart

Description automatically generated

# Solving with Python and SciPy

## Python code

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import least\_squares

def model(B, x):

    return B[0]+B[1]\*np.exp(-B[2]\*x)

def fun(B, x, y):

    return model(B, x) - y

*# measurements*

xm = np.array([0.0,0.0,1.0,1.0,2.0,2.0,3.0,3.0,4.0,4.0,5.0,5.0])

ym = np.array([2.86,2.64,1.57,1.24,0.45,1.02,0.65,0.18,0.15,0.01,0.04,0.36])

*#initial guess*

B0 = np.array([1,1,1])

res = least\_squares(fun, B0, bounds=(0, 100), args=(xm, ym), verbose=1)

print(res)

print(f'Optimized B1 = {res.x[0]}')

print(f'Optimized B2 = {res.x[1]}')

print(f'Optimized B3 = {res.x[2]}')

x\_modelled = np.linspace(0, 5)

y\_modelled = model(res.x, x\_modelled)

plt.plot(xm, ym, 'o', markersize=4, label='data')

plt.plot(x\_modelled, y\_modelled, label='fitted model')

plt.xlabel("x")

plt.ylabel("y")

plt.legend(loc='best')

plt.show()

## Console result

Text

Description automatically generated

## Plot result

Chart

Description automatically generated