

ExerciseSineFit: Introduction

Least Squares Fit (LSF) method gives **solution for free parameters** $\bar{\beta}$. This method solves $\bar{\beta}$ values that

- **Minimize** χ^2 when errors σ_i are **known**
- **Minimize** R when errors σ_i are **unknown**

Download program from **LineModel.py** from homepage. The model is

$$g(t) = \beta_0 + \beta_1 t$$

LineModel.py first creates simulated data with this model, and then performs LSF to these data. The subroutines are

Data(n) produces simulated data

- $n = \mathbf{n}$ random time points $t_i = \mathbf{T}$ drawn from an even distribution between 0 and 10
- Two random free parameter values $\bar{\beta} = [\beta_1, \beta_2] = \mathbf{SimBETA}$ drawn from an even distribution between -1 and 1
- $n = \mathbf{n}$ random errors $\sigma_i = \mathbf{EY}$ drawn from normal distribution $N(m_y = 0, s_y = 0.1)$, where m_y is the mean and s_y is the standard deviation.
- Simulated data $y_i = \mathbf{Y} = g_i + \sigma_i = \mathbf{G} + \mathbf{EY}$

IndexSortOrder(y) gives indices $\mathbf{k}$ that are used to re-arrange $\mathbf{y}$ values into ascending order, which in this case are the time points $t_i = \mathbf{T}$.

Write1(T,Y,EY) stores simulated data to file **LineModel.dat**

Model(T,BETA) computes model values $g_i = \mathbf{G}$

Funct(BETA,T,Y,EY) gives variable $(\mathbf{Y}-\mathbf{G})/\mathbf{EY}$ minimized in LSF $\equiv$ **optimize.leastsq**

LSF(T,Y,EY) performs LSF.

Plot1(T,Y,EY,TT,GG) plots results into **LineFit.eps**

```

# =====
# /home/jetsu/opetus/both/programs/LineModel.py
# =====
# - Keep computations and plots apart.
# =====
import os
import numpy as np
import pylab as pl
from scipy import optimize
os.system( 'clear ' )
# =====
def Data(n):
    T=np.random.uniform(0,10,n)
    k=IndexSortOrder(T)
    T=T[k]
    BETAsim=-1.+np.random.uniform(0.,2.,2)
    EY=np.random.normal(0,0.1,n)
    G=Model(T,BETAsim)
    Y=G+EY
    Write1(T,Y,EY)
    return T,Y,EY,BETAsim
# =====
def IndexSortOrder(y):
    k=sorted(range(len(y)), key=lambda i:y[i])
    return k
# =====
def Write1(T,Y,EY):
    file1=open( 'LineModel.dat' ,"w")
    for i in range(np.size(T)):
        file1.write( "%10.5f_%10.5f_%10.5f\n" %(T[i],Y[i],EY[i]))
    file1.close()
    return
# =====
def Model(T,BETA):
    G=BETA[0]+BETA[1]*T

```

```

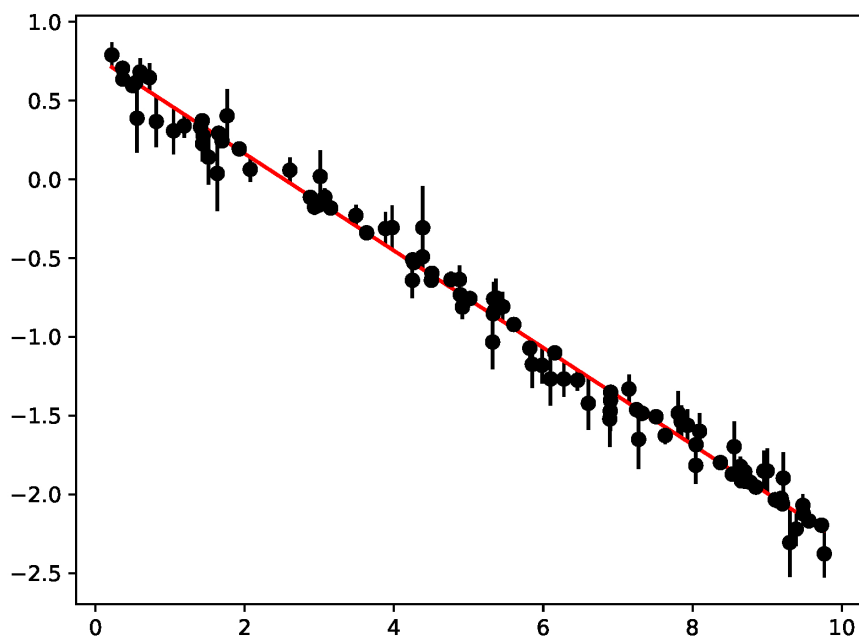
    return G
# =====
def Funct(BETA,T,Y,EY):
    G=Model(T,BETA)
    return (Y-G)/EY
# =====
def LSF(T,Y,EY):
    BETAtrial=np.ones(2)
    ny=optimize.leastsq(Funct,BETAtrial,args=(T,Y,EY))
    BETA=ny[0]
    return BETA
# =====
def Plot1(T,Y,EY,TT,GG):
    pl.axes([0.1,0.1,0.8,0.8])
    pl.errorbar(T,Y,EY,fmt='ok',ms=6)
    pl.plot(TT,GG,'r')
    pl.savefig('LineFit.eps')
# =====
#                                     Main program
# - Computations =====
n=100
T,Y,EY,BETAsim=Data(n)
BETAfinal=LSF(T,Y,EY)
print('Simulated_data_..._BETAsim=',BETAsim)
print('Least_Squares_Fit_..._BETAfinal=',BETAfinal)
DT=np.max(T)-np.min(T)
TT=np.min(T)+(np.arange(101)/100.)*DT
GG=Model(TT,BETAfinal)
# - Plots =====
Plot1(T,Y,EY,TT,GG)

```

The **scipy** subroutine **optimize.leastsq** minimizes “the sum of squares of a set of equations”. We use **(Y-G)/EY**, which minimizes χ^2 .

Notation **ny** is adopted in referring to a variable that is not used later in the program. Note that the first component of this variable, **ny[0]**, contains the final free parameter values **BETA**. Any trial values **BETAtrial** will give the same result, because the model is **linear**.

One example of **LineFit.eps** is shown below. Note that **LineModel.py** always produces a different figure, because it always analyses a different sample of random data.



ExerciseSineFit: Problem

The model is

$$g(t) = \beta_1 + \beta_2 \cos(2\pi ft) + \beta_3 \sin(2\pi ft),$$

where the constant frequency is $f = 1/P$ and $P = 2.4$. Edit a **python** program **ExerciseSineFit.py** that produces $n = 100$ simulated observations

$$y(t_i) = g(t_i) + \sigma_i$$

Draw the simulated

Random time points $t_i = \mathbf{T}$ from a uniform distribution $U(0, 10, n)$

Random free parameters $\beta_i = \mathbf{BETA}$ from a uniform distribution $U(-1, 1, 3)$

Random errors $\sigma_i = \mathbf{EY}$ from a normal distribution $N(0, 0.1, n)$

Store your simulated data into **SineModel.dat**. Perform a LSF to these simulated data and show your results in figure **SineFit.eps**. Your figure should resemble the one shown below. Send your files **ExerciseSineFit.py** and **SineFit.eps** to the assistant.

