L9: Rayleigh testi

- **Laskuharjoitus: Aikapisteet** `t` $= t_1, t_2, ..., t_n$ ovat sarake 1. tiedostossa `Rayleighdata.dat` ($n = 528$). **Laske ja plottaa** testattaville frekvenseille f_j Rayleigh testiparametri

$$z(f_j) = \{[\sum_{i=1}^{n} \cos 2\pi f_j (t_i - t_0)]^2 + [\sum_{i=1}^{n} \sin 2\pi f_j (t_i - t_0)]^2\}/n, \quad (1)$$

välillä `fmin` $= f_{min} = 1/P_{max}$ ja `fmax` $= f_{max} = 1/P_{min}$, missä `pmin` $= P_{min} = 1.^d5$ ja `pmax` $= P_{max} = 90.^d0$. **Testattavien frekvenssien** f_j välinen etäisyys `fstep` on

$$f_{step} = f_0/\mathrm{OFAC} \quad (2)$$

missä $f_0 = 1/\Delta T$, $\Delta T = t_n - t_1$ ja `ofac`$=$ OFAC $= 10$. Testattavien frekvenssien f_j **määrä** `M` välillä f_{min} ja f_{max} on

$$M = \mathrm{INT}[(f_{max} - f_{min})/f_{step}], \quad (3)$$

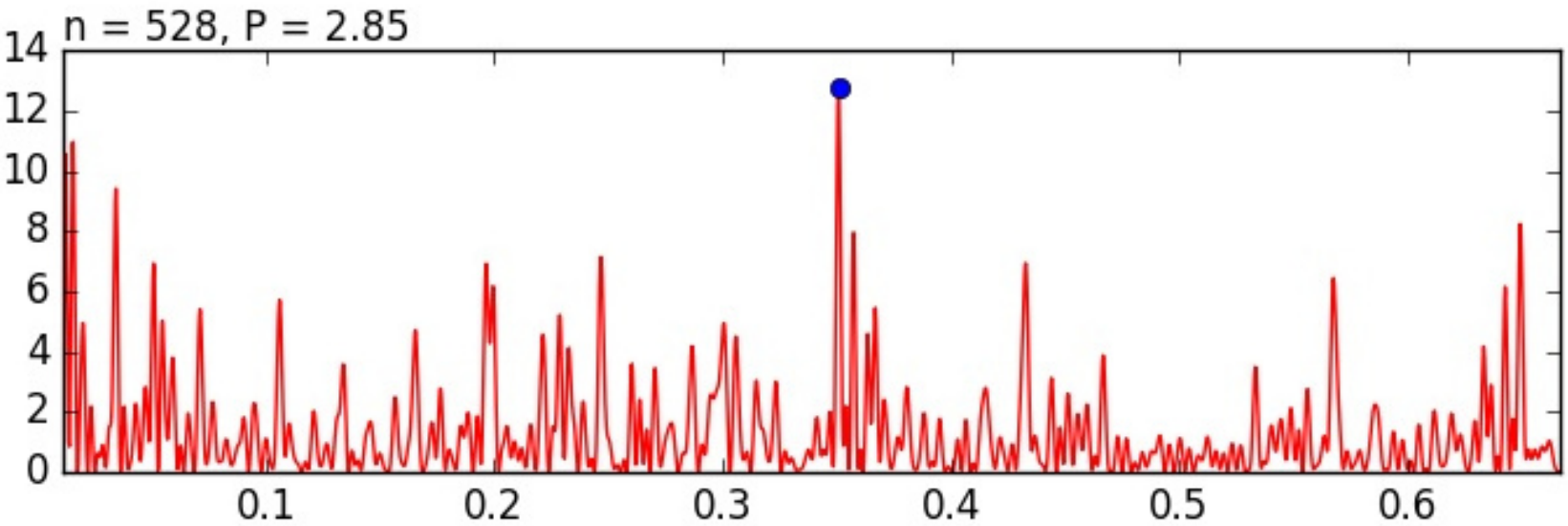
missä INT poistaa argumentin desimaaliosan (Esim: $\mathrm{INT}[12.34] = 12$).

- **Testattavat frekvenssit** ovat siis `f`$= f_j = f_{min} + j\, f_{step}$, missä $j = 0, 1, 2, 3, ..., M$
- **Vihje**: Kotisivun ohjelmassa `RayleighAliOhjelma1.py` laskettiin jo $z(f_j)$ yhdelle testattavalle frekvenssille

Tarkennuksia

1. Lue `t` vektori tiedostosta `Rayleighdata.dat`
2. Kiinnitä `pmin`$= 1.5$, `pmax`$= 90$, ja `ofac`$= 10$ arvot
3. Laske yllä mainituista arvoista muuttujien `fmin`, `fmax`, `fstep` ja `M` arvot
4. Luo `M`=M testattavaa frekvenssiä vektoriin `f`$= f_{\mathrm{j}}$
5. Luo "tyhjän periodogrammin" vektori `z=0.0*f`$= z(f_{\mathrm{j}}))$, jossa on yhtä monta komponenttia kuin `f`:ssä
6. Tee `f` looppi, joka laskee arvot $z(f_{\mathrm{j}})$ arvoille f_{j} ja tallentaa tuloksen vektoriin `z`
7. Plottaa `z` versus `f`
8. Ratkaise ja merkitse korkein piikki `zbest`$= z(f_{\mathrm{best}})$ kohdassa `fbest`$= f_{best}$. **Esimerkki** ratkaisu
   ```
   j=(z==max(z)).nonzero()
   fbest=f[j] ; zbest=z[j]
   ```
   selitetään seuraavalla sivulla
9. Kirjoita kuvaan $n = 528$ ja $P_{\mathrm{best}} = 1/f_{\mathrm{best}} = 2.85$ tiedot. Malli toivotusta lopputuloksesta on annettu seuraavalla sivulla.

- **Kuvassa** piste `(fbest,zbest)` on merkitty korkeimman `z` piikin huipussa sinisellä pallolla



L9: Etsintä vektorista: Valinta indeksien avulla

- `numpy.arange(3)` elementit ovat `[0 1 2]`
  - **Tapa 1**: Maksimiarvon indeksin etsintä
    `i=(a==max(a)) ; print(i)`
    `[False False True]`
  - Tarkistus `print(a[i])` antaa `[2]`. Ehto `a==max(a)` toteutuu elementille `a[2]=2`
  - **Tapa 2**: `j=(a==max(a)).nonzero() ; print(j)` antaa indeksin arvon `(array([2]),)`
  - Tarkistus `print(a[j])` tulostaa myös `[2]`. Ehto `a==max(a)` toteutuu indeksin arvolla `j=2` elementille `a[2]=2`
- **Etsintä kahdella ehdolla**: `k=((a>0) & (a<2))` antaa `[False True False]`. Ehdot `a>0` ja `a<2` vain elementillä `a[1]=1`
- **Logiikka**: `&` = **JA**, `|` = **TAI**
  `(a<1) | (a>2))` toteutuu vain arvolle `a[0]=0`
- **Laskuharjoituksen** korkein piikki komennoilla
  `j=(z==max(z)).nonzero()`
  `fbest=f[j] ; zbest=z[j]`

```
jetsu@dx5-flspa-02: ~
File  Edit  View  Search  Terminal  Help
>>> import numpy as np ; a=np.arange(3)
>>> print(a)
[0 1 2]
>>> i=(a==np.max(a))   ; print(i)
[False False  True]
>>> print(a[i])
[2]
>>> j=(a==max(a)).nonzero() ; print(j)
(array([2]),)
>>> print(a[i])
[2]
>>> k=((a>0) & (a<2)) ; print(k)
[False  True False]
>>> print(a[k])
[1]
>>> k=((a>0) | (a<2)).nonzero() ; print(k)
(array([0, 1, 2]),)
>>> print(a[k])
[0 1 2]
>>> k=((a<1) | (a>2)).nonzero() ; print(k)
(array([0]),)
>>> print(a[k])
[0]
>>> k=((a<1) | (a>2)) ; print(k)
[ True False False]
>>> print(a[k])
[0]
>>>
```

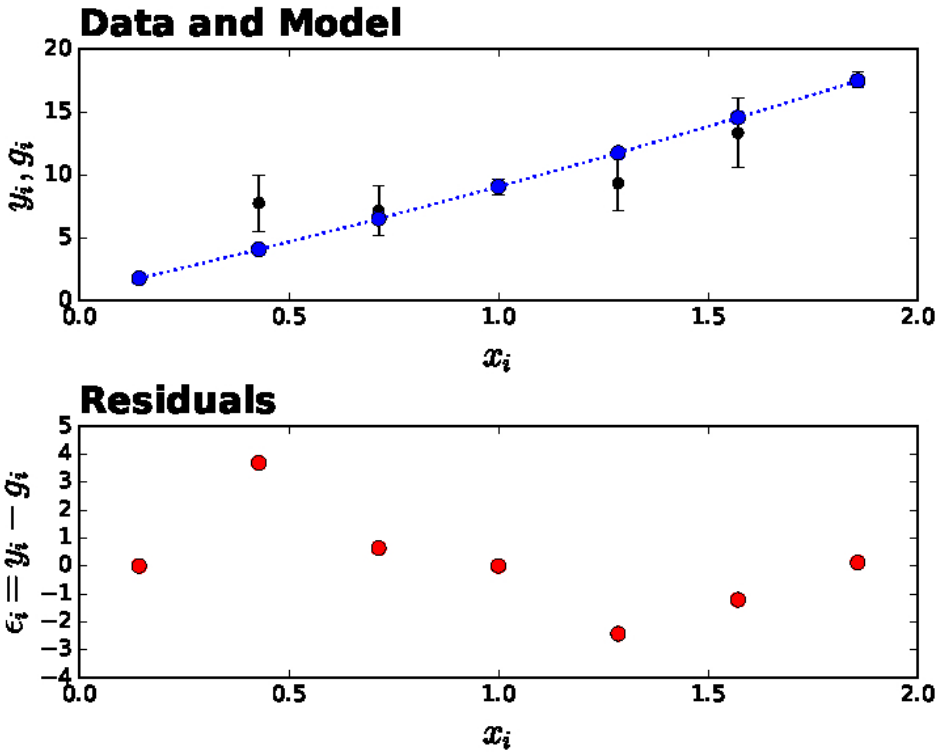
Pienimmän Neliösumman Sovitus (PNS)

n = havaintoa: **Kuvassa** mustat pallot

x_i = Havaintojen **paikat**

$y_\mathrm{i} = y(x_\mathrm{i})$ = Havaintojen **arvot**

σ_i = Havaintojen **tarkkuus**



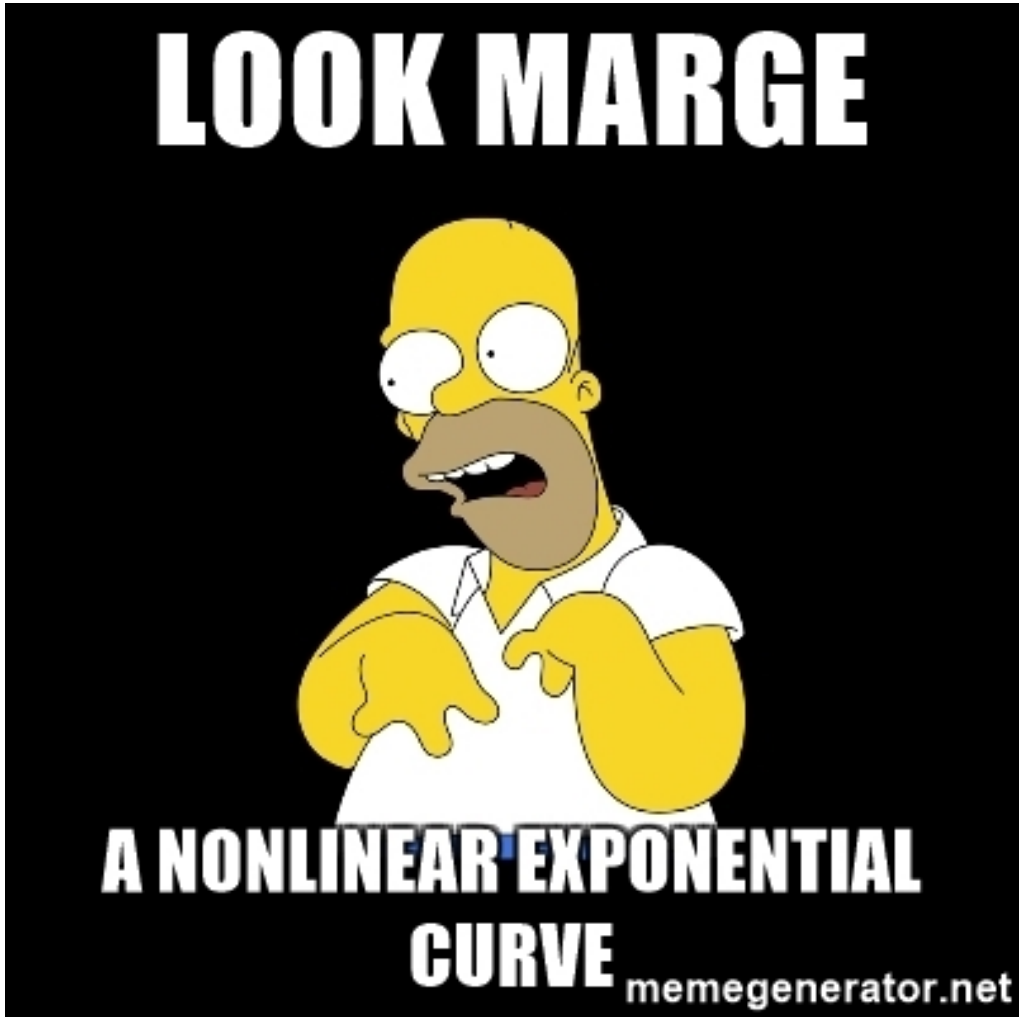
- **Malli** havainnoille $g(\bar{\beta}) = g(x, \bar{\beta})$
- Mallissa on Q **vapaata parametria** $\bar{\beta} = [\beta_1, \beta_2, \ldots, \beta_Q]$
- **Kuvan** malli on paraabeli $g(x, \bar{\beta}) = a0 + a1\, x + a2\, x^2$ eli $\bar{\beta} = [a0, a1, a2]$ eli $Q = 3$
- **Residuaalit** ovat $\epsilon_\mathrm{i} = y_\mathrm{i} - g(x_i, \bar{\beta}) = y_\mathrm{i} - g_\mathrm{i}$
  = **Havainnot** miinus **malli** = **mustat pallot** miinus **siniset pallot** = **punaiset pallot**
- **PNS** etsii **parhaat** $\bar{\beta}$ arvot $\beta_1, \ldots \beta_Q$, jotka **minimoivat**

$$\chi^2(\bar{\beta}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \left[\frac{y_\mathrm{i} - g(t_i, \bar{\beta})}{\sigma_\mathrm{i}}\right]^2 = \sum_{\mathrm{i}=1}^{\mathrm{n}} \left[\frac{y_\mathrm{i} - g_\mathrm{i}}{\sigma_\mathrm{i}}\right]^2 = \sum_{\mathrm{i}=1}^{\mathrm{n}} \left[\frac{\epsilon_i}{\sigma_i}\right]^2$$

- Summattu residuaalit jaettuina virheillään korotettuna toiseen potenssiin. Siitä nimi PNS.
- Hyvässä mallissa $\chi^2 \approx n$. **Miksi?** Hyvässä mallissa $n \gtrsim Q$. **Miksi?**

Malleista kuva: @memegenerator.net

- Malli $\bar{g}$ on **lineaarinen**, jos sen kaikki osoittaisderivaatat $\partial g/\partial \beta_i$ ovat riippumattomia kaikista vapaista parametreista β_j.
  Jos näin ei ole,
  malli on **epälineaarinen**.
- Esimerkki: **lineaarinen** malli
  $g(\bar{\beta}, x) = M + A \sin x + B \cos x$,
  $\bar{\beta} = [M, A, B]$ eli $Q = 3$
  $\partial g/\partial M = 1$,
  $\partial g/\partial A = \sin x$,
  $\partial g/\partial B = \cos x$
- Esimerkki: **epälineearinen** malli
  $g = A\mathrm{e}^{\mathrm{Bx}}$,
  $\bar{\beta} = [A, B]$ eli $Q = 2$
  $\partial g/\partial A = \mathrm{e}^{\mathrm{Bx}}$,
  $\partial g/\partial B = AB\mathrm{e}^{\mathrm{Bx}}$



- **Lineaarisilla** malleilla PNS antaa $\bar{\beta}$:lle yksikäsitteisen ratkaisun $\bar{\beta}_1$
- **Epälineaarisilla** malleilla $\bar{\beta}$:n ratkaisu $\bar{\beta}_1$ riippuu siitä, mikä annetaan (etsitään tavalla tai toisella) ratkaisun **alkuarvoksi** $\bar{\beta}_0$

python: Pienimmän neliösumman sovitus

- Data: **PNSdata1.dat** ($n = 3$)

```
1 2.42 0.9
2 5.47 1.6
3 7.24 0.8
```

```
# ____________________________________________________
# Tama on ohjelmani PNSmalli1.py
# - Pienimman neliosumman sovitus
# ____________________________________________________
import os              ; os.system('clear')   # Tyhjennetaan
import numpy as np ; import pylab as pl   # Importoidaan
from scipy.optimize import curve_fit      #-"-
# ____________________________________________________
def funct(x,a,b):
    g=a+b*x
    return g
# ____________________________________________________
file='PNSdata1.dat'                          #Data
x=np.loadtxt(file,skiprows=0,usecols=(0,)) #Data: x
y=np.loadtxt(file,skiprows=0,usecols=(1,)) #Data: y
e=np.loadtxt(file,skiprows=0,usecols=(2,)) #Data: e
beta0=np.ones(2) ; print('beta0=',beta0)    # Alkuarvo
beta1,covmatrix=curve_fit(funct,x,y,beta0,sigma=e)
print('beta1=',beta1)                        # Loppuarvo
pl.axes([0.15,0.15,0.70,0.80])               # paikka
pl.xlim([0,4]) ; pl.ylim([0,9])              # xy rajat
pl.errorbar(x,y,e,fmt='ok',markersize=9)     # Data plot
xx=0.5+np.arange(21)/20.*3                   # Mallin xx
g=beta1[0]+beta1[1]*xx ; pl.plot(xx,g)       # Malli g(xx)
pl.xlabel('$x_i$',fontsize=20)               # x-label
pl.ylabel('$y_i,g(x)$',fontsize=20)          # y-label
pl.savefig('PNSmalli1.jpg')                  # Kuva&Nayta
```

- Tekee kuvan **PNSmalli1.jpg** (oikealla)

- **python3 PNSmalli1.py** tulostaa

```
beta0= [ 1.  1.]
beta1= [ 0.10799829  2.4006929 ]
```

$\bar{\beta}_0 =$ **beta0** = sovituksen alkuarvo

$\bar{\beta}_1 =$ **beta1** = sovituksen loppuarvo

Lineaarinen malli: Loppuarvo **beta1** ei riipu alkuarvosta **beta0**

$\bar{x} = [x_1, x_2, x_3] =$ **x** = Datan paikka

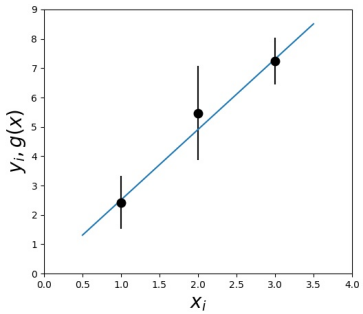
$\bar{y} = [y_1, y_2, y_3] =$ **y** = Datan arvot

$\bar{e} = [e_1, e_2, e_3] =$ **y** = Datan virheet

$\bar{g} = g(x, \bar{\beta}) = a + bx =$ **g** = Malli

$\bar{\beta} = [a, b] =$ Vapaat parametrit

curve_fit tarvitsee aliohjelman **funct**



python: Pienimmän neliösumman sovitus: ILMAN VIRHEITÄ!

- Data: **PNSdata1a.dat** ($n = 3$)

```
 1 2.42
 2 5.47
 3 7.24
```

```
# ____________________________________________________
# Tama on ohjelmani PNSmalli1a.py
# — Pienimman neliosumman sovitus kun VIRHEET puuttuu tms.
# ____________________________________________________
import os              ; os.system('clear')   # Tyhjennetaan
import numpy as np ; import pylab as pl      # Importoidaan
from scipy.optimize import curve_fit         #—"—
# ____________________________________________________
def funct(x,a,b):
    g=a+b*x
    return g
# ____________________________________________________
file='PNSdata1a.dat'                          # Data
x=np.loadtxt(file,skiprows=0,usecols=(0,))    # Data: x
y=np.loadtxt(file,skiprows=0,usecols=(1,))    # Data: y
e=np.ones(np.size(y))                         # Data: e puuttuu
beta0=np.ones(2) ; print('beta0=',beta0)      # Alkuarvo
beta1,covmatrix=curve_fit(funct,x,y,beta0,sigma=e)
print('beta1=',beta1)                         # Loppuarvo
pl.axes([0.15,0.15,0.70,0.80])                # paikka
pl.xlim([0,4]) ; pl.ylim([0,9])               # xy rajat
pl.plot(x,y,'ok',markersize=9)                # Data plot
xx=0.5+np.arange(21)/20.*3                    # Mallin xx
g=beta1[0]+beta1[1]*xx ; pl.plot(xx,g)        # Malli g(xx)
pl.xlabel('$x_i$',fontsize=20)                # x—label
pl.ylabel('$y_i,g(x)$',fontsize=20)           # y—label
pl.savefig('PNSmalli1a.jpg')                  # Kuva&Nayta
```

- Tekee kuvan **PNSmalli1a.jpg** (oikealla)

- **python3 PNSmalli1a.py** tulostaa

```
beta0= [ 1.  1.]
beta1= [ 0.22333333  2.41      ]
```

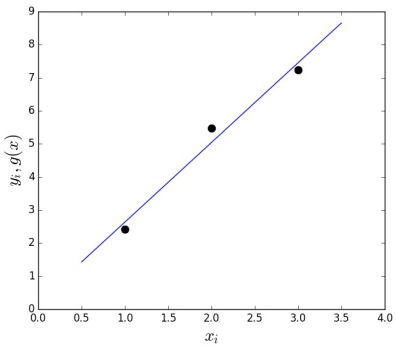
$\bar{\beta}_0 =$ **beta0** = sovituksen alkuarvo

$\bar{\beta}_1 =$ **beta1** = sovituksen loppuarvo

Lineaarinen malli: Loppuarvo **beta1** ei riipu alkuarvosta **beta0**

Ohjelman **PNSmalli1.py** lopputulos **ei ole sama** kuin ohjelman **PNSmalli1.py** lopputulos, vaikka molemmissa sama data **x** ja **y**

Miksi?



Virheet e tunnettu

```
beta1,covmatrix=curve_fit(funct,x,y,beta0,sigma=e)
```

Virheet e ei tunnettu
tai samat

```
e=np.ones(np.size(y))

beta1,covmatrix=curve_fit(funct,x,y,beta0,sigma=e)
```

python: Pienimmän neliösumman sovitus

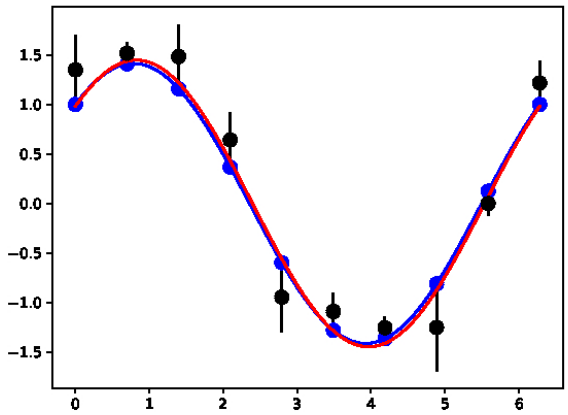
```
# ------------------------------------------------------------
# Tama on ohjelmani PNSmalli2.py
# - Pienimman neliosumman sovitus simuloituun dataan y=y(x)+e
# ------------------------------------------------------------
import os ; os.system('clear')              # Tyhjennetaan naytto
import numpy as np  ; import pylab as pl # Importoidaan
from scipy.optimize import curve_fit        #-"-
# ------------------------------------------------------------
def funct(x,a,b):
    g=a*np.cos(x)+b*np.sin(x)
    return g
# ------------------------------------------------------------
n = input('Anna datan maara = n = ')  ; n=int(n)
s = input('Anna datan virhe = s = ')  ; s=float(s)
# ------------------------------------------------------------
x=2.*np.pi*np.arange(1.*n)/(n-1)          # Simuloidun datan x
beta0=np.ones(2)                          # Alkuarvo beta0
g0=beta0[0]*np.cos(x)+beta0[1]*np.sin(x)  # Alkuperainen malli
e=s*np.random.normal(0.,s,n)              # Gaussian error
y=g0+e                                    # Keinotekoisen datan y
beta1,covmatrix=curve_fit(funct,x,y,beta0,sigma=e) # PNS
print('Alku  beta0 = ', beta0) ; print('Loppu beta1 = ', beta1)
xx=2.*np.pi*(np.arange(101.)/100)         # Sovituksen xx
gg=beta0[0]*np.cos(xx)+beta0[1]*np.sin(xx)# Alkuperainen malli
yy=beta1[0]*np.cos(xx)+beta1[1]*np.sin(xx)# Sovituksen malli
# ------------------------------------------------------------
pl.plot(x,g0,'ob',markersize=9) # Alkuperainen malli: siniset pisteet
pl.plot(xx,gg,'b',markersize=9) # Alkuperainen malli: sininen viiva
pl.errorbar(x,y,e,fmt='ok',markersize=9) # Simuloitu data: mustat pisteet
pl.plot(xx,yy,'r')   # Simuloituun sovitettu malli: punainen viiva
# ------------------------------------------------------------
pl.savefig('PNSmalli2.eps')               # Kuva tiedosto
```

PNSmalli2.py tulostaa

```
Anna datan maara = n = 10
Anna datan virhe = s = 0.5
beta0 =  [ 1.  1.]
beta1 =  [ 0.97844502  1.23098811]
```

sekä tekee alla näkyvän kuvan **PNSmalli2.eps**

- **Malli** $g(x, \bar{\beta}) = a \cos x + b \sin x$
- **Vapaat parametrit** $\bar{\beta} = [a, b]$
- Kokeillaan eri **n** ja **s** yhdistelmiä
- Tulos on aina erilainen
- **Mitä tästä opitaan?**



python: Pienimmän neliösumman sovitus

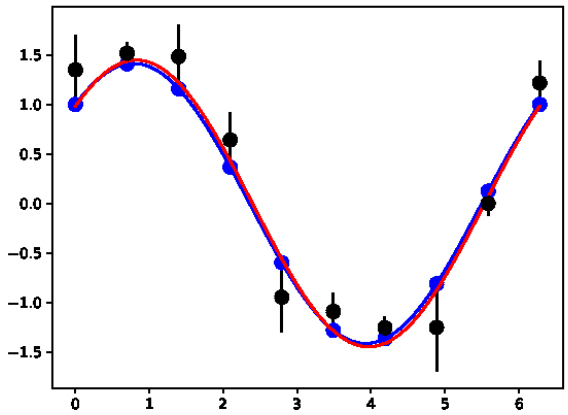
```
# ------------------------------------------------------------
# Tama on ohjelmani PNSmalli2.py
# - Pienimman neliosumman sovitus simuloituun dataan y=y(x)+e
# ------------------------------------------------------------
import os ; os.system('clear')              # Tyhjennetaan naytto
import numpy as np  ; import pylab as pl # Importoidaan
from scipy.optimize import curve_fit        #-"-
# ------------------------------------------------------------
def funct(x,a,b):
    g=a*np.cos(x)+b*np.sin(x)
    return g
# ------------------------------------------------------------
n = input('Anna datan maara = n = ')  ; n=int(n)
s = input('Anna datan virhe = s = ')  ; s=float(s)
# ------------------------------------------------------------
x=2.*np.pi*np.arange(1.*n)/(n-1)          # Simuloidun datan x
beta0=np.ones(2)                          # Alkuarvo beta0
g0=beta0[0]*np.cos(x)+beta0[1]*np.sin(x)  # Alkuperainen malli
e=s*np.random.normal(0.,s,n)              # Gaussian error
y=g0+e                                    # Keinotekoisen datan y
beta1,covmatrix=curve_fit(funct,x,y,beta0,sigma=e) # PNS
print('Alku  beta0 = ', beta0) ; print('Loppu beta1 = ', beta1)
xx=2.*np.pi*(np.arange(101.)/100)         # Sovituksen xx
gg=beta0[0]*np.cos(xx)+beta0[1]*np.sin(xx)# Alkuperainen malli
yy=beta1[0]*np.cos(xx)+beta1[1]*np.sin(xx)# Sovituksen malli
# ------------------------------------------------------------
pl.plot(x,g0,'ob',markersize=9) # Alkuperainen malli: siniset pisteet
pl.plot(xx,gg,'b',markersize=9) # Alkuperainen malli: sininen viiva
pl.errorbar(x,y,e,fmt='ok',markersize=9) # Simuloitu data: mustat pisteet
pl.plot(xx,yy,'r')   # Simuloituun sovitettu malli: punainen viiva
# ------------------------------------------------------------
pl.savefig('PNSmalli2.eps')               # Kuva tiedosto
```

PNSmalli2.py tulostaa

```
Anna datan maara = n = 10
Anna datan virhe = s = 0.5
beta0 =  [ 1.  1.]
beta1 =  [ 0.97844502  1.23098811]
```

sekä tekee alla näkyvän kuvan **PNSmalli2.eps**

- **Malli** $g(x, \bar{\beta}) = a \cos x + b \sin x$
- **Vapaat parametrit** $\bar{\beta} = [a, b]$
- Kokeillaan eri **n** ja **s** yhdistelmiä
- Tulos on aina erilainen
- **Mitä tästä opitaan?**



python: Pienimmän neliösumman sovitus

- Toinen näyttö: kotisivun `PNSmalli3.py`
- Loopin `for i in range(4):` **sisällä**
- **Plotattavat käyrät** `yy` **ja** `gg`

  `xx` $= 0 \leq xx_1, \ldots, xx_{101} \leq 2\pi =$ tiheä tasavälinen plottien argumentti

  `yy` = $g(xx, \bar{\beta}_1)$ sovitetty käyrä

  Kuvat: `yy` = punainen viiva

  `gg` = $g(xx, \bar{\beta}_{SIM})$ simulointimallin käyrä

  Kuvat: `gg` = sininen viiva
- **Varsinaiset plotit**

  `pl.axes([z1[i],z2[i],lev,kor])` = neljän **kuvan** paikat

  `pl.xlim([x1,x2])` = x–rajat

  `pl.ylim([y1,y2])` = y–rajat

- Loopin `for i in range(4):` **sisällä**

  `pl.plot(x0,g0,'ob')` = siniset ympyrät

  `pl.plot(xx,gg,'b')` = sin... jatkuva viiva

  `pl.errorbar(x0,y0,e0,fmt='or')` = punaiset ympyrät ja virheet

  `pl.plot(xx,yy,'r')` = pun... jatkuva viiva

  `txt1='...` = formatoitu stringi

  `pl.text(x1...` = stringin tulostus kuvaan
- Loopin `for i in range(4):` **jälkeen**

  `pl.savefig('PNSmalli3.eps')` kuvan tallennus `eps` muodossa
- **loopin sisällä muuttuvat vain** `n0=n[i]`, `s0=s[i]` ja `pl.axes([z1[i],z2[i],lev,kor])`
- Kuva `PNSmalli3.eps` seuraavalla sivulla

python: Pienimmän neliösumman sovitus

`PNSmalli3.eps`

- $n = 10$, $\sigma = 0.1$

  Vähän tarkkaa dataa

  PNS tulos OK
- $n = 10$, $\sigma = 1.0$

  Vähän epätarkkaa dataa

  PNS tulos “sinne päin”
- $n = 30$, $\sigma = 1.0$

  Paljon epätarkkaa dataa

  PNS tulos OK
- $n = 30$, $\sigma = 10.0$

  Paljon huonoa dataa

  PNS tulos “sinne päin”
- Sovita väärä malli tähän dataan (esim. suora)
  $\Rightarrow$ **“Musta laatikko”**