

```

#rotina porca para determinar um ajuste gaussiano de dados

import math
import numpy
from numpy import *

def gaussian(r,ro,s):
    y=(r-ro)/(s)
    p=math.exp(-y*y/2.0)
    return p

def estimative(x,y):
    A= max(y) #maximo mais provavel
    xi=min(numpy.where(x>=A))[0]
    x0=x[xi] #posisao do maximo
    s=abs(x[0] - x[-1])
    for i in xrange( 0,xi):
        if y[i] >= A/2:
            s= abs(x[i] - x0 ) # largura
    m=sum(y)/len(y) # valor medio
    return A,x0,s,m

def gauss_non_linear(x,y, max_inter=40 ):
    # Return values: A , x0 , s , m
    # y(x) = A*Exp( ((x-x0)^2)/( 2* s^2) ) +m
    x=numpy.array(x)
    y=numpy.array(y)
    #ajusta uma gaussiana sobre x,y
    #parametros a serem ajustados A,x0,s
    A,x0,s,m = estimative(x,y)
    X= array([A,x0,s,m+0 ])
    sh= (len(X), len( y ))
    jm= zeros(sh)
    repeat = True
    q=0.1
    dn=0.01
    nn=0
    while repeat:
        y0=array([ X[3]+ X[0]*gaussian(k,X[1],X[2] ) for k in x ])
        err=sum( (y0 - y)*( y0-y) )
        jm= zeros(sh)
        for i in range(sh[0]):
            N=copy(X)
            N[i]+= dn
            y1=array([ N[0]*gaussian(k,N[1],N[2] )+N[3] for k in x ])
            for j in range(sh[1]):
                jm[i,j]= (y1[j] - y0[j])/dn
        jm=jm.transpose()
        u=dot(jm.transpose(),jm)
        ui= numpy.linalg.inv(u + ( q )*numpy.identity(len(X) ) )
        p2= dot( jm.transpose() , y-y0 )
        p1= ui
        dX=dot( ui,p2)
        X2= X + dX
        y2=array([ X2[0]*gaussian(i,X2[1],X2[2] )+X2[3] for i in x ])
        nn+=1
        err2=sum( (y2 - y)*( y2-y) )
        if nn > max_inter: repeat =False
        if err2 > err:
            q=q*10
            if q > 1e4 : q=1e4
        else:
            q=q/10
            if abs(err2 -err)/err <0.001 :
                repeat=False
            X=copy(X2)
    return X

```

```

def two_gauss_non_linear(x,y,max_inter=40 ):
    x=numpy.array(x)
    y=numpy.array(y)
    #ajusta uma gaussiana sobre x,y
    #parametros a serem ajustados A,x0,s
    A,x0,s,m = gauss_non_linear(x,y,max_inter) #fist try
    y0=array([ A*gaussian(k,x0,s ) for k in x ])
    y2= y- y0
    A2,x02,s2,m2 = estimative(x,y2)
    X= array([A,x0,s, A2,x02,s2,m+m2 ])
    sh= (len(X), len( y ))
    jm= zeros(sh)
    repeat = True
    q=0.1
    dn=0.01
    nn=0
    while repeat:
        y0=array([ X[0]*gaussian(k,X[1],X[2]) + X[3]*gaussian(k,X[4],X[5] )+X[6] for k
in x ])
        err=sum( (y0 - y)*( y0-y) )
        jm= zeros(sh)
        for i in range(sh[0]):
            N=copy(X)
            if random.random() < 0.5: dn=-dn #optional
            N[i]+= dn
            y1=array([ (N[0]*gaussian(k,N[1],N[2]) + N[3]*gaussian(k,N[4],N[5]) +N[6]) for k
in x ])
            for j in range(sh[1]):
                jm[i,j]= (y1[j] - y0[j])/dn
            jm=jm.transpose()
            u=dot(jm.transpose(),jm)
            ui= numpy.linalg.inv(u + ( q )*numpy.identity(len(X) ) )
            p2= dot( jm.transpose() , y-y0 )
            p1= ui
            dX=dot( ui,p2)
            X2= X + dX
            y2=array([ X2[0]*gaussian(i,X2[1],X2[2] ) + X2[3]*gaussian(i,X2[4],X2[5] )+X2[6]
for i in x ])
            err2=sum( (y2 - y)*( y2-y) )
            #print "ERR2=", err2
            nn+=1
            if nn > max_inter: repeat =False
            if err2 > err:
                q=q*10
                if q > 1e2:
                    q=1e2
                dn=dn*2 #optional
            else:
                X=copy(X2)
                q=q/10.0
                dn=dn/2 #optional
                if abs(err2 -err)/err <0.001 : # 0.001 is the variable, change if you wish.

                    repeat=False
    return X

```