

opdracht fitten

November 21, 2018

Opdracht fitten data

Pieter luyten

r0708257

De dataset die in dit document gebruikt wordt is “voormiddag_Data12.txt”. We willen volgende vergelijking fitten door deze data:

$$\mathcal{L}(x|\mu, \gamma, \delta, A) = A \cdot \frac{\gamma}{((x - \mu)^2 + \gamma^2)} + \delta$$

Dit geef de fit in figuur 1 voor de data

Figuur 1: de data met fit

De waardes van de parameters voor deze beste fit zijn:

$$\mu = 0.3 \pm 0.3$$

$$\gamma = 3.0^{0.3}_{-0.2}$$

$$\delta = 96.8 \pm 1.2$$

$$A = (32 \pm 2) \cdot 10$$

De fouten op de gefitte parameters komen uit de χ^2 -verdeling van deze parameters, het is de afstand van het minimum waarop de waarde van de χ^2 verdeling juist 1 meer is dan in het minimum zoals je kan zien in figuur 2. De minimale waarde is 58,6 met een p-waarde van 73%. Het is dus aannemelijk dat de fit een correct model is van de data

Figuur 2: De verdeling van de parameters rond hun minimum

1 De code

1.1 Importeren packages en definiëren functies

```
In [1]: import numpy as np
import scipy.stats as nst
import scipy.optimize as opt
import pandas as pd
import matplotlib.pyplot as plt
```

1.1.1 definiëren functie om fit te berekenen

```
In [2]: #
        # hulpfuncties voor de fit-functie
        #

        # functie die de Least-squares functie uitrekent
        # f is de te fitten functie, x en y de gemeten waarden met sigma de fout op y
        # (normaal verdeeld). theta zijn de parameters voor f
        # die geoptimaliseerd moeten worden
        def LS(theta, f, x, y, sigma):
            return sum([(y[i] - f(x[i], theta))*2/sigma[i]**2 for i in range(len(x))])

        # de vectorversie van LS, hierin moet theta een matrix zijn met als rijen de
        # verschillende waarden voor de vector theta
        def LS_vectorized(theta, f, x, y, sigma):
            Y = np.array([y_val for t in theta for y_val in y])
            X = np.array([x_val for t in theta for x_val in x])
            S = np.array([s_val for t in theta for s_val in sigma])
            squares = (Y - f(X, (theta[:,0], theta[:,1], theta[:,2], theta[:,3])))**2 / (S**2)
            return np.sum(squares, axis=0)

        # minimaliseert de least-squares functie om de optimale
        # parameters theta voor f te vinden
        # zodat f een zo goed mogelijke fit is door de datapunten (x_i, y_i)
        def find_theta(f, theta0, x, y, sigma):
            return opt.minimize(LS, theta0, args=(f, x, y, sigma))

        # maakt van de LS functie een 1D functie in de variabele theta_index LS_i(t) en
        # evalueert deze in de vector T
        def LS_i_vec(T, LS_vec, theta, index, f, x, y, sigma):
            dt = np.zeros_like(theta)
            dt[index] = 1
            THETA = np.array([theta + dt*t for t in T])
            return LS_vec(THETA, f, x, y, sigma)

        # de functie die van LS een functie maakt in 1 variabele, -val zodat de punten
        # waar LS gelijk is aan val nulpunten worden
        def LS_i(t, LS, theta, index, f, x, y, sigma, val):
            dt = np.zeros_like(theta)
            dt[index] = t
            return LS(theta+dt, f, x, y, sigma) - val

        # zoekt de standaardafwijkingen op theta_index in de functie chi
        # van de minimale waarde chi(theta)
        def find_sigma(chi, theta, index, f, x, y, sigma, delta, val):
            # los de vergelijking op
```

```

    args = (chi, theta, index, f, x, y, sigma, val)
    dt1 = opt.fsolve(LS_i, delta, args=args)
    dt2 = opt.fsolve(LS_i, -delta, args=args)
    return (dt2, dt1)

# -----
# de fit functie

# fit de functie f door de datapunten (x_i, y_i) met fout dy op y
# met de least-squares methode. theta0 is een eerste gok voor de parameters
# van f. delta is een gok op de procentuele afwijking
# van de gefitte parameters (array met zelfde dimensies als theta0)
def fit_function(f, x, y, dy, theta0, delta):
    # zoek de optimale waarde voor theta
    theta = find_theta(f, theta0, x, y, dy)
    minimum = theta['fun']
    # bereken de fout op theta
    sigma = [find_sigma(LS, theta['x'], i, f, x, y, dy, delta[i]*theta['x'][i], minimum)
              for i in range(len(theta['x']))]

    # bereken de p-waarde van de fit
    vrijheid = len(x)-len(theta0)
    p = 1-nst.chi2.cdf(theta['fun'], vrijheid)
    return {'param' : theta['x'], 'stdv' : sigma, 'p' : p}

```

1.1.2 te fitten functie

```

In [3]: def lorentz(x, param):
        (mu, gamma, delta, A) = param
        return A * gamma/((x-mu)**2 + gamma**2) + delta

```

1.2 importeren data

```

In [4]: data = pd.read_csv("voormiddag_Data12.txt", header=None, sep=" ")
        x = data[0].values
        I = data[1].values
        dI = np.sqrt(I)
        names = ["\mu", "\gamma", "\delta", "A"]

```

1.3 fitten functie

```

In [5]: fit = fit_function(lorentz, x, I, dI, (0, 1, 1, 1), (0.2, 0.2, 0.2, 0.2))

In [6]: fig, ax = plt.subplots(1,1,figsize=(8, 6))

        ax.errorbar(x, I, dI, fmt='o', capsize=3, label="datapunten")
        X = np.linspace(-40, 40, 1000)
        ax.plot(X, lorentz(X, fit['param']), label="fit")

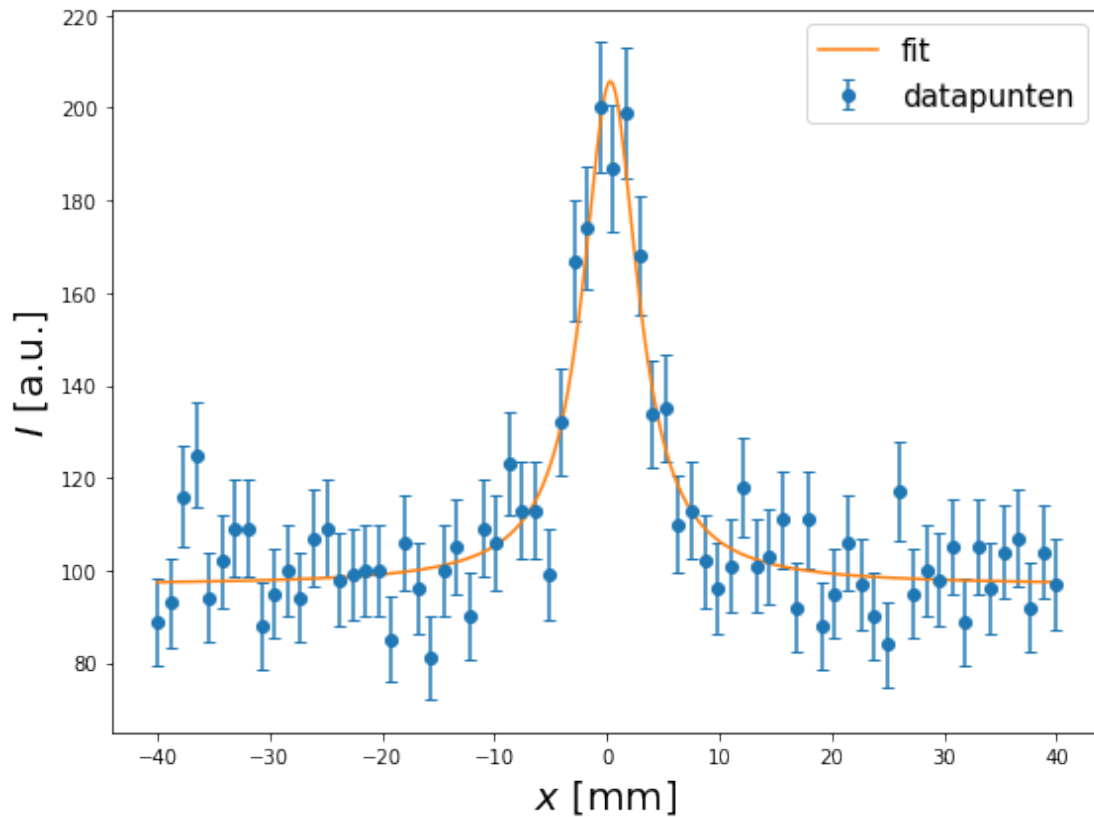
```

```

ax.set_xlabel('$x$ [mm]', size=20)
ax.set_ylabel('$I$ [a.u.]', size=20)
plt.legend(fontsize=15)
fig.tight_layout()

plt.show()
fig.savefig("fit.png", dpi=600)

```



1.4 plots chi2 functie

```

In [7]: fig = plt.figure(figsize=(12, 10))
        mini = LS(fit['param'], lorentz, x, I, dI)

        for i in range(4):
            # add subplot
            ax = fig.add_subplot(2,2,i+1)
            # initialize data to plot
            t = fit['param'][i]
            dtn = fit['stdv'][i][0][0]
            dtp = fit['stdv'][i][1][0]

```

```

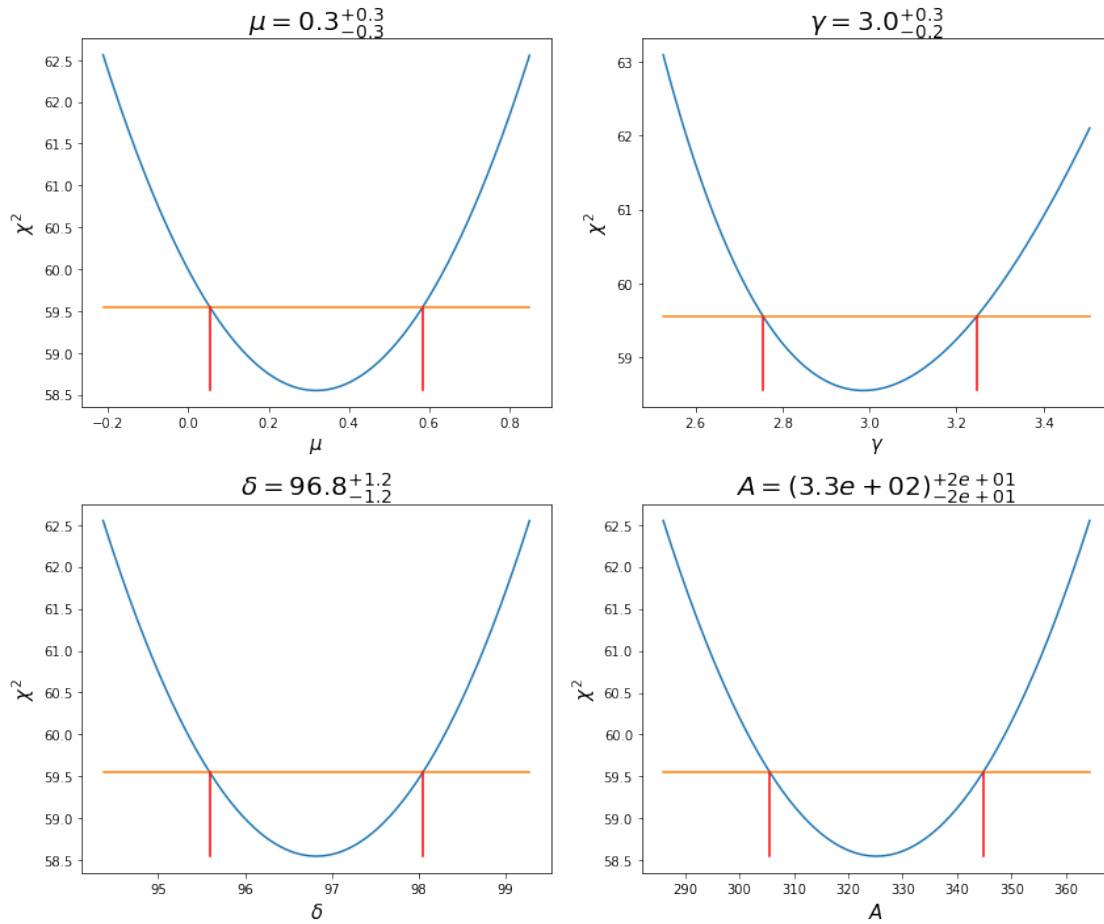
dX = np.linspace(2*dtn, 2*ntp, 200)
Y = LS_i_vec(dX, LS_vectorized, fit['param'], i, lorentz, x, I, dI)
X = t*np.ones_like(dX) + dX

# plot the data
ax.plot(X, Y)
ax.plot(X, (mini+1)*np.ones_like(X))
ax.plot([t+dt, t+dt], [mini, mini+1], color='r')
ax.plot([t+ntp, t+ntp], [mini, mini+1], color='r')

# lay-out and labels of the plot
ax.set_title("$s = %.1f_{%.1f}^{+%.1f}$" \
             %(names[i], t, dt, ntp), size=20)
ax.set_xlabel("$s" %names[i], size=15)
ax.set_ylabel("$\chi^2$", size=15)

ax = plt.gca()
ax.set_title("$s = (%2.1e)_{%2.0e}^{+%2.0e}$" \
             %(names[i], t, dt, ntp), size=20)
fig.tight_layout()
plt.show()
fig.savefig("chi2s.png", dpi=600)

```



1.5 print parameters met fout in latex code

```
In [8]: for i in range(4):
        print("%s: %.1f_{%.1f}^{%.1f}" %(names[i], fit['param'][i], \
                                         fit['stdv'][i][0][0], fit['stdv'][i][1][0]))
```

```
\mu: 0.3_{-0.3}^{0.3}
\gamma: 3.0_{-0.2}^{0.3}
\delta: 96.8_{-1.2}^{1.2}
A: 325.2_{-19.6}^{19.6}
```

```
In [9]: for i in range(4):
        print("%s: %.1f \pm %.1f" %(names[i], fit['param'][i], fit['stdv'][i][1][0]))
```

```
\mu: 0.3 \pm 0.3
\gamma: 3.0 \pm 0.3
\delta: 96.8 \pm 1.2
A: 325.2 \pm 19.6
```

```
In [10]: fit['stdv']
```

```
Out[10]: [(array([-0.26563882]), array([0.26594516])),  
          (array([-0.22991659]), array([0.25992544])),  
          (array([-1.22830267]), array([1.22830275])),  
          (array([-19.57804298]), array([19.57792733]))]
```

```
In [11]: display(fit['p'], mini)
```

0.7310094656298058

58.55086634925484