1.
A) (In python)

B) Als er geen B was geweest zouden de punten een stijgende lijn rondom 0 of dalende lijn rondom de 0 hebben. Dat gebeurt nu niet en dus is het verstandiger om y = ax+b toe te passen.

C) In curvefitgui komt de volgende info naar boven: value: 3.3241195246 g/(cm^3) & stderr: 1.1927199018e-02 g/(cm^3). Oftewel rho = 3,324 +- 0,011 g/(cm^3)

D) in curvefitgui komt de volgende info naar boven: value: 8.5510732032e-01 g & stderr: 1.3475082074e-01 g. Oftewel m = 0,86 +- 0.13g

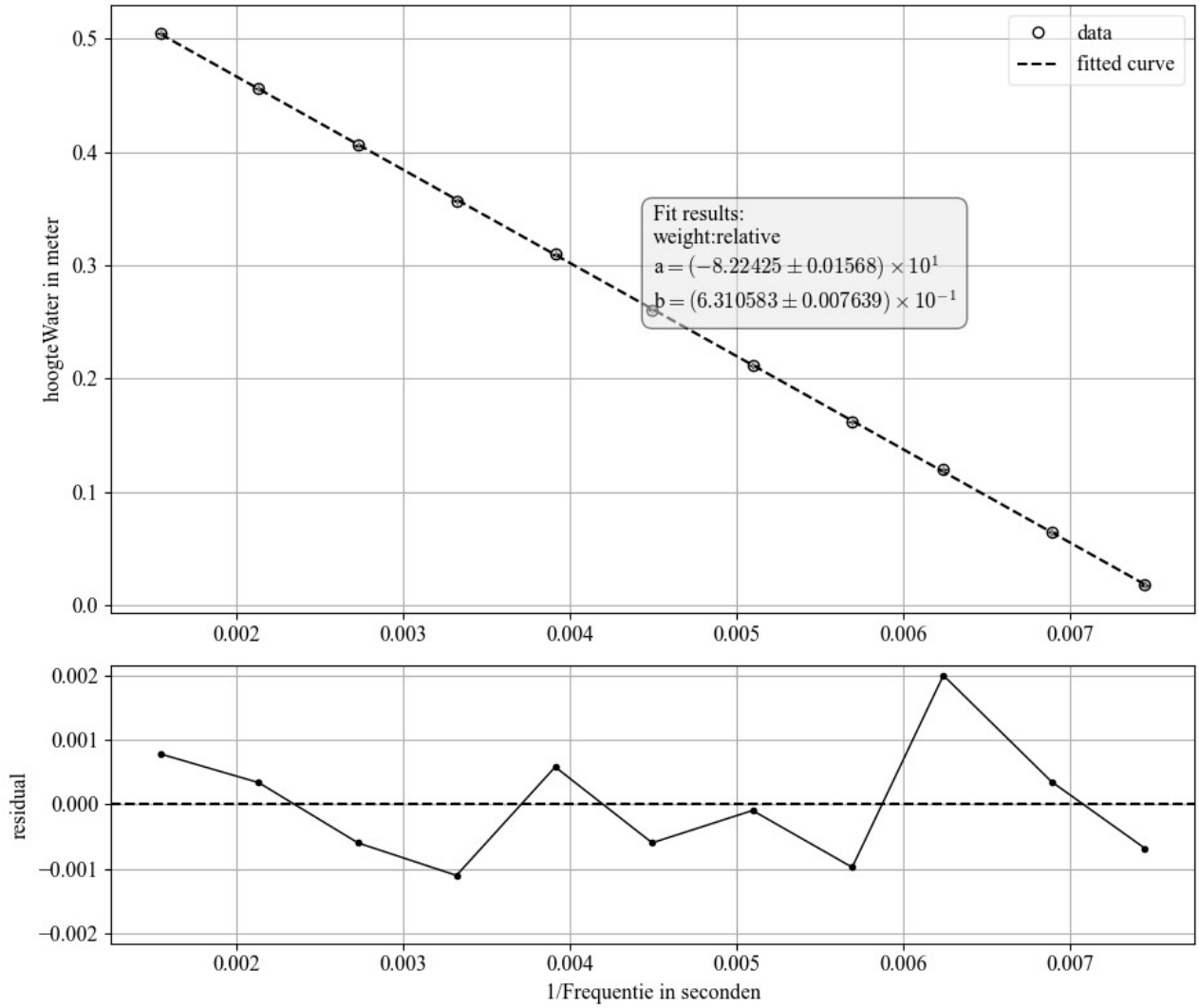
E)Omdat je dan voor a de dichtheid kan nemen. Als je dit niet doet krijg je 1/a en dat is onhandiger af te lezen.

F) S_min = 4.0228921059e-01, volgens curvefitgui. Vervolgens is Smin = 12 – 2, want er zijn 12 datapunten en 2 parameters. Wi is dan 10/4.0228921059e-01 = 24,8578 en dat betekent dat dm = 1/sqrt(24,8578) = 0.2

G) (In python)

H) None en relative zijn gelijk want bij none is er geen foutmarge in de error en bij relative is die er ook niet omdat overal de foutmarge 2 is. Bij absolute is de error 10x zo groot. Dit komt omdat hier de standaard deviatie wordt meegenomen en de uitersten best ver van de gemiddelde lijn liggen.

2. Snelheid van het geluid is (-8.22425 *10)*-4 = 328.97 m/s en dat onbekende stukkie is 0.631 – 0.612 = 0.0191 m. M’n foutmarges in mn programmatje zijn stuk dus als je kunt checken wat er daar mis is kan ik dat ook wel ff fixen <3



$$$$$$$$$$$$$$$$$$4
Hieronder staan trouwens nog mn code en een paar super vette fitjes en in dat andere pdfje een mooie berekening ding voor opgave 2 :D

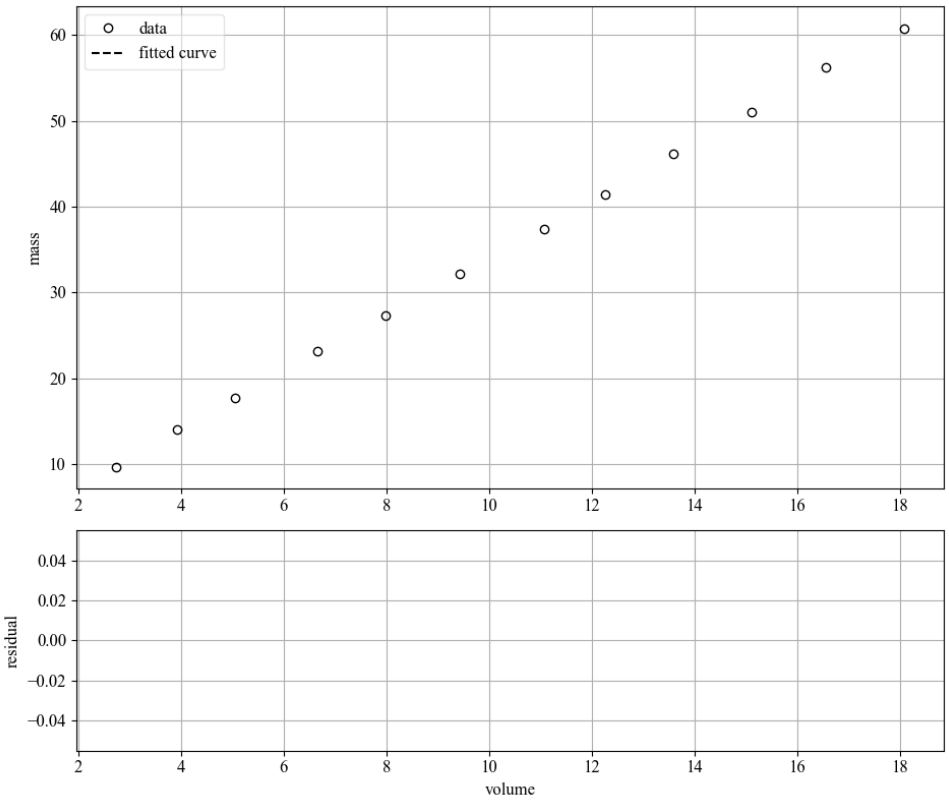
Code opgave 1:

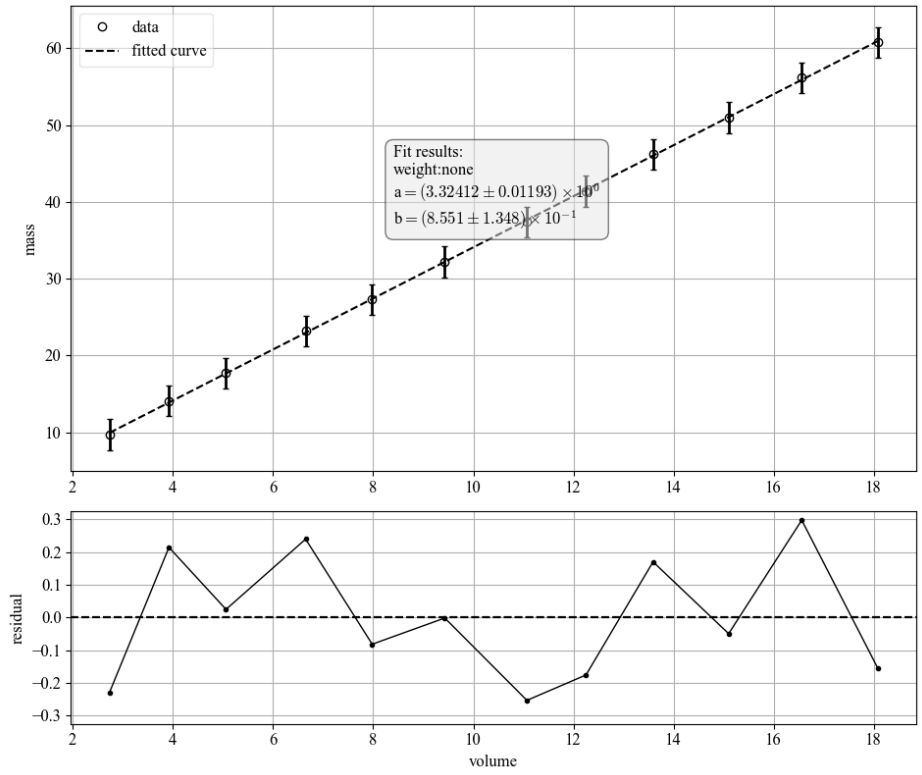
```
import numpy as np
import curvefitgui as cfg
from scipy.stats import sem

volume = np.array(
    [2.73, 3.92, 5.06, 6.65, 7.98, 9.43, 11.07, 12.25, 13.59, 15.1, 16.56, 18.08]
)  # cm3
mass = np.array(
    [9.7, 14.1, 17.7, 23.2, 27.3, 32.2, 37.4, 41.4, 46.2, 51.0, 56.2, 60.8]
)  # g

print(sem(mass))

mylist = []
for i in mass:
    mylist.append(2)
err = np.array(mylist)

print(err)
cfg.linear_fit_gui(volume,mass, xlabel='volume', ylabel='mass', yerr=err)
```

Code opgave 2:

```
# Consider the following experiment in which we want to measure the speed of sound. In a
# cylinder, that is open on one side and contains some water, we excite a resonance by blowing
# across the opening (see figure). The frequency of the produced sound is measured. This
# procedure is repeated for different amounts of water. The acquired data is collected in the
# table. When we blow across the opening a standing wave is created that has a node (zero
# amplitude) at the water-air interface and an anti-node (maximum amplitude) just above the
# opening of the cylinder. The distance between node and anti-node is 1 / 4 λ, with λ the wavelength
# of the produced sound. The anti-node is an unknown distance d above the opening. If the
# volume of water is increased, sound with a higher pitch is produced as the distance between
# node and anti-node is reduced. The speed of sound v is related to the frequency (pitch) f and
# the wavelength λ as: v = f λ.

# Assignment: Determine the speed of sound v and the distance d using the data given. Also
# include their standard errors. Make sure you do all the checks to determine if the fit is valid.

import numpy as np
import curvefitgui as cfg

hoogteWater = np.array(
    [0.018, 0.065, 0.120, 0.162, 0.212, 0.261, 0.310, 0.357, 0.406, 0.456, 0.505]
)  # m
frequentie = np.array(
    [134.3, 145.2, 160.3, 175.7, 196.3, 222.6, 255.7, 301.3, 366.4, 468.9, 648.4]
)  # Hz

hwl = 0.001 / np.sqrt(len(hoogteWater))
fsl = 0.1 / np.sqrt(len(frequentie))

hoogteWater_sigma = np.ones(len(hoogteWater)) * hwl  # m
frequentie_sigma = fsl / frequentie**2  # Hz

print(hwl)
print(fsl)


cfg.linear_fit_gui(
    1 / frequentie,
    hoogteWater,
    xlabel="1/Frequentie in seconden",
    ylabel="hoogteWater in meter",
    xerr=frequentie_sigma,
    yerr=hoogteWater_sigma,
)
```

Fitjes:


data
fitted curve
mass
residual
volume


data
fitted curve
Fit results:
weight:none
mass
residual
volume