PH2255 Course:
Introduction to Statistical Methods
Exercise 3

Thomas Bass

29 January 2021

Abstract

Exercise 3 of PH2255 introduces us to another real-world historical application of the methods we have learned. By applying the χ^2_{min} method to one of Ptolemy's experiments - investigating the refraction of light through an air-water medium - we can compare it to a common hypothesis at the time, the hypothesis Ptolemy used, and one first discovered by Ibn Sahl in the 10th Century (later re-discovered by Snell in 1621).

First, we start by ingesting the data given in the lab script, using Numpt's np.array for easy data manipulation later on, especially with trig functions. The errors for each measurement were not recorded, so we assume $\sigma = \frac{1}{2}$ deg for θ_r, as Ptolemy's measurements were all given in half-degree increments. We do not use errors on θ_i, as such errors would be absorbed into θ_r

```
t_i = np.array([10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0])
t_r = np.array([8.0,  15.5, 22.5, 29.0, 35.0, 40.5, 45.5, 50.0])
sig = np.array([0.5,  0.5,  0.5,  0.5,  0.5,  0.5,  0.5,  0.5])
```

Next, we set up the two hypotheses used in part a, the first being the commonly accepted hypothesis at the time, and the second being Ptolemy's new hypothesis:

$$\theta_r = \alpha\theta_i \tag{1}$$

$$\theta_r = \alpha\theta_i - \beta\theta_i^2 \tag{2}$$

Defining these in Python, we get the following:

```
def h1(t_i, *vars):
    a, = vars
    return a*t_i

def h2(t_i, *vars):
    a, b = vars
    return a*t_i - b*t_i**2
```

We use the same techniques as previous weeks to fit these functions to the data points: by using SciPy's curve_fit function, we get the fit's parameters and

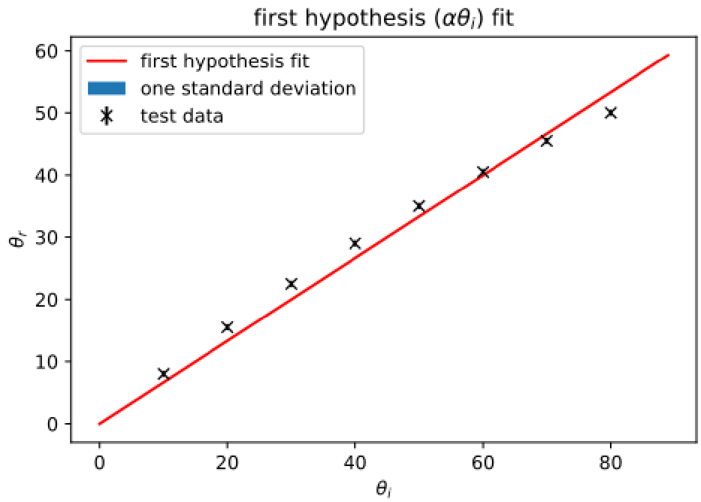

Figure 1: First hypothesis fit, showing the data points, fit curve, and one standard deviation above and below the fit.

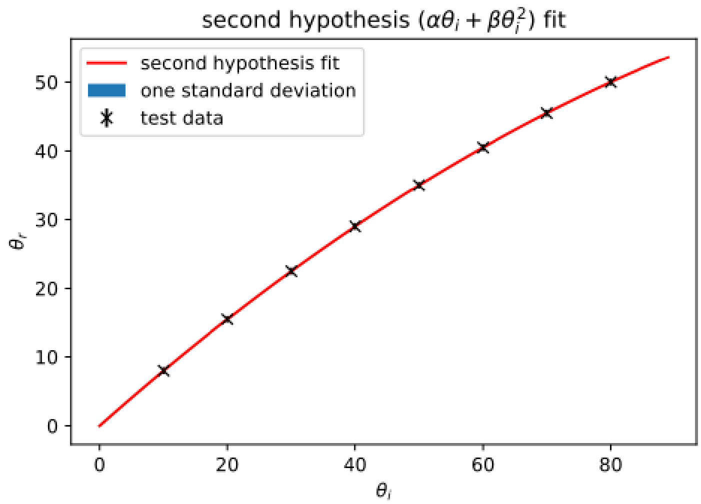

Figure 2: Second hypothesis fit, showing the data points, fit curve, and one standard deviation above and below the fit.

Hypothesis Fit	α	β
1	0.666	n/a
2	0.825	0.0025

Table 1: Parameters for the fitted functions of the first two hypotheses.

covariances. From these, we can plot the fits along with one standard deviation above and below the fit, resulting in the following plots:

The parameters for these fits are listed below in Table 1

To quantify the *"goodness-of-fit"* for these, we again employ the χ^2_{min} value, and the associated *p-value*. To obtain these values, we use the following Python functions, employing SciPy's `stats.chi2` function:

```
chi_sq_min = sum(((d - f1(h, *theta_hat))/sig)**2)
ndof = len(h)-len(p0)
p_val = scipy.stats.chi2.sf(chi_sq_min, df=ndof)
```

From this, we generate the following χ^2_{min} and p-values:

Hypothesis Fit	χ^2_{min}/NDoF	*p-value*
1	19.24	6.71×10^{-26}
2	0.0	1.0

Table 2: Parameters for the fitted functions of the first two hypotheses.

The results of χ^2_{min}=0 for the second hypothesis is extremely unusual - it indicates that the fit matches the data *far better* than expected, and does not have the expected standard deviation given by our choice of errors. This indicates to us that some of the data may have been inferred and not measured - perhaps the angles below 45 degrees were merely doubled by Ptolemy and recorded as the angles above 45 degrees.

To analyse this further, and to use a third hypothesis, we use the following equation first discovered by Ibn Sahl in the 10th century, later rediscovered by Snell in 1621, and commonly known as Snell's Law:

$$\theta_r = \sin^{-1}\left(\frac{\sin\theta_i}{r}\right) \tag{3}$$

We use the exact same analysis for this third hypothesis as the first two: using the minimisation of residuals method offered by `curve_fit`, and calculating the χ^2_{min} and p-value for the fit. Plotting the fitted parameters gives us the following figure:

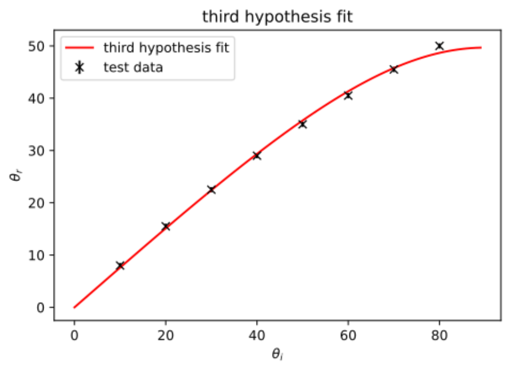

Figure 3: Third hypothesis fit, showing the data points, fit curve, and one standard deviation above and below the fit.

With the calculated parameter $r = 1.31$

By applying the χ^2_{min} method, we find $\chi^2_{\text{min}} = 2.00$ and *p-value* $= 0.05$. These values are far more within the realm of expectation for our error bars, and methodology. From looking at the plot, we can see that the final data point is by far the outlier: from this we could reject the hypothesis, but the rest of the data points hold well. Instead, we could challenge the assumption of $\sigma = \frac{1}{2}$ deg, and instead investigate the results with a larger error width.

A Python Code

```
import numpy as np
from scipy.optimize import curve_fit
from scipy.stats import chi2
import matplotlib
import matplotlib.pyplot as plt

# Ex 3a

# Define the data set

t_i = np.array([10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0])
t_r = np.array([8.0,  15.5, 22.5, 29.0, 35.0, 40.5, 45.5, 50.0])
sig = np.array([0.5,  0.5,  0.5,  0.5,  0.5,  0.5,  0.5,  0.5])


# Define Hypotheses

def h1(t_i, *vars):
    a, = vars
    return a*t_i

def h2(t_i, *vars):
    a, b = vars
    return a*t_i - b*t_i**2

# Curve fitting

p0_1 = np.array([1.0])
p0_2 = np.array([1.0, 1.0])

params_1, covariance_1 = curve_fit(h1, t_i, t_r, p0_1, sig, absolute_sigma=True)
params_2, covariance_2 = curve_fit(h2, t_i, t_r, p0_2, sig, absolute_sigma=True)

# Parameter standard deviation

params_1_std_dev = np.sqrt(np.diag(covariance_1))
params_2_std_dev = np.sqrt(np.diag(covariance_2))

# Plotting

fig1 = plt.figure()
fig2 = plt.figure()
ax1 = fig1.add_subplot()
ax1.set_title('first hypothesis ($\\alpha \\theta_i$) fit')
ax2 = fig2.add_subplot()
ax2.set_title('second hypothesis ($\\alpha \\theta_i+\\beta \\theta_i^2$) fit')
ax1.set_ylabel("$\\theta_r$")
```

```python
ax2.set_ylabel("$\\theta_r$")
ax1.set_xlabel("$\\theta_i$")
ax2.set_xlabel("$\\theta_i$")

t_i_val = np.arange(0, 90, 1)

def std_dev(t_i, cov): return np.sqrt(sum([(t_i**(i+j))*cov[i][j] for i in range(len(cov)) for j in range(len(cov))]))

## Given Data
ax1.errorbar(t_i, t_r, yerr=sig, fmt='kx', label="test data")
ax2.errorbar(t_i, t_r, yerr=sig, fmt='kx', label="test data")

## Plot the two fits
ax1.plot(t_i_val, h1(t_i_val, *params_1), 'r', label="first hypothesis fit")
ax1.fill_between(t_i_val, h1(t_i_val, *params_1)-std_dev(t_i_val, covariance_1), h1(t_i_val, *params_1)+std_dev(t_i_val, covariance_1), label="one standard deviation")
ax1.legend()

ax2.plot(t_i_val, h2(t_i_val, *params_2), 'r', label="second hypothesis fit")
ax2.fill_between(t_i_val, h2(t_i_val, *params_2)-std_dev(t_i_val, covariance_2), h2(t_i_val, *params_2)+std_dev(t_i_val, covariance_2), label="one standard deviation")
ax2.legend()

plt.show()

# Chi-Square

## Hypothesis 1
chi_sq_min_1 = sum(((t_r - h1(t_i, *params_1))/sig)**2)
ndof_1 = len(t_i)-len(p0_1)
print(f"---Hypothesis 1---\nChi-sq-min/NDoF:\t{chi_sq_min_1/ndof_1}\np-value:\t\t{chi2.sf(chi_sq_min_1, df=ndof_1)}")

## Hypothesis 2
chi_sq_min_2 = sum(((t_r - h2(t_i, *params_2))/sig)**2)
ndof_2 = len(t_i)-len(p0_2)
print(f"---Hypothesis 2---\nChi-sq-min/NDoF:\t{chi_sq_min_2/ndof_2}\np-value:\t\t{chi2.sf(chi_sq_min_2, df=ndof_2)}")

# Ex 3b

# Define Sahl formula

def h3(t_i, *vars):
    r, = vars
    return np.rad2deg(np.arcsin( np.sin(np.deg2rad(t_i)) / r ))

p0_3 = np.array([1.0])

params_3, covariance_3 = curve_fit(h3, t_i, t_r, p0_3, sig, absolute_sigma=True)

chi_sq_min_3 = sum(((t_r - h3(t_i, *params_3))/sig)**2)
ndof_3 = len(t_i)-len(p0_3)
print(f"---Hypothesis 3---\nChi-sq-min/NDoF:\t{chi_sq_min_3/ndof_3}\np-value:\t\t{chi2.sf(chi_sq_min_3, df=ndof_3)}")
```

```
fig3 = plt.figure()
ax3 = fig3.add_subplot()
ax3.set_title('third hypothesis fit')
ax3.set_ylabel("$\\theta_r$")
ax3.set_xlabel("$\\theta_i$")

ax3.errorbar(t_i, t_r, yerr=sig, fmt='kx', label="test data")

ax3.plot(t_i_val, h3(t_i_val, *params_3), 'r', label="third
    hypothesis fit")
ax3.legend()

plt.show()

print(f"\n\nHypothesis 1:\t a = {params_1[0]}")
print(f"\n\nHypothesis 2:\t a = {params_2[0]}\tb = {params_2[1]}")
print(f"\n\nHypothesis 3:\t r = {params_3[0]}")
```

py.py

```
---Hypothesis 1---
Chi-sq-min/NDoF: 19.235294117647054
p-value: 6.711003202955456e-26
---Hypothesis 2---
Chi-sq-min/NDoF: 0.0
p-value: 1.0
---Hypothesis 3---
Chi-sq-min/NDoF: 2.0003105594271977
p-value: 0.05114270171746438

Hypothesis 1:  a = 0.6661764705875071
Hypothesis 2:  a = 0.825 b = 0.0024999999999999996
Hypothesis 3:  r = 1.3116118503798777
```