Exercise 2.1

(a) No, $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is here a singular matrix which is rank deficient.

(b) Yes, but we need to be more specific. The determinant can be above zero, but too small to give a stable solution. Usually in such cases the matrix is really singular but noise is making its determinant different from zero.

(c) Yes, but same considerations as in the previous statement.

(d) No, since perfectly correlated variables (or rows) will give rise to a singular matrix. Again, numerical issues must be considered.

(e) Yes, because orthogonal variable space will have a maximum rank. No problems with inverting $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

(f) No, since the smallest of the dimensions will set the upper limit to the rank. If the rank is half this, it means that we have correlated rows or columns.

(g) No, We know the rank must be less or equal to N and since $M > N$ we know that $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ does not have rank M and thus cannot be inverted.

(h) Maybe. The fact that we have more rows than columns does not guarantee the invertibility of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, but it makes it more likely or at least it is possible.

Exercise 2.2

The projection matrix, $\mathbf{H}$, is given by,

$$\mathbf{H} = \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}.$$

Show that:

(a) $\mathbf{H}$ is symmetric. This means that $\mathbf{H} = \mathbf{H}^{\mathrm{T}}$:

$$\begin{aligned}
\mathbf{H}^{\mathrm{T}} &= \left(\mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\right)^{\mathrm{T}} \\
&= \left(\mathbf{X}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\right)^{\mathrm{T}}\mathbf{X}^{\mathrm{T}} \\
&= \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}} \\
&= \mathbf{H}.
\end{aligned}$$

(b) $\mathbf{H}^k = \mathbf{H}$ where the integer $k > 0$: For $k = 0$ and $k = 1$ this is true. Consider now the case $k = 2$, $\mathbf{HH} = \mathbf{H}^2$:

$$\begin{aligned}\mathbf{HH} &= \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}} \\ &= \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}} \\ &= \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}} = \mathbf{H}.\end{aligned}$$

Assume now that we have shown that this is true up to the value n, $\mathbf{H}^n = \mathbf{H}$. Then,

$$\mathbf{H}^{n+1} = \mathbf{HH}^n = \mathbf{HH} = \mathbf{H},$$

so the proof follows by induction.

Exercise 2.3

In this case we can solve the problem directly:

$$\begin{aligned}\mathbf{y} &= \mathbf{X}\hat{\mathbf{b}}, \\ \mathbf{X}^{-1}\mathbf{y} &= \mathbf{X}^{-1}\mathbf{X}\hat{\mathbf{b}}, \\ \hat{\mathbf{b}} &= \mathbf{X}^{-1}\mathbf{y}.\end{aligned}$$

Exercise 2.4

We formulate the problem as:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{e},$$

where the matrix $\mathbf{X}$ has the following form:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{11}^2 & x_{11}^3 \\ 1 & x_{21} & x_{21}^2 & x_{21}^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n1}^2 & x_{n1}^3 \end{bmatrix}.$$

The least-squares solution is then:

$$\hat{\mathbf{b}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}.$$

Exercise 2.5

A Python script performing the required polynomial fitting is given below. The resulting figures are shown in Fig. 1–6. We see that a polynomial of order 3 is suitable to model the temperature in this case.

```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.style.use('seaborn-talk')


def fit_ploynomial(xdata, ydata, degree):
    params = np.polyfit(xdata, ydata, degree)
    yre = np.polyval(params, xdata)
    residuals = ydata - yre
    return yre, residuals


def plot_xy(xdata, ydata, yre=None, degree=None, residuals=None, output=None):
    fig = plt.figure()
    if residuals is None:
        ax1 = fig.add_subplot(111)
        ax2 = None
    else:
        ax1 = fig.add_subplot(121)
        ax2 = fig.add_subplot(122)
        ax1.set_title('Fitted curve')
        ax2.set_title('Residuals')
    ax1.scatter(xdata, ydata)
    ax1.set_xlabel('Time (hour)')
    ax1.set_ylabel('Temperature (°C)')
    if yre is not None:
        lab = 'Fitted curve'
        if degree is not None:
            lab = 'Polynomial of order {}'.format(degree)
        ax1.plot(
            xdata,
            yre,
            label=lab,
            color='#ff7f00'
        )
        ax1.legend()
    if residuals is not None and ax2 is not None:
        ax2.scatter(xdata, residuals)
        ax2.set_xlabel('Time (hour)')
        ax2.set_ylabel('Temperature (°C)')
    fig.tight_layout()
    if output is not None:
        fig.savefig(output)


def plot_xy_res(xdata, ydata, output=None):
    fig = plt.figure()
    ax1 = fig.add_subplot(111)
    ax1.set_title('Sum of squared residuals')
```

```
    ax1.scatter(xdata, ydata, s=200)
    ax1.plot(xdata, ydata, ls='--')
    ax1.set_xlabel('Polynomial degree')
    ax1.set_ylabel(r'Sum of squared residuals (°C)$^2$')
    ax1.set_yscale('log')
    fig.tight_layout()
    if output is not None:
        fig.savefig(output)


def main():
    data = pd.read_csv('Data/data_exercise_2.txt', delim_whitespace=True)
    xdata = data['hour']
    ydata = data['yobs']
    plot_xy(xdata, ydata, output='raw_data.pdf')
    degrees = [1, 2, 3, 4, 5]
    residual_sum = []
    for degree in degrees:
        yre, residuals = fit_ploynomial(xdata, ydata, degree)
        plot_xy(xdata, ydata, yre=yre, degree=degree, residuals=residuals,
                output='degree-{}.pdf'.format(degree))
        residual_sum.append(sum(residuals**2))
    plot_xy_res(degrees, residual_sum, 'residuals.pdf')
    plt.show()


if __name__ == '__main__':
    main()
```

Listing 1: *Python code for performing the fitting.*

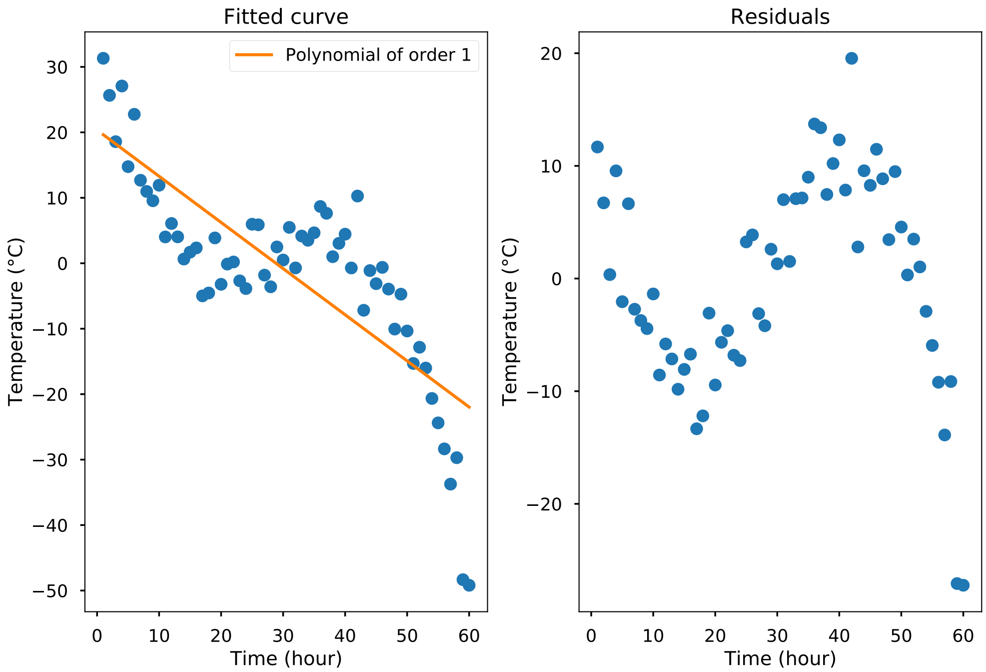

Figure 1: *Polynomial fit of degree* 1.

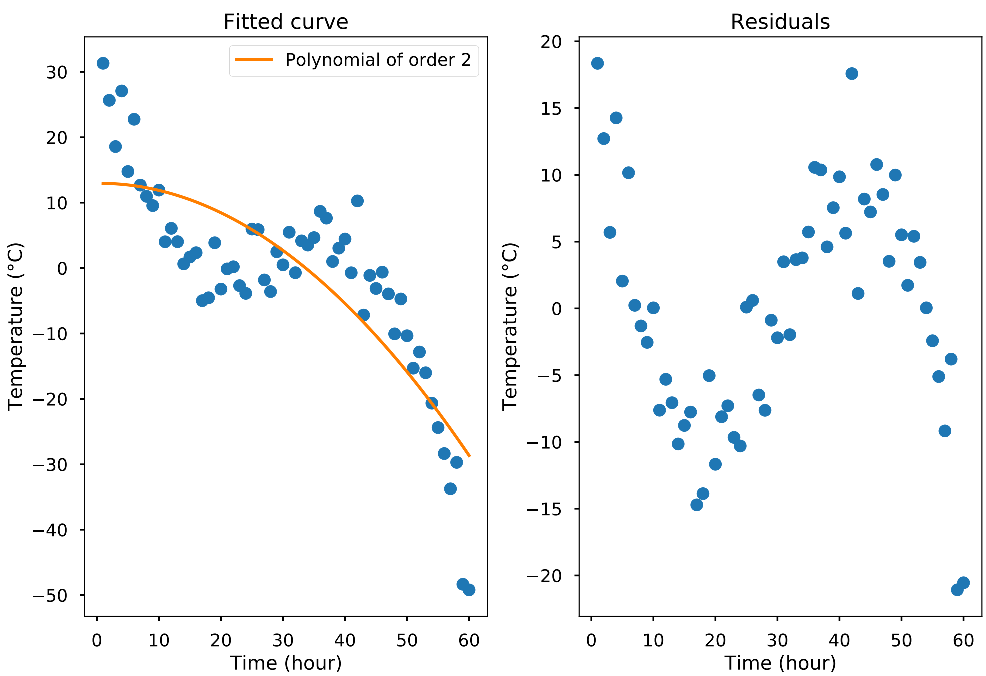

Figure 2: *Polynomial fit of degree* 2.

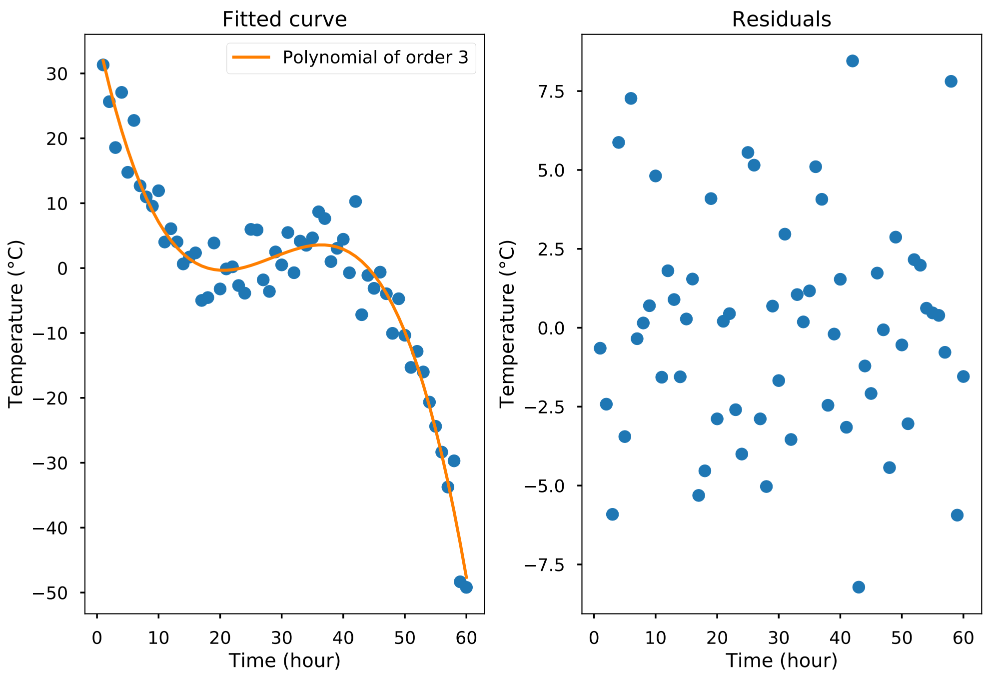

Figure 3: *Polynomial fit of degree* 3.

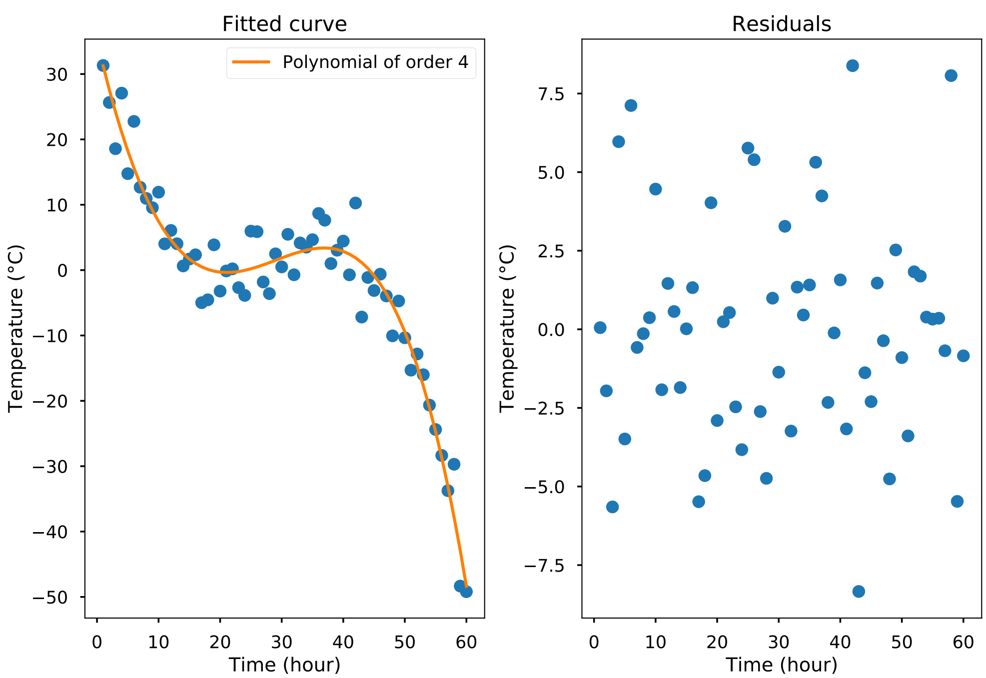

Figure 4: *Polynomial fit of degree* 4.

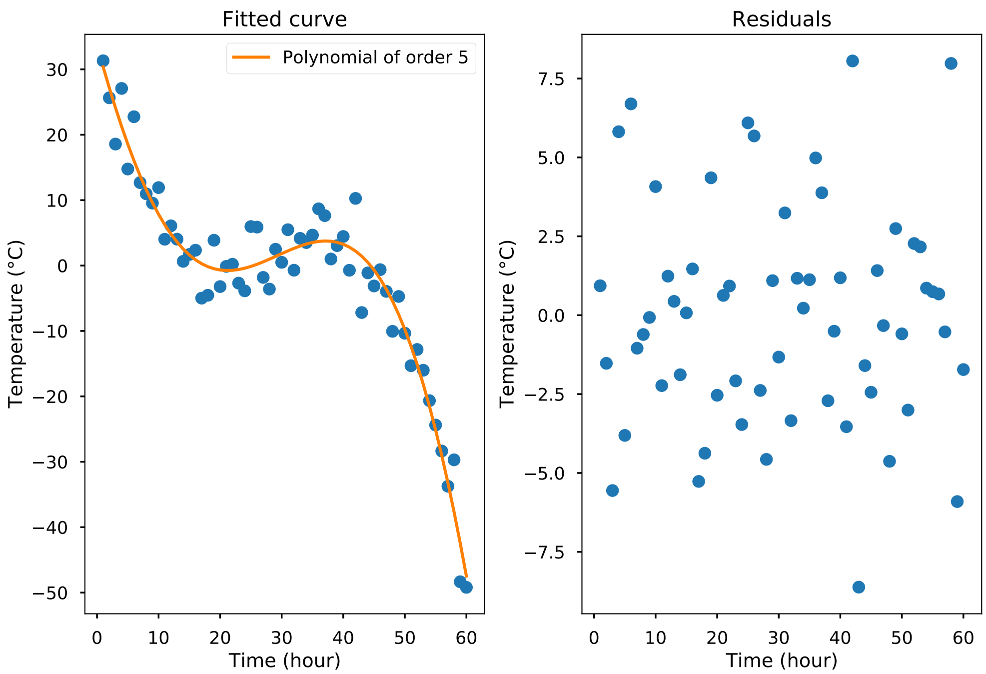

Figure 5: *Polynomial fit of degree* 5*.*

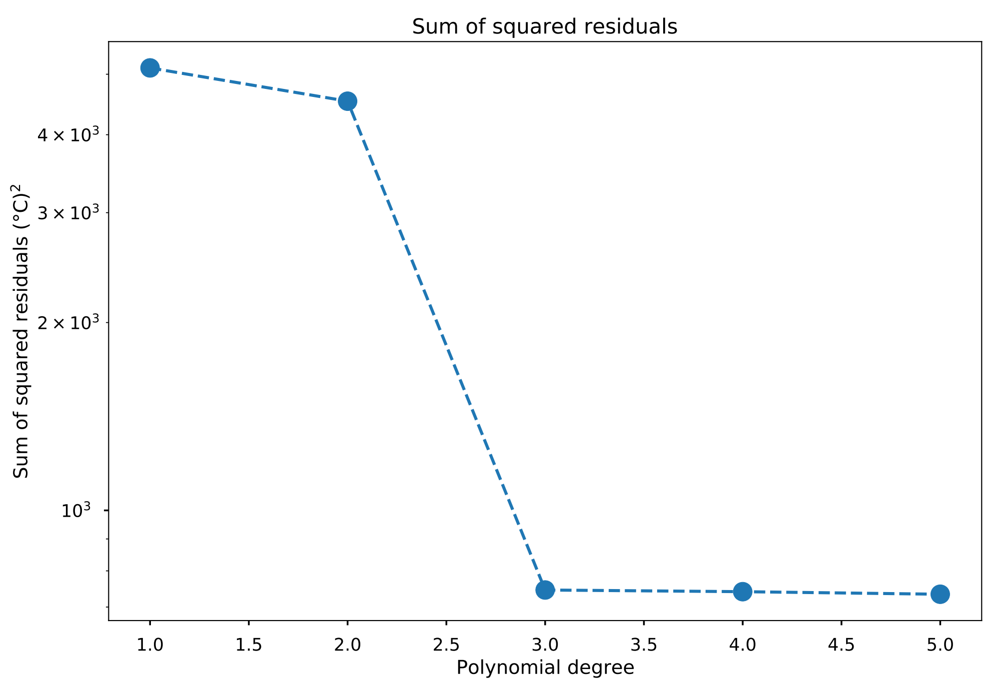

Figure 6: *Sum of residuals as a function of polynomial degree.*