Newtons Law of Cooling

Kaylin Shanahan 2022

This is a python programme to plot the natural log of the temperature data, as a function of time, including error bars, from which the slope m and y-intercept c can be determined, allowing us to determine the value of the cooling constant k along with its error.
Using Newtons Law of Cooling we can describe the temperature of a cooling body:

$$T(t) - T_s = (T_0 - T_s)e^{-kt}$$

where T is the temperature of the object, T_s is the temperature of the surrounding, T_0 is the initial temperature of the object, t is time and k is a cooling constant.
We can simplify this equation (as $T_0 = 0$) further to:

$$T(t) = T_0 e^{-kt}$$

Finally, by taking the natural logs of both sides, we have an equation which will provide a graph of ln(T) against t, with a straight line with a slope of $-k$, allowing us to determine the cooling constant k.

$$ln[T(t)] = ln(T_0) - kt$$

- Input time and temperature data arrays
- Calculate the natural log of temperature data
- Output graph plotting the natural log of the temperature data as a function of time, including error bars
- Calculate errors in slope m, intercept c and cooling constant k
- Output slope m, intercept c and cooling constant k values

In [28]:

```
# Newtons Law of Cooling - fitting straight lines with errors
# Using realistic data and errorbars

# Numpy is needed for our calculations
import numpy as np
```

```python
# Matplotlib is needed to plot our straight-line graph
import matplotlib.pyplot as plt

# Store the data in two NumPy arrays
t = np.array([100, 200, 300, 400, 500, 600, 700, 800, 900, 1000])
Temp = np.array([95.5, 74.2, 70.3, 68.7, 56.5, 55.2, 40.4, 40.5, 32.9, 31.9])

# Calculate the Natural Log of Temperature and its error
LnTemp = np.log(Temp)
LnTemp_err = 0.1

# Plot LnTemp as a function of t with errorbars
plt.errorbar(t, LnTemp, yerr=LnTemp_err, fmt="ro", capsize=5)
plt.title("Graph of Natural Log of Temperature vs Time")
plt.xlabel("Time $(s)$")
plt.ylabel("LnTemp")
plt.grid()

# Call te NumPy polynomial order 1 fitter, with covariance matix
[m,c], Covmat = np.polyfit(t, LnTemp, 1, cov=True)

# Calculate the errors in m and c from covariance matrix
m_err = np.sqrt(Covmat[0,0])
c_err = np.sqrt(Covmat[1,1])

# Calculate the fractional error in m
m_frac_err = m_err / m

# Print the slope and intercept
print("slope = {0:6.3f}".format(m, m_err))
print("intercept = {0:6.3f}".format(c, c_err))

# Calculate k using the slope
kslope = -m
# Fractional error in g is same as fractional error in m
kslope_err = kslope * m_frac_err
print("k = {0:5.2f} +/-{1:6.3}".format(kslope, kslope_err))


# Define an array of 10 temperature values
t10 = np.linspace(100, 1000, 1001)
# Use the slope and intercept to calculate the 10 temperature values
LnTemp_fit = m * t10 + c
```

```
# Plot the fitted straight line
plt.plot(t10, LnTemp_fit)
plt.show()
```

```
slope = -0.001
intercept =  4.639
k =  0.00 +/--7.73e-05
```