Temperature Modelling

In [175…
```
%pip install numpy
%pip install pandas
%pip install scipy
```

```
Requirement already satisfied: numpy in ./.venv/lib/python3.13/site-packages
(2.3.3)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: pandas in ./.venv/lib/python3.13/site-package
s (2.3.3)
Requirement already satisfied: numpy>=1.26.0 in ./.venv/lib/python3.13/site-
packages (from pandas) (2.3.3)
Requirement already satisfied: python-dateutil>=2.8.2 in ./.venv/lib/python
3.13/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in ./.venv/lib/python3.13/site-p
ackages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in ./.venv/lib/python3.13/site
-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in ./.venv/lib/python3.13/site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: scipy in ./.venv/lib/python3.13/site-packages
(1.16.2)
Requirement already satisfied: numpy<2.6,>=1.25.2 in ./.venv/lib/python3.13/
site-packages (from scipy) (2.3.3)
Note: you may need to restart the kernel to use updated packages.
```

In [176…
```
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
```

Newton's Law of Cooling

The rate of change of temperature of an object is proportional to the difference between its own temperature and the ambient temperature (the temperature of its surroundings).

$$\frac{dT}{dt} = -k(T - T_a)$$

Where:

- (T) is the temperature of the object at time (t)
- (T_a) is the ambient temperature
- (k) is a positive constant that depends on the characteristics of the object and its surroundings

Solution to the Differential Equation

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

Where:

- (T_0) is the initial temperature of the object at (t = 0)
- (e) is the base of the natural logarithm
- (t) is the time elapsed (in seconds)
- (k) is the cooling constant

Another solution is with time constant τ.

$$T(t) = T_a + (T_0 - T_a)e^{-t/\tau}$$

where:

- τ is the time constant that depends on the insulation characteristics of the object

In [177…
```
# Newton's Law of Cooling
def newtons_law_of_cooling(t, T_0, T_amb, k):
    return T_amb + (T_0 - T_amb) * np.exp(-k * t)
```

Data

In [178…
```
df = pd.read_csv('dataset.csv')
df.head()
```

Out[178…

	no	timestamp	T_amb	Temp
0	0	05:45	29.2	97.3
1	1	06:00	29.2	94.8
2	2	06:15	29.1	92.8
3	3	06:30	29.0	90.7
4	4	06:45	29.0	88.9

In [179…
```
df.columns = df.columns.str.strip()
n = len(df)
df['Elapsed_Hours'] = np.arange(n) * 0.25
```

In [180…
```
# Declare Variables

ambient_temperature = df['T_amb'].mean()
temperature = df['Temp'].values
time = df['Elapsed_Hours'].values
```

Model

In [181…
```
def cooling_model(t, tau):
    return ambient_temperature + (temperature[0] - ambient_temperature) * np
```

In [182…
```
tau_opt, _ = curve_fit(cooling_model, time, temperature, bounds=(0, np.inf))
temps_fitted = cooling_model(time, tau_opt[0])

print(tau_opt[0])
```

```
8.140576905063138
```

This approach uses the curve fitting method to estimate the cooling constant k or time constant τ by minimizing the difference between the observed temperatures and the temperatures predicted by the model. This is more accurate than using just two data points.

What `curve_fit` does is it tries many values of τ and computes:

$$\text{Error} = \sum_i [\text{Observed}_i - \text{Predicted}_i]^2$$

It then picks the τ that minimizes this squared error, that is the best fitting exponential curve to the data.

Model Fitting

In [183…
```
results_df = pd.DataFrame({
    "Time (hours)": time,
    "Observed Temp (°C)": temperature,
    "Fitted Temp (°C)": temps_fitted,
    "Difference (°C)" : (temperature-temps_fitted)
})
results_df
```

Out[183…

	Time (hours)	Observed Temp (°C)	Fitted Temp (°C)	Difference (°C)
0	0.00	97.3	97.300000	0.000000
1	0.25	94.8	95.234363	-0.434363
2	0.50	92.8	93.231199	-0.431199
3	0.75	90.7	91.288617	-0.588617
4	1.00	88.9	89.404787	-0.504787
5	1.25	87.3	87.577929	-0.277929
6	1.50	85.6	85.806323	-0.206323
7	1.75	84.0	84.088296	-0.088296
8	2.00	82.0	82.422229	-0.422229
9	2.25	81.0	80.806549	0.193451
10	2.50	79.8	79.239734	0.560266
11	2.75	78.3	77.720304	0.579696

Extrapolation

In [184…

```
extrapolate_time = np.arange(0, 300.25, 0.25)
extrapolate_temperature = cooling_model(extrapolate_time, tau_opt[0])

extrapolate_df = pd.DataFrame({
    "Time (hours)" : extrapolate_time,
    "Temperature" : extrapolate_temperature
})
extrapolate_df.head()
```

Out[184…

	Time (hours)	Temperature
0	0.00	97.300000
1	0.25	95.234363
2	0.50	93.231199
3	0.75	91.288617
4	1.00	89.404787

In [185…

```
# General Information

print("tau = ", tau_opt[0])
print("Ambient Temperature = ", ambient_temperature)
print("Initial Temperature = ", temperature[0])
print("Temperature after 12 hours = ", extrapolate_temperature[48])
```

```
tau =  8.140576905063138
Ambient Temperature =  28.999999999999996
Initial Temperature =  97.3
Temperature after 12 hours =  44.63970332335817
```

In [199…

```python
def temperature_at_time_t(time_value):
    time_value = float(time_value)
    if not (extrapolate_time.min() <= time_value <= extrapolate_time.max()):
        raise ValueError(
            f"time must be within [{extrapolate_time.min()}, {extrapolate_ti
        )
    return float(cooling_model(time_value, tau_opt[0]))

def time_at_temperature_T(temperature_value):
    temperature_value = float(temperature_value)

    max_temp = float(temperature[0])
    min_temp = float(extrapolate_temperature.min())

    if not (min_temp <= temperature_value <= max_temp):
        raise ValueError(
            f"temperature must be within [{min_temp}, {max_temp}] °C."
        )

    temps_sorted = extrapolate_temperature[::-1]
    times_sorted = extrapolate_time[::-1]
    time_value = np.interp(temperature_value, temps_sorted, times_sorted)
    return float(time_value)


TIME = 0.75
TEMPERATURE = 90
temperature_at_time_t(TIME), time_at_temperature_T(TEMPERATURE)
```

Out[199…

```
(91.2886174741422, 0.9210102376477249)
```

Limit of The model as $t \to \infty$