# What is the relationship between the length of the filars and the period of a bifilar pendulum?

Personal code: _____

## 1 Introduction

A bifilar pendulum is widely used to measure the moment of inertia of objects, which quantifies the amount of torque required to cause a specific angular acceleration along an axis. By observing the minute irregularities in the moment of inertia of the Earth, geologists can measure seismic events in remote locations, changes in the Earth's magnetic inclination and the effect of tides on the Earth's gravitational acceleration (Davison, 1894). Due to its high accuracy, bifilar pendulums are now used to measure the moment of inertia of complex geometries, such as aircrafts and ships that would otherwise be infeasible to solve by hand (Jardin and Mueller, 2007). Since the moment of inertia is a critical parameter to aerodynamic and hydrodynamic stability, understanding the dynamics of a bifilar pendulum can lead to better designs in aircrafts, enhancing the safety in transportation systems.

Although the general equation for bifilar pendulums have been well established, Klopsteg (1930) acknowledged the difficulty of determining the centre of mass of the test object, Kane and Tseng (1967) determined that minor inequalities of filar lengths will exhibit strong non-linear effects and chaotic side-sway motion, while Denman (1992) observed that torsional oscillations and vibrations in the filars cause significant deviations in the amplitude and period of bifilar pendulums. Given the wide range of unfulfilled assumptions and the variability of experimental results, the formulation of an accurate relationship will better aid real-life problems.

## 2 Framework

The following section aims to derive the equation of the bifilar pendulum, suspended by a rod (Amrozia and Muhammad, 2017; Then, 1965; Wouter, 2016):

$$T = \frac{2\pi l_{rod}}{r}\sqrt{\frac{l}{12g}} \tag{1}$$

Consider a bifilar pendulum with a rod MP suspended by filars AM and BP, disturbed from its equilibrium position MP to NQ and allowed to oscillate about point O along the $\hat{z}$ axis:

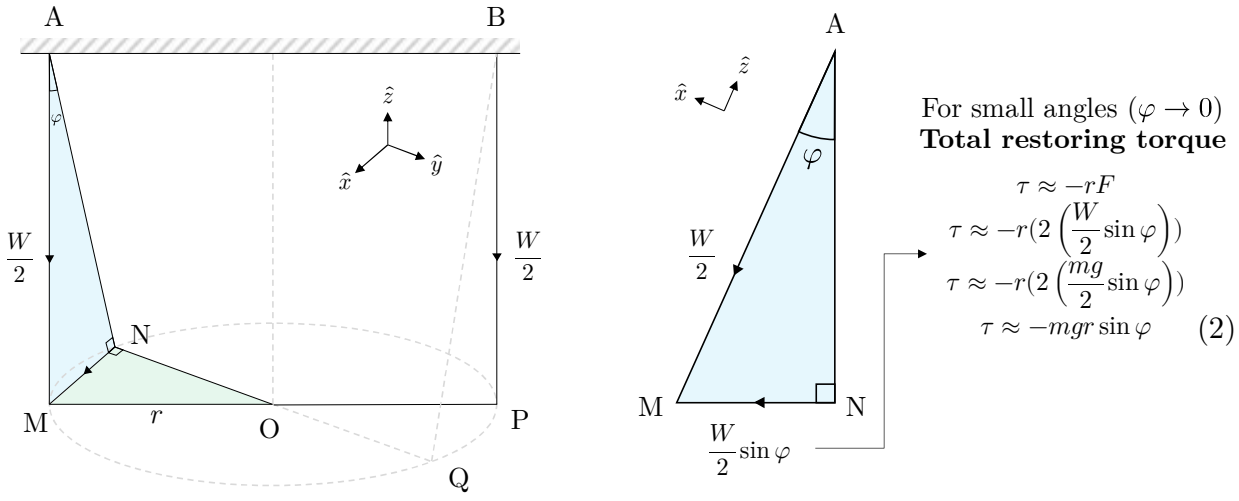
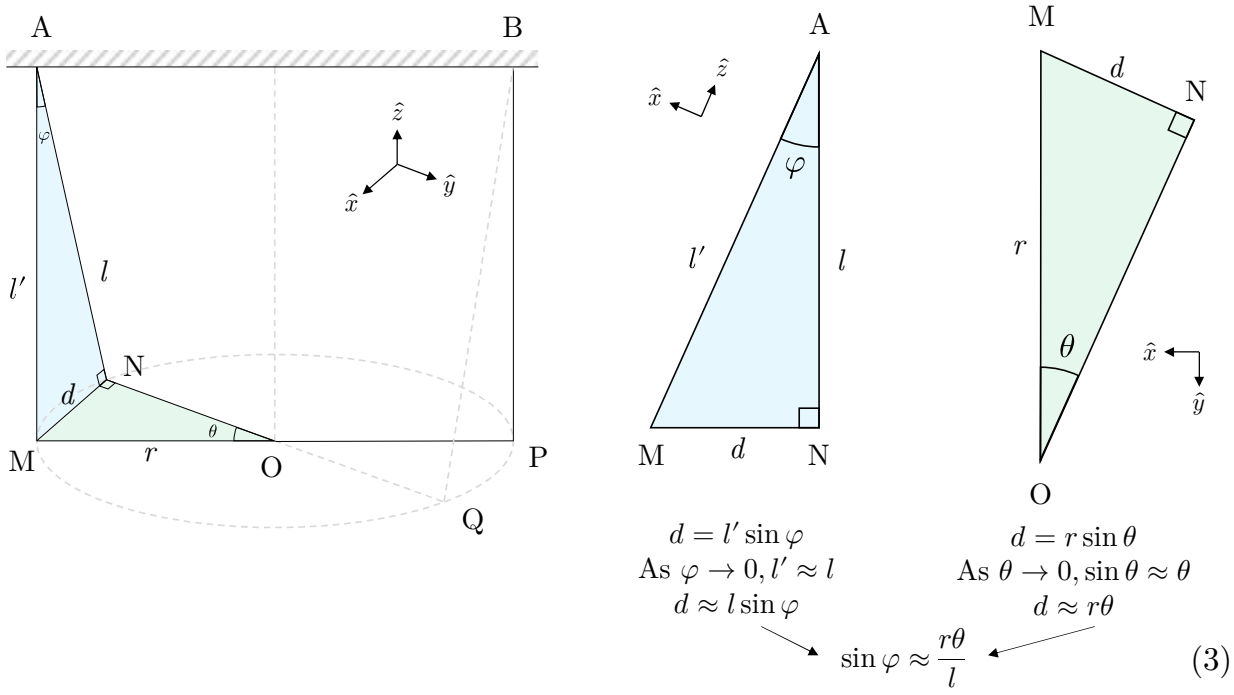



**Figure 1**. The decomposition of forces of a bifilar pendulum disturbed from its equilibrium position.

The total restoring torque due to the weight of the rod is obtained in Equation 2. To better represent the motion of the rod, the relationship between $\varphi$ and $\theta$ is found to be:

For small angles ($\theta \to 0, \varphi \to 0$):




$$d = l' \sin\varphi$$
$$\text{As } \varphi \to 0, l' \approx l$$
$$d \approx l \sin\varphi$$

$$d = r \sin\theta$$
$$\text{As } \theta \to 0, \sin\theta \approx \theta$$
$$d \approx r\theta$$

$$\sin\varphi \approx \frac{r\theta}{l} \tag{3}$$

**Figure 2**. The approximate relationship between $\varphi$ and $\theta$ using trigonometric identities.

Substituting Equation 3 into Equation 2, the total restoring torque expressed in terms of $\theta$ is:

$$\tau = -\frac{mgr^2\theta}{l} \tag{4}$$

Substituting Equation 4 into Newton's Second Law of Angular motion yields:

$$\tau = I\ddot{\theta}$$
$$\ddot{\theta} + \frac{mgr^2}{Il}\theta = 0 \tag{5}$$

Solving the second-order differential equation in Equation 5 with respect to time $t$, with initial angle $\theta_i = \theta_0$ and initial angular velocity $\dot{\theta}_i = 0$, the solution is:

$$\theta = \theta_0 \cos\left(\sqrt{\frac{mgr^2}{Il}}t\right) \tag{6}$$

Therefore, the period of the cosine curve in Equation 6 is:

$$T = \frac{2\pi}{\sqrt{\frac{mgr^2}{Il}}} = \frac{2\pi}{r}\sqrt{\frac{Il}{mg}} \tag{7}$$

The moment of inertia of a rod about its centre of mass is given by (Weisstein, 1996):

$$I = \frac{1}{12} m {l_{rod}}^2 \tag{8}$$

Substituting (8) into (7) and simplifying the equation yields the final relationship:

$$T = \frac{2\pi l_{rod}}{r}\sqrt{\frac{l}{12g}} \tag{9}$$

There are several required assumptions to support Equation 9:

1. The angle $\theta$ is sufficiently small, such that the vertical translational kinetic energy is negligible when compared to the horizontal kinetic energy, otherwise the small angle approximation will not hold, and the error in the period will increase by the order of $O(\theta^3)$.
2. There is no loss in rotational kinetic energy, such as work done to the rod due to air resistance, which will cause the angle $\theta$ to decrease.
3. The filars are massless, otherwise it may cause the centre of mass of the test subject to be shifted upwards, decreasing the effective filar length ($l$) and increase the mass ($m$), overall decreasing the period of the oscillations.
4. The filars are assumed to be non-rigid, inextensible and have no torsional rigidity (Uhler, 1923). Microscopically, the strong intermolecular attraction between molecules of a particularly rigid filar may cause resistance to deformation, causing movements to require additional tension, thus resulting in damped responses.
5. The filars (AM, BP) are vertically parallel to each other and have the same length ($l$). The rod (MP) is horizontally parallel to the top (AB) and have the same length ($2r$). The axis of rotation (O along $\hat{z}$) should pass through the midpoint and centre of mass of the rod, otherwise slight deviations can cause chaotic motion (Kane and Tseng, 1967).

Additionally, Equation 7 can be linearised by expressing it in the slope-intercept form:

$$\underbrace{T}_{\substack{\text{Plotted}\\ \text{on y-axis}}} = \underbrace{\frac{2\pi l_{rod}}{r}\sqrt{\frac{1}{12g}}}_{\textbf{Slope of graph}}\underbrace{\sqrt{l}}_{\substack{\text{Plotted}\\ \text{on x-axis}}} + \underbrace{0}_{\textbf{y-intercept}} \tag{10}$$

When $T$ is plotted against $\sqrt{l}$, the expected slope and y-intercept is $\frac{2\pi l_{ro}}{r}\sqrt{\frac{1}{12g}}$ and 0 respectively.

## 3 Variables

false

# 4 Pilot Study

As the percentage error in the period of the equation of the bifilar pendulum ($T = \frac{2\pi l_{rod}}{r}\sqrt{\frac{l}{12g}}$) is $\frac{\Delta T}{T_0}$, the denominator $T_0$ must be sufficiently large to reduce the percentage error in $T$. There are three ways in which the range of the other variables can be adjusted to produce the least error:

1. Increasing the length of the rod ($l_{rod}$)
2. Increasing the length of the filars ($l$)
3. Decreasing radius of gyration ($r$)

As explored in Assumption 2, the loss in horizontal rotational kinetic energy should be minimised. Hence, to minimise the drag, a long slender rod with low cross-sectional area, of length $l_{rod} = 0.1958 \pm 0.0001$m and diameter $d_{rod} = 0.00678 \pm 0.00001$m has been selected as the test object.

However, if the radius of gyration ($r$) is too low, the percentage error $\frac{\Delta r}{r_0}$ will increase. Hence, to better assess the effects of $l$ and $r$ on the percentage error, a 2D heatmap is created:

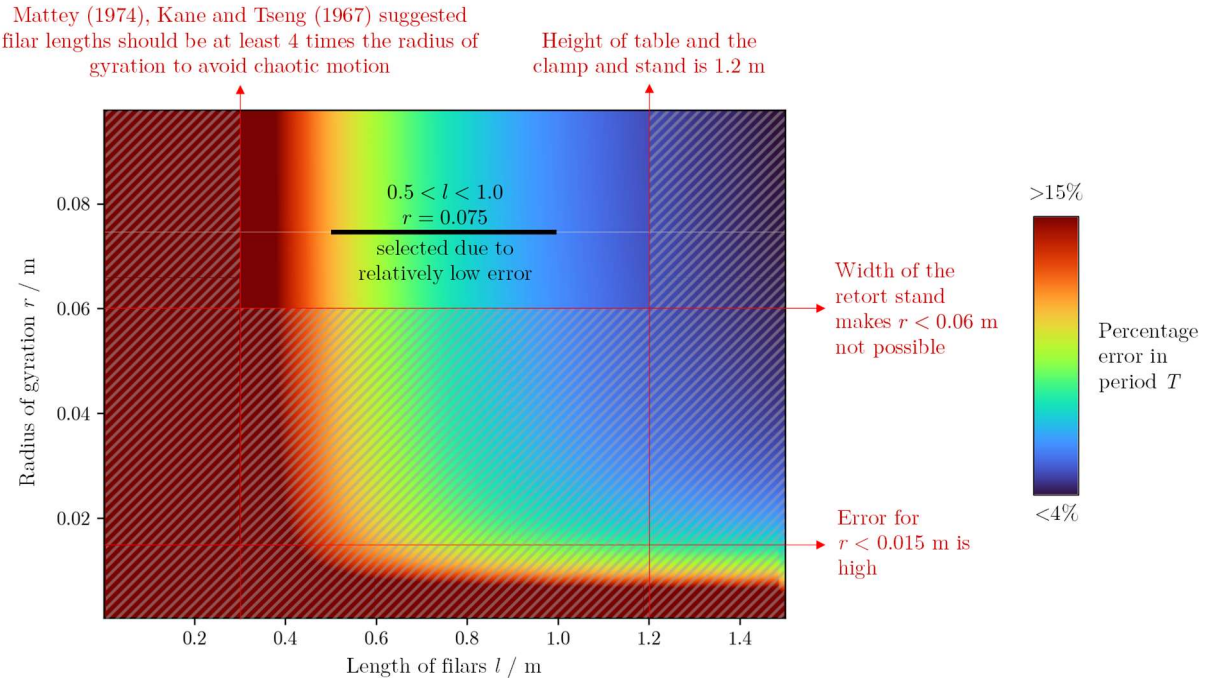



**Figure 3**. A heatmap of the percentage error in $T$ for $0 < l < 1.5$ s and $0 < r < \frac{l_{rod}}{2}$ m.[1]

As observed in Fig. 3, due to experimental setup limitations, the radius of gyration $r$ is selected to be 0.075m. As for the range of the independent variable, length of filars $l$, it has been decided to perform 7 regular intervals between $0.5 < l < 1.0$ m.

Moreover, because $T \propto \sqrt{l}$, when $l \to \infty$, $\frac{dT}{dl} \to 0$, meaning that for increasing filar lengths, the corresponding magnitude of change in $T$ is significantly decreasing. Hence, in order to measure significant changes in $\Delta T$, the length of filars $l$ is controlled to be below 1.0m, as demonstrated in Fig. 4.

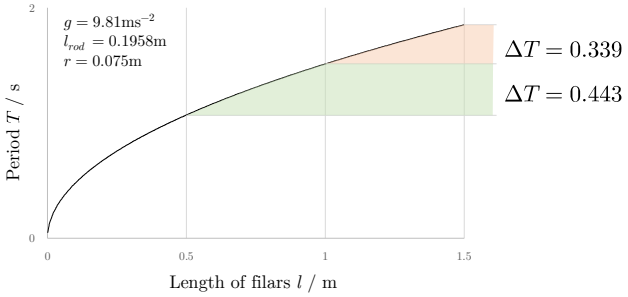



**Figure 4**. A graph of $T$ against $l$.

[1] The detailed formulae and code used to generate the visualisation can be found in Appendix 17.3.

## 5 Apparatus

## 6 Procedure and Precautions

## 7 Ethical, safety and environmental concerns

## 8 Raw Data Table

## 9 Qualitative Observation

## 10 Processed Data Table

$$\delta f(n,\dots) = \sqrt{\sum\nolimits_n \left(\frac{\partial f}{\partial n}\delta n\right)^2} \qquad (\quad)$$

$$l = l_1 - \frac{1}{2}l_2 + \frac{1}{2}l_3$$

$$\delta\sqrt{l} = \sqrt{(\frac{\delta l_1}{2\sqrt{l_1 - \frac{1}{2}l_2 + \frac{1}{2}l_3}})^2 + (-\frac{\delta l_1}{4\sqrt{l_1 - \frac{1}{2}l_2 + \frac{1}{2}l_3}})^2 + (\frac{\delta l_1}{4\sqrt{l_1 - \frac{1}{2}l_2 + \frac{1}{2}l_3}})^2}$$

## 11 Graph

## 12 Introduction

## 13 Conclusion

## 14 Sources of error and limitations

## 15 Improvement and Extensions

## 16 References

# 17 Appendix

## 17.1 Formula for the percentage error in the period

Given the formula for a bifilar pendulum suspended by a rod:

$$T = \frac{2\pi l_{rod}}{r}\sqrt{\frac{l}{12g}} \qquad (\ )$$

The error in $T$ is therefore:

$$\delta T = \sqrt{\left(\frac{\partial T}{\partial l_{rod}}\delta l_{rod}\right)^2 + \left(\frac{\partial T}{\partial r}\delta r\right)^2 + \left(\frac{\partial T}{\partial l}\delta l\right)^2}$$

$$\delta T = \sqrt{\left(\frac{2\pi}{r}\sqrt{\frac{l}{12g}}\delta l_{rod}\right)^2 + \left(-\frac{2\pi l_{rod}}{r^2}\sqrt{\frac{l}{12g}}\delta r\right)^2 + \left(\frac{\pi l_{rod}}{r}\sqrt{\frac{1}{12gl}}\delta l\right)^2} \qquad (\ )$$

## 17.2 Code for pilot study

Using the equation obtained in Appendix 17.1, the following code creates Fig. 3 and Fig. 4:
Filename: `pilot.py`
Language: `Python 3.8.5`
Packages used:

- `numpy 1.8.5`
- `pandas 1.1.2`
- `matplotlib 2.2.5`

```
'''
This script does the following:
- calculate T for 0.001 < l < 1.500, with increments of 0.001
- save the raw data to a CSV file.
- for 0.001 < l < 1.500, with increments of 0.001:
    - for 0.0010 < r < 0.0979, with increments of 0.0001:
        - calculate the percentage error in T
        - plot on a 2D axis with the 'turbo' colormap
- store the graph in a PNG image.
'''

import math
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

g = 9.81

def getT(l_rod, r, l):
    return 2*math.pi*l_rod/r*math.sqrt(l/12/g)

def getTerr(l_rod, δl_rod, r, δr, l, δl):
    return math.sqrt(
        math.pow(δl_rod*((2*math.pi/r) * (math.sqrt(l/12/g))),2) +
        math.pow(δr*((2*math.pi*l_rod/math.pow(r,2)) * (math.sqrt(l/12/g))),2) +
        math.pow(δl*((2*math.pi*l_rod/r) * (.5/math.sqrt(l/12/g))),2)
    )

df0 = pd.DataFrame([(l, getT(.1958, .075, l)) for l in np.arange(.001, 1.500, .001)], columns=['l', 'T'])
df0.to_csv('output1/pilot.csv', index=False)

data = []
```

```
l_rod, δl_rod = .1958, .0001
δl = .001
δr = .001
for l in np.arange(.001, 1.500, .001):
    for r in np.arange(.001, .0979, .0001):
        pctδT = min(getTerr(l_rod, δl_rod, r, δr, l, δl) / getT(l_rod, r, l), .15)
        data.append([l, r, pctδT])

df = pd.DataFrame(data, columns=['l', 'r', 'pctδT'])
plt.rcParams['font.family'] = 'Latin Modern Roman'
plt.scatter(x=df['l'],y=df['r'],c=df['pctδT'], cmap='turbo')
plt.xlim([.001, 1.500])
plt.ylim([.001, .0979])
plt.savefig('output1/pilot.png', dpi=300)
```

## 17.3 Code for parsing raw video

Filename: `parse.py`

Language: `Python 3.8.5`

Packages used:

- `rich 9.10.0`
- `numpy 1.8.5`
- `pandas 1.1.2`
- `opencv-python 4.5.3.56`

```
'''
This script does the following:
- read the list of videos from a specific folder
- for each video, process each frame:
    - for each frame, convert the frame to greyscale
    - perform canny edge detection
    - perform probabilistic hough line transform
    - for each possible location of the rod:
        - select the longest line
        - calculate the length of the rod (in pixels) using Pythagoras\
        - calculate the angle of the rod (in radians) using atan2
    - aggregate the temporal changes of the rod
- store the data in multiple CSV files for further processing
'''

from rich.progress import Progress
import numpy as np
import pandas as pd
import cv2
import os

class Line:
    def __init__(self, line):
        self.p1 = np.array(line[0][:2]).astype(int)
        self.p2 = np.array(line[0][2:4]).astype(int)

    def getLen(self):
        self.length = np.sqrt(np.sum((self.p1 - self.p2) ** 2, axis=0))
        return self

    def getAngle(self):
        self.angle = np.arctan2(
            self.p2[1] - self.p1[1],
            self.p2[0] - self.p1[0] if self.p1[0] < self.p2[0] else self.p1[0] - self.p2[0]
        )
        return self
```

```
with Progress() as progress:
    files = os.listdir('src')
    task0 = progress.add_task(f"[green]Processing files...", total=len(files))
    for filename in files: # process each video in the specified folder
        progress.update(task0, advance=1, description=f"[green]Processing {filename}...")
        capture = cv2.VideoCapture(os.path.join("src", filename))

        totalframecount = int(capture.get(cv2.CAP_PROP_FRAME_COUNT))
        task1 = progress.add_task(f"[green]Parsing frames...", total=totalframecount)
        vals, framecount = [], 0
        while 1:
            success, frame = capture.read()
            if not success:
                break

            image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            edges = cv2.Canny(image, 100, 100, L2gradient=True)
            lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100, maxLineGap=200)

            if lines is not None:
                l = Line(sorted(lines, key=lambda x:Line(x).getLen().length, reverse=True)[0]).getLen().getA
ngle()
                vals.append([framecount, l.length, l.angle])

            progress.update(task1, advance=1)
            framecount += 1

        df = pd.DataFrame(vals, columns=['frame', 'length', 'angle'])
        df.to_csv(os.path.join('output', f'{os.path.splitext(filename)[0]}.csv'), index=False)
```

## 17.4 Code for parsing raw data

Filename: `analyse.py`[2]

Language: `Python 3.8.5`

Packages used:

- `rich 9.10.0`
- `numpy 1.8.5`
- `pandas 1.1.2`
- `scipy 1.18.5`

```
'''
This script does the following:
- read raw data from a specific folder
- for each raw data
    - smoothen noisy data using a non-uniform Savitzky-Golay filter
    - obtain the peaks of the smoothened data
    - for each peak:
        - calculate the average time
        - calculate the difference in time
        - calculate the average length of the rod (in pixels)
        - calculate the average angle of the rod (in radians)
- aggregate all processed data
- store the data in one CSV file for further processing
'''

import os
import numpy as np
import pandas as pd
from scipy.signal import argrelextrema
from rich.progress import Progress
```

[2] The code for the non-uniform Savitzy-Golay filter is obtained from https://dsp.stackexchange.com/a/64313.

```
def non_uniform_savgol_filter(x, y, window, polynom):
    half_window = window // 2
    polynom += 1

    A = np.empty((window, polynom))
    tA = np.empty((polynom, window))
    t = np.empty(window)
    y_smoothed = np.full(len(y), np.nan)

    for i in range(half_window, len(x) - half_window, 1):
        for j in range(0, window, 1):
            t[j] = x[i + j - half_window] - x[i]

        for j in range(0, window, 1):
            r = 1.0
            for k in range(0, polynom, 1):
                A[j, k] = r
                tA[k, j] = r
                r *= t[j]

        tAA = np.matmul(tA, A)
        tAA = np.linalg.inv(tAA)
        coeffs = np.matmul(tAA, tA)

        y_smoothed[i] = 0
        for j in range(0, window, 1):
            y_smoothed[i] += coeffs[0, j] * y[i + j - half_window]

        if i == half_window:
            first_coeffs = np.zeros(polynom)
            for j in range(0, window, 1):
                for k in range(polynom):
                    first_coeffs[k] += coeffs[k, j] * y[j]
        elif i == len(x) - half_window - 1:
            last_coeffs = np.zeros(polynom)
            for j in range(0, window, 1):
                for k in range(polynom):
                    last_coeffs[k] += coeffs[k, j] * y[len(y) - window + j]

    for i in range(0, half_window, 1):
        y_smoothed[i] = 0
        x_i = 1
        for j in range(0, polynom, 1):
            y_smoothed[i] += first_coeffs[j] * x_i
            x_i *= x[i] - x[half_window]

    for i in range(len(x) - half_window, len(x), 1):
        y_smoothed[i] = 0
        x_i = 1
        for j in range(0, polynom, 1):
            y_smoothed[i] += last_coeffs[j] * x_i
            x_i *= x[i] - x[-half_window - 1]

    return y_smoothed

df2s = []
with Progress() as progress:
    files = os.listdir('output')
    task = progress.add_task(f"[green]Processing files...", total=len(files))
    for filename in files:
        progress.update(task, advance=1, description=f'[green]Processing {filename}...')
        df = pd.read_csv(os.path.join("output", filename))
        smoothed = non_uniform_savgol_filter(df.frame.values, df.angle.values, 31, 5)
        idx = argrelextrema(smoothed, np.greater, order=5)[0]

        df1 = pd.DataFrame(columns=["frame", "length", "turning_smoothed"])
```

```
        df1['frame'] = df.iloc[idx].frame
        df1['length'] = df.iloc[idx].length
        df1['turning_smoothed'] = smoothed[idx]

        df1l, df2l = df1.values.tolist(), []
        for i, j in enumerate(df1l[:-1]):
            r0, r1 = df1l[i], df1l[i+1]
            df2l.append([
                (r1[0] + r0[0]) / 2 / 60,
                (r1[0] - r0[0]) / 60,
                (r1[1] + r0[1]) / 2,
                (r1[2] + r1[2]) / 2
            ])

        identifier = os.path.splitext(filename)[0]
        df2s.append(pd.DataFrame(df2l, columns=[
            f"{identifier}_avg_frame",
            f"{identifier}_frame_diff",
            f"{identifier}_avg_length",
            f"{identifier}_avg_turning_smoothed"
        ]))

df2 = pd.concat(df2s, axis=1)
df2.to_csv(os.path.join("output1", "all.csv"), index=False)
```

## 17.5 Other

null