# Assignment 5 5079519

September 17, 2024

Import relevant packages here.

[1]:
```
import matplotlib.pyplot as plt
import numpy
import pandas as pd
import math
```

Load the data and verify it is loaded correctly.

Print it (head, tail, or specific rows, choose a sensible number of rows).

Compare it to the source file.

[2]:
```
import os
print(os.getcwd())

file_path = 'cf_data.csv'
df = pd.read_csv(file_path)
df.head()
```

```
c:\Users\xyria\Downloads
```

[2]:

| | dv | s | a |
|---|---|---|---|
| 0 | -0.743240 | 53.5427 | 1.242570 |
| 1 | -0.557230 | 53.6120 | 1.777920 |
| 2 | -0.454769 | 53.6541 | 0.544107 |
| 3 | -0.525396 | 53.7030 | -0.294755 |
| 4 | -0.601285 | 53.7592 | -0.290961 |

In the ensuing, you will use numpy.

Let's create a grid for the values to plot. But first create two arrays named dv and s using numpy.linspace that hold the grid values at the relevant indices in their respective dimension of the grid.

Create a grid named a with zeros using numpy.zeros in to which calculated acceleration values can be stored. Let the grid span:

Speed difference dv [m/s]

From -10 till 10

With 41 evenly spaced values

```
</li>
<li>Headway <code>s</code> [m]
    <ul>
        <li>From 0 till 200</li>
        <li>With 21 evenly spaced values</li>
    </ul>
</li>
```

[3]:
```
dv = numpy.linspace(-10, 10, num=41, endpoint=True, retstep=False, dtype=None,␣
 ↪axis=0,)
s = numpy.linspace(0, 200, num=21, endpoint=True, retstep=False, dtype=None,␣
 ↪axis=0,)
a = numpy.zeros([41, 21])
```

Create from the imported data 3 separate numpy arrays for each column dv, s and a. (We do this for speed reasons later.)

Make sure to name them differently from the arrays that belong to the grid as above.

You can access the data of each column in a DataFrame using data.xxx where xxx is the column name (not as a string).

Use the method to_numpy() to convert a column to a numpy array.

[9]:
```
DV = df.dv.to_numpy()
S = df.s.to_numpy()
A = df.a.to_numpy()
```

Create an algorithm that calculates all the acceleration values and stores them in the grid. The algorithm is described visually in the last part of the lecture. At each grid point, it calculates a weighted mean of all measurements. The weights are given by an exponential function, based on the 'distance' between the grid point, and the measurement values of dv and s. To get you started, how many for-loops do you need? For this you will need math. Use an upsilon of 1.5m/s and a sigma of 30m. Warning: This calculation may take some time. So:

Print a line for each iteration of the outer-most for-loop that shows you the progress.

Test you code by running it only on the first 50 measurements of the data.

[22]:
```
epsilon = 1.5
sigma = 30

for i in range(len(dv)):
    print(f"Now calculating gridpoint {i+1}/41.")

    for j in range(len(s)):
        sum_omega = 0
        total_omega = 0

        for k in range(len(DV)):
            omega_dv = math.exp(-abs((dv[i] - DV[k])) / epsilon)
```

```
            omega_s = math.exp(-abs((s[j] - S[k])) / sigma)
            weight = omega_dv * omega_s
            sum_omega += weight * A[k]
            total_omega += weight
        a[i, j] = sum_omega / total_omega
```

```
Now calculating gridpoint 1/41.
Now calculating gridpoint 2/41.
Now calculating gridpoint 3/41.
Now calculating gridpoint 4/41.
Now calculating gridpoint 5/41.
Now calculating gridpoint 6/41.
Now calculating gridpoint 7/41.
Now calculating gridpoint 8/41.
Now calculating gridpoint 9/41.
Now calculating gridpoint 10/41.
Now calculating gridpoint 11/41.
Now calculating gridpoint 12/41.
Now calculating gridpoint 13/41.
Now calculating gridpoint 14/41.
Now calculating gridpoint 15/41.
Now calculating gridpoint 16/41.
Now calculating gridpoint 17/41.
Now calculating gridpoint 18/41.
Now calculating gridpoint 19/41.
Now calculating gridpoint 20/41.
Now calculating gridpoint 21/41.
Now calculating gridpoint 22/41.
Now calculating gridpoint 23/41.
Now calculating gridpoint 24/41.
Now calculating gridpoint 25/41.
Now calculating gridpoint 26/41.
Now calculating gridpoint 27/41.
Now calculating gridpoint 28/41.
Now calculating gridpoint 29/41.
Now calculating gridpoint 30/41.
Now calculating gridpoint 31/41.
Now calculating gridpoint 32/41.
Now calculating gridpoint 33/41.
Now calculating gridpoint 34/41.
Now calculating gridpoint 35/41.
Now calculating gridpoint 36/41.
Now calculating gridpoint 37/41.
Now calculating gridpoint 38/41.
Now calculating gridpoint 39/41.
Now calculating gridpoint 40/41.
Now calculating gridpoint 41/41.
```

The following code will plot the data for you. Does it make sense when considering:

Negative (slower than leader) and positive (faster than leader) speed differences?

Small and large headways?

[21]:
```
X, Y = numpy.meshgrid(s, dv)
axs = plt.axes()
p = axs.pcolor(Y, X, a, shading='nearest')
axs.set_title('Acceleration [m/s/s]')
axs.set_xlabel('Speed difference [m/s]')
axs.set_ylabel('Headway [m]')
axs.figure.colorbar(p)
axs.figure.set_size_inches(10, 7)
```