Import relevant packages here.

In [ ]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import math as m
```

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.

In [ ]:
```
data = pd.read_csv('cf_data.csv')

data.head()
```

Out[ ]:

	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
- Headway `s` [m]
  - From 0 till 200
  - With 21 evenly spaced values

In [ ]:
```
dv = np.linspace(-10 , 10 , 41)
s = np.linspace(0 , 200 , 21)
a = np.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.

In [ ]:
```
DV = (data.dv).to_numpy()
S = (data.s).to_numpy()
A = (data.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.

In [ ]:

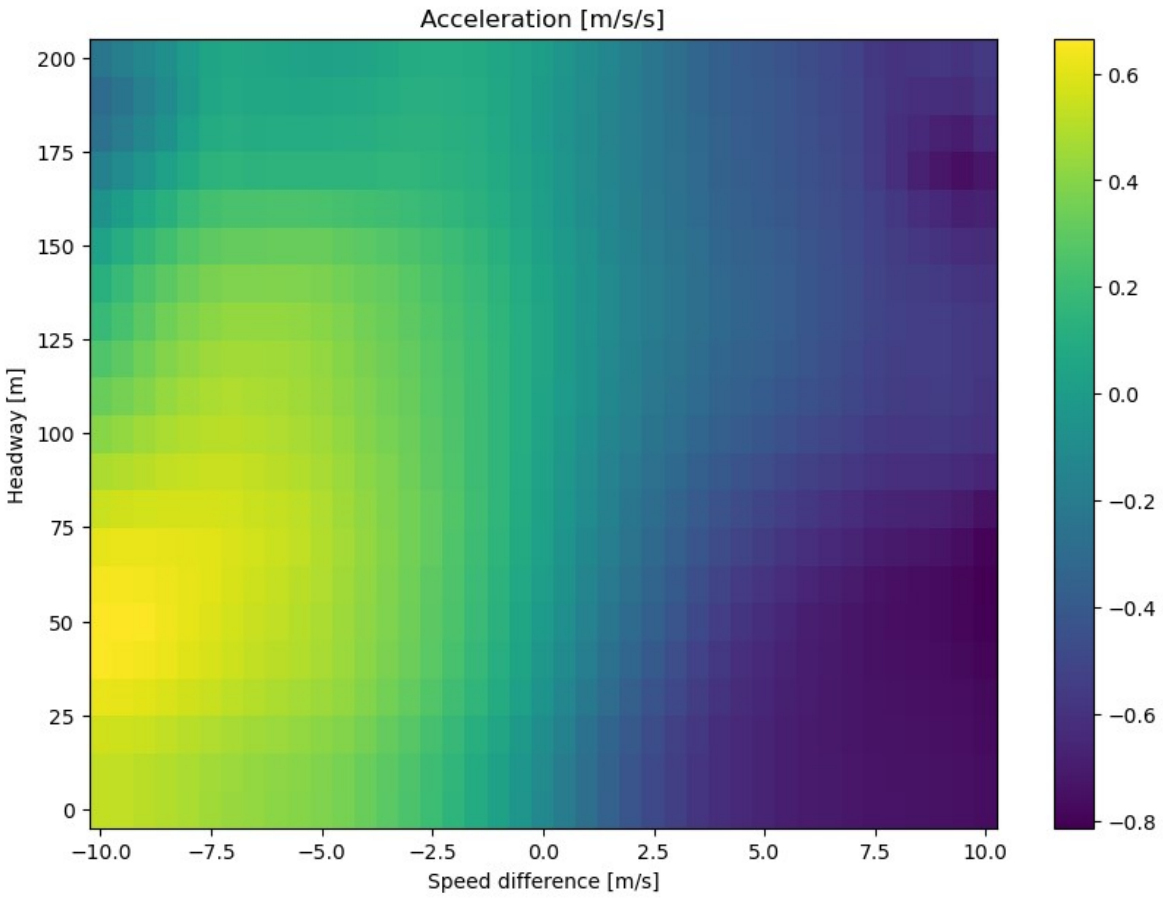
```
# ...
upsilon = 1.5
sigma = 30

for dv_index , speed_diff in enumerate(dv):
    for s_index , headway in enumerate(s):

        weighted_A = 0
        sum_of_weights = 0

        for data_index in range(data.shape[0]):

            measured_dv = data.loc[data_index , 'dv']
            measured_s = data.loc[data_index , 's']
            measured_a = data.loc[data_index , 'a']

            weight_dv = m.exp(-abs(measured_dv - speed_diff) / upsilon)
            weight_s = m.exp(-abs(measured_s - headway) / sigma)
            weight = weight_dv * weight_s

            weighted_A += weight * measured_a
            sum_of_weights += weight

        a[dv_index , s_index] = weighted_A / sum_of_weights
    print('loop %d out of %d' % (dv_index+1 , len(dv)))
```

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

In [ ]:
```
X, Y = np.meshgrid(dv, s)
axs = plt.axes()
p = axs.pcolor(X, Y, a.T, 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)
```


Acceleration [m/s/s]
Headway [m]
200
175
150
125
100
75
50
25
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Speed difference [m/s]
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8