Import relevant packages here.

In [1]:
```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

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 [2]:
```python
data = pd.read_csv('cf_data.csv')
display(data.head(10))
```

	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
5	-0.682448	53.8232	-0.283414
6	-0.768859	53.8957	-0.271604
7	-0.860452	53.9770	-0.133532
8	-0.832777	54.0678	0.243356
9	-0.576125	54.1436	0.406759

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

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 [4]: 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 [5]: upsilon = 1.5
        sigma = 30

        for i in range(len(dv)):
            print(f'Calculating point number {i}')

            for j in range(len(s)):

                som = 0
                totalsom = 0

                for k in range(len(DV)):
                    omega_dv = np.exp(-abs(DV[k] - dv[i]) / upsilon)
                    omega_s = np.exp(-abs(S[k] - s[j]) / sigma)
                    omega = omega_dv * omega_s
                    som += omega
                    totalsom += omega * A[k]

                a[j, i] = totalsom / som
```
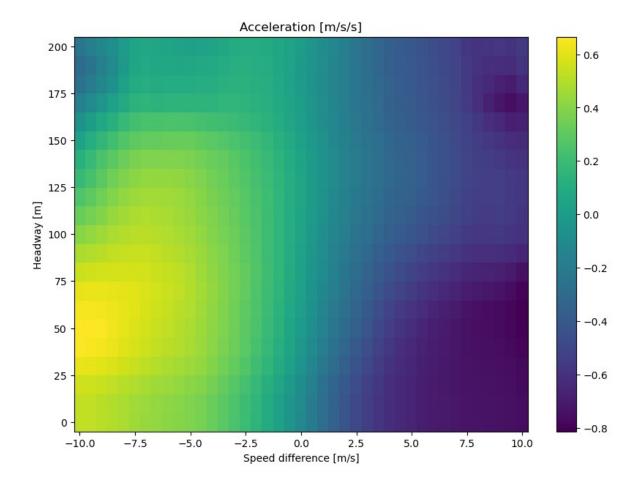
```
Calculating point number 0
Calculating point number 1
Calculating point number 2
Calculating point number 3
Calculating point number 4
Calculating point number 5
Calculating point number 6
Calculating point number 7
Calculating point number 8
Calculating point number 9
Calculating point number 10
Calculating point number 11
Calculating point number 12
Calculating point number 13
Calculating point number 14
Calculating point number 15
Calculating point number 16
Calculating point number 17
Calculating point number 18
Calculating point number 19
Calculating point number 20
Calculating point number 21
Calculating point number 22
Calculating point number 23
Calculating point number 24
Calculating point number 25
Calculating point number 26
Calculating point number 27
Calculating point number 28
Calculating point number 29
Calculating point number 30
Calculating point number 31
Calculating point number 32
Calculating point number 33
Calculating point number 34
Calculating point number 35
Calculating point number 36
Calculating point number 37
Calculating point number 38
Calculating point number 39
Calculating point number 40
```

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?

```python
In [6]:  X, Y = np.meshgrid(dv, s)
         axs = plt.axes()
         p = axs.pcolor(X, Y, 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)
```

Acceleration [m/s/s]