Import relevant packages here.

In [87]:

```
import matplotlib.pyplot as plt
import numpy as np
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.

In [88]:

```
file_path = 'cf_data.csv'

data = pd.read_csv(file_path, encoding='ISO-8859-1')

print(data.head())
print(data.tail())

```

```
         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
            dv        s         a
73903  5.19874  116.139 -0.795081
73904  5.10428  115.627 -0.314263
73905  5.13764  115.118  0.232283
73906  5.15348  114.599  0.262078
73907  5.25868  113.112 -0.612440
```

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 [99]:

```
dv = np.linspace(-10, 10, num=41)
s = np.linspace(0, 200, num=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 [100]:

```
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 [ ]:

```

```

In [101]:

```
# Define constants
upsilon = 1.5  # m/s
sigma = 30.0   # m

# Loop over the grid points

for i in range(41):
    print(f'Iteration number {i}')
    for j in range(21):


        # Define the lists for omega * A and omega
        omega_A = np.zeros(len(DV))
        omega = np.zeros(len(DV))

        # Calculate the omega * A and omega values
        for k in range(len(DV)):
            omega_A[k] = (np.exp(- abs(dv[i] - DV[k]) / upsilon - abs(s[j] - S[k
            omega[k] = np.exp(- abs(dv[i] - DV[k]) / upsilon - abs(s[j] - S[k])

        # Calculate the weighted mean and put it into the matrix a
        a[i, j] = np.sum(omega_A) / np.sum(omega)

```

```
Iteration number 0
Iteration number 1
Iteration number 2
Iteration number 3
Iteration number 4
Iteration number 5
Iteration number 6
Iteration number 7
Iteration number 8
Iteration number 9
Iteration number 10
Iteration number 11
Iteration number 12
Iteration number 13
Iteration number 14
Iteration number 15
Iteration number 16
Iteration number 17
Iteration number 18
Iteration number 19
Iteration number 20
Iteration number 21
Iteration number 22
Iteration number 23
Iteration number 24
Iteration number 25
Iteration number 26
Iteration number 27
Iteration number 28
Iteration number 29
Iteration number 30
Iteration number 31
Iteration number 32
Iteration number 33
Iteration number 34
Iteration number 35
Iteration number 36
Iteration number 37
Iteration number 38
Iteration number 39
Iteration number 40
```

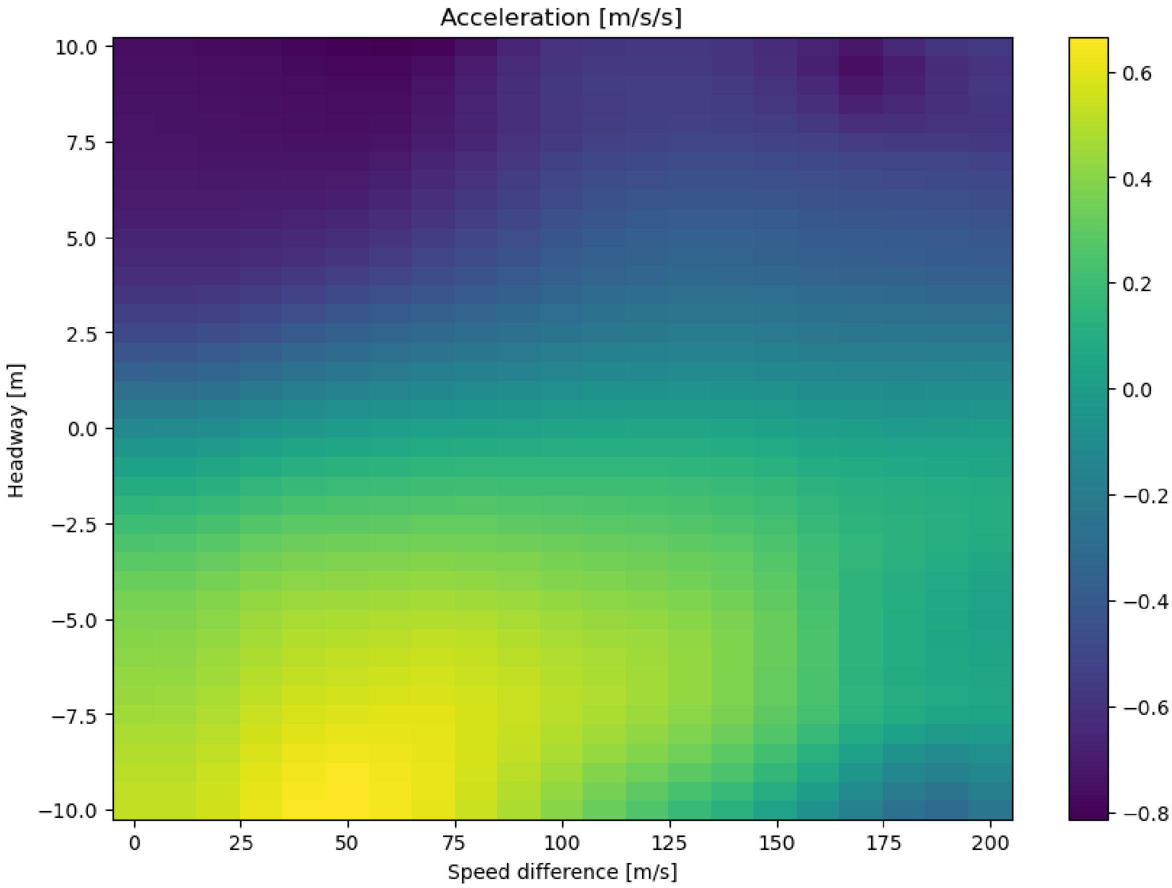
The following code will plot the data for you. Does it make sense when considering:

- Negative (slower than leader) and positive (faster than leader) speed differences?
- Small and large headways?

In [102]:

```
X, Y = np.meshgrid(s, dv)
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)

```



In [ ]:

```

```