Import relevant packages here.

In [15]:
```
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 [16]:
```
data = pd.read_csv('cf_data.csv')

print(data.tail(20))
```

```
             dv         s         a
73888  0.159417   16.3755  0.059915
73889  0.049829   16.3611 -0.471343
73890 -0.001790   16.3655 -0.286011
73891  0.186518   16.3615  0.022933
73892  0.224040   16.3282 -0.530671
73893  0.067944   16.3166 -0.375556
73894  0.066033   16.3146  0.023560
73895  0.011580   16.3034 -0.502929
73896 -0.173098   16.3123 -0.300304
73897 -0.161517   16.3381  0.724566
73898  0.141326   16.3446  0.594131
73899  0.201311   16.3098 -0.332118
73900  0.157068   16.3044 -0.215962
73901  5.199570  117.1910  0.820016
73902  5.262130  116.6660 -0.008254
73903  5.198740  116.1390 -0.795081
73904  5.104280  115.6270 -0.314263
73905  5.137640  115.1180  0.232283
73906  5.153480  114.5990  0.262078
73907  5.258680  113.1120 -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 [17]:
```
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 [18]:
```
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 [19]:
```
# ...

upsilon = 1.5
sigma = 30
for i in range(len(s)):
    #loop through distance
    for j in range (len(dv)):
        #loop through delta speed

        weight = 0
        weighted_A = 0
        weight_sum = 0

        #these first two loops basically make it so we go through all 41x21 cell
        for k in range (len(DV)):
            weight = math.exp(-abs(DV[k] - dv[j])/upsilon - abs(S[k] - s[i])/sig
            weighted_A += weight * A[k]
```

```
            weight_sum += weight
            weighted_mean = weighted_A/weight_sum
            #this inner loop calculates the weighted means for every single cell


        a[i, j] = weighted_mean
```

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 [20]:
```
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)
```