Import relevant packages here.

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np
import math
import pandas as pd
```

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 [8]:
```
data = pd.read_csv('cf_data.csv')
print(data.head(3))
print(data.tail(3))
print(data.loc[9:11])
```

```
         dv        s         a
0 -0.743240   53.5427  1.242570
1 -0.557230   53.6120  1.777920
2 -0.454769   53.6541  0.544107
             dv         s         a
73905   5.13764   115.118  0.232283
73906   5.15348   114.599  0.262078
73907   5.25868   113.112 -0.612440
           dv        s         a
9   -0.576125   54.1436  0.406759
10 -0.423120   54.1830  0.132934
11 -0.482842   54.2282 -0.036750
```

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 [9]:
```
dv = np.linspace(-10, 10, 41)
s = np.linspace(0, 200, 21)
a = np.zeros((len(s), len(dv)))
```

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 [10]:
```
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 [11]:
```
# ...
SIGMA=30
UPSILON=1.5
for i,s_i in enumerate(s):
    print(f"processing row{i+1}of {len(s)}")
    for j,dv_j in enumerate(dv):
        weights=np.exp(-(abs(DV-dv_j))/UPSILON-abs(S-s_i)/SIGMA)
        a[i,j]=np.sum(weights*A)/np.sum(weights)
```

```
processing row1of 21
processing row2of 21
processing row3of 21
processing row4of 21
processing row5of 21
processing row6of 21
processing row7of 21
processing row8of 21
processing row9of 21
processing row10of 21
processing row11of 21
processing row12of 21
processing row13of 21
processing row14of 21
processing row15of 21
processing row16of 21
processing row17of 21
processing row18of 21
processing row19of 21
processing row20of 21
processing row21of 21
```

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 [12]:
```
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)
```