TIL6010
TIL Programming | Python

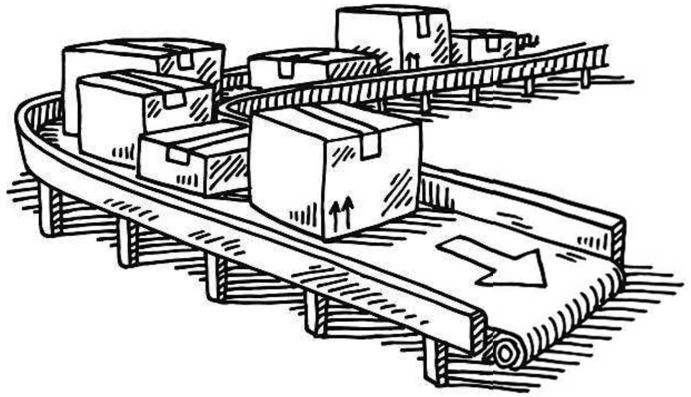
Wouter Schakel

27 September 2021

Bin Packing

Bin Packing

- Pack items in boxes (bins)
  - Weight
  - Dimensions
  - Shape
  - Strength
- We assume only weight is relevant
- Goal: minimize number of boxes required



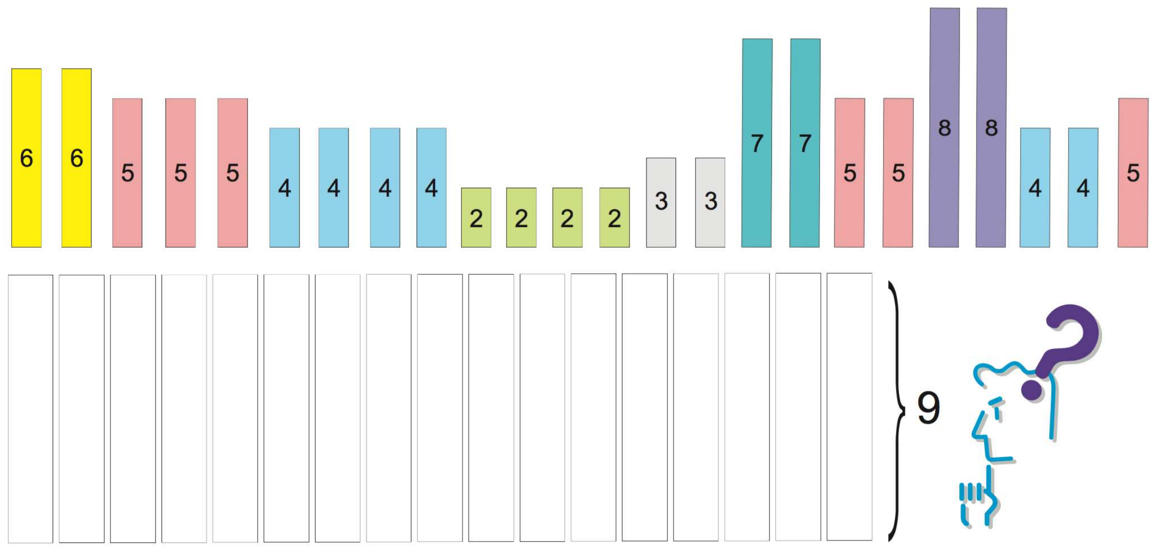
Bin Packing

- Items come ‘one at a time’
- Each box has a capacity
- Limited number of open boxes (closed boxed are not re-opened)

- Items are packed using a *heuristic*:
  - Procedure to obtain a solution
  - Not guaranteed to be optimal

Bin Packing

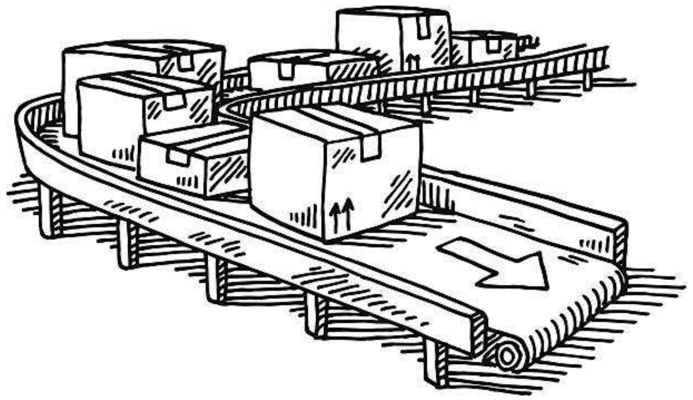
- Come up with a heuristic for packing
- Assume infinite space for open boxes
- One item at a time



Heuristics

Next Fit

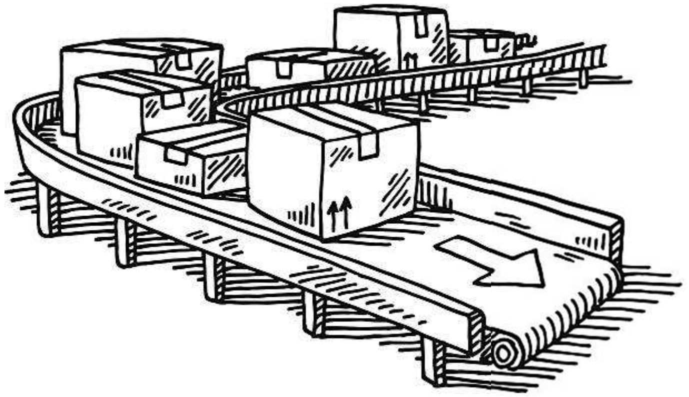
- 1 open box at any time
- If item does not fit, new box
    - Previous box is closed



Heuristics

Next k Fit

- k open boxes at any time
- Place in 1st box it fits in
- If item does not fit, new box
    - Oldest is closed



For-else

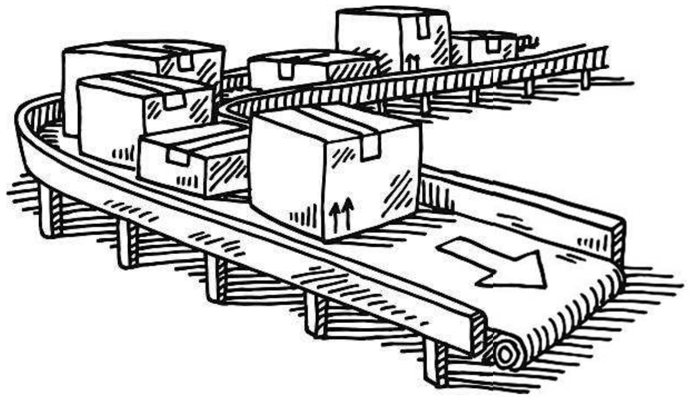
```
if (for-loop is successful):
    do the for-loop
else:
    what needs to happen if the for-loop is not successful
```

```
for (...):
    ...
    break # for-loop was successful
else:
    what needs to happen if the for-loop is not successful
```

Heuristics

First Fit

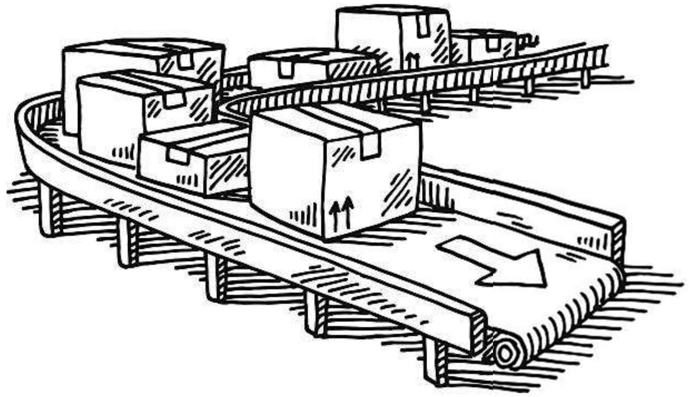
- All boxes open
- Place in 1st box it fits in
- If item does not fit, new box



Heuristics

Best Fit

- All boxes open
- Place in **fullest** box it fits in
- If item does not fit, new box



Heuristics

Offline vs. Online

- Online
  - One item at a time
- Offline
  - All items available
  - Partial heuristic: take largest remaining item

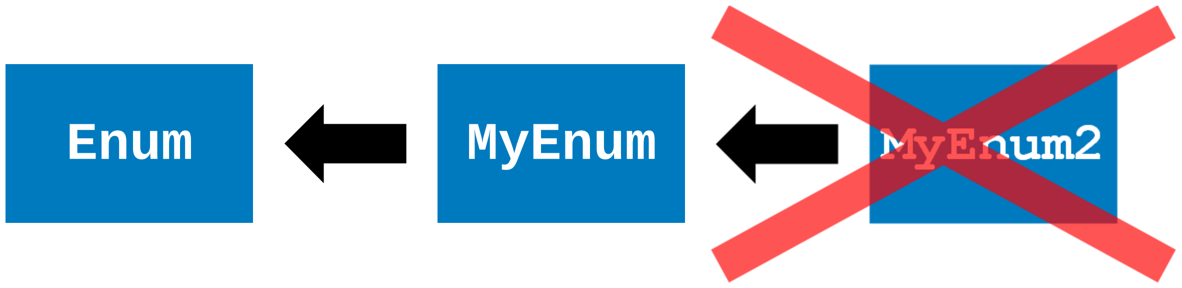
- **Challenge**: how to implement offline algorithms?

Enums

Enums

A special kind of Class

- Confined list of unique and constant values
- Values defined at Class-level
- No other instances can be created



Enums

Examples

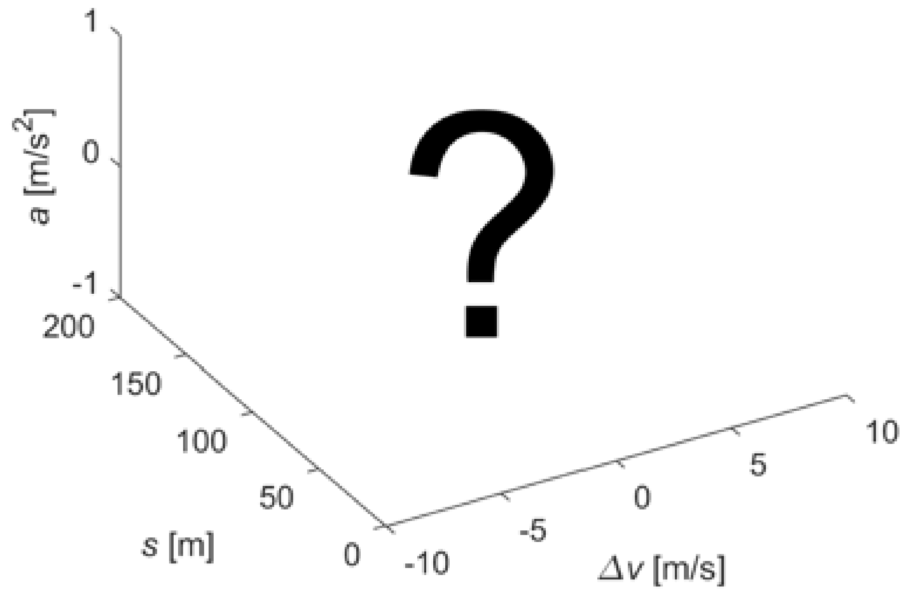
- Traffic light colors
  - RED, AMBER, GREEN
- Smurfs
  - SMURFETTE, PAPA_SMURF, CLUMSY_SMURF, BRAINY_SMURF, …
- Line types
  - CONTINUOS, DASHED, DASH_DOT, DOTTED, DOUBLE, …

Lab session: data smoothing filter

Data smoothing filter

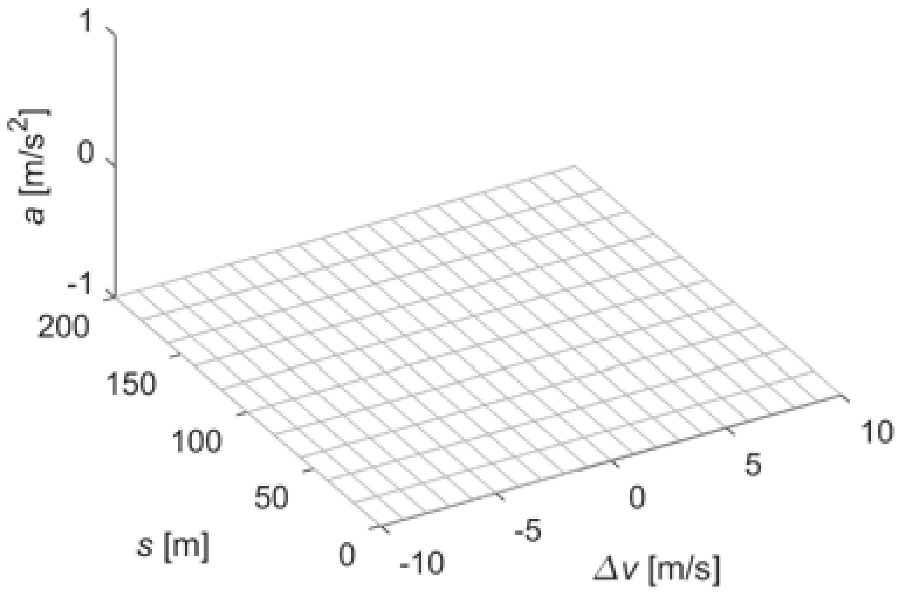
- Acceleration behavior while driving

- Depending on:
  - Speed difference with leader Δv
  - Distance to leader s



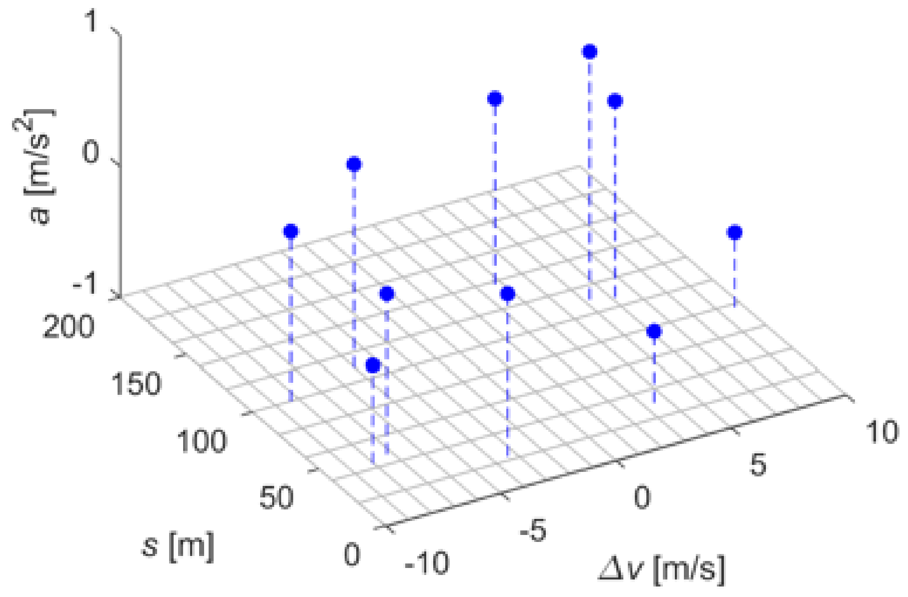
Data smoothing filter

- Numerical approach:
  - We define a grid
  - At each point we calculate a value



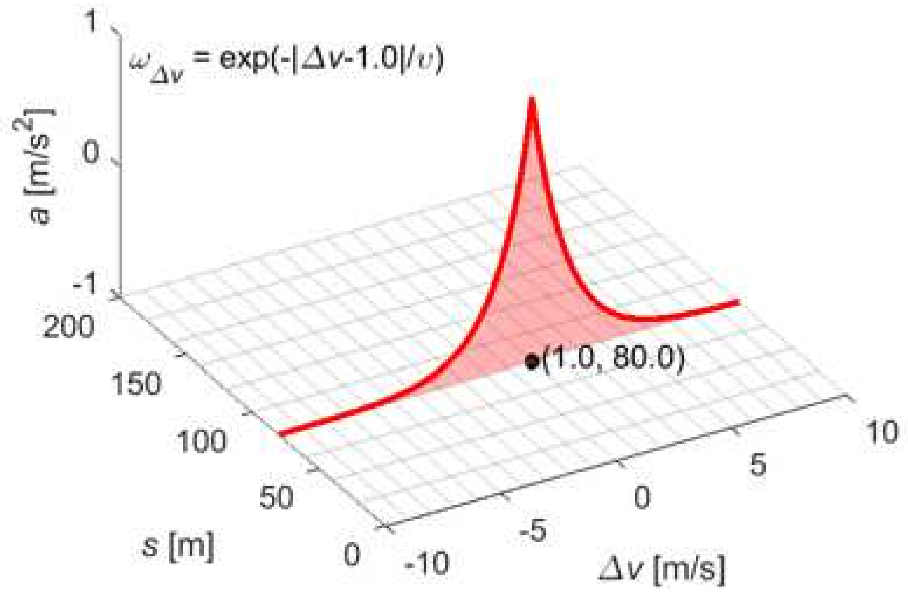
Data smoothing filter

- Suppose we have 10 measurements
- At each point:
  - Calculate weighted mean
  - Weight ~ proximity



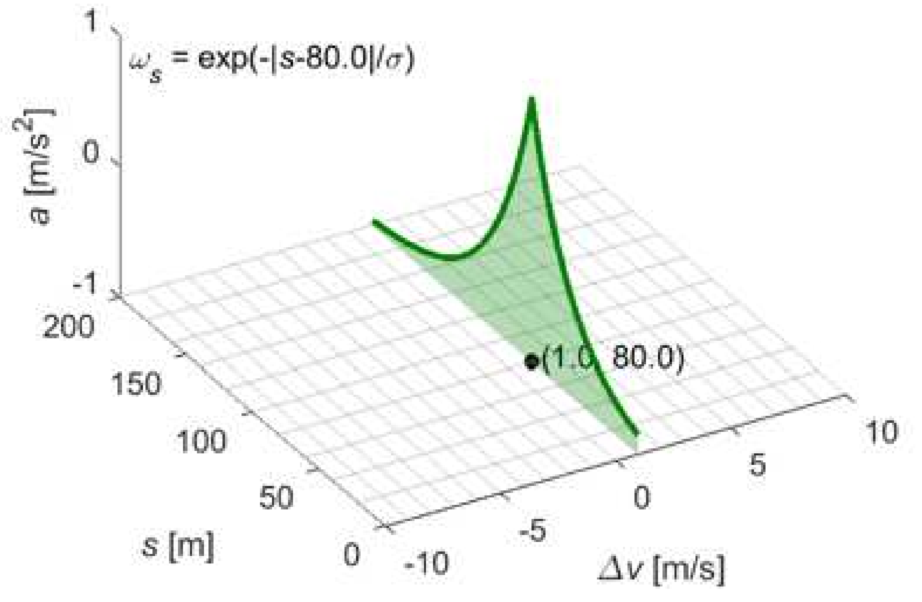
Data smoothing filter

- Exponential weight function *ω*
- Width determined by *υ*



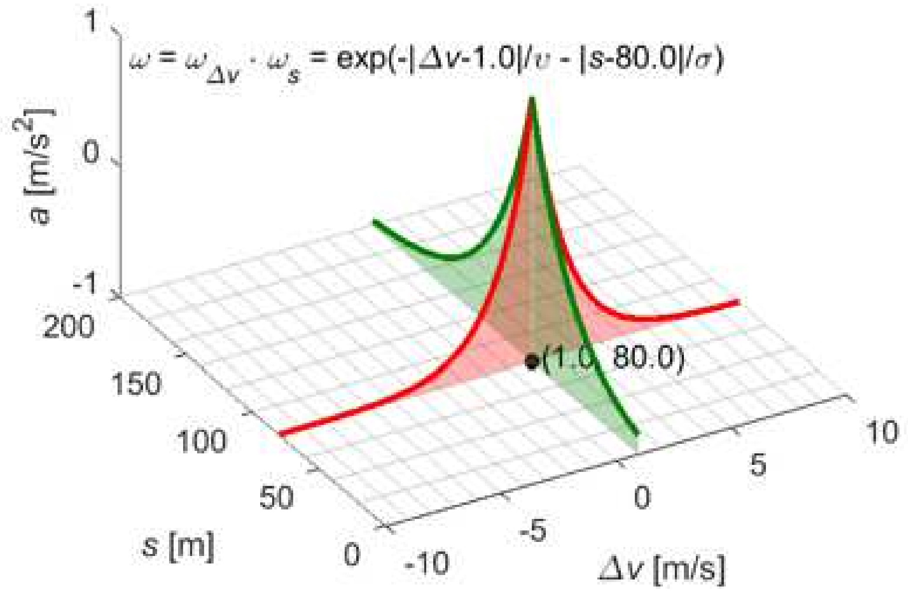
Data smoothing filter

- Similar in other dimension
- Width determined by σ



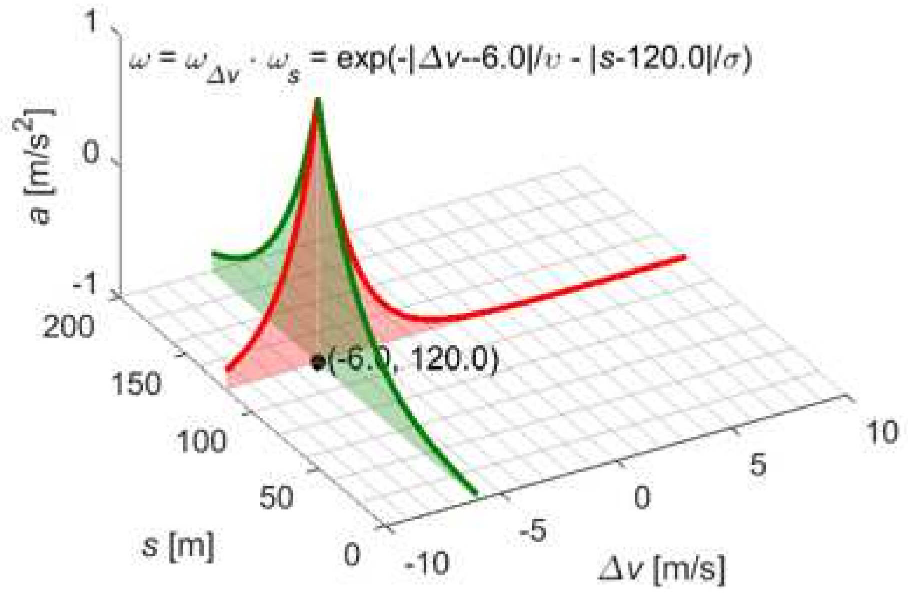
Data smoothing filter

- We multiply weights of both dimensions
- This 3D function is a *kernel*



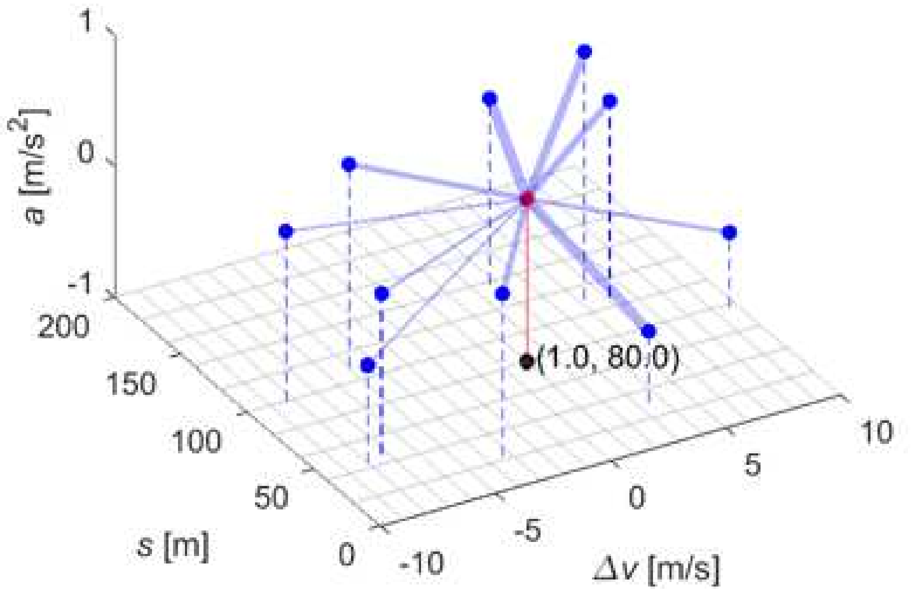
Data smoothing filter

- For another point, the kernel is moved



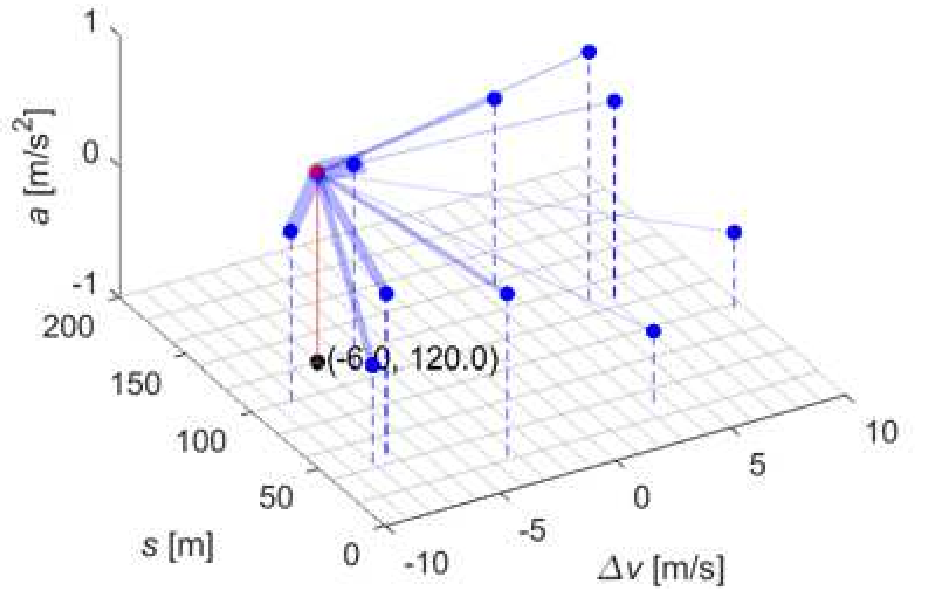
Data smoothing filter

- Calculation at one point
- Thickness of transparent lines indicates weight



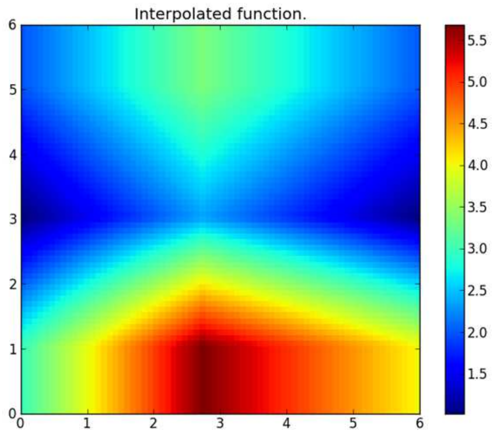
Data smoothing filter

- And for the other point



Data smoothing filter

- Data will be provided
- Goal: 2D pseudo-color plot
  - Color indicates the 3rd dimension (acceleration)
  - Plotting code will be provided


Interpolated function.