Problem description – example 1

You need ego (own) vehicle speed at 10 Hz for an engineering task. Unfortunately CAN-data is missing and only GPS data (incl. speed) of 1 Hz is available. However, longitudinal acceleration is available at 10Hz.

Solution: Use integrated acceleration together with GPS to reconstruct speed. (Alternatives, such as using Kalman filters, are possible, but not covered here).

Files: Use *OffsetAdjustAccX.m*, and *oDAll.mat*

Process:

- Integrate acceleration and plot GPS-speed and integrated acceleration on top of each other
- Choose the point in time where GPS speed can be used to create the offset for the integrated acceleration. Considerations:
    - Choose a point as close to the point of interest for your where you need the speed data (e.g., it you want to study the emergency braking behavior, choose the offset relatively close to the initialization of braking).
    - Choose a point where the surrounding area (~+-0.5s) is as flat/constant as possible. Avoid being in or close to large slopes (high jerk) – this is a conflicting consideration to the one above about choosing a point close to the "interest" area. The reason for wanting to choose a flat/constant part is to avoid that issues with GPS/accelerometer synchronization. That is, if there is a synchronization error there is a risk that the synchronization affects the offset much (highly sensitive to synchronization error)
- Add the GPS speed at the offset point to the integrated acceleration.
- Plot GPS and the new speed.
- Correct for the error/offset in the acceleration by iterating ("for" loop) from some negative acceleration offset, in small steps, to some positive acceleration
    - For each iteration, do the offset adjustment of the integrated acceleration.
    - Create an error metric for the comparison between the GPS speed and the new integrated acceleration.
- Find which offset that produced the minimum error
- Plot it all again….
- Do it for all events you want to use.

Problem description – example 2

You want to get a model for drivers braking in critical situations when a lead-vehicle that the ego vehicle is following is braking hard. You have a number of crashes (or near-crashes). You have the idea to use a three-piece linear model, with an initial acceleration (a_0), a linear jerk (j_B), and a maximum acceleration (a_m), see Figure 1. The task is now to find the parameter values of a_{init}, j_B, T_{init}, and a_{max}, that define the best fit to the data.

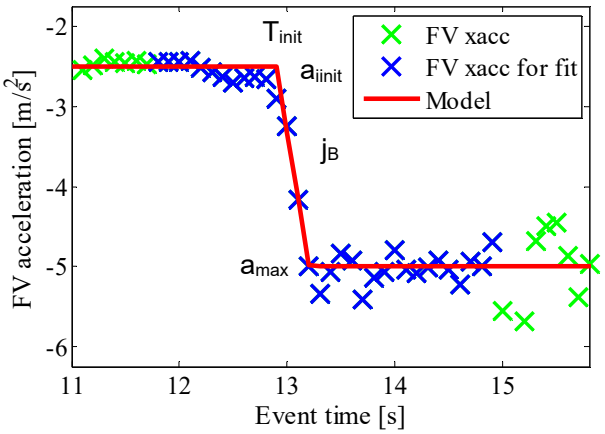

Figure 1: Model of driver acceleration (deceleration) in a critical situation. FV = following vehicle = ego/own vehicle.

Files: Files: Use *FindAccModelParamValues.m*, and the output from Problem 1 (or oDResultsAccOffset.*mat)*

Solution: Do a full parameter sweep for the parameters. The brake-points needs to be parameterized too.

Process:

- Define the data points that should be used for the model fit
- Define the sweeps of the "grid search" (=searching across all possible values, with some resolution within some range) of all parameters.
- Do the grid search through looping
    - Create a brake profile with the given parameter values
    - Compare the values of the created profile with the original one
    - Calculate an error metric and store the corresponding parameter values so that it is possible to retrieve the values for the optimal parameter setting.
- When all error values have been calculated. Find the minimum error value parameter set.
- Create the optimal model and plot together with the original data. Does it look OK?

Extra (more advanced) tasks for those who want to

Synchronization error as well

The same task as for Problem 1, but now there is also a time synchronization error between the GPS and accelerometer data. Correct the synchronization error. Consider how to deal with the combination of acceleration offset and time synchronization to get the 10Hz speed.

Files: Same .m as in Problem 1 as a basis, but use oDAllTimeSync.m instead as the data.

Use a faster way to find the optimum parameter settings

Same as for Problem 2, but use an alternative approach for finding the optimal parameters. Use for example Matlab's "genetic algorithm" called *ga*. (e.g., `x = ga(fitnessfcn,nvars)`). This is a relatively tough assignment.