VEHICULAR MOBILITY MODELLING

MOBMOD COURSE - FALL 2022

PROF. JÉRÔME HÄRRI AND JIN YAN AND ALI NADAR

OBJECTIVES

This lab aims at understanding the behavior of platooning mechanisms (CACC) and the impact of various CACC algorithms related to flow changes. This lab is conducted on the traffic simulator SUMO in addition to the PLEXE extension.
We will investigate:

1. The differences between CFM, ACC and CACC mechanisms in a platoon configuration.
2. The main differences between two leading CACC algorithms on platooning.
3. The impact of sudden break and reacceleration on the platoon parameters.

This lab is based on Ubuntu and SUMO 1.15.0, and it's available at **`/usr/share/sumo`**

GETTING STARTED

We are going to use **SUMO 1.15**, which is already installed on your computer.

SUMO is a complex simulator and its documentation can be found at http://sumo.dlr.de/wiki/Simulation_of_Urban_MObility_-_Wiki.

In this lab we are going not going to use SUMO but its extension for platooning: PLEXE: https://plexe.car2x.org/

PLEXE is a CACC extension module, which connects to SUMO or to network simulators as OMNET++. However, in this lab, we will only use the PLEXE APIs for SUMO available at: https://github.com/michele-segata/plexe-pyapi.

You need to install PLEXE Python APIs for SUMO as follows:

```
#!/bin/bash
cd Lab3
git clone  https://github.com/michele-segata/plexe-pyapi.git
cd plexe-pyapi
pip install --user .
```

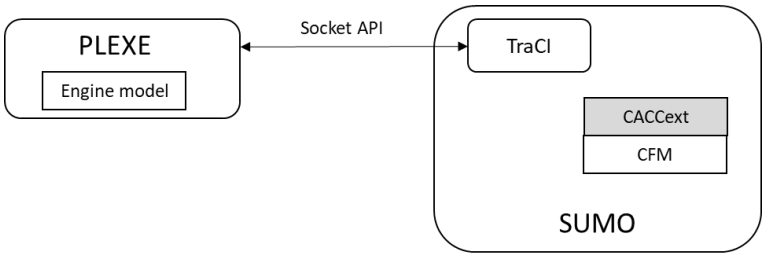
You can run PLEXE as:

```
#!/bin/bash
python3 YOURSCRIPT.py
```

The analysis of the simulation results can be done with a tool of your choice.

PLEXE OVERVIEW

PLEXE consists of a micro-mobility model extension to SUMO supporting platooning. In addition, PLEXE includes a set of Python APIs to control vehicles according to this new model in a highly configurable way.



PLEXE communicates with SUMO via the TraCI (Traffic Communication Interface), which allows it to command SUMO. The socket API also allows PLEXE to run anywhere, independently to SUMO.

PLEXE also extends SUMO with the support of a more realistic Engine model, which aims at reproducing the deviation between the desired acceleration and the real acceleration.

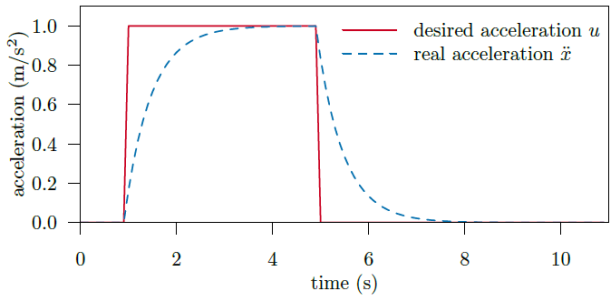


Figure 1 Difference between desired and real acceleration in PLEXE (source: M. Segata).

PLEXE provides a set of Python APIs, which are translated to SUMO instructions. PLEXE requires a SUMO scenario, on top of which it will control one of more vehicles. Accordingly, PLEXE requires SUMO configuration files, .cfg, .rou.xml and .net.xml files to be reachable from the PLEXE script.

PLEXE APIS

We list here the main PLEXE APIs that will be used during this lab. The full list of PLEXE APIs are available here: **/plexe-pyapi/plexe/plexe.py**

`set_active_controller(…)`- specifies which controller (human, ACC, CACC) should be included in a car.

`add_vehicle(…)`- adds a vehicle in SUMO of a configurable type (vtype).

`set_path_cacc_parameters(…)`- sets CACC parameter of the PATH CACC controller.

`set_path_ploeg_parameters(…)`- sets CACC parameter of the PLOEG CACC controller.

`set_cc_desired_speed(…)`- sets the cruise control desired speed, usually the platoon leader

`set_acc_headway_time(…)`- sets the cruise control default headway time.

`set_fixed_lane(…)`- sets the lane on which the PLEXE vehicle will drive.

`use_controller_acceleration`- sets the type of controller to use to define the acceleration.

`get_radar_data(…)`- Returns data measured by a radar sensor (common for ACC and CACC).

`get_distance(…)`- returns the distance between two vehicles.

`communicate(…)`- artificially share the current SUMO topology with the vehicles. In theory, this should be the role of a V2X communication system.

`add_platooning_vehicle(…)`- a helper function to add one vehicle in a platoon calling other APIs.

PLEXE CONTROLLERS

PLEXE has 1 ACC controller and 3 different types of CACC controllers. We list here the main controllers:

DRIVER – considering the car driven by default CFM (Krauss)

ACC – Adaptive Cruise Control (ACC) aim at adapting the speed to maintain a fixed gap, measured by a radar.

CACC – Cooperative Adaptive Cruise Control (CACC) aim at enabling cooperation between vehicles (either driven by a leader, or totally distributive) to maintain a stable platoon/automated driving. CACC controllers have different strategies to reach this objective.

- **PATH –** the US California PATH is the first CACC controller developed 25 years ago. Similar to ACC, it aims at keeping a fixed gap through inter-vehicular cooperation (and communication).
- **PLOEG –** This CACC controller aims instead to maintain a fixed time-headway between vehicles.

In this lab, we will compare PATH and PLOEG CACC controllers in a dynamic platoon.

HOW TO CONFIGURE A PLATOON WITH PLEXE

A platoon needs to be generated on a given road and controlled externally by a TraCI client, which will have PLEXE extensions. PLEXE APIs therefore need to be aThe traffic demand is mainly defined using trips and flows. A trip is associated to a vehicle and it has an origin, a destination and a departure time. Flows are defined once for many vehicles, over a period of time, with origin and destination. Trips and flows are used to generate routes for each vehicle in the simulation.

PLEXE SNIPET FOR TRACI

The APIs are provided by a single class (plexe.Plexe) (which MUST be imported by your script). Plexe must be added as a step listener so that SUMO knows it needs to wait for PLEXE input at each SUMO simulation step.

Look at the following snippet as example:

```
from plexe import Plexe, ACC
import traci

start_sumo("cfg/freeway.sumo.cfg", False)
plexe = Plexe()
traci.addStepListener(plexe)

traci.simulationStep()
plexe.set_active_controller("vehicle.0", ACC)
plexe.set_cc_desired_speed("vehicle.0", 30)
plexe.set_fixed_lane("vehicle.0", 0)
```

Note: this snippet imports Plexe as well as the PATH CACC controller to the script. It starts SUMO with a default 'freeway' scenario. It creates one vehicle on the right-most lane, with a default ACC controller, which speed is set to 30m/s.

PLATOON LEADER

Within a platoon, the leader does not specifically need to have a CACC controller (unless it needs to control the driving dynamics of the whole platoon (not in this lab). A simple ACC controller is enough.

The vehicle which should be the leader is defined as follows:

```
if i == 0
      plexe.set_active_controller(vid, ACC)
```

PLATOON MEMBERS

For each vehicle not considered to be the leader, add a CACC controller of a given type:

```
else:
       plexe.set_active_controller(vid, CACC)
```

Where CACC is the PATH CACC controller.

PLEXE SIMULATION

One you have configured your PLEXE script and the SUMO configuration file, you can run PLEXE as follows:

```
#!/bin/bash
python3 YOURSCRIPT.py
```

If you want to also use a PLEXE extension displaying a dashboard:

```
#!/bin/bash
python3 dashboard-demo.py YOURSCRIPT.py
```

RESULT EVALUATION

Simulation results are mostly extracted from SUMO using the Floating Car Data (cfd) data extraction in XML format. You need to add (extend) the SUMO .cfg file as follows:

```
<output>
     #after other output requests..
     <fcd-output value="fcd.xml"/>
     <fcd-output.acceleration value="true"/>
</output>
```

Acceleration is explicitly requested in order to plot the impact of various controllers on it.

SUMO output is saved in XML format. Among the tools provided by SUMO, you can find a XML to CSV converter: **SUMO_HOME/tools/xml/xml2csv.py -i file.xml -o file.xml**

Once you have the results in CSV format, you can evaluate them with your tool of choice.

SUMO provides some visualization tools too:

```
SUMO_HOME/tools/visualization/plotXMLAttributes.py fcd.xml -x dist -y speed -s
sumo_0.31.0/tools/plot_trajectories.py fcd.xml -t td -o plot.png –s
```

where 'td' represent on combination of data to plot among the following ones:

td: time vs distance
ts: time vs speed
ta: time vs acceleration
ds: distance vs speed
da: distance vs acceleration
xy: Spatial plot of driving path

Alternatively, PLEXE logs data in a file. Check the name and format to extract what you would need.

ASSIGNMENT

Generate a 5 vehicles platoon a multi-lane freeway scenario driving by a human driver.

1. Open the Lab3.py script and apply the required changes.
2. Modify the freeway.sumo.cfg to include CFD output data.
3. Run your script, export data and analyze the speed, the exact inter-vehicular distance as well as the inter-vehicular distance error as a function of the time (or distance). Why is the distance increased at the beginning?
    a. The distance error is defined as the difference between the configured interdistance and the one observed during simulation. Check your lecture nodes for the target interdistance for KRAUSS.
4. Now, replace all vehicles by an ACC controller. And do step 3 again. How is the initial inter-distance adapted?
    a. ACC defines a headway time of 1.5s. You need to transform it to distance according to the speed.
5. Now, replace all vehicles (except the first) with the default (PATH) CACC controller. Do step 3 again. Why is the inter-distance so small? What is the risk?
6. Finally, replace all CACC controller with the PLOEG controller. Do step 3 again. Why is the inter-distance increased again?
7. Compare the graphs and analyze the impact of the various controllers.

Now, you will evaluate the impact of a flow perturbation on the platoon dynamics.

1. Open the Lab3.py script and uncomment the line triggering the leader to briefly break and reaccelerate.
    a. This will make the leader breaks at the 1000 time interval and then at time 1100, accelerate again.
2. Run the script for considering CACC and PLOEG controllers, export data as step 3.
3. Compare the graphs and analyze the impact of the various controllers. What can you say about the shockwave created by the breaking vehicle considering various controllers?