User’s manual for the FMUs

This brief document contains a description on how the various components of the simulation operate and how their behavior is altered by values given to their parameters. The underlined names represent the parameters, monospaced text represents either MATLAB source code, parameters’ names or mathematical formulae.

1 The Leader

The behavior of the leader is modeled by the “Leadcar” FMU. It has been programmed in MATLAB using the Simulink package and the *Vehicle Dynamics Blockset*. It’s mainly composed by three logical units

1. The **acceleration controller**, that uses the FMU’s parameters to generate an acceleration signal – the *desired acceleration* – used to control the car
2. The **vehicle body**, implemented with the aforementioned *blockset*
3. The **attack function**, that’s used to simulate an attack on the vehicle’s sensors

The Leadcar is modeled as show in Figure 5. First the desired acceleration is passed through the vehicle body that generates, for each simulation step, the car’s position using the *bicycle model*; finally the car’s velocity and position outputed by the vehicle body are sent as output to the rest of the simulation. If an attack function is enabled, the aforementioned values are altered to simulate an attack on the car’s sensors.

Name	Type	Unit of measurement	Description
`initial_position`	PARAMETER	meters	Initial position. It’s always 0
`initial_velocity`	PARAMETER	m/s	Initial velocity. It’s always 0
`acceleration_value`	PARAMETER	m/s^2	See below
`deceleration_value`	PARAMETER	m/s^2	See below
`frequency`	PARAMETER	rad/s	See below
`low_frequency`	PARAMETER	Hz	See below
`high_frequency`	PARAMETER	Hz	See below
`operational_mode`	PARAMETER		See below
`amplitude`	PARAMETER	m/s^3	See below
`attack`	PARAMETER		See below
`attack_amplitude`	PARAMETER	m/s^3	See below
`attack_time`	PARAMETER	s	See below
`offset`	PARAMETER	m/s^2	See below
`phase`	PARAMETER	1/s	See below

slope	PARAMETER	m/s^3	See below
sprint_period	PARAMETER		
position_x	OUTPUT	m	Car’s sensor
speed	OUTPUT	m/s	Car’s sensor
acceleration	OUTPUT	m/s^2	Car’s sensor
position_y	OUTPUT	m	Always 0
real_x	OUTPUT	m	Real car’s position, unmodified by any attack on the sensors
real_v	OUTPUT	m/s	As real_x but for the speed
real_a	OUTPUT	m/s^2	As real_x but for the acc.
~~attacked~~	~~OUTPUT~~	~~bool~~	NOT USED
time	INPUT	s	Simulator’s clock

Acceleration controller

The generated output signal $a(t)$ is shaped by certain FMU’s parameters.

Let us define the following functions:

- `fun1(t) = amplitude*sin(frequency * t + phase) + offset`
- `fun2(t) = slope*t`
- `final_steps(t) = 0.03*sin(1.5 * t) + 0.005`

Note how the `frequency` parameter is expressed in rad/s instead of Hertz. The final steps function is a «[s]ine function with very low amplitude and high frequency, to model the fact that in a real scenario it’s not possible to obtain a constant speed, thus the acceleration is almost never set to 0 constantly» (Deliverable D2.1, pg 9)

The parameter `operational_mode` can be defined as 0, 1 or 2.

In **operational mode 0**, the leader accelerates at constant acceleration `acceleration_value` for the first 10 seconds of the simulations then follows `fun1(t)`.

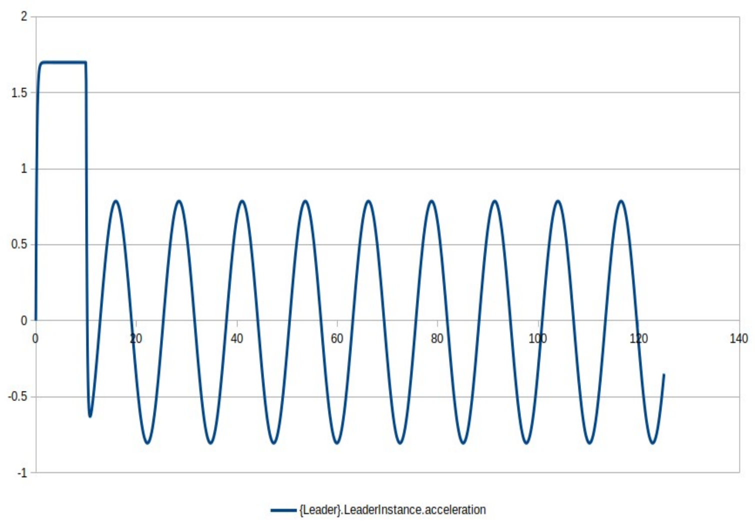
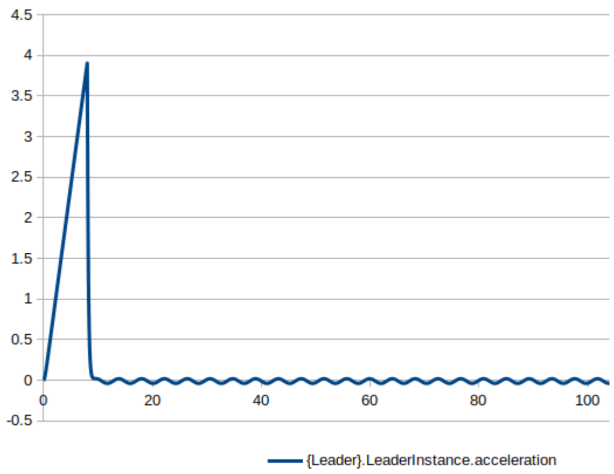


Figura 1: Acceleration function in mode 0

In **operational mode 1**, the leader accelerates following `fun2(t)` for the first 8 seconds then switches to `fun1(t)`.

In **operational mode 2**, let us call `T = sprint_period + 30`, the leader accelerates and decelerates following a periodic function of period `T` defined as:

```
a(t) =    acceleration_value      if t % T < sprint_period
          final_steps(t)          if sprint_period <= t % T < 30
          deceleration_value      if t % T < 30 + sprint_period
```



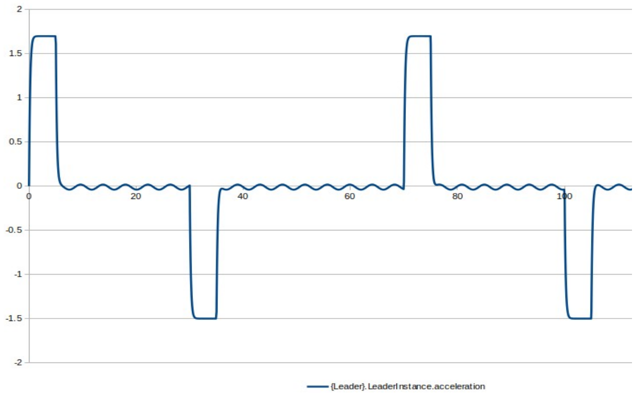


Figura 2: Acceleration in mode 2

Figura 3: Acceleration in mode 1

Note how the signs of the acceleration and deceleration values must be opposite to each other, otherwise the velocity will diverge to infinity. Note also that it is wise to never set `sprint_period >= 30`.

Attack function

The attack function, when enabled, adds a spurious signal on top of the sensors' signal – that are just the outputs of the vehicle dynamics block, – resulting in erroneous or noisy values being communicated. The **type of the attack** is specified by the `attack` parameter and can be defined as 0, 1, 2 or 3. The attack begins when `attack_time > t` and continues for the rest of the simulation.

Let us define the following acceleration functions:

- `low_freq_sine(t)  = attack_amplitude*sin(low_frequency * (2*π) * t)`
- `high_freq_sine(t) = attack_amplitude*sin(high_frequency * (2*π) * t)`

These accelerations are then added to the acceleration, velocity and position in a coherent way.

- In **attack mode 0**, no attack takes place.
- In **attack mode 1**, the low frequency sine is added.
- In **attack mode 2**, the high frequency sine is added.
- In **attack mode 3**, both functions are added.

Nota bene: the default value for `attack` is 1, so if its value is not explicitly set in your config file, the attack will take place.

2 The follower car

The behavior of the follower cars is modeled by the “SimpleCar” FMU. It has been programmed in MATLAB using the Simulink package and the *Vehicle Dynamics Blockset*. It’s mainly composed by three logical units.

1. The **attack function on the actuator**, it alters the value of the desired acceleration computed by the cruise control
2. The **vehicle body**, implemented with the aforementioned *blockset*
3. The **attack function on the sensors**, it works the same as the leader’s

The follower car is modeled as shown in Figure 6.

Name	Type	Unit of measurement	Description
`desired_acceleration`	INPUT	m/s^2	Target acceleration
`position_x`	OUTPUT	m	Car’s sensor
`speed`	OUTPUT	m/s	Car’s sensor
`acceleration`	OUTPUT	m/s^2	Car’s sensor
`position_y`	OUTPUT	m	It’s always 0
`real_x`	OUTPUT	m	Real car’s position, unmodified by any attack on the sensors
`real_v`	OUTPUT	m/s	As real_x but for the speed
`real_a`	OUTPUT	m/s^2	As real_x but for the acc.
~~`attacked`~~	~~OUTPUT~~	~~bool~~	Not used
`time`	INPUT	s	Simulator’s clock
`attack`	PARAMETER		See below
`attack_amplitude`	PARAMETER	m/s^3	See below
`attack_time`	PARAMETER	S	See below
`high_frequency`	PARAMETER	Hz	See below
`initial_position`	PARAMETER	m	Starting position 0
`initial_velocity`	PARAMETER	m/s	Starting velocity 0
`low_frequency`	PARAMETER	Hz	See below
`vehicle_starting_time`	PARAMETER	s	Start time

Attack function

The attack function, when enabled, adds a spurious signal on top of the sensors’ signal, resulting in erroneous or noisy values being communicated. The **type of the attack** is specified by the `attack` parameter and can be defined as 0, 1, 2, 3, 4 or 5. The attack begins when `attack_time > t` and continues for the rest of the simulation. The attacks 0, 1, 2 and 3 operate on the **sensors** and operate the same way as the leader's, refer to Section 1.

- In **attack mode 4** the acceleration value is altered as follows: `a_(t) = a(t) * (1 + attack_amplitude)`, that is the car's acceleration is amplified by a certain factor
- In **attack mode 5** the acceleration value is altered as follow: `a_(t) = a(t) + attack_amplitude`, that is the car's acceleration is slightly increased or decreased

Nota bene: the default value for `attack` is 1, so if its value is not explicitly set in your config file, the attack will take place.

Vehicle body

As for the leader, the physics of the vehicle are handled by the *Vehicle Dynamics* block; however the parameter `vehicle_starting_time` keeps the `a_(t) = 0` until `vehicle_starting_time < t`.

3 Platoon MEC

This FMU models the network and the CACC system. MEC stands for Multi-access Edge Computing. The following table describes the FMU's variables, the for brevity's sake all the cars' variables have been shorten with a variable **i** that varies from 0 to 9, where 0 represent the leader, the remaining the follower cars.

Name	**Type**	**Unit of measurement**	**Description**
`platoon_size`	PARAMETER		Self explanatory
`network_uplink_delay`	PARAMETER	ms	Distribution of the uplink delay channel. This figure represents the average
`network_downlink_delay`	PARAMETER	ms	Distribution of the downlink delay channel. This figure represents the average
`platoon_distance_strategy`	PARAMETER	m	The distance to keep between vehicles
~~`cacc_target_distance`~~	~~INPUT~~	m	CACC controller target distance. NOT USED
`platoon_0_i_length`	PARAMETER	m	Vehicle i's length
`platoon_0_i_pos_x`	INPUT	m	Vehicle i's X position
`platoon_0_i_pos_y`	INPUT	m	Vehicle i's Y position (0)
`platoon_0_i_speed`	INPUT	m/s	Vehicle i's speed
`platoon_0_i_acceleration`	INPUT	m/s²	Vehicle i's linear acceleration
`platoon_0_i_des_acc`	OUTPUT	m/s²	Desired acceleration for i. Unused for i = 0

The CACC’s desired accelerations for the platoon’s vehicles are computed by the following procedure:

```
def compute_follower_acceleration_decoration(self, target_distance, *vehicles_data):
      current_vehicle, front_vehicle, platoon_leader_vehicle = vehicles_data
      # xi = 1
      # w_n = 0.2  # 2 Hz
      alpha_1 = 1 - self.C1
      alpha_2 = self.C1
      c1_xi = self.C1 * (self.xi + math.sqrt(math.pow(self.xi, 2) - 1))
      alpha_3 = - (2 * self.xi - c1_xi) * self.w_n
      alpha_4 = - c1_xi * self.w_n
      alpha_5 = - math.pow(self.w_n, 2)

      if self.current_vehicle.front_vehicle_distance is None:
            [...]
      else:
            follower_front_vehicle_distance = current_vehicle.front_vehicle_distance

      eps_space = - follower_front_vehicle_distance + target_distance
      eps_speed = current_vehicle.speed - front_vehicle.speed

      desired_acc = alpha_1 * front_vehicle.acceleration
      desired_acc += alpha_2 * platoon_leader_vehicle.acceleration
      desired_acc += alpha_3 * eps_speed
      desired_acc += alpha_5 * eps_space
      desired_acc += alpha_4 * (current_vehicle.speed - platoon_leader_vehicle.speed)

      desired_acc = max(-current_vehicle.max_deceleration,
                        min(current_vehicle.max_acceleration, desired_acc))
      return desired_acc, cs.DEFAULT_PLATOON_MANEUVER_TIME
```

4 The simulations and their parameters

Figura 4 shows how the various elements of the simulation are linked togheter.

Ogni cartella contiene una serie di cartelle il cui nome rappresenta i valori scelti per ciascun parametro che definisce la simulazione. I parametri di interesse sono descritti sotto.

Ogni cartella è relativa a una tipologia di macro scenario, in particolare abbiamo:

- **No attack**: simulazioni del sistema nel caso nominale.
- **Attack leader**: vengono simulati gli attacchi ai sensori del leader.
- **Attack middle**: diviso in parte 1 e 2, rappresentano rispettivamente gli attacchi ai sensori (1,2 e 3) e all’attuatore (4 e 5) del veicolo numero 4 (il quinto del platoon).
- **Attack first**: come sopra, ma relativi al veicolo 1, ovvero il secondo nel platoon, quello subito dietro al leader.

I valori nello specifico come già detto si possono trovare nella cartella, che conterrà anche il file results.csv, contenente i dati di simulazione.

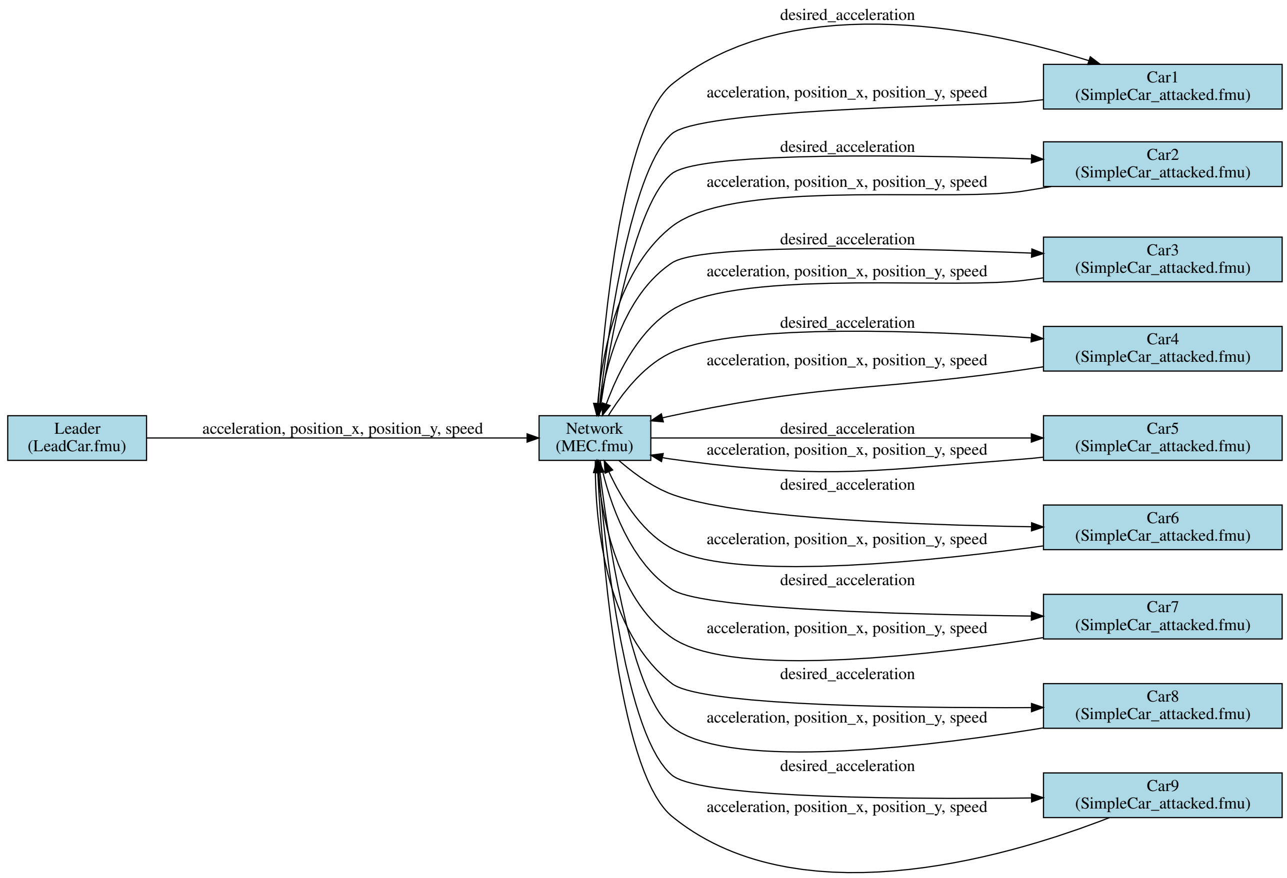


Figura 4: Topology of the simulation

For each batch of simulations, an `index.html` file is provided, it provides a table with the parameters used for each simulation and links to the folder, the results and config files; you can open said HTML file in your web browser.

The generated CSV stores the following parameters:

1. Leader

- **Leader.position_x**
- **Leader.speed**
- **Leader.acceleration**
- **Leader.position_y** It should be 0

2. CarX (veicoli da Car1 a Car9)

- For each vehicle **Car1**, **Car2**, ..., **Car9**
  - **CarX.position_x**
  - **CarX.speed**
  - **CarX.acceleration**
  - **CarX.position_y** It should be 0

3. Network (platoon acc)

- **Network.platoon_0_X_des_acc** Each of these is the value that the CACC algorithm in the edge decided for each vehicle

The following table reports how the various parameters are set for each batch of simulations. The Cartesian product is performed when multiple lists of parameters are given. The table has been generated from the various `dse.json` files, each batch folder has one and, once merged with the `mm.json` file located in the multi-models' folder, represent the configuration used to run the batch.

All cars start in position 0, each car waits 4 seconds for the car in front to start moving, so the first car will start moving after 4 seconds from the simulation start, the second after 8 and so on…

Parameter	Attack First p.1	Attack First p.2	Attack Leader	Attack Middle p.1	Attack Middle p.2	No Attack
{Car1}.CarInstance_1.attack	1, 2, 3	4, 5	0	0	0	0
{Car1}.CarInstance_1.attack_amplitude	0.08	0.08				
{Car1}.CarInstance_1.attack_time	30	30				
{Car1}.CarInstance_1.high_frequency	172	172				
{Car1}.CarInstance_1.low_frequency	0.1	0.1				
{Car1}.CarInstance_1.vehicle_starting_time	4	4				4
{Car2}.CarInstance_2.attack	0	0	0	0	0	0
{Car3}.CarInstance_3.attack	0	0	0	0	0	0
{Car4}.CarInstance_4.attack	0	0	0	1, 2, 3	4, 5	0
{Car4}.CarInstance_4.attack_amplitude				0.08	0.08, -0.08	
{Car4}.CarInstance_4.attack_time				30	30	
{Car4}.CarInstance_4.high_frequency	172	172		172	172	172
{Car4}.CarInstance_4.low_frequency	0.1	0.1		0.1	0.1	0.1
{Car4}.CarInstance_4.vehicle_starting_time	16	16	16	16	16	16
{Car5}.CarInstance_5.attack	0	0	0	0	0	0
{Car6}.CarInstance_6.attack	0	0	0	0	0	0
{Car7}.CarInstance_7.attack	0	0	0	0	0	0
{Car8}.CarInstance_8.attack	0	0	0	0	0	0
{Car9}.CarInstance_9.attack	0	0	0	0	0	0
{Leader}.LeaderInstance.acceleration_value	1.7, 2	1.7, 2	1.7, 2	1.7, 2	1.7, 2	1.7, 2
{Leader}.LeaderInstance.amplitude	0.8	0.8	0.8	0.8	0.8	0.8
{Leader}.LeaderInstance.attack	0	0	1, 2, 3	0	0	0
{Leader}.LeaderInstance.attack_amplitude			0.08, 1			
{Leader}.LeaderInstance.attack_time			30			
{Leader}.LeaderInstance.deceleration_value	-1.5, -1.7	-1.5, -1.7	-1.5, -1.7	-1.5, -1.7	-1.5, -1.7	-1.5, -1.7
{Leader}.LeaderInstance.frequency	0.628, 0.5	0.628, 0.5	0.628, 0.5	0.628, 0.5	0.628, 0.5	0.628, 0.5
{Leader}.LeaderInstance.high_frequency			172			
{Leader}.LeaderInstance.initial_position	0	0	0	0	0	0
{Leader}.LeaderInstance.initial_velocity	0	0	0	0	0	0
{Leader}.LeaderInstance.low_frequency			0.1			

Parameter	Attack First p.1	Attack First p.2	Attack Leader	Attack Middle p.1	Attack Middle p.2	No Attack
{Leader}.LeaderInstance.offset	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
{Leader}.LeaderInstance.operational_mode	0, 1, 2	0, 1, 2	0, 1, 2	0, 1, 2	0, 1, 2	0, 1, 2
{Leader}.LeaderInstance.phase	0	0	0	0	0	0
{Leader}.LeaderInstance.slope	0.5	0.5	0.5	0.5	0.5	0.5
{Leader}.LeaderInstance.sprint_period	5	5	5	5	5	5
{Network}...network_downlink_delay	3, 5	3, 5	3, 5	3, 5	3, 5	3, 5
{Network}...network_uplink_delay	8, 10	8, 10	8, 10	8, 10	8, 10	8, 10
{Network}...platoon_0_0_length	4	4	4	4	4	4
{Network}...platoon_0_1_length	4	4	4	4	4	4
{Network}...platoon_0_2_length	4	4	4	4	4	4
{Network}...platoon_0_3_length	4	4	4	4	4	4
{Network}...platoon_0_4_length	4	4	4	4	4	4
{Network}...platoon_0_5_length	4	4	4	4	4	4
{Network}...platoon_0_6_length	4	4	4	4	4	4
{Network}...platoon_0_7_length	4	4	4	4	4	4
{Network}...platoon_0_8_length	4	4	4	4	4	4
{Network}...platoon_0_9_length	4	4	4	4	4	4
{Network}...platoon_size	10	10	10	10	10	10
{Car1}.CarInstance_1.initial_position	0	0	0	0	0	0
{Car1}.CarInstance_1.initial_velocity	0	0	0	0	0	0
{Car1}.CarInstance_1.vehicle_starting_time	4	4	4	4	4	4
{Car2}.CarInstance_2.initial_position	0	0	0	0	0	0
{Car2}.CarInstance_2.initial_velocity	0	0	0	0	0	0
{Car2}.CarInstance_2.vehicle_starting_time	8	8	8	8	8	8
{Car3}.CarInstance_3.initial_position	0	0	0	0	0	0
{Car3}.CarInstance_3.initial_velocity	0	0	0	0	0	0
{Car3}.CarInstance_3.vehicle_starting_time	12	12	12	12	12	12
{Car4}.CarInstance_4.initial_position	0	0	0	0	0	0
{Car4}.CarInstance_4.initial_velocity	0	0	0	0	0	0
{Car4}.CarInstance_4.vehicle_starting_time	16	16	16	16	16	16
{Car5}.CarInstance_5.initial_position	0	0	0	0	0	0
{Car5}.CarInstance_5.initial_velocity	0	0	0	0	0	0
{Car5}.CarInstance_5.vehicle_starting_time	20	20	20	20	20	20
{Car6}.CarInstance_6.initial_position	0	0	0	0	0	0
{Car6}.CarInstance_6.initial_velocity	0	0	0	0	0	0
{Car6}.CarInstance_6.vehicle_starting_time	24	24	24	24	24	24
{Car7}.CarInstance_7.initial_position	0	0	0	0	0	0

Parameter	Attack First p.1	Attack First p.2	Attack Leader	Attack Middle p.1	Attack Middle p.2	No Attack
{Car7}.CarInstance_7.initial_velocity	0	0	0	0	0	0
{Car7}.CarInstance_7.vehicle_starting_time	28	28	28	28	28	28
{Car8}.CarInstance_8.initial_position	0	0	0	0	0	0
{Car8}.CarInstance_8.initial_velocity	0	0	0	0	0	0
{Car8}.CarInstance_8.vehicle_starting_time	32	32	32	32	32	32
{Car9}.CarInstance_9.initial_position	0	0	0	0	0	0
{Car9}.CarInstance_9.initial_velocity	0	0	0	0	0	0
{Car9}.CarInstance_9.vehicle_starting_time	36	36	36	36	36	36

5 Appendix

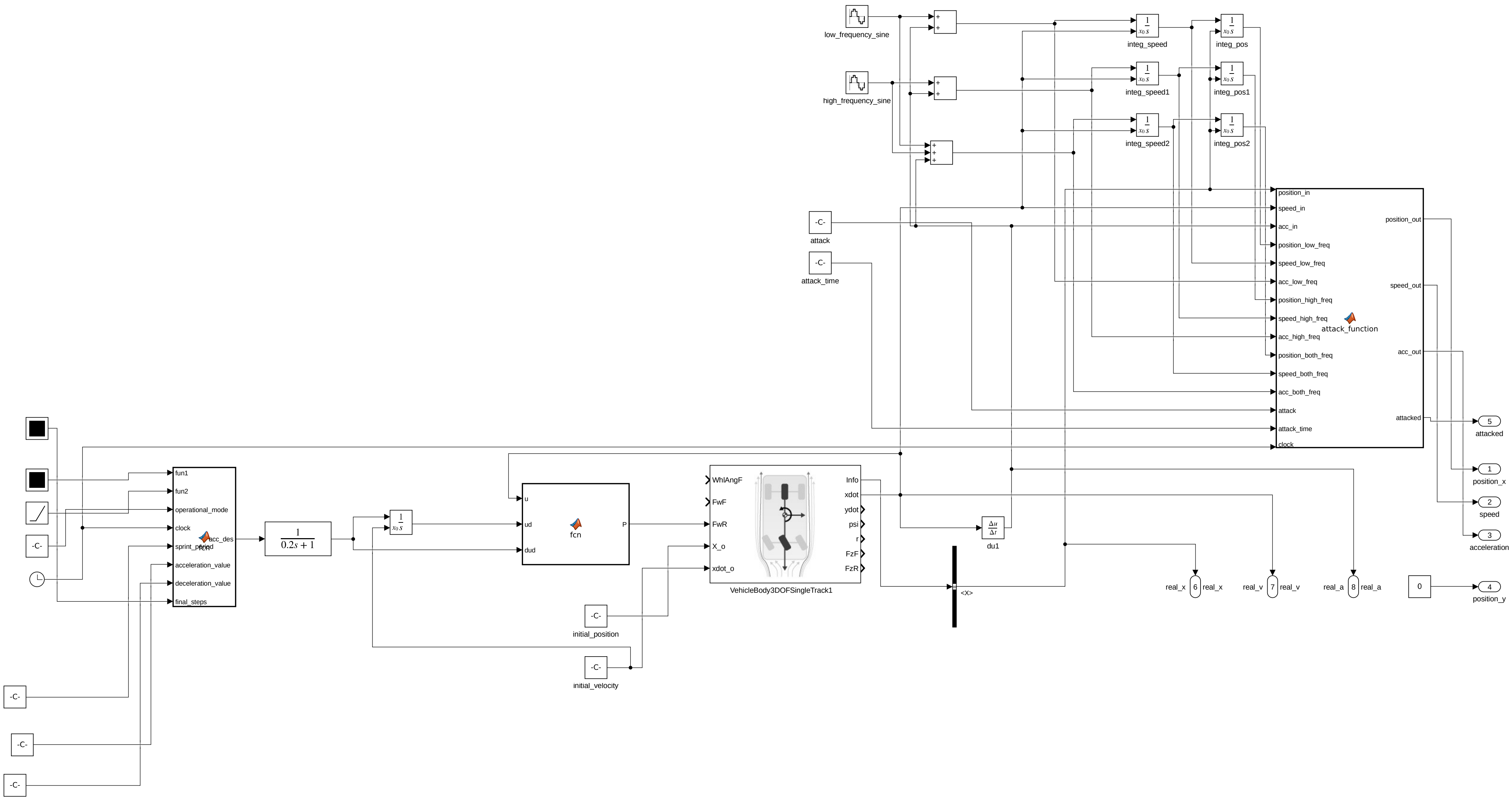


Figure 5: FMU of the leader

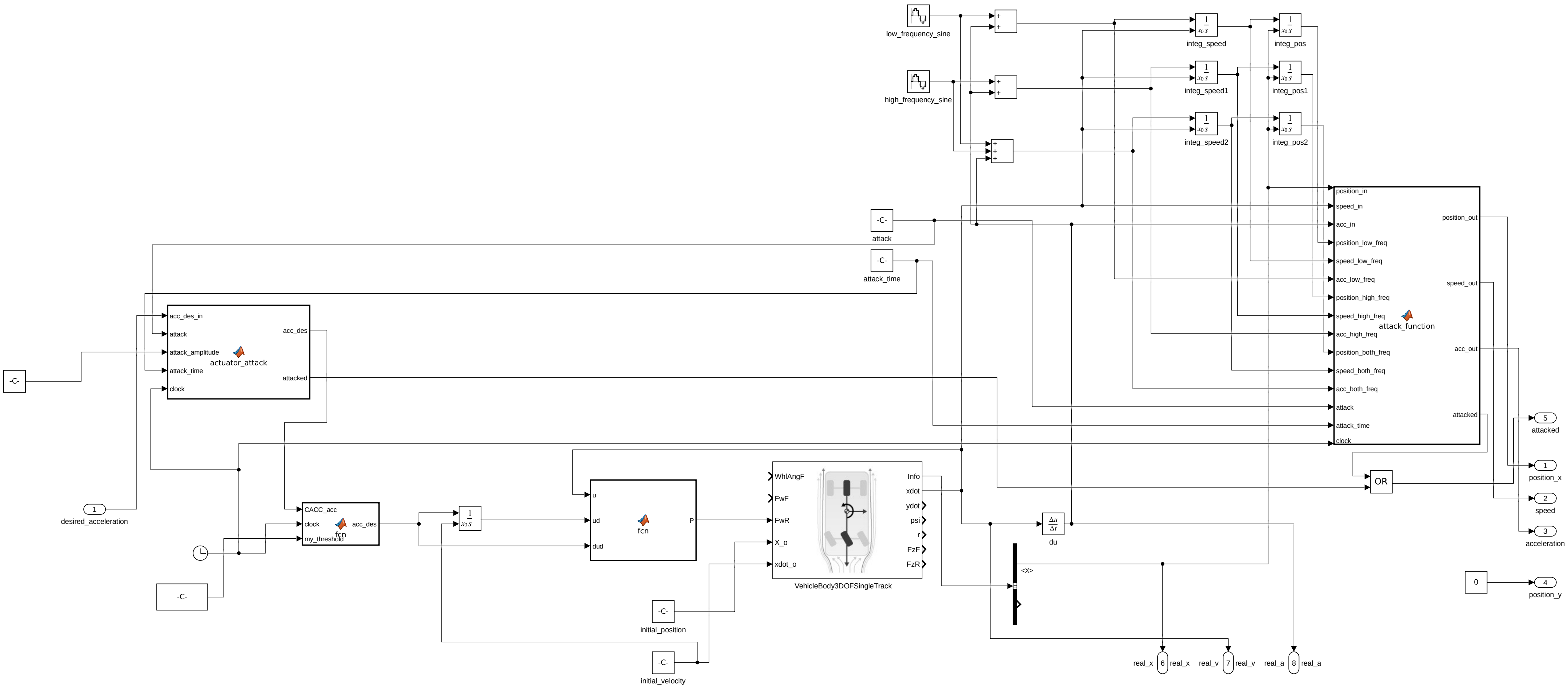


Figure 6: FMU of the follower car