Models for Distributed Real-Time Simulation in a Vehicle Co-Simulator Setup

Anders Andersson[1] Peter Fritzson[2]

[1]Swedish National Road and Transportation Research Institute, Sweden, anders.andersson@vti.se
[2]IDA, Linköping University, Sweden, peter.fritzson@liu.se

Abstract

A car model in Modelica has been developed to be used in a new setup for distributed real-time simulation where a moving base car simulator is connected with a real car in a chassis dynamometer via a 500m fiber optic communication link. The new co-simulator set-up can be used in a number of configurations where hardware in the loop can be interchanged with software in the loop. The models presented in this paper are the basic blocks chosen for modeling the system in the context of a distributed real-time simulation, estimating parameters for the powertrain model, the choice of numeric solver, and the interaction with the solver for real-time properties.

Keywords: Modelica, real-time, distributed, communications link

1. Introduction

Vehicles are today becoming increasingly complex systems. An important part of a vehicle is the powertrain which converts energy in a stored form (fuel) to kinetic energy in order to move the vehicle. Recent developments to cope with increasing demands on low fuel consumption and other environmental aspects have introduced several new concepts including e.g. hybrid (combined electrical and fuel) vehicles on the market.

The new vehicles create new challenges. Questions like the following can be relevant:

- "How do we control the charging of the batteries optimally?"
- "Does an automatic gearbox change gears in the way the driver desires?"
- "Does our Human Machine Interface which should help the driver to drive more economically achieve its goal?"

One way to get answers to such questions rather quickly and in a cost efficient way is to use simulation and simulators. Simulation scenarios can be built in many different ways and one approach is to include hardware in the simulation to increase its accuracy [1].

With these benefits in mind cooperation was started between the Swedish National Road and Transport Research Institute (VTI), and Linköping University, including the Vehicular systems group at ISY and the PELAB group at IDA.

The VTI Simulator III (Figure 1) hardware at VTI has been connected via a fiber optic link to the Vehicle propulsion laboratory at Linköping University. A more detailed description of the hardware can be found in [2].

Using the VTI Simulator III it is possible to do a driving experiment, e.g. adjust the road environment or change vehicle models. Since the driving experiment is a controlled experiment, repeatable driving scenarios can be achieved. It is also possible to test safety critical situations in a controlled and safe way.

In the Vehicle propulsion laboratory it is possible to equip different cars with a chassis dynamometers setup where the dynamometers can be used for both accelerating and decelerating the vehicle. The chassis dynamometers are mobile and one person can fit them on a car by moving them around, thus enabling a fast switch between different cars. Connecting these facilities constitute a good platform for testing new powertrain solutions and also improve the fidelity of the VTI Simulator III by using a powertrain in the loop.

Even though new cars could be equipped quickly in the Vehicle propulsion laboratory there is also a need for a model of the setup with which different ideas can be tested in the simulators at VTI before using a vehicle in the chassis dynamometers setup.

Modelica [3], [4], is a modeling language which features properties such as acausal and object-oriented modeling. It has been demonstrated that it is possible to create models applicable for real-time simulation using Modelica [5], [6]. The goal of this work is to build upon an existing hardware model by splitting it into sub-systems and porting parts of it to the Modelica language. The long term aim in this work is to model the complete setup with the car and the system in the Vehicle propulsion laboratory providing a flexible framework for testing distributed simulation scenarios and to run the models in real-time together with VTI Simulator III.

This paper is organized as follows: in Section 2 we give an introduction to the different hardware facilities and the connection between them. In Section 3 we present the Modelica models for the different hardware parts and in

5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, 19 April, 2013, University of Nottingham, UK.

EOOLT 2013 website:
http://www.eoolt.org/2013/

Section 4 we show performance tests on the models. The work conducted so far is then summarized in Section 5 which also includes some future work.

2. Hardware Facilities

The facilities used in this study are the VTI Simulator III and the chassis dynamometer lab at Linköping University. To control the vehicle in the chassis dynamometer lab a pedal robot has been constructed. These physical systems will be described in the following sections.

2.1 VTI Simulator III

The Swedish National Road and Transport Research Institute (VTI) is an independent Swedish research institute. The main research areas at VTI are infrastructure, traffic, transportation systems. To conduct research within these areas one important tool is vehicle simulators. The history of moving base simulators at VTI goes back to the late 1970's, and today VTI has three advanced moving base simulators.

The moving base simulators are mostly involved in behavioral studies where research questions such as "How does a driver react in this critical situation?" and "Can my invention detect if a driver is sleepy?" can be investigated.

One of these advanced moving base simulators is the VTI Simulator III which is located in Linköping. This simulator is the one used in this work. A picture of the VTI Simulator III can be seen in Figure 1.

Figure 1. The VTI moving base Simulator III in Linköping.

The motion system in VTI Simulator III has four degrees of freedom where the large outer linear motion can be used for lateral or longitudinal motion. On this motion system the driver is positioned inside a dome in a car cabin which is a production car that has been cut in half to fit the dome. The car cabin is mounted on a vibration table able to produce higher frequency noise reproducing road deformations, e.g. potholes and cracks [7].

The driver view is presented on a 120 degrees arched screen where six projectors are used to produce the front view. Small screens are used as rear view mirrors. The graphics software used is developed at VTI and includes the environments. Sound for the driver is presented using the vehicle speakers complemented with a few extra speakers resulting in a surround sound setup.

The software for controlling the simulation is developed at VTI including scenario logics, interfaces to hardware, graphics and sound, vehicle models and traffic models. The vehicle model used in most of the studies when car driving is of interest is a Fortran model. This model has been developed over several decades and its functionality has been tested in several studies over the years. Thus, the validity of the model has been confirmed repeatedly. The problem with the model is that it takes more and more time to integrate new functions and thus there is a desire to migrate the model to another language.

2.2 Vehicle Propulsion Laboratory

At the Vehicular Systems group at Linköping University, a new chassis dynamometer lab was built in 2011, see [8]. The dynamometers in this lab are mobile and can be adjusted to fit different vehicles sizes and different propulsion systems, e.g. front wheel drive or all-wheel drive. These dynamometers can provide both positive and negative torque while measuring the torque output from the equipped vehicle within 0.1 percent accuracy.

When running the system, the car is attached to the dynamometers by removing the wheels and connecting the wheel hubs to the dynamometers, see Figure 2. When all driving wheel hubs have been connected the exhaust is connected to the ventilation system. At this stage it is now possible to start the system and place a driver in the car. He can now start to drive by turning the car key and pressing the accelerator pedal.

Figure 2. Vehicle mounted at Vehicular Systems chassis dynamometer laboratory at LiU.

2.3 Chassis Dynamometer Vehicle Model

The measured torque output from the vehicle results in vehicle speed and acceleration. Therefore, the dynamometers have to simulate resistance forces equal to those experienced when driving on a road.

These resistance forces come from rolling resistance, F_{roll}, air resistance, F_{air}, and incline resistance (gravity when going uphill or downhill), F_{climb}. Thus, the incline forces are not only resistance uphill but also acceleration downhill, since the chassis dynamometers can provide

both a negative and a positive torque to the wheels. The acceleration, a, of the vehicle is calculated as

where m is the mass of the vehicle and F_{tot} are the total forces acting on the vehicle. F_{prop} is calculated from the measured torques at the wheels using

where T_i are the measured torques at the wheel hubs and r_w is the radius of the wheel.

The resistance forces are given as:

Here c_r is the rolling resistance coefficient, g is the gravitational constant, c_d is aerodynamic resistance constant, A_f is the vehicle front area, A_{air} is the density of the air, v is the speed of the vehicle, v_0 is the relative wind speed and p is the incline of the road. Values for parameters used in this study are shown in Table 1.

Table 1. Used values for the vehicle model parameters.

Parameter	*Value*
M	1401 [kg]
c_d	0.320 [-]
A_f	2.0 [m^2]
c_r	0.01 [-]
r_w	0.3 [m]

Signals measured from the chassis dynamometer are then sent to the VTI Simulator III using the UDP protocol. The list of signals is presented in Table 2.

Table 2. Data sent from the chassis dynamometers.

Signal	*Unit*	*Description*
n_i	[rpm]	Vehicle wheel speed
T_i	[Nm]	Vehicle wheel torque
v_l	[km/h]	Longitudinal velocity
v_v	[km/h]	Vertical vehicle speed
r_{road}	[m]	Road curvature radius
H	[°]	Heading, relative origin
h	[m]	Elevation of road
p	[°]	Incline
d_{TP}	[m]	Distance since start
t_{TP}	[s]	Time since start
T_i	[°C]	Dynamometer temperature
$S_{d,i}$	[-]	Dynamometer status
S	[-]	System status

2.4 Pedal Robot

The driver positioned in the VTI Simulator III can control the vehicle by pressing the accelerator or the brake pedal. The signals emitted by the simulator also have to influence the vehicle in the chassis dynamometers lab and thus a pedal robot was constructed. The pedal robot mimics the driver input in the VTI Simulator III so that the vehicle will receive the driver input. The control parameters are the accelerator pedal position and the brake pressure. The pedal robot is depicted in Figure 3.

One design choice during the construction of the pedal robot was to only include accelerator and brake pedals. Thus, only vehicles with automatic gear are considered. This design choice limits testing of vehicles but the effort needed to add manual gear change to the pedal robot was considered to be too time-consuming. This design choice limits the input signals needed by the pedal robot to accelerator pedal position and brake pedal pressure.

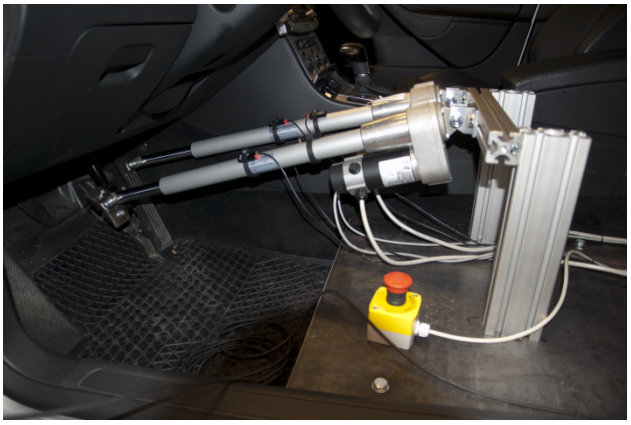

Figure 3. Pedal robot installed in the vehicle in the chassis dynamometer lab.

2.5 Network Performance

To connect the two hardware facilities together, the VTI Simulator III at VTI and the chassis dynamometers at Linköping University, it was decided to use optical fiber. Since the distance between the facilities is approximately 500 m, an optical fiber link was a viable option. The resulting network can thus be considered as a local network.

To test the network connection a round trip time test was performed. Packets which resemble the packages that are sent from the VTI Simulator III software were sent at 200 Hz which is the speed the VTI Simulator III kernel loop runs at. The result from sending one million packets is shown in Table 3.

Table 3. Statistics from the connection between facilities.

Number of packages	1 000 000
Minimum delay	0.20 ms
Maximum delay	2.17 ms
Median delay	0.22 ms
Dropped packets	none
Spikes above 0.5 ms	18

From Table 3 it can be seen that when sending one million packets no packets were lost. It can also be seen that there were very few delays longer than 0.5 ms. Based on the measured performance of the network it was decided to use the efficient UDP packet switching protocol for communication between the involved hardware setups and models during the tests.

3. Modelica Models

Modelica is a language for modeling and simulating complex physical and technical systems. The initiative to start the design of the Modelica modeling language was taken in 1996 in an international effort [15]. The main features of the language are:

- Object oriented approach
- Acausal equation-based modeling
- Hybrid (continuous-time and discrete-time) modeling

Since Modelica allows acausal modeling the model code is close to the physical equations describing the system behavior. This supports a more intuitive modeling process and provides a higher level of abstraction. Various commercial and open-source tools support the design, compilation and simulation of Modelica models. In the course of this work we have tested our models with both OpenModelica [10] and Dymola [11].

When porting the model to Modelica, an important question is what parts of the model to export to which separate subsystems. As for any large model there are alternative choices regarding how to split it into submodels. In this work the solution to this issue was more or less dictated by the hardware we want to model.

Thus the model uses the same UDP network interface as the chassis dynamometers lab, and network packets to and from the VTI Simulator III will have the same format when using a model. The resulting model partitioning consists of a Modelica car model used in the VTI Simulator III and of another Modelica model of the chassis dynamometers setup.

3.1 Powertrain Model

The powertrain model is split into two parts, the engine and the gearbox. To model the engine in the Fortran car model in the VTI Simulator III an engine map is used. This engine map consists of a 12 by 12 matrix where the engine rotational speed and throttle are taken as input and the output is engine torque.

Using such a matrix has been sufficient for many performed studies while still being simple. Thus, such an engine model is implemented in Modelica using a `Modelica.Blocks.Tables.CombiTable2D` model from the Modelica standard library, MSL, with smoothness set to `Modelica.Blocks.Types.Smoothness.LinearSegments`. Instead of using throttle as input to the engine map it was changed to pedal position resulting in

Here is the output torque from the engine, is the accelerator pedal position scaled between zero to one and is the engine rotational speed.

The engine rotational speed is also limited in the model to prevent the engine to infinitely increase rotational speed.

Looking at the gearbox inside the Fortran car model it is constructed using for-loops and `break` statements. The possibility to translate these parts, loops with `break` statements, which are quite nonphysical, to a Modelica model was investigated, but it was decided to create a new basic model instead. The created model uses the following equations between the gearbox and the wheel hubs:

Here $\dot{E}_{clutch}$ is the rotational speed at the clutch, $\dot{E}_{fl}$ and $\dot{E}_{fr}$ are the rotational speeds at the wheel hubs, i_{gear} is the gear ratio from the gearbox and final drive, T_{clutch} is the torque at the clutch and T_{fl} and T_{fr} are the torques at the wheels.

The clutch model's rotational speed depends on the clutch position. The following equations are used for the clutch:

Here is the inertia in the engine, is a modified clutch adjusted dead zones in the clutch and is a first order response time.

This model for the gearbox models a manual gear and the logic for automatic gear changes has to be applied. This would mean that instead of having clutch pedal and gear stick handled by a driver, physical or modeled, the clutch and gear signals are handled by logics with accelerator pedal as input from the driver.

3.2 Estimating Powertrain Model Parameters

In the chassis dynamometer lab there exist signals measuring the responses at the wheels, e.g. engine rotational speed and torques. To add other necessary signals for the parameterization of a powertrain model, an OBD II sensor was used. The OBD II sensor is capable of

logging 5 parameters at a speed of 2-4 Hz which might be too slow for dynamic testing, but for static tests it was deemed sufficient. As the two systems used for measuring data both save time stamps a time synchronization using the Network Time Protocol [12] was performed before the measurements.

Starting with estimating the gear ratios, the chassis dynamometer setup wheel rotational speed and time stamps were logged. From the OBD II sensor engine rotational speed and time stamps were logged. For every gear the driver started at a low speed which was maintained for approximately one minute. The driver then increased speed to another stationary speed. This procedure of increasing speed was repeated two to three times giving a measurement of the gear ratios with both low and high engine rotational speeds.

The measurements from the chassis dynamometers are made at a higher frequency. Thus, to achieve equal positions in time, linear interpolation is used to get logged wheel rotational speed at the same time instances as the engine rotational speed. A least square approximation is then used to estimate the gear ratios and the estimated ratios are shown in Table 4.

Table 4. Measured gear ratios from car mounted in the chassis dynamometer lab.

Gear	*Ratio*
1	16.70
2	10.08
3	6.79
4	4.97
5	3.79
6	3.06

In Figure 4 the relation between the engine rotational speed and the wheel rotational speed using measured gear ratio for gear one is shown.

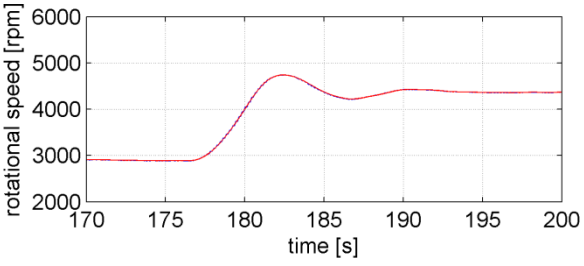

Figure 4. Comparison of engine rotational speed, blue curve, and vehicle wheel speeds multiplied with the gear ratio, red curve, at gear one.

We continue by measuring a static engine map. In this setup we use the OBD II sensor to measure time, engine rotational speed, and accelerator pedal position. Before starting any measurements the engine was run at high load to reduce variations in temperature during the measurements. Measurements for each operating point were done during approximately 30 seconds when the engine torque output had stabilized. The resulting engine map is shown in Figure 5.

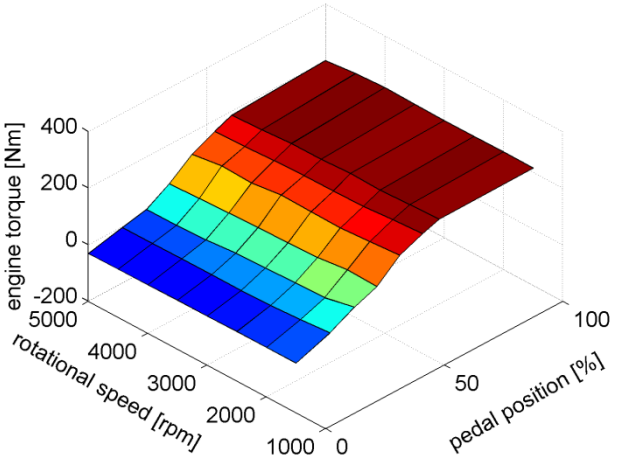

Figure 5. Measured engine map from a car mounted in the chassis dynamometer lab.

In the used cabin in the VTI Simulator III the red zone of the engine rotational speed starts at 6400 rpm with a stop at 7000 rpm. To take this into account the engine map is modified by extrapolating the engine map from 5000 rpm to 6400 rpm and after that linearly reduce output torque to 7000 rpm. At 7000 rpm the output engine torque is independent of pedal position and the engine torque where the driver has released the accelerator pedal is used for every pedal position. In this case -35.5 Nm.

3.3 Pedal Robot Model

We considered the performance of the pedal robot to be accurate and fast enough to neglect the effects from it. Thus, the pedal robot has not been modeled and instead the pedal signals are sent directly to the chassis dynamometers model.

3.4 Model of the Chassis Dynamometer Lab

The complete model of the chassis dynamometer lab consists of three parts. One part handles the longitudinal vehicle model used to calculate the vehicle speed as described earlier in the hardware section. The second part is the vehicle powertrain which from driver pedal input models wheel torque and rotational speed responses as shown. The third part is the brake dynamics. To simulate this setup the complete model had to be extended with a driver model. A simulation of the complete setup is shown in Figure 6 where a calm acceleration is performed.

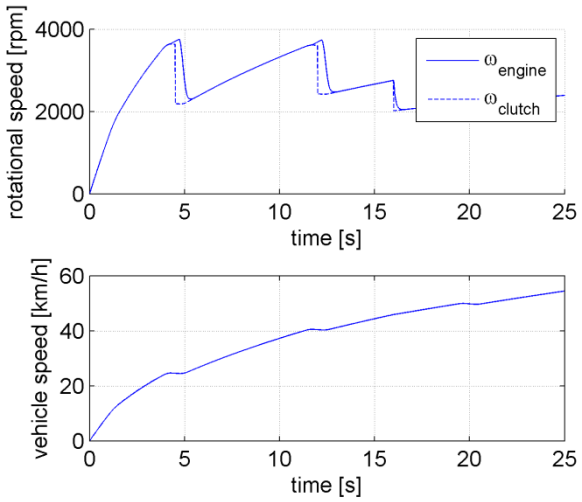

Figure 6. A simulation of the complete chassis dynamometers lab model during a calm acceleration.

The minimum requirement of the complete model is that it has to run at least at 100 Hz because the chassis dynamometers send torque and rotational speed data at this frequency. These signals are important and thus we want the model to send these signals in the same way the hardware would do.

The final chassis dynamometer model has 63 equations where 40 of these are trivial equations. The final model has 8 continuous states.

3.5 Simulator Car Model

In the VTI Simulator III simulator environment different vehicle models can be used. One of these models is a ten degrees of freedom Modelica car model, see [13]. The model has been compiled from Dymola to Simulink in Matlab 7.5 where it was further compiled to be used in an xPC-Target 3.3 environment which is a real-time environment. For further information about xPC-Target see [14].

In the Modelica car model the parts regarding the powertrain have been modified to have the same connections as to the chassis dynamometers model. Since the connections are the same it is possible to use the estimated powertrain model without adjustments in both the Modelica car model and the chassis dynamometers model. The connections also make it possible to run the simulator connected to the chassis dynamometers model in the same way as if it would be connected to the hardware in the chassis dynamometers lab.

3.6 Complete Simulator Setup

The complete simulator setup consists of several components of hardware and Modelica models. An overview of these components is shown in Figure 7.

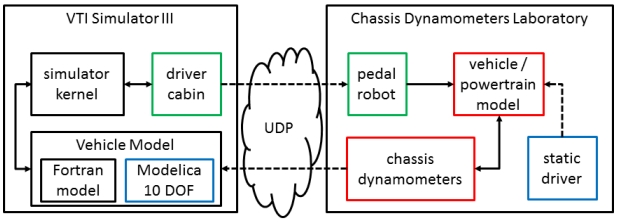

Figure 7. An overview of the components in the simulator environment. Green boxes picture hardware components and blue boxes picture Modelica model components. Red boxes picture components where either hardware or a Modelica model can be used.

Here it can be seen that it is possible to combine these components in different ways. Examples of combinations are:

- VTI Simulator III connected to the pedal robot and the chassis dynamometers.
- VTI Simulator III with the Modelica powertrain model included in the 10 DOF Modelica car model.

It should be noted that the chassis dynamometers and connected vehicle either both are in hardware or software as it is not possible to connect a Modelica powertrain model to the chassis dynamometers.

One component shown in Figure 7 which has not been discussed much is the static driver. By static driver we here mean that the driver has a predefined way of driving, e.g. change gear from gear 1 to 2 at time 5 s. Another typical term used for this kind of component is a drive cycle. Thus, this Modelica model relies heavily on the `time` variable and on `if` statements for controlling driver output such as the accelerator pedal, the clutch pedal, the brake pedal and the gear.

4. Results

For the models we have created there are many aspects to investigate. Our main concern during the initial stages has been the feasibility of real-time implementation. We primarily consider two questions:

- How accurately different solvers simulate the model?
- Will the model manage desired time steps?

In the following sections we discuss how our models perform with respect to these questions.

4.1 Performance of the Chassis Dynamometers Model

We start here by looking at the first question: "How accurately different solvers simulate the model?". As a baseline for comparison the DASSL solver has been used which is compared to the Euler forward solver. Since some signals from the chassis dynamometers are sent at 100 Hz this sets the minimal required speed for the model.

Figure 8 shows the difference in acceleration when using Euler forward or DASSL during the acceleration maneuver shown in Figure 6. Settings used were a tolerance of 1e-6 and 2500 intervals. Simulation environment used was OpenModelica.

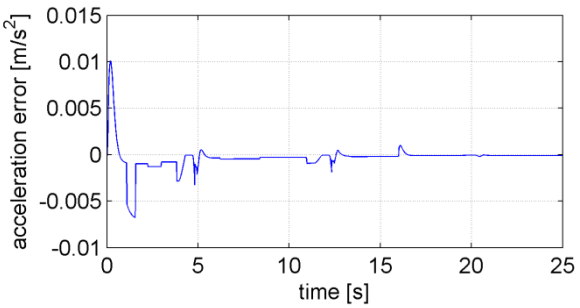

Figure 8. Difference in acceleration when simulations are using Euler forward or DASSL as solvers during an acceleration maneuver.

Here it can be seen that the difference between using DASSL and Euler forward is comparably small. Since the difference between DASSL and Euler forward is so small and we want to keep the setup as simple as possible Euler forward has been chosen for real-time simulation.

The next concern is if it will be possible to obtain 100 Hz performance during a real-time simulation. To give an estimate of this the profiler in OpenModelica, [15], has been used. The results from a run with Euler forward with a time step of 0.01 s are shown in Table 5.

Table 5. Time measurements from model simulation.

Task	*Time*	*Fraction*
Pre-Initialization	0.000464	1.12%
Initialization	0.000118	0.29%
Event-handling	0.000671	1.62%
Creating output file	0.030664	74.24%
Linearization	0.000000	0.00%
Time steps	0.004043	9.79%
Overhead	0.003650	8.84%
Unknown	0.00216	4.11%
Total simulation time	0.041306	100.00%

In Table 5 we can see that event handling and the time steps take approximately 0.0047 s which means that it should be possible to run the model in real-time using Euler forward with a time step of 0.01 s.

5. Conclusion

The established connection between the VTI Simulator III and the chassis dynamometer lab has been developed and this setup shows promising results. For applications it will be interesting to perform more detailed investigations regarding modeling and performance issues.

In this paper we have described and presented our first results on the technical side. A Modelica car model of appropriate complexity has been adjusted for use in both OpenModelica and Dymola. A chassis dynamometer model has been developed which runs in both OpenModelica and Dymola.

The performance profiler in OpenModelica indicates that it is possible to run the model in real-time. This shows that it is promising to continue this work to achieve a real-time simulation with the VTI Simulator III together with a Modelica model of the chassis dynamometer setup.

5.1 Future Work

One main future work is to run the parameterized models in real-time together with the hardware. This should be investigated using the different possible combinations of hardware and software. Additionally, some parts of the model should be improved. For instance, the models for the pedal robot and the brake dynamics have been over-simplified and can be enhanced. A model from accelerator pedal to throttle input could also be further investigated.

Another option to investigate is the use of FMI for real-time simulation [16]. FMI is a standard for exporting a compiled model to C-code packaged into a so-called FMU (Functional Mockup Unit) which can be imported for simulation within another tool.

The chassis dynamometer model has already been compiled to a FMU using OpenModelica. A possible future work is to integrate this FMU with a small cross platform application. The intention is to run the model on a stripped Linux computer through this small cross platform application.

Acknowledgements

The authors would like to thank Tobias Lindell and Per Öberg for the help with measurements in the chassis dynamometers lab. The authors would also like to thank Lena Buffoni for valuable comments and help with the manuscript.

Partial support for this work has been received from SSF in the project HiPo, and from Vinnova in the project RTSIM and in the ITEA2 project MODRIO.

References

[1] X. Hu and E. Azarnasab. From virtual to real – A progressive simulation-based design framework. In *Discrete-Event Modeling and Simulation*, *CRC Press,* 2010.

[2] A. Andersson, P. Nyberg, H. Sehammar, and P. Öberg. Vehicle Powertrain Test Bench Co-Simulation with a Moving Base Simulator Using a Pedal Robot. In SAE World Congress, number 2013-01-0410, Detroit, USA, 2013.

[3] Modelica Association. Modelica Language Specification 3.3, www.modelica.org, May 2012.

[4] Peter Fritzson. *Principles of Object Oriented Modeling and Simulation with Modelica 2.1*, 940 pages, ISBN 0-471-471631, Wiley-IEEE Press. January 2004.

[5] Elmqvist, H., Mattsson, S., & Olsson, H. Real-time simulation of detailed vehicle and powertrain dynamics. *SAE SP, 2004*.

[6] Otter, M., Schlegel, C., & Elmqvist, H. Modeling and realtime simulation of an automatic gearbox using Modelica. *Proceedings of ESS,* 1997.

[7] A. Bolling, J. Jansson, M. Hjort, M. Lidström, et al. An approach for realistic simulation of real road condition in a moving base driving simulator. *ASME/Journal of Computing and Information Science in Engineering*, 11(4), 2011.

[8] P. Öberg, P. Nyberg, and L. Nielsen. A new chassis dynamometer laboratory for vehicle research. In SAE World Congress, number 2013-01-0402, Detroit, USA, 2013.
[9] S. Mattson, H. Elmqvist and M. Otter. Physical system modeling with Modelica. *Control Engineering Practice*, 6(4), 501–510, 1998.
[10] Open Source Modelica Consortium. *OpenModelica Users Guide version 1.9.0beta*, February 2013. www.openmodelica.org.
[11] Dassault Systems. Dymola Users Guide, version 7.1, February 2013. www.dymola.com.
[12] D. L. Mills. Internet Time Synchronization: The Network Time Protocol, IEEE Transactions on Communications, vol. 39, no. 10, 1991.
[13] J. G. Fernández. *A Vehicle Dynamics Model for Driving Simulators.* Master's thesis at Chalmers University of Technology, Göteborg, Sweden, 2012.
[14] P. J. Mosterman, S. Prabhu, A. Dowd, J. Glass, T. Erkkinen, J. Kluza, R. Shenoy. Embedded Real-Time Control via MATLAB, Simulink, and xPC Target.
[15] M. Huhn, M. Sjölund, W. Chen, C. Schulze & P. Fritzson. Tool Support for Modelica Real-time Models In *Proceedings of the 8th Modelica Conference*, Dresden, Germany, March 20-22, 2011.
[16] T. Blochwitz, M. Otter, & M. Arnold. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *Proceedings of the 8th Modelica Conference,* (pp. 105–114), 2011.

Appendix

Modelica models for most of the simulated system.

```
model ChassisDynamometerSystem
  StaticDriver driver;
  ChassisDynamometerVehicleModel
   chassis_dynamometer_vehicle_model;
  volvos40.volvos40powertrain powertrain;
equation
  powertrain.throttle = driver.throttle;
  powertrain.clutch = driver.clutch;
  powertrain.gear = driver.gear;
  powertrain.long_vel =
   chassis_dynamometer_vehicle_model.vl;
  connect(powertrain.fl,
    chassis_dynamometer_vehicle_model.fl);
  connect(powertrain.fr,
    chassis_dynamometer_vehicle_model.fr);
end ChassisDynamometerSystem;
```

```
model StaticDriver "driver with
    pre-defined output"
  output Real throttle
   "throttle position scaled [0.0-1.0]";
  output Real clutch
   "clutch positio scaled [0.0-1.0]";
  output Real brake "brake pressure";
  output Integer gear "chosen gear";
  output Real stw_ang
   "steering wheel angle";
protected
  constant Real pi=Modelica.Constants.pi;
  constant Real stwamp = (110 * pi) / 180
    "amplitude of the swd manouvre";
equation
  der(throttle) = if time < 17 then 10 *
    (0.5 - throttle) else 10 * (0.3 -
    throttle);
  clutch = if abs(time - 5) < 1 then
    1 - max(0, min(1, abs(time - 5)))
    elseif abs(time - 10) < 1 then
      1 - max(0, min(1, abs(time - 10)))
    elseif abs(time - 20) < 1 then
      1 - max(0, min(1, abs(time - 20)))
    else 0;
  brake = if time < 20 then 0
    elseif time < 30 then 15000
    elseif time < 40 then 0
    elseif time < 55 then 15000
    else 0;
  gear = if time < 5 then 1
    elseif time < 10 then 2
    elseif time < 20 then 3
    elseif time < 35 then 4
    else 3;
  stw_ang = if time < 10 then 0
    elseif time < 10 + (1 / 0.7 * 3) / 4
      then stwamp * sin(2 * pi * 0.7 *
       (time - 10))
    elseif time < 10 + (1 / 0.7 * 3) / 4 +
      0.5 then -stwamp
    elseif time < 10 + (1 / 0.7 * 4) / 4 +
      0.5 then stwamp * sin(2 * pi * 0.7 *
       (time - 10 - 0.5))
    else 0;

end StaticDriver;
```

```
model ChassisDynamometerVehicleModel
  "Vehicle model used in the Chassis
   Dynamometer setup at LiU"
  package Interfaces =
Modelica.Mechanics.Rotational.Interfaces;
  Interfaces.Flange_a fl "mechanical
    connection to front left wheel";
  Interfaces.Flange_a fr "mechanical
    connection to front right wheel";
  Interfaces.Flange_b rl "mechanical
    connection to rear left wheel";
  Interfaces.Flange_b rr "mechanical
    connection to rear right wheel";
  Modelica.SIunits.Acceleration a(start =
    0) "vehicle acceleration";
  Modelica.SIunits.Velocity v(start = 0)
   "vehicle speed";
  output Real[4] n "wheel rotational
    speeds";
  output Real[4] M "wheel torque";
  output Real vl "vehicle longitudinal
    speed";
  output Real vv "vehicle lateral speed";
  output Real rroad "road curvature
    radius";
  output Real H "vehicle heading";
  output Real h "elevation of road";
  output Real p "incline";
```

```
  output Real d_TP "distance since start";
  output Modelica.SIunits.Time t_TP "time
    since start";
  output Modelica.SIunits.Temperature[4] T
    "dynamometer temperature";
protected
  package SI = Modelica.SIunits;
  constant SI.Mass m = 1401 "vehicle
    mass";
  constant SI.CoefficientOfFriction c_d =
    0.32
    "aerodynamic resistance coefficient";
  constant SI.Area A_f = 2.0 "vehicle
    front area";
  constant SI.CoefficientOfFriction c_r =
     0.001 "rolling friction coefficient";
  constant SI.Length r_w = 0.3 "wheel
    radius";
  constant SI.Acceleration g =
    Modelica.Constants.g_n "gravitational
      constant";
  constant SI.Density rho_air = 1.202 "air
    density at an altitude of 200m";
  Real Ftot "total amount of forces acting
    on the vehicle";
  Real Fprop "propulsion forces";
  Real Froll "rolling resistance forces";
  Real Fair "air resistance forces";
  Real Fclimb "vehicle incline forces";
equation
  a = der(v);
  Ftot = m * a;
  Ftot = Fprop - Froll - Fair - Fclimb;
  Fprop = -(fl.tau + fr.tau + rl.tau +
    rr.tau) / r_w;
  Froll = c_r * m * g;
  Fair = (c_d * A_f * rho_air * v * v)/2;
  Fclimb = 0.0;
  der(fl.phi) = v / r_w;
  der(fr.phi) = v / r_w;
  der(rl.phi) = v / r_w;
  der(rr.phi) = v / r_w;
  //Output
  n[1] = v / r_w;
  n[2] = v / r_w;
  n[3] = v / r_w;
  n[4] = v / r_w;
  M[1] = fl.tau;
  M[2] = fr.tau;
  M[3] = rl.tau;
  M[4] = rr.tau;
  vl = v;
  vv = 0.0 "dummy value";
  rroad = 0.0 "dummy value";
  H = 0.0 "dummy value";
  h = 0.0 "dummy value";
  p = 0.0 "dummy value";
  der(d_TP) = v;
  t_TP = time;
  T[1] = 300.0 "dummy value";
  T[2] = 300.0 "dummy value";
  T[3] = 300.0 "dummy value";
  T[4] = 300.0 "dummy value";
end ChassisDynamometerVehicleModel;
```