Using Modelica models for Driver-in-the-loop simulators

Mike Dempsey Garron Fish Alessandro Picarelli
Claytex Services Limited
Edmund House, Rugby Road, Leamington Spa, UK
mike.dempsey@claytex.com garron.fish@claytex.com alessandro.picarelli@claytex.com

Abstract

Driver-in-the-loop simulators are increasingly used in Motorsport and Automotive companies to enable engineers and drivers to experience a new vehicle design in a realistic environment before it is built. The use of simulators enables drivers to test a new vehicle and/or control system without having to build a prototype and to carry out those tests in complete safety and in repeatable conditions.

Using Modelica as the development language for the vehicle model within these systems enables rapid model development and the fast evaluation of vehicle concepts. This enables more vehicle concepts to be tested before committing to a prototype build. The use of physical models also ensures that geometry changes and other physical modifications to the concept can be evaluated on the simulator at an early stage.

Keywords: driving simulator, vehicle dynamics, real-time simulation

1 Introduction

Driver-in-the-loop simulators come in a wide range of different formats ranging from the basic workstation simulator through to the high end simulators with multiple projectors and a motion platform with up to 6 degrees of freedom. The aim is to provide a driver with a realistic environment and accurate vehicle response to enable them and the engineers they are working with to evaluate the behaviour and performance of the vehicle design.

Driving simulators are used for a variety of reasons in automotive and motorsport companies. These include driver training, vehicle attribute perception, new designs and the evaluation of new technologies that may affect the concentration/driving pleasure of the driver. Recently there has been an increasing interest in using simulators earlier in the design process to assess the behaviour of new vehicle designs and technologies and to understand how well these systems work together in a vehicle. To meet this requirement it is necessary to improve the model development process so that new vehicle concepts can be quickly modelled using physical models. Dymola and Modelica are ideal for this application and have been used in this way by a number of Automotive and Motorsport users.

2 The Vehicle Model

2.1 Overview

The vehicle model used in a driving simulator has to represent the complete vehicle, accurately predict its behaviour and run in real-time. There is often also a desire to be able to use the same vehicle model in other parts of the engineering process outside of the simulator. Using Modelica to define the vehicle model and Dymola to compile this enables us to meet these requirements as this paper explains.

The vehicle model itself needs to be able to accurately predict the transient performance of the real car. To achieve this it needs to include models of the tyres, suspension, powertrain including both the physical and control aspects of these systems.

A number of commercial libraries have been used in the development of vehicle models for use in driving simulators. These are the Engines and VDLMotorsports Libraries and the following sections describe the use of these libraries to develop a model of a 1.8 litre, 4 cylinder turbo-charged gasoline direct injection engine fitted with a manual gearbox and double wishbone suspension.

This paper will detail 3 of the key aspects to the vehicle model that have been developed to meet these needs: the chassis model; the engine model; and the tyre-road contact model.

2.2 Chassis model

The VDLMotorsports Library [14] is an extension to the Vehicle Dynamics Library [15] developed by Modelon. This library was originally developed for modelling open-wheel race cars where the double

wishbone suspension setup is commonly used together with either pushrod or pullrod actuation of the inboard springs and dampers. The main objective of the library was to provide suspension models for this application that could run in real-time without additional detailed work from the end-users. As part of the intended use of these models was for driving simulators and other trackside tools it was necessary to make sure that the geometry of the compiled models, including all of the physical adjustments normally possible with these suspensions, could be applied through parameter changes without requiring the model to be recompiled. An annotated view of a pushrod suspension showing all the physical adjustments required within the suspension is shown in Figure 1. The shims identified can be defined as a thickness and make a change to the suspension geometry.

To achieve this we developed new implementations of the double wishbone suspension models that shared no common components, but a common architecture with those in the Vehicle Dynamics Library. The major problem to be overcome in defining these suspension models was the non-linear systems of equations that are normally formed when the suspension is created using the Modelica MultiBody library and individual joints (revolute, universal and spherical).

New combinations of aggregated joints [1] were developed to provide an analytic solution to the suspension degrees of freedom which has enabled this mechanism to be implemented without any of the usual non-linear systems of equations. These new joint combinations have been used to define the outboard suspension mechanism as shown in Figure 2 (the animation of this mechanism is shown in Figure 1). The pushrod is defined with an aggregated joint of the form Revolute-Prismatic-Spherical-Prismatic-Universal and is based on the JointUSR in the Modelica Standard Library. The upper wishbone and steering link are a more complex joint structure shown in Figure 3.

In both Figure 2 and Figure 3 the cyan bars on various components represent prismatic adjustments that can be applied to the mechanism to adjust the overall suspension geometry.

In the original baseline model that was created using the Modelica MultiBody library each instance of the suspension model contained 2 non-linear systems of equations of sizes 73 and 57, before symbolic manipulation. In the new suspension models both these non-linear systems of equations are eliminated.

The VDLMotorsports Library also supports the double wishbone suspension setup more commonly found in road cars. These models make use of the

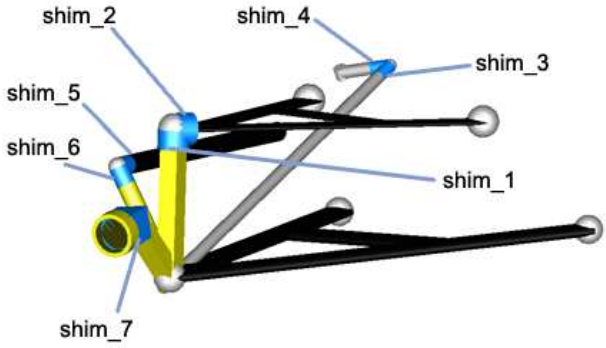

Figure 1: Annotated view of a pushrod suspension highlighting the physcial adjustments in the suspension

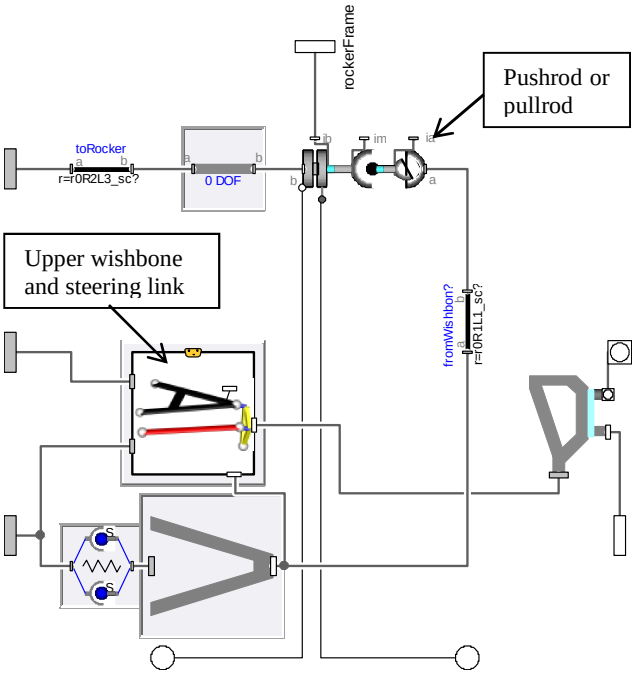

Figure 2: Double wishbone suspension with pushrod/pullrod

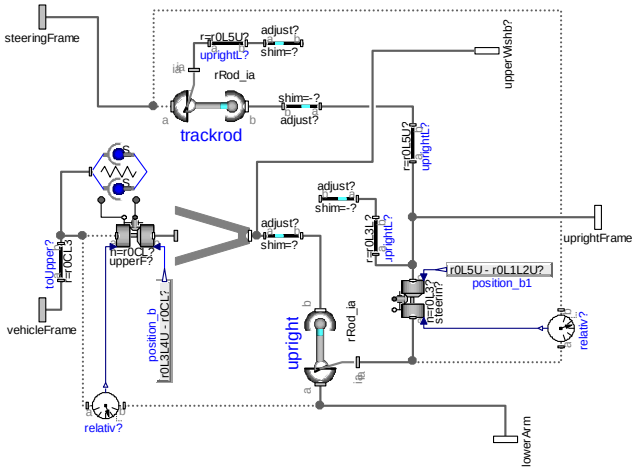

Figure 3: Detailed view of the aggregated joint mechanism defining the upper wishbone and steering link degrees of freedom

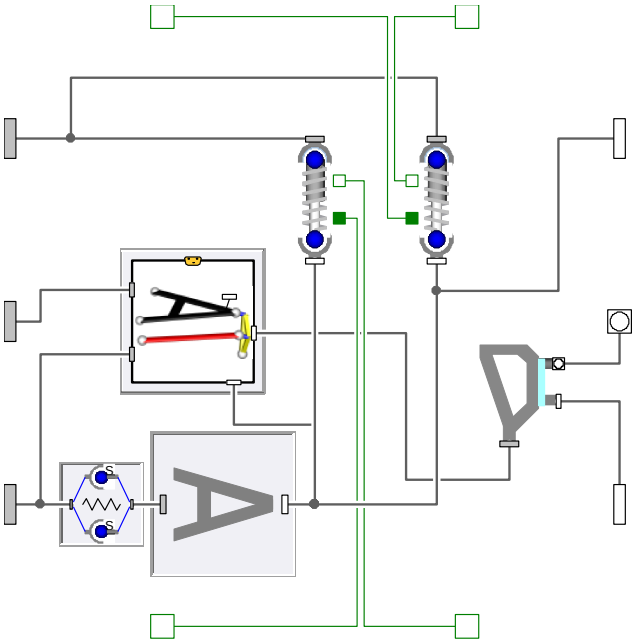

Figure 4: Double wishbone suspension with outboard springs and dampers

same efficient implementation of the suspension but now connect the spring and damper units to the lower wishbone (as shown in Figure 4) or other suspensions links as required.

For this example the vehicle model has been created using the double wishbone suspension with outboard springs and dampers connected to the lower wishbone. This suspension model fits within the Vehicle Dynamics Library suspension architecture and enables the standard steering and anti-roll bar models to be used when applicable. The suspension models are fully adjustable which would enable geometry changes as well as spring and damper rate changes to be assessed when using this model on the simulator.

2.3 Tyre and road contact

The tyre models typically used in driving simulators are based on the Pacejka tyre slip model and as such are essentially single point of contact handling models. The vertical dynamics models are usually implemented as some form of nonlinear spring to more accurately capture the vertical dynamics of the tyre and its input to the suspension. Whilst these single point of contact tyre models give a good prediction of vehicle handling on smooth surfaces they are not really adequate for use on rough surfaces such as those used in driving simulators.

In many cases the simulator road data is based on high-fidelity LiDAR scans of a real road or track surface that captures all the bumps and surface details. This level of detail in the track surface is not compatible with a single point of contact tyre model because it leads to a lot of noise feeding into the suspension and steering.

In order to make good use of the detailed road surface data a filtering method is introduced into the contact point calculation so that the rough surface is reduced to a single effective contact point. One such method looks at a number of potential contact points underneath the tyre and calculates a resultant contact point and surface normal at which all of the tyre forces are applied.

The screenshot in Figure 5 is taken from a single tyre test rig that is using 5 potential contact points (shown in red) and filtering these to determine one effective contact point (shown in yellow). The filtering is achieved by considering the amount of tyre compression at each potential contact point and then adjusting the position and surface normal of the resultant contact point accordingly. When used with triangular meshed road data some further filtering is then required to avoid sudden jumps in the surface normal as the edges of the different triangles are crossed.

The introduction of these filtering methods in the tyre model is done by replacing the standard Vehicle Dynamics Library contact block with a customized contact model. The development of these filtering methods has driven several enhancements in the Vehicle Dynamics Library ground models which have been implemented by Modelon. These new options now enable the standard implicit contact model, which generates nonlinear systems of equations, to be replaced with an explicit contact model which does not include nonlinear systems of equations.

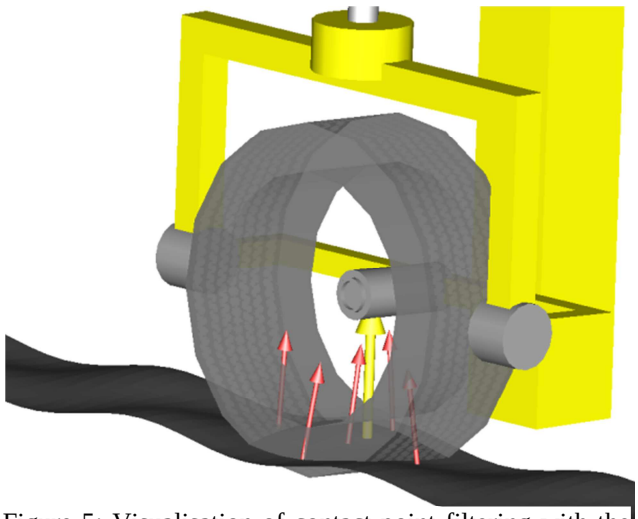

Figure 5: Visualisation of contact point filtering with the potential contact points in red and the resultant contact point in yellow

2.4 Engine model

The engine model used in the test vehicle was developed using the Engines Library [2]. The Engines Library comes in two versions with different capabilities: Mean Value Engine Models (MVEM) which means cycle averaged torque and emissions; and Crank Angle Resolved Engine Models (CAREM) which means crank angle resolution of torque, heat release and emissions. Both versions model the intake and exhaust manifold fluid dynamics and heat transfer with varying levels of detail. The heat transfer models can range from models with no thermal resistance to ones which take into account the flow regime within the particular component [5]. The fluid flow models can range from simple volume models to ones which include the fluid momentum dynamics to analyse fluid pressure pulsations propagating throughout the system and study their effects on the system performance.

For the purposes of CPU time reduction, a surrogate mode is available for both MVEM and CAREM [2],[3] models. The surrogate mode allows the engine model to be reduced in complexity whilst maintaining a high level of accuracy when compared to the non-surrogate mode. Figure 6 shows the comparison of a MVEM model running in both modes and shows that there is little deviation between the results.

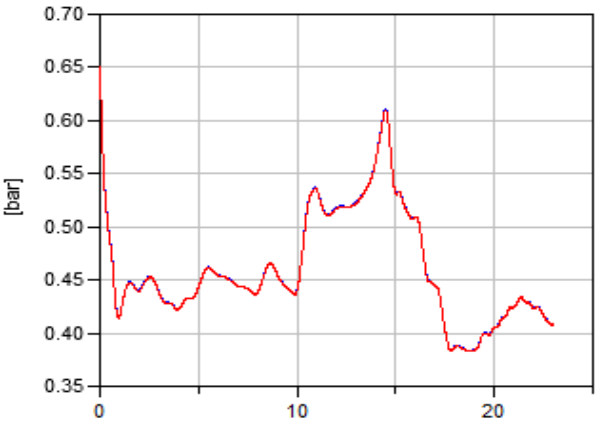

Figure 6: Plot of plenum pressure for a naturally aspirated I4 SI engine with surrogate mode enabled (blue) and disabled (red). In this case the two lines lie almost exactly on top of one another.

All the fluid components in the Engines library are based on the Modelica Fluid library [4] which ensures compatibility with the latter and all derived libraries. The medium model is based on the Modelica Media library but it has been necessary to provide some additional customisations to achieve the level of performance required in this library. The medium model tracks 7 species throughout the air path of the engine so that fuel mass and the emissions composition can be traced through the engine.

The engine used in this example is a mean value 4 cylinder 1.8l turbocharged spark-ignited gasoline engine with direct injection and producing a peak power of 160kw and peak torque of 225Nm. The transients of the turbocharger, fluids and heat transfer in the air-path and torque output are captured in the results as well as the multi-body behaviour of the system. This model is shown in Figure 7.

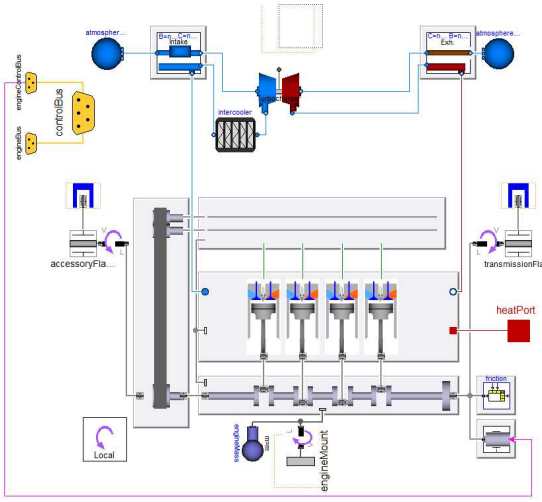

Figure 7: I4 Turbo Mean Value Engine Model diagram layer showing the engine sub-systems. Note that fluid and mechanical connector arrays are used between the components to simplify the diagram layer complexity

In the above model the intake manifold comprises an air flow mass sensor, a turbocharger compressor including ducting, an air to air intercooler, a throttle body, a plenum volume and cylinder head port volumes. The exhaust manifold comprises of the cylinder head ports and primary exhaust system coupled to the turbocharger turbine. Emissions after-treatment systems have been omitted from this engine model as they were not required for the purposes of this experiment; however, their associated pressure drops have been taken into account within the exhaust system. Both intake and exhaust volume models include heat transfer effects from the fluid to the pipe walls.

The Engines library contains two main types of turbocharger turbines and compressors models [6][7][8][10]. There are purely map based models where the mass flow rate and efficiency are functions of the speed and pressure ratio. These models rely on having very good map data available throughout the operating range of the turbine and compressor. The second type of model are based on ellipse curves (e.g. Stodola's law for turbines). In these models ellipse curves are fitted to the available map data to

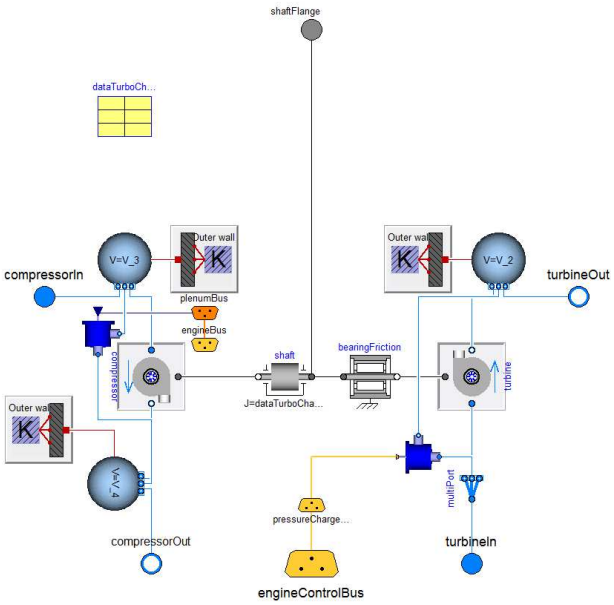

Figure 8: Turbocharger diagram layer showing the layout of the system

describe the mass flow rate through the component. The advantage of these ellipse models is that the characteristics are guaranteed to be smooth.

The engine model described in this paper utilises a turbocharger turbine model based on Stodola's law which calculates the mass-flow rate as a function of the pressure ratio and essentially follows the ellipse law:

$$m\_flow_{corrected} = K \cdot \sqrt{1 - \left(\frac{1}{pressureRatio}\right)^2}$$

where K is a constant that scales the turbine characteristics.

The turbocharger compressor uses a similar ellipse law that relates the mass flow rate and pressure ratio through the ellipse equation:

$$1 = \left(\frac{pressureRatio}{a}\right)^z + \left(\frac{m\_flow_{corrected}}{b}\right)^z$$

The coefficients a, b and z are defined in tables that are dependent on rotational speed.

The complete turbocharger model is shown in Figure 8. The compressor side (left hand side of the figure) includes intake and outlet ducting with associated replaceable heat transfer models and a pneumatically controlled recirculating pressure relief valve. The turbine side (right hand side) includes outlet ducting and an electronically controlled wastegate to limit the turbine performance. The shaft including the impellers is modelled as a 1D rotational system with bearing friction model.

The ellipse based compressor and turbine improve the model robustness and can rely on less finely resolved compressor and turbine maps than the purely table based variants to achieve similar functionality. Compressor surge is taken care of by supplying a surge line to the model. Special considerations for impeller torque are implemented for operation beyond the compressor surge line.

The basic turbocharger model shown in Figure 8 has been used for both low pressure and high pressure turbocharging applications

2.5 Results

The whole vehicle model including the powertrain was compiled in Dymola using a fixed step solver with inline integration. The inline integration method chosen was a mixed implicit/explicit Euler with step size of 1.25ms. The mixed method was chosen as it gave the best results in terms of expected system behaviour, model robustness and simulation performance. The step size was determined by considering the requirements of the model, the pc processor speed and the frequency that the rest of the simulator system needs to operate at.

In this experiment, the model is compiled with a driver and simulated in Dymola to verify the behaviour. The driver model is open loop for both longitudinal and lateral control and the test consists of an acceleration from rest, shifting through the gears to test the extreme situation for both chassis, wheels and powertrain:

Test sequence:

1. t=0, vehicle standing at rest with engine idling
2. t=2, launch starts, i.e. clutch engagement starts
3. The vehicle accelerates at full throttle through the gears up to a speed of 250km/h

The results of this acceleration test are shown in Figure 9. The main user input required to get such complex models to function in real time is good initialisation. This is assuming that all the work at the component level has already been done to eliminate numerical jacobians and minimise the number of non-linear systems of equations.

The critical areas in these vehicle models are the tyres, suspension and engine initial conditions. Within the VDLMotorsports Library a number of experiments and functions exist that enable the start conditions for the tyres and suspensions so be determined and automatically extracted for use in other simulations. For the engine model, the initial pressures and temperatures within the intake and exhaust system were determined by running the engine at a

constant speed and load operating point that is then used as the start point for the real-time model. The final temperatures and pressures from this steady state experiment are used to define the initial conditions in the real-time experiment.

The results in Figure 9 show the vehicle longitudinal dynamics and the dynamics of the engine components when running the model in real time. We have focused on the intake pressure and the turbocharger and wastegate [9] performance with the latter used to limit the intake manifold pressure. With such models running in real time, swapping turbocharger models or tweaking the turbocharger characteristics, enable the user to test different boosting configurations for driveability and performance within a vehicle simulator.

Such models as the one used in this paper provide invaluable and easily accessible detailed information for the engineers in the development phase of the vehicle when there is need to test a variety of options in a short space of time. The transient detail exhibited in the results for the torque generation are of sufficient detail to provide the effects necessary for the evaluation of different engine and powertrain solutions in vehicle simulators.

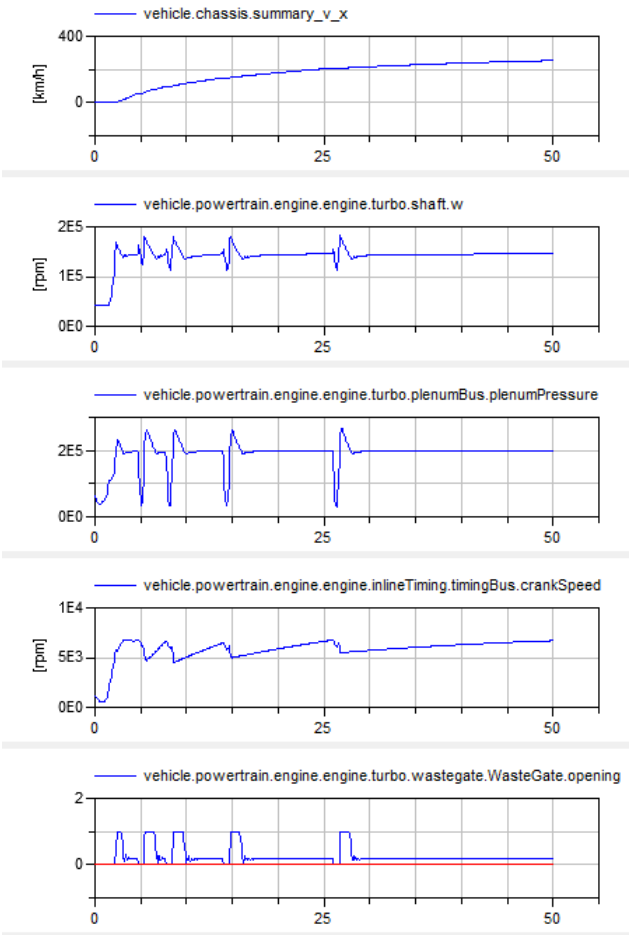

Figure 9: System simulation engine behaviour during the full throttle acceleration test. From top to bottom: Vehicle speed (km/h), turbocharger shaft speed (rpm), absolute plenum pressure (Pa), crank speed (rpm), wastegate (blue) and dump valve (red) opening

3 Integrating the physics model and simulator system

3.1 System Overview

A typical driving simulator will consist of several computers each with responsibility for a different aspect of the system. Figure 10 shows the basic architecture of a typical system.

The motion platform usually has a dedicated computer to decide how it should move and receives inputs from the physics model such as vehicle orientation and accelerations.

There is usually a PC (or real-time computer) that is dedicated to the physics model. The physics model receives inputs from both the motion platform via the motion controller and the vision system. The data coming from the motion platform are the driver inputs such as steering, pedal positions, gear, etc. The data coming from the vision system usually includes the environmental conditions and road surface information.

It is also possible to incorporate the real control systems with the physics model. The exact approach varies depending on the objective of the simulator and the hardware available. In some instances a model of the control system is compiled to run on a

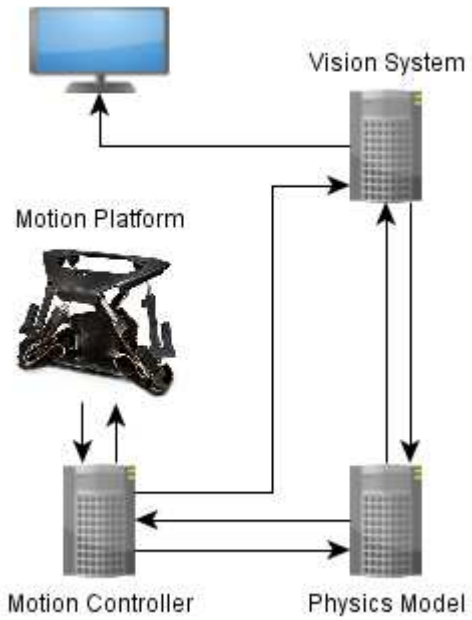

Figure 10: Basic configuration of a typical driving simulator including a motion platform

real-time computer alongside the physics models and in other cases a Hardware-in-the-loop approach is adopted with the real controller connected in to the system.

The vision system itself can consist of multiple computers depending on the actual system configuration. Typically there will be one master computer and then a number of slave machines with at least one computer per projector used.

For a desktop simulator the motion platform and controller is replaced with a steering wheel and pedals that could be as basic as a gaming system or a more specialised system such as steering wheels with high accuracy motors for steering torque assessment.

3.2 Test System Specification

For the example in this paper we have integrated the model described in section 2 within a desktop driving simulator system. We are using rFactor Pro [13] for the vision system and the physics model is running in the McLaren Electronics vTag 310 tool [11]. A Logitech G27 [12] steering wheel and pedals is used for the driving controls together with a single monitor. This configuration makes the driving simulator compact enough to use in the office environment whilst providing a useful tool for evaluating baseline capability of a vehicle or detailed assessment of a control system.

By running the model in the vTag environment we are able to make use of the telemetry system built in to this tool. This means that we can expose the model variables to the telemetry stream and view them in real-time using Atlas [16] (also from McLaren Electronics). This enables logging of the model behaviour for offline analysis.

The use of the vTag environment also enables the control system model to be run alongside the physics model. In Formula 1, for example, it is mandated by the governing body that the complete vehicle control system be developed using the McLaren Electronics tool chain which means it can easily be compiled and run in the vTag environment.

rFactor Pro has an extensive library of scenarios that can be used to test the vehicle. These include LiDAR based race tracks for most of the Formula 1 circuits, North American Indy & NASCAR circuits as well as La Sarthe, Nordschleife and a virtual proving ground with lane-changes, split Mu and low-Mu surfaces, a handling circuit with inclines and programmable surfaces. The environmental conditions (temperature, pressure, humidity and weather) can all be controlled within rFactor Pro. The virtual proving ground includes a wide range of different roads and track surface sections.

3.3 Model build process

To compile the model for use in the vTag environment we first have to define all of the input and output signals at the top level of the model. We then utilize the Source Code Export feature of Dymola to compile the model using inline integration and export the model as c-code.

When Dymola exports the model in this way the code exposes a number of methods that enable the model equations to be coupled to a solver. As inline integration is used with the models the solver doesn't have to integrate any states but it does still have to handle events. The implementation of this solver has been optimized to run the model as efficiently as possible.

The interface between the physics model and the rest of the system is defined in additional c-files that are compiled with the solver and model equations to create the executable model. This interface contains a number of functions that are called by the system to handle the initialization of the model and the calculation of each time step. As the vision system will typically run at a lower frequency than the model the interface supports running multiple model steps each time the vision system asks for a step to be run.

For instance, this means the vision system can run at 500Hz but the model could run at 1000Hz or higher as appropriate. In this test system, the vision system is running at 400Hz and the model is running at 800Hz.

4 Conclusions

Using Modelica to define the vehicle model and Dymola to compile and export the model as c-code suitable for real-time simulation means that the physics model in a driving simulator can be very easily updated to test new concepts as well as explore setup variations of an existing design. The speed with which design ideas can be implemented in Dymola and compiled ready for the simulator means that it is possible for real drivers to start evaluating these ideas at a very early stage in the development process.

References

[1] Otter M., Elmqvist H., and Mattsson S.E.: The New Modelica MultiBody Library. Modelica 2003 Conference, Linkping, Sweden, pp. 311-330, Nov. 3-4, 2003.

[2] Dempsey M. Picarelli A. Investigating the MultiBody Dynamics of the Complete Powertrain System. Como, Italy: Proceedings 7th Modelica Conference, 2009.

[3] John J. Batteh Charles E. Newman. *"Detailed Simulation of Turbocharged Engines with Modelica"* Modelica Conference, 2008

[4] Casella. F. et al. *"The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks"* Modelica Conference, 2006

[5] Christopher Depcik and Dennis Assanis *"A Universal Heat Transfer Correlation for Intake and Exhaust Flows in an Spark-Ignition Internal Combustion Engine"* SAE 2002-01-0372

[6] Stodola, A. (1945). *Steam and Gas Turbines.* McGraw-Hill, New York. Reprinted by Peter Smith.

[7] Eriksson, L. et al *"Modeling of a turbocharged SI engine"* Annual reviews in control 2002.

[8] Heywood, B. *Internal Combustion Engine Fundamentals* McGraw-Hill

[9] Robert Bosch Gmbh *Gasoline Engine Management* Bentley Publishers 2006.

[10] Stone, R. *Introduction to internal Combustion Engines* SAE International 1999.

[11] http://www.mclarenelectronics.com/Products/Product/vTAG

[12] http://www.logitech.com/en-gb/gaming/wheels/5184

[13] http://www.rfactor-pro.com/

[14] http://www.claytex.com/products/claytex-libraries/#vdlmotorsports-library

[15] http://www.modelon.com/products/modelica-libraries/vehicle-dynamics-library/

[16] http://www.mclarenelectronics.com/Products/Product/ATLAS