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

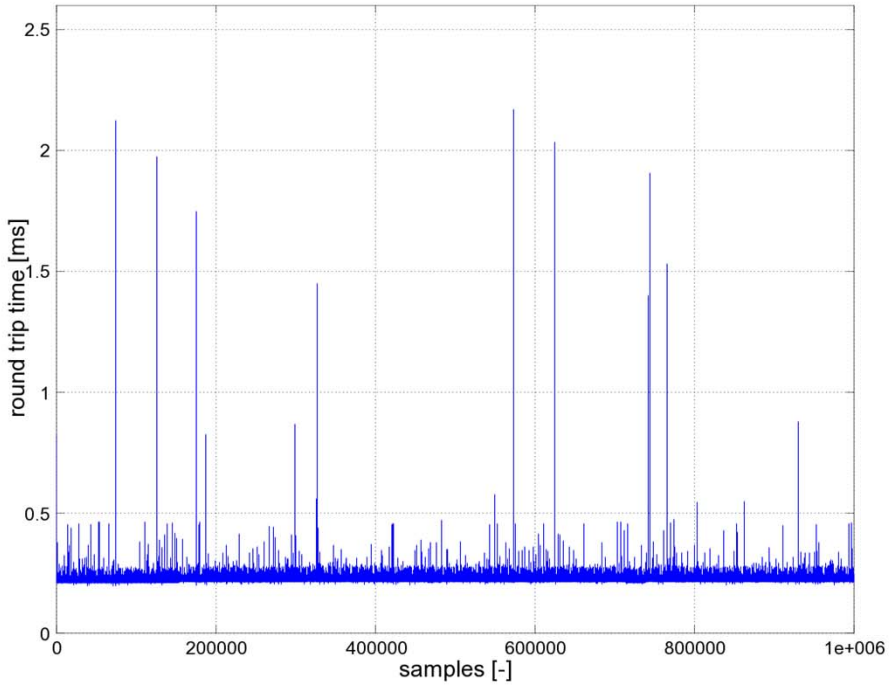
Anders Andersson
VTI - (Swedish Road and Traffic Institute)
Peter Fritzson
LIU - (Linköping University)

Motivation and Research Questions

- Move from Fortran models to Modelica
- - Understandability and maintainability
- - Plug and play of controllers, model components
- Investigate distributed hardware-in-the-loop (HiL) real-time simulation distributed over 500m link
- How to configure two hard-ware simulators in conjunction with Modelica models for HiL simulation
- Validating and tuning Modelica versions of the VTI vehicle training simulator models

Hardware – Network connection

VTI simulator – Linköping university vehicular lab



- ~500 m distance
- optical fiber
- round trip time test with simulator packages

Introduction - Background

- In 2011 Linköping University, LiU, built a new chassis dynamometer lab. With this lab only 500m from the moving base simulator at the Swedish National Road and Transport Research Institute, VTI, it became realistic to connect the facilities.
- As a part of this connection one part is to model the complete distributed setup. The models should be possible to run together with the hardware in different configurations.
- We aim to increase the simulator fidelity and the amount of different simulator setups that can be used.

Introduction – Complete Setup Schematic

VTI training simulator

Driver input via fiber link

Vehicle response via fiber link
(acceleration, speed)

Control vehicle pedals

Car in LIU vehicular lab

Hardware – Sim3
VTI Vehicle Training Simulator

- 4 DOF moving base simulator
    - larger outer motion
- vibration table
- 120 degrees arched screen
- rear view mirrors
- surround sound
- vehicle cabin for driver

Constructed for vehicle dynamics studies but are currently mostly used for behavioral studies.

Hardware – Chassis Dynamometers
at LIU Vehicular Lab

- 4 mobile dynamometers
  - here two are used for FWD
- connects to car
- measure at wheels
- car driver
  - here a robot
- longitudinal vehicle model

A newly built lab at LiU where powertrain dynamics are of interest, e.g. control strategies in hybrid vehicles. Also research in the area of driving cycles.

Hardware – Pedal Robot
at LIU Vehicular Lab

- installed at driver seat
- controls brake and acceleration
  - Sim3 accelerator position
  - Sim3 brake pressure
- UDP communication
- added brake pressure sensor
- use with automatic gearbox

Prototype constructed to test the feasibility of distributed simulation between Sim3 and the chassis dynamometers lab.

Existing Models

- Fortran vehicle model
  - developed and extended over 30 years
  - well known behavior from several studies
  - several different datasets (gearbox, vibration dynamics)
  - modified to receive input from chassis dynamometers
- Modelica car model
  - first version developed in 2012 in a Masters thesis
  - not yet as accurate behavior as the Fortran model
  - model under development for further improvements
  - compiled by Dymola to S-function C-code included by Simulink to xPC-Target for real-time simulation
  - currently experimental testing with OpenModelica

Hardware – Co-Simulation Setup

Co-Simulation

This is the setup we want to model.

Modelica Models

Why Modelica:

- Acausal modeling for a natural model description and easier maintenance.
- Object oriented modeling, we divide the system to components, plug and play of model components

Typical challenges:

- Include sub-models in complete vehicle model, e.g. “We have a new powertrain model, can you put it into your vehicle model?“
- Interfaces towards hardware and software, e.g. “Can we have the ESC production code (black box) connected to the vehicle model?”.

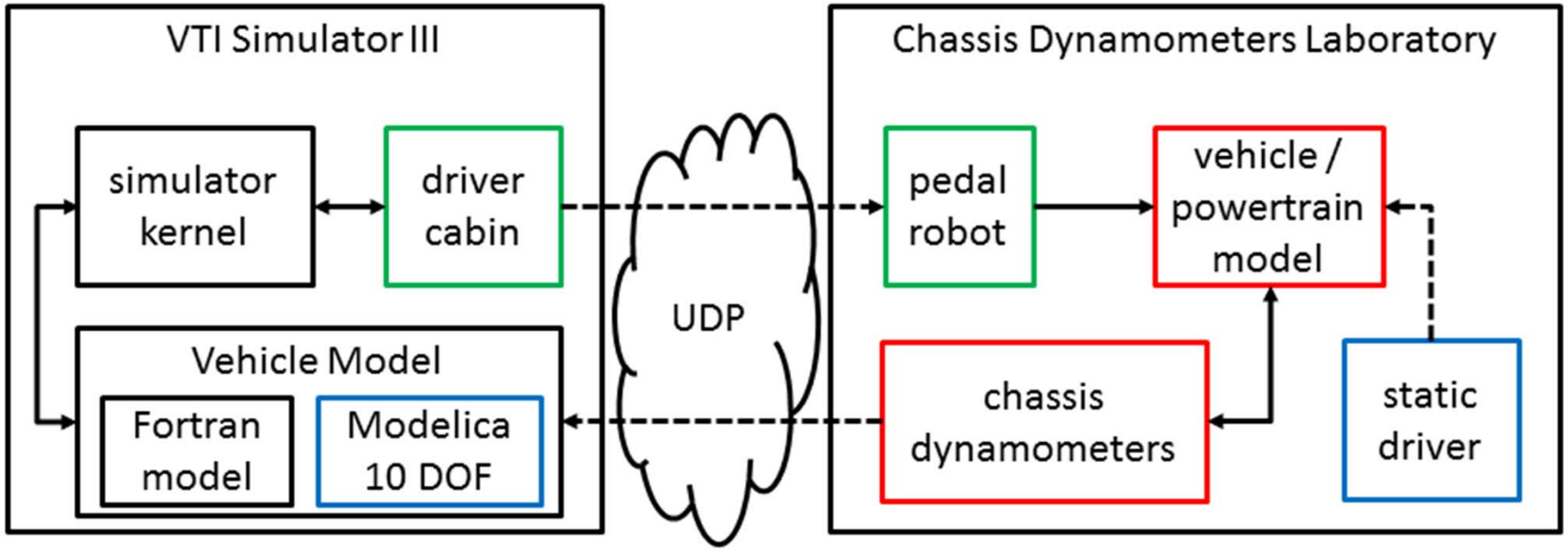
Modelica Models

Main features we aim for here:

- Models should be **parameterized** with open vehicle data since we want to share the models within partners.
- We want the ability to **change vehicles** and thus it should be easy to measure a new vehicle.
- Models for **real-time** simulation.
- Replace the current powertrain model in the Modelica car model.

An OBD II sensor was added to measure powertrain data.

System Architecture and Modelica Models



500m fiber-optic link
UDP communication protocol

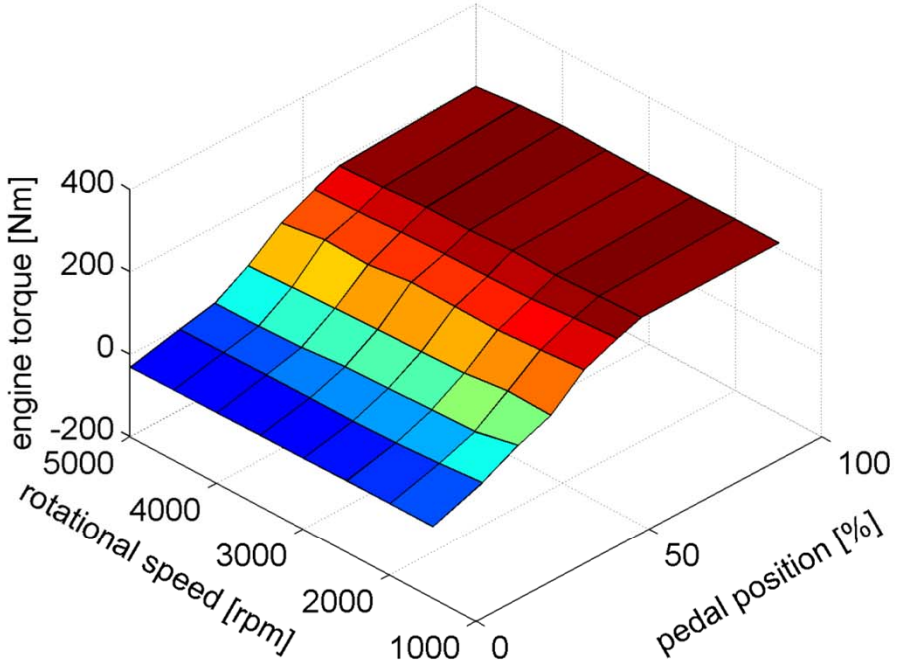
Green boxes – hardware components

Blue boxes – Modelica models

Red boxes – hardware components or Modelica models

Modelica Models - Engine

Measured a static map from accelerator pedal and engine rotational speed to torque output. Added idle and maximum rpm responses.



Notes:

- Measurement gets harder on high and low engine rpm due to the car forcing a gear change (in manual gear mode) and the chassis dynamometers control system.
- It takes time for the engine to reach static levels
- Took 30-60 min for the complete measurement.

vti

Modelica Models - Gearbox

Static measure of gears.

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

Model:

$$\dot{\omega}_{engine} = \frac{(\omega_{clutch} - \omega_{engine})}{\tau_s} * (1 - \Delta_{clutch}) + \dot{\omega}_{clutch} * (1 - \Delta_{clutch}) + \frac{T_{engine} - T_{clutch}}{I_{engine}} * \Delta_{clutch}$$

$$T_{clutch} = T_{engine} * (1 - \Delta_{clutch})$$

Notes:

- Better performance of OBD II sensor needed to measure dynamics during gear changes.
- An accurate sensor for engine rpm.

Modelica Models – Dynamometers and Driver

Chassis dynamometers vehicle model

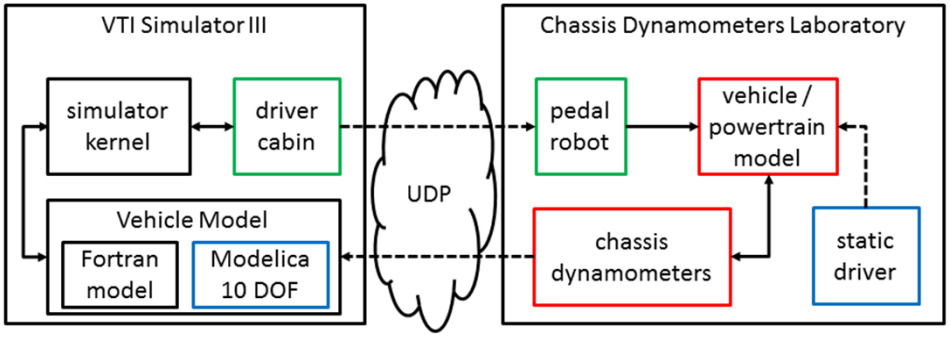
- The chassis dynamometers has a longitudinal vehicle model for vehicle dynamics.
- The output from this model is matched to the output from the hardware chassis dynamometers.
- The connection points in the Modelica model is also at the wheels to correlate to the hardware.

Static driver

- To test the complete setup of models a driver was used.
- The pedal robot not included at this stage (introduced error with pedal robot is about 1 percent).

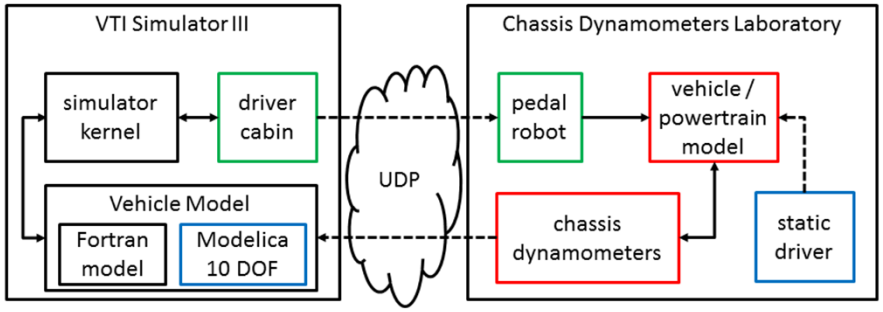
Chassis Dynamometer Modelica Model

```
model ChassisDynamometerSystem
  StaticDriver                        driver;
  ChassisDynamometerVehicleModel      chassis_dynamometer_vehicle_model;
  Powertrain                          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;
```



Modelica Model – Static driver

```
model StaticDriver "driver with pre-defined output"
  output Real throttle "throttle position scaled [0.0-1.0]";
  output Real clutch "clutch position 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;
equation
  der(throttle) = if time < 17 then 10 * (0.25 - throttle)
                  else 10 * (0.25 - throttle);
clutch = if abs(time - 4.5) < 1 then 1 - max(0, min(1, abs(time - 4.5)))
         elseif abs(time - 12) < 1 then 1 - max(0, min(1, abs(time - 12)))
         elseif abs(time - 20) < 1 then 1 - max(0, min(1, abs(time - 20)))
         else 0;
  brake = 0;
  gear = if time < 4.5 then 1
         elseif time < 12 then 2
         elseif time < 16 then 3
         elseif time < 35 then 4
         else 3;
  stw_ang = 0;
end StaticDriver;
```



Modelica Model – Chassis Dynamometer Vehicle

```
model ChassisDynamometerVehicleModel
  package Interfaces =
Modelica.Mechanics.Rotational.Interfaces;
  Interfaces.Flange_a fl;
  Interfaces.Flange_a fr;
  Interfaces.Flange_b rl;
  Interfaces.Flange_b rr;
  Modelica.SIunits.Acceleration a(start = 0);
  Modelica.SIunits.Velocity v(start = 0);
  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;
protected
  package SI = Modelica.SIunits;
  constant SI.Mass m = 1401 "vehicle mass";
  constant SI.CoefficientOfFriction c_d = 0.32;
  constant SI.Area A_f = 2.0 "vehicle front area";
  constant SI.CoefficientOfFriction c_r = 0.001;
  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
....
```

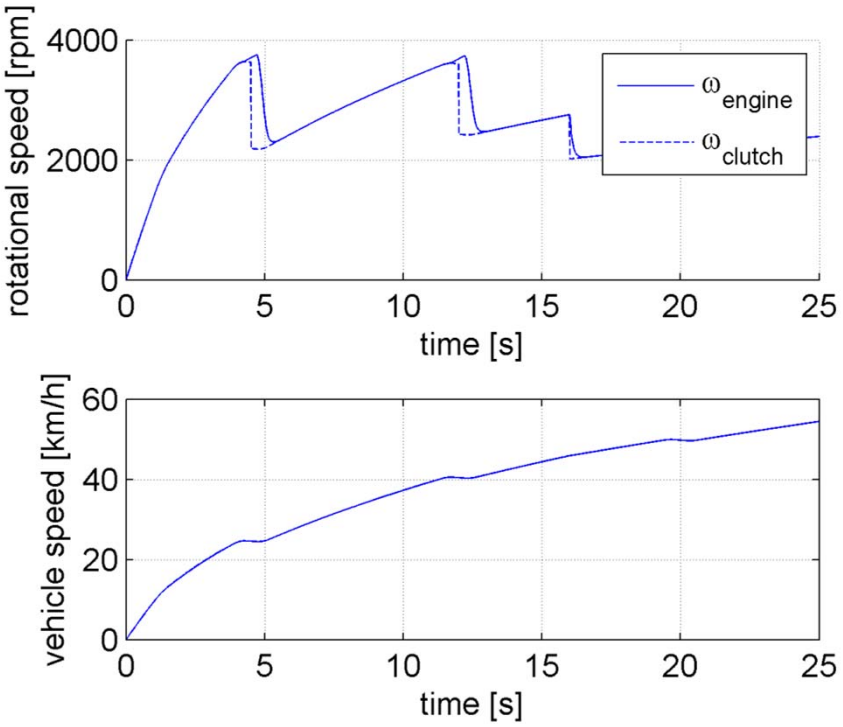
vti

Modelica Model – Chassis Dynamometer Vehicle

```
...
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;
  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;
```

Modelica Models – Powertrain Performance

- The figure shows a slow acceleration when running the engine, gear (manual), chassis dynamometers and static driver together.
- The difference in rotational speed is from the driver.
- Pressing and releasing the clutch takes about 1 secondfor the driver.



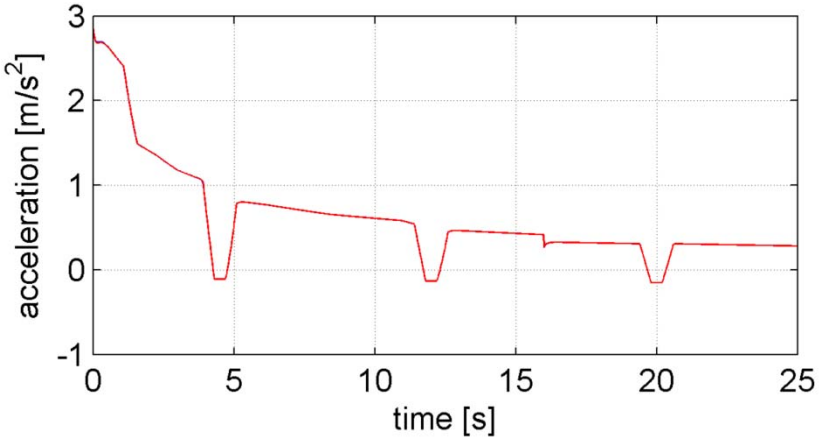
Simulation done in OpenModelica with models of the chassis dynamometers, static driver, and the measured powertrain.

vti

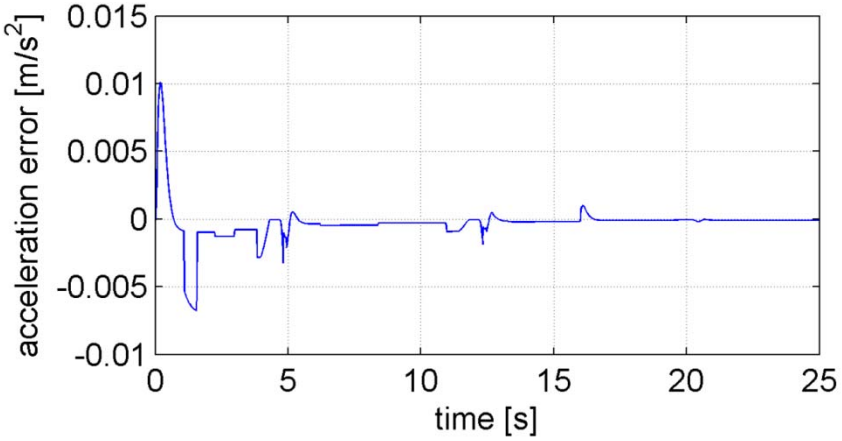
Modelica Models – Powertrain Performance

The difference between solvers is small (below 1%).

Shown here is the difference between DASSL and Euler forward. Simulation done in OpenModelica.



Acceleration Value



Diff between solvers DASSL and Euler

vti

Modelica Models – Powertrain Performance

- Measurements using a real-time profiler in OpenModelica
- Should be possible to run them in real-time when aiming for at least an update speed of 100 Hz.

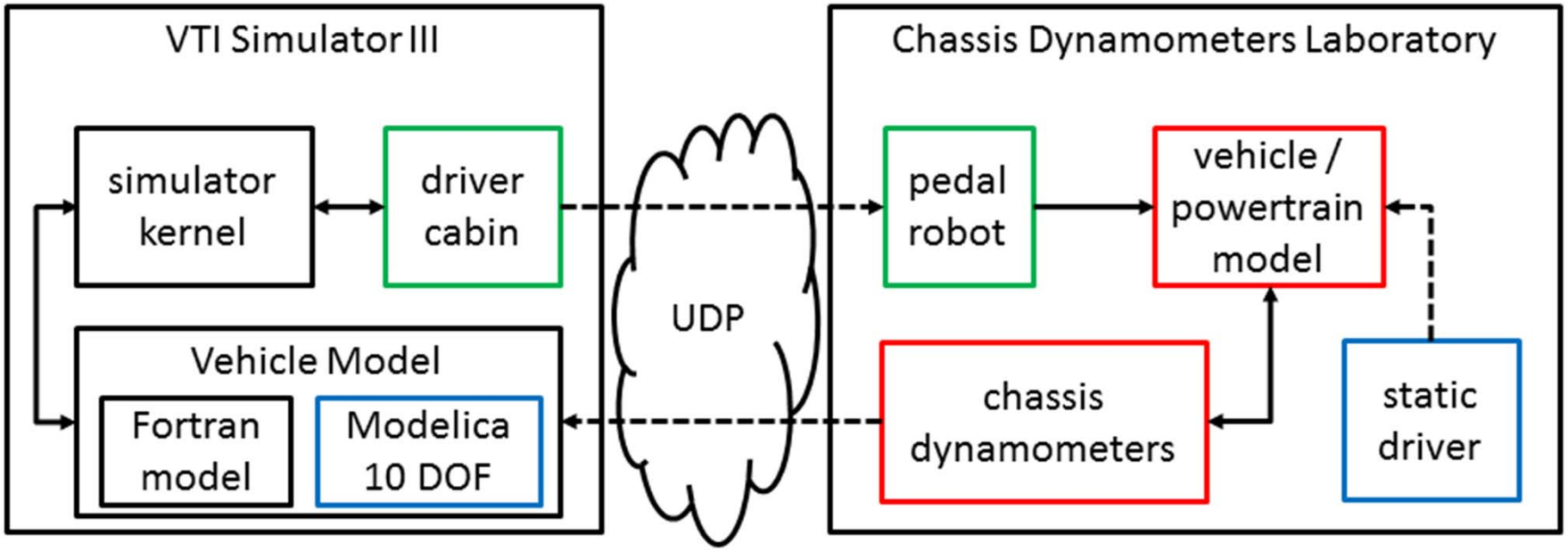
(The test run is the same maneuver shown in previous slides.)

Steps	Total time	Fraction	Average Time	Max Time	Deviation
2508	0.0712	67.61%	0.0000284	0.000432	14.22x

- These measurements are made on a 2.6 GHz Intel Core i5 computer with 8 Gb of RAM.
- As can be seen there should be plenty of margin for 100 Hz (goal will probably be to run it in 1000 Hz).

Conclusion

The final configuration provides a basis for several different simulation configurations,
e.g. experience a driving cycle using a static driver and the chassis dynamometers connected to VTI Simulator III.



Conclusion

- Models have been **parameterized from hardware**.
- New model parameters can be measured in one day and probably faster.
- Profiler shows that it should be **possible** to run the models in **real-time**, in the **distributed** setup
- Desire to have faster sensors for dynamic measurements.
- Measured engine and gearbox parameters have been used in the Modelica car model.

Future work

- Run the models in real-time maybe using models exported using FMI.
- Improve models, e.g. add automatic gear.
- Run complete setup in different configurations using hardware and models.