A Modular Technique for Automotive System Simulation

Felix Günther, Georg Mallebrein, Heinz Ulbrich[*]
Robert Bosch GmbH, Stuttgart, Germany
[*]Technische Universität München, Institute of Applied Mechanics
{felix.guenther, georg.mallebrein}@de.bosch.com, [*]ulbrich@amm.mw.tum.de

Abstract

Increasingly challenging requirements such as environmental and safety legislation as well as increasing development costs are leading to a need for more overall system understanding in the automotive sector. Modelica, as a suitable way for multi-physics modeling, is therefore applied by Bosch, e.g. to investigate energy flows amongst domains.

We present a modular approach consisting of two parts to handle complexity and increase the performance: a modular library for the different domains and a co-simulation framework. To begin with, coupling aspects such as causality and communication are discussed in this context and their implementation is shown. A further focus is the variable macro step size that we developed within the framework for the automotive drive cycle simulation. The results of the modular approach are described and analyzed regarding error and performance aspects. Finally, challenges of the work are mentioned and an outlook, including FMI [2], [10], is given.

Keywords: co-simulation; automotive system simulation; multi-domain

1 Introduction

Scarcity of resources, legislation and customer demands continue to be the main driver for automotive manufacturers. New technologies such as hybridization or full electrification but also systems and components to increase the efficiency of the conventional powertrain help to reduce CO_2-emissions. Especially a supplier such as Bosch, providing a broad range of components as well as system solutions, requires a profound overall system understanding in all development stages. This is achieved by simulation, applying acceptably complex but comprehensive vehicle models. In contrast to signal-oriented or domain specific tools, Modelica proved to be a suitable way for physical, multi-domain and object-oriented modelling and is therefore applied, currently using DYMOLA2012 [5] as simulation environment. Besides the physical domains, the vehicle controllers complete the models and hence a forward oriented, robust drive cycle system simulation is done. The resulting energy flows amongst the domains during a drive cycle give potential assessments of a certain system or component. This is compared to vehicle measurements (as e.g. in [16]).

A drawback of including all vehicle domains in one model is that the generated hybrid DAE system causes a high computational effort. Putting together the powertrain model with a detailed thermal and exhaust system model leads to simulation times several times slower than real-time on a Windows PC system.

To avoid this effect of complexity, the vehicle model is partitioned into subsystem models which are simulated in parallel using their locally adapted solvers. In a first approach, three subsystems were coupled, performing a co-simulation via TISC [17], at the expense of a precision loss but resulting in speed-up by factor of 5 on a single core PC. Additionally, using simulator coupling, the possibility to introduce existing MATLAB/SIMULINK or AMESIM models is given.

To reduce the numerical error introduced by partitioning and coupling, further development on coupling aspects was done. This was realized by applying MDPCosim [11] as master-slave co-simulation environment and expanding it with approximation methods and a variable macro step size control. In comparison to the approach with error estimation by repeating steps [4], a different approach with heuristic methods is being developed and in use for the presented drive cycle simulations.

2 Modular Simulation of Automotive Models

In order to optimize the energy consumption of a vehicle, simulation of the overall system is needed. This includes, besides the model of the powertrain with driver, engine, transmission, brakes and driving resistances, all energy-relevant subsystems, namely the exhaust system, cooling and oil circuit as well as the electric power net and the system control. Accordingly, the models contain the mechanic, hydraulic, thermal, electric and boolean logic domain.

A modular realization was consequently chosen. On the one hand modularity is used to derive defined interfaces between the subsystems. This is obligatory for a collaboration of several developers or even several tools. On the other hand, a partition into modules for a possible co-simulation is prepared.

2.1 Motivation for modular simulation

One reason for modular simulation is to assemble models, developed in different tools. [7] shows different approaches, thus model coupling could be e.g. realized via FMI for model exchange [2]. Another reason is to partition the system to benefit of multi-rate time integration [1], [14], using different solvers for different dynamic behavior of subsystems. This can be used to meet real-time requirements for HiL simulation in the automotive domain, such as achieved in [9]. For HiL simulation fixed-step solvers are being used though, in contrast to the presented overall system simulation with variable DAE solvers.

In order to measure the computation time (t_{CPU}) overhead by assembling i subsystem models to the overall vehicle, a factor θ_A (1) is introduced.

$$\theta_A = \frac{t_{CPU,overall\ model}}{\sum_i t_{CPU,\ partial\ model\ i}} \tag{1}$$

During development of the subsystem models, e.g. the thermal system including the fluid circuits, the behavior of the vehicle is introduced by measured timetables and a standalone simulation is possible. Adding the thermal system model to the residual model of the vehicle containing already the other domains (i = 2), a θ_A of 12.1 was observed (for a more complex thermal model 19.2). Adding as an example a detailed model of the battery (i = 3) will again increase this factor. Therefore, instead of simulating the model with one solver, the modular approach is chosen, which additionally provides efficiency by simulating in parallel. Here, θ_A can be seen as an upper limit for the speed-up achievable by multi-rate advantages. For stability and accuracy reasons a partition leading to preferably weak coupling is useful, also giving the possibility to apply larger macro steps H. One possible partition method is described in [14]. Another method is the TLM (Transmission Line Modelling) approach, as presented in the Modelica context in [13]. TLM creates a modular simulation by adding a solver to each component. Though, the advantage of symbolic manipulation of the equations for multiple components within one technical subsystem would disappear. Hence, in our work the nearby application along technical domain boundaries is chosen.

2.2 Coupling aspects

For a co-simulation of the modular vehicle model the implementation of different coupling aspects is necessary. In the following, those aspects are categorized and their application is described:

- **Synchronization:** different communication strategies between the solvers are possible. Figure 1 shows in its upper part a sequential asynchronous solution, e.g. described in [15] with advantages in accuracy and no necessity to define macro step sizes. The lower part presents the parallel synchronous solution, which is more efficient and therefore used here. The MDPCosim master controls the slaves including the models, who can run in parallel
- **Causality:** the advantage of Modelica with physical modeling and equation preprocessing disappears at the coupling interfaces, where causal, directed signals have to be used. MDPCosim covers the possibility for connecting slaves with coupling laws in the master [12] (flow-flow-coupling) and e.g. a reaction torque is retrieved in the master. For the applied step sizes and partitions in the vehicle simulation the more conducive technique of potential-flow-coupling is adopted. Details of the causal interface are commented in section 3.
- **Approximation:** the discretization introduced at the interface is adding an additional error to the simulation that can be reduced by approximation methods. Depending on the chosen synchronization scheme, different methods are possible: extrapolation, interpolation or even iterative such as described in [3]. It can be implemented in the master (constant), the slave (time varying) or both. On the present, parallel case, extrapolation including smoothing is chosen, see section 3.
- **Macro step size:** using synchronous coupling, a suitable macro step size has to be chosen. An efficiency gain for the overall simulation with acceptable co-simulation error needs to be combined. Therefore, investigation for fixed, predefined (timetable) and in conclusion variable macro step sizes was

done. As a result, a heuristic method was developed, which is described in detail in section 4.

- **Event handling:** occurring events in the subsystems will cause severe errors if the coupling values are affected, e.g. in a start-stop-strategy. This must be avoided thus no discontinuous signals are chosen in the present interfaces. Still, detecting and treating events during co-simulation is important for future work and one viable solution could be using FMI for co-simulation [10].

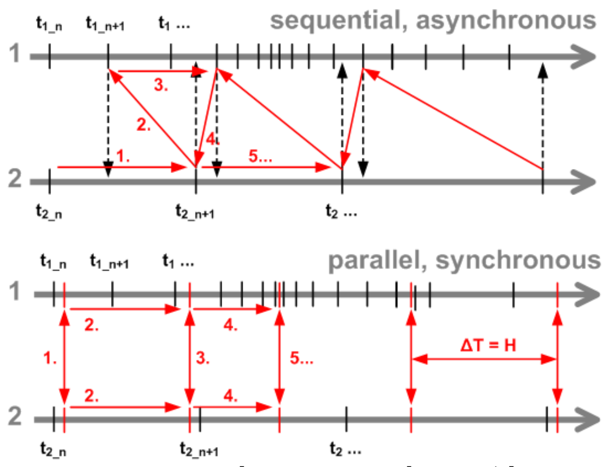


Figure 1: Two synchronization schemes (slaves 1, 2)

2.3 Evaluation of modular simulation

Illustrating the implementation of some coupling aspects and signal routing, figure 2 shows the implemented structure for the slaves in Modelica.

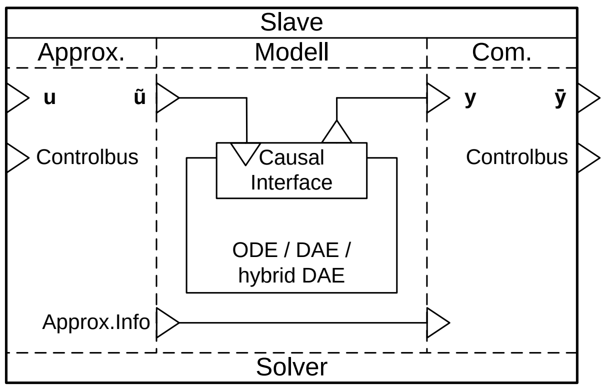


Figure 2: Structure of a slave model with signals

The input signals $\boldsymbol{u}$ are extrapolated in the approximation section to $\tilde{\boldsymbol{u}}$. As wrapper to convert $\tilde{\boldsymbol{u}}$ and the output signals $\boldsymbol{y}$ to the physical proper model the causal interface for different domains is modelled. The continuous $\boldsymbol{y}$ then are communicated to the master as discrete signals $\bar{\boldsymbol{y}}$. Additional, modular specific information is written to the control bus as explained later.

In order to evaluate the accuracy of the modular approach, a discretization error $\bar{\tau}_{CUM}$ is defined by (2).

$$\bar{\tau}_{CUM} = \frac{\int \left| \bar{y} - y \right| dt}{t} \tag{2}$$

Notable at the physical interfaces this error is fed back and influences the behavior of $\boldsymbol{y}$, compared to the same states $\boldsymbol{y}^*$ in a monolithic simulation (one solver). The correlation in the master between slave inputs $\boldsymbol{u}$ and outputs $\boldsymbol{y}$ is given as incidence matrix $\boldsymbol{I}$:

$$\boldsymbol{u} := \boldsymbol{I}\boldsymbol{y} \tag{3}$$

In order to take into account the accuracy augmentation by approximation, as well as the cumulated feedback influences, the co-simulation error

$$\tilde{\tau}^*_{CUM} = \frac{\int \left| \tilde{\boldsymbol{u}} - (\boldsymbol{I}\boldsymbol{y}^*) \right| dt}{t} \tag{4}$$

is introduced. In (3) and (5) t represents the simulated time.

Both accuracy and efficiency of the modular approach with co-simulation have to be regarded. Besides cumulating events and F-evaluations the speed-up factor S_{CS} (5) is important, having the reduction of simulation time of the overall vehicle model as motivation.

$$S_{CS} = \frac{t_{CPU,monolith.}}{t_{CPU,CoSim}} \tag{5}$$

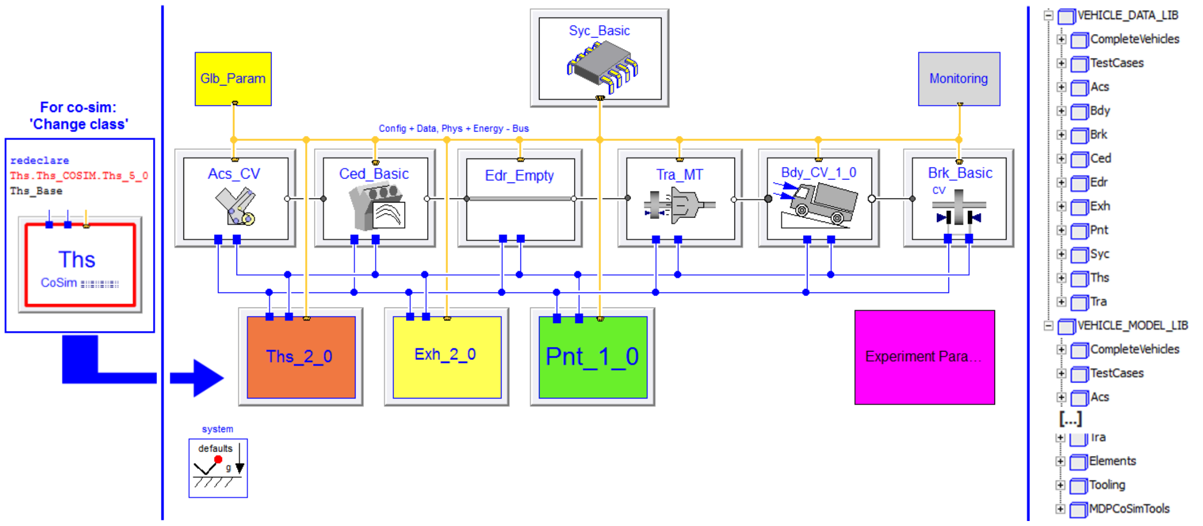
The accuracy and efficiency of co-simulation was observed during the development of the library and the framework described below.

3 Implementation

As mentioned in the introduction, the modular automotive system simulation relies on two parts, the modular library and the co-simulation framework.

3.1 Modular library

The development of the library was based on using the Powertrain Library [6]. In addition, detailed models of the oil circuit, cooling circuit, HVAC, the exhaust system and the power net were developed. In order to enable configuring multiple classes of vehicles in different model granularity and combine them with existing in-house data libraries, the modular library was developed.



```
redeclare
Ths.Ths_COSIM.Ths_5_0
Ths_Base
```

Figure 3: Example: overall vehicle model / data and model structure in the library

It contains a '_MODEL_LIB' part for the model development and, mirroring the same structure, a separate '_DATA_LIB' part, where vehicles and subsystems are configured, parameterized and set up for different (drive cycle) simulation experiments. Figure 3 shows a top-level model of a commercial vehicle. On the right part, the library structure with model and data part is shown. The different subsystems, e.g. for combustion engines ('Ced') or thermal systems ('Ths'), can be changed by redeclaring the class with other data lib models. In the same way, a subsystem can be set up to be co-simulated, as shown in the left part of figure 3: The Ths-model is replaced by an interface ('Ths CoSim') directing to the thermal system co-simulation slave, which can be found as standalone model in the library structure and will be run in parallel.

Modularity is also represented in the subsystem models. Figure 4 depicts the thermal system. It contains replaceable models for the energy balance, combustion engine heat, cooling circuit, oil circuit and HVAC.

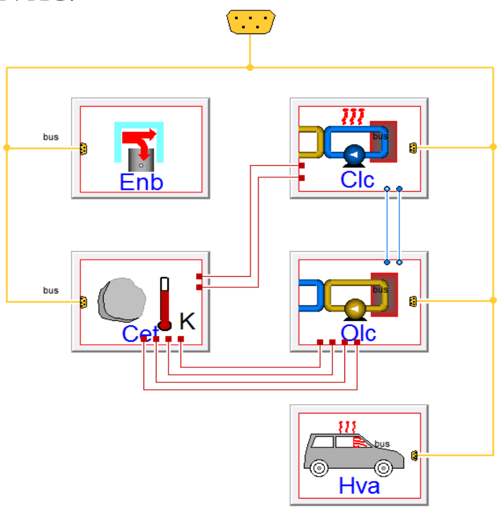


Figure 4: The thermal system model

In order to allow maximum modularity accompanied by physical coupling between the subsystems, causal interface models are introduced for different domains enabling signal exchange with a 'causalSubBus'. In such manner e.g. the oil pump is coupled to the powertrain part. The related flange interface in figure 5 shows the crank part.

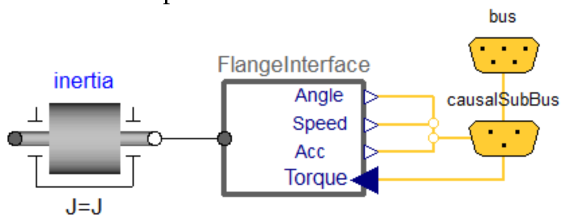


Figure 5: Causal interface for flanges

Depending on the macro step sizes, a flow-flow or a potential-flow-coupling can be chosen. Similar interfaces are used for the thermal part. For communication with MDPCosim and signal approximation, a configurable interface model is in the library, figure 6.

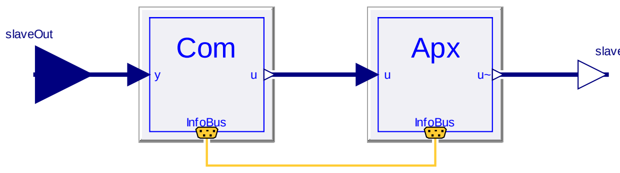


Figure 6: Co-simulation interface model

Different slave approximation methods, partly in combination with master approximation of flow variables, have been tested on a two-mass-oscillator model as well as in the vehicle context. Following, different methods, such as Taylor, Lagrange and Hermite polynomials and a transition method, smoothing signal jumps, are implemented for the library and applied in the 'Apx' block. All methods can handle a variable macro step size. Additionally, an 'infoBus' is introduced containing information about the signals approximation.

3.2 MDPCosim framework

The latter information can be fed via a control bus and used for master algorithms. This architecture is described in figure 7. It shows the configuration of the MDPCosim framework [12] and its adaption as vehicle co-simulation environment in C++. The abovementioned overall vehicle model is represented as slave 1 … *N*. An overhead process actuates the master and slave processes. These run in parallel, while the co-simulation is controlled by the master. This includes synchronization, connecting the signals (feedthrough or coupling law with approximation [12]) and the macro step size algorithm.

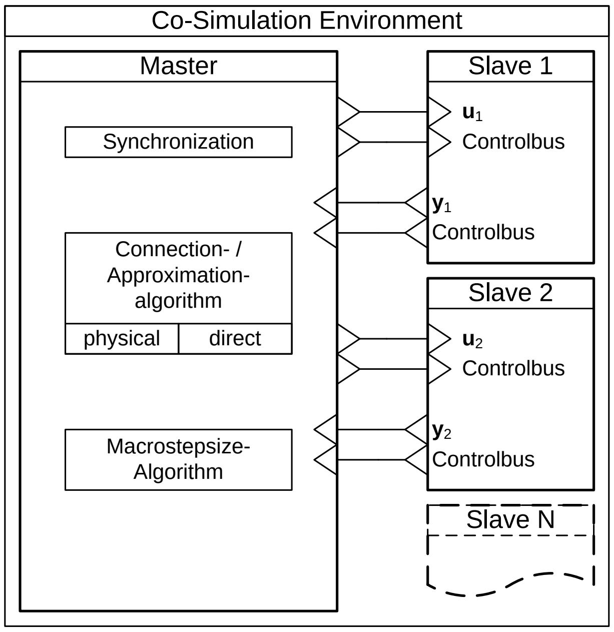


Figure 7: Architecture of the adapted environment MDPCosim

As inter process communication, shared memory is used; TCP/IP is planned for future work. Besides the coupling signals ($\boldsymbol{u}, \boldsymbol{y}$) the control bus connects master and slaves. It contains: derivatives of $\boldsymbol{y}$, step size H, T_{next}, $\bar{\boldsymbol{\tau}}_{CUM}$ and information about approximation, local step sizes $\boldsymbol{h}$ and events. This information is handled within the master algorithms.

The inner layout of the slaves is shown in figure 2 and figure 6 respectively.

3.3 Batch co-simulation

As a tool for the development of modular methods, MDPCosim was expanded with a superimposed algorithm that allows automated batch runs. This is used to sweep parameters of the master as well as the slave models. Hence, fitting of model parameters is possible or a variety of co-simulations needed for requirements engineering of a certain component in the overall system context can be run.

Figure 8 demonstrates the data and process flow of a batch run. It is configured, using a file that contains the following information: number of runs, type, identifier of a parameter (file), (min and max) values. The different types are 'variants', 'parameter', and 'autoParaVari'. The type 'variants' is followed by a file identifier defining numbered versions of a model or master parameter file or different model files. The other types allow naming a parameter to be varied, giving all values or giving a minimal and maximal value.

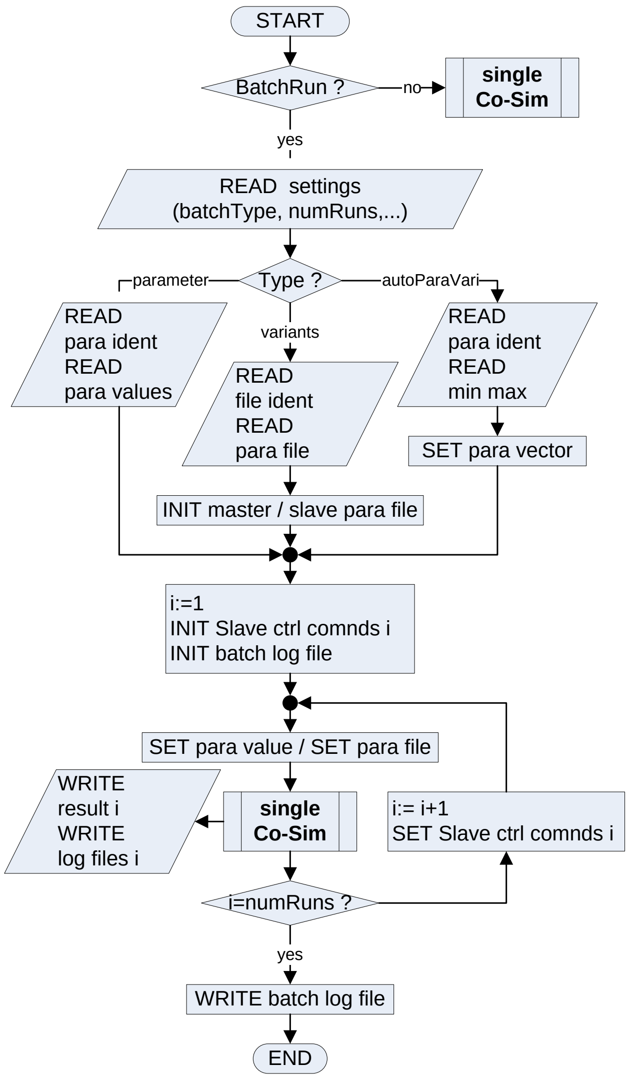

START
BatchRun ?
no
single Co-Sim
yes
READ settings (batchType, numRuns,...)
Type ?
parameter
autoParaVari
variants
READ para ident READ para values
READ file ident READ para file
READ para ident READ min max
SET para vector
INIT master / slave para file
i:=1 INIT Slave ctrl comnds i INIT batch log file
SET para value / SET para file
WRITE result i WRITE log files i
single Co-Sim
i:= i+1 SET Slave ctrl comnds i
i=numRuns ?
yes
WRITE batch log file
END

Figure 8: Flow chart of the batch architecture

The result files and a batch log file can be imported in MATLAB to be commonly evaluated for the aspects described in section 2.3. By means of this evaluation the development of approximation and macro step size algorithms is done.

4 Macro Step Size Control for Drive Cycle Simulation

The objective of modular automotive simulation is to increase simulation speed, while keeping the co-simulation error acceptable. In contrast to the sequential asynchronous approach, see 2.2, for the parallel synchronous technique, defining the macro step size *H* is necessary. Setting $H = h_i$ (with h_i: local step size of slave *i*) is not conducive, if we consider the effort for each macro step with an event in all slaves, waiting for synchronization and the master algorithm for coupling. Thus relatively large *H* are pursued. To reduce the following discretization error, the extrapolation methods are adopted and an algorithm for variable *H* is developed.

State of the art of simulation tools and the abovementioned context (section 2 and 3) lead to the following boundary conditions for the macro step size control: large subsystem models are solved with a commercial simulation tool and are therefore seen as black boxes for the master; overall, the coupling is kept weak (slow changing temperatures in the thermal model / small pump inertia compared to the powertrain inertia.); the drive cycles (e.g. [8] or [18]) span more than 1000 s and macro steps in the range of 0.1 s and 10 s are chosen; the drive cycles provide an approximate predictive behaviour of the overall vehicle. Embedded methods or the Richardson method such as presented in [4] are neither conducive (for efficiency reasons) in the present use-case nor possible (yet), since a macro step cannot be repeated. Thus, the methods, described in this paper are heuristic and strictly monotone procedures based on indicators.

The chosen approaches go without the need to repeat steps and partly establish the general correlation:

$$H = H(\boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{h}, \bar{\boldsymbol{\tau}}, \tilde{\boldsymbol{u}}(t), t\_event, ..., \boldsymbol{p}, ...) \tag{6}$$

With ***p*** as parameters to be defined by the user and ***ũ**(t)* derived by the knowledge about the used approximation method for each signal. Based on (6) different approaches can be combined:

- H is determined basing on additional user input parameters ***p*** or simply the common RTOL/ATOL user limits.
- H is determined by one leading slave output y, by multiple or all outputs of all slaves ***y***.
- H is determined with local slave information about derivatives $\dot{\boldsymbol{y}}$, error $\bar{\boldsymbol{\tau}}$, local step size(s) ***h***, approximation and events. This information is provided via the control bus (Figure 2, 7).
- H is determined with or without quasi-error estimation based on $\bar{\boldsymbol{\tau}}$ and $\tilde{\boldsymbol{u}}$.

In figure 9 the master algorithm with the step size control (‘H algorithm’) is explained. If the parameter for *H* is set to <0, a table file with *H* = *f(t)* is used and if *H* is set to 0 the *H* controlling algorithm is initiated. After initializing the slaves and the master including the *H* algorithm initialization, the co-simulation cycle starts. Within each cycle, after executing each macro step, the H algorithm is called and can set a new value for *H*.

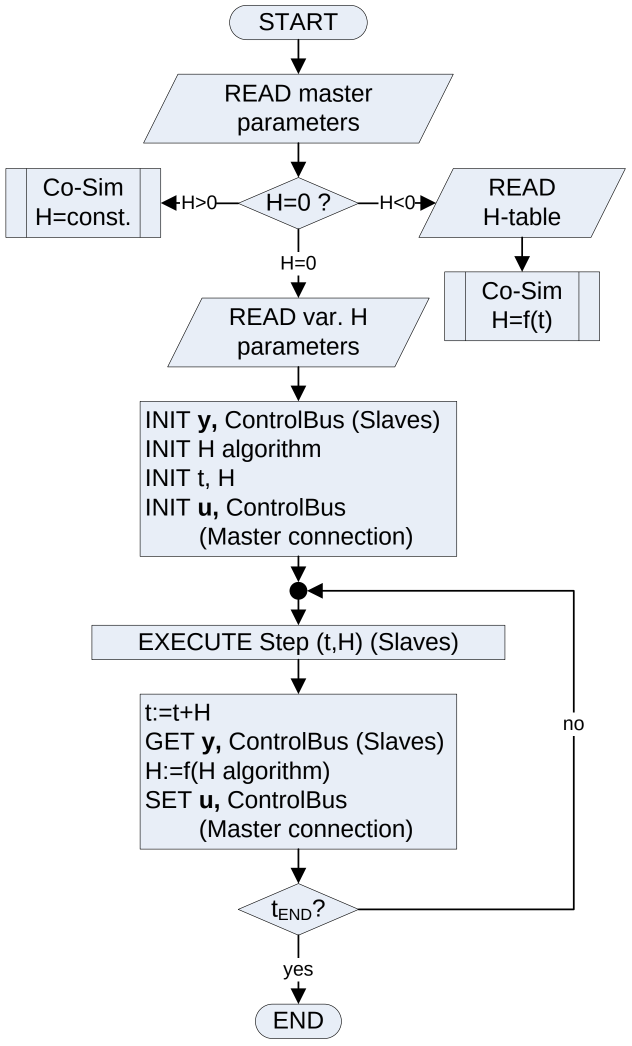
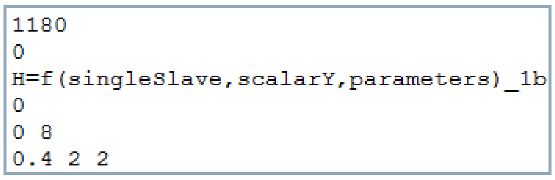


Figure 9: Flow chart of the master algorithm

In figure 10 an example of a master parameter file is shown. Starting with the entry for the co-simulation end time, the second line defines *H*. If it is 0, the algorithm continues reading the parameter file with a line for the chosen H algorithm type and a line for start value for *H* (optional), followed by type specific parameters (see next section).



```
1180
0
H=f(singleSlave,scalarY,parameters)_1b
0
0 8
0.4 2 2
```

Figure 10: Example: Master parameter file for variable macro step size

Currently, the development of the more sophisticated H algorithms is still ongoing. However, first algorithms are implemented and in use. A state-lead algorithm is described in 4.1.

4.1 Implementation

In a drive cycle, the desired vehicle speed v and the gear is given as *v = f(t)* and gear *= f(t)*. Additionally, the resulting kinematic states in the powertrain dominate the overall vehicle model behavior. Accordingly, the macro step sizes are based on gradients of the vehicle speed ($\dot v$) or the engine speed ($\dot\omega_{eng}$) as leading states and indicator for most changing rates of the model states. Thus the macro step size *H* is set inversely proportional to the last gradient value.

As shown the example in figure 10, line 5, the user has to provide parameters for the index number of the slave *i* and the belonging index number *j* of the leading state. This has to be completed with the last line of parameters with values for H_{min}, H_{max}, method tuning parameters and minimum and maximum gradient values. To reduce the user input, a method without the latter entries was developed. The correlation follows with

$$H = H_{\min} + [1 - \min(1, \dot y_{dl})]^{p_0} (H_{\max} - H_{\min}) \qquad (7)$$

with a dimensionless $\dot y_{dl}$:

$$\dot y_{dl} = \frac{\left|\dot y_{i,j}\right|}{p_1 \dot y_{avg}} \qquad (8)$$

depending on a weighted mean value over the current simulation time. The algorithm can be optimized with the remaining parameters: H_{min}, H_{max}, p_0, p_1. For this purpose, batch runs for each parameter are done with representing use case models and evaluated using MATLAB. One of the results is shown in figure 11, where a variation of H_{min} from 0.05 s to 0.5 s, holding all other master parameters constant, is presented. It covers the evaluation for number of steps, speed-up S_{cs} (5) and the error $\tilde\tau^*_{CUM}$ (4). In that manner, the parameters were optimized for a certain drive cycle.

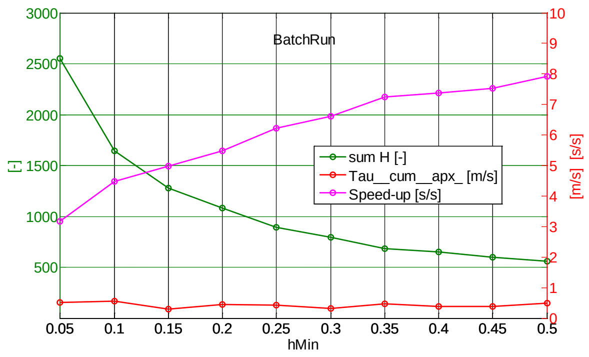


Figure 11: Evaluated batch run varying H_{min}

The macro step sizes during a drive cycle (NEDC) in figure 12 are in the range of 0.4 s and 2 s and result in an average step size of 0.92 s.

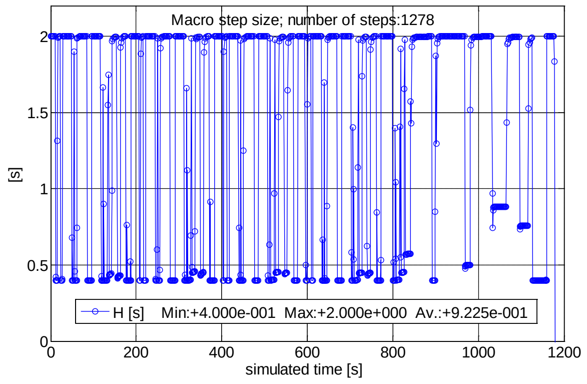


Figure 12: macro step sizes in a drive cycle

To evaluate the algorithm, this is compared to a fixed H co-simulation with this average value. The comparison is shown in table 1.

Table 1: macro step size control method evaluation

	variable H	fixed H (0.92 s)
S_{cs} [s/s]	4.516	4.502
$\bar\tau_{CUM}$ [m/s]	0.036	0.060

With a comparable speed-up factor the cumulated discretization error could be significantly reduced. With adapted parameters, the method was also successfully applied for the two-mass-oscillator test case. However, the user needs knowledge about the model and the coupling method. Therefore ongoing investigation is done on methods with less user inputs on the one hand and embedding more local information as well as a control strategy for approximations on the other hand.

5 Use-Cases and Results

All models presented in this paper are simulated using DYMOLA [5] and co-simulation is done by coupling several DYMOLA processes. The vehicle hybrid DAE models require the use of the ‘dassl’ solver and the same integration settings are used for all experiments. The modular simulation was investigated on test use-cases, a two mass oscillator (TMO) and simple thermal modal as well as in the overall automotive context. For completeness, some results of the approximation tests with the TMO are given in the following.

5.1 Two-mass-oscillator test case

To develop the causal flange interface, an undamped rotating TMO is modeled with high frequency of the left and low frequency in the right mass and simulated, using MDPCosim. Thus the direction and the combination of approximations can be distinguished. Table 2 shows some results. The co-simulation error after 20 s with H = 5 ms is compared for potential-flow-coupling (angle φ_L to the right side) and flow-flow-coupling with different approximation methods. It could be reduced by more than two orders.

Table 2: TMO: improvement with approximation

	$\tilde{\tau}^*_{\mathrm{CUM},\varphi_L}$ [rad]
potential-flow-coupling: no approx. (0. order extrapolation)	4.43e-1
potential-flow-coupling: phi_left: first order transition tau_right: 4-point-lagrange	2.90e-3
flow-flow-coupling: master: 2nd order method slaves: first order transition	1.38e-3

5.2 NEDC: vehicle with detailed thermal system

Following the conditions for overall vehicle simulation with larger step sizes, in the current state of the modular library only potential-flow-coupling and signals without direct physical reaction are used, such as the fuel mass flow. For temperature signals a 4-point-lagrange polynomial and for (rotational) speed signals a first order taylor or the transition method is configured.

Here, as an example a conventional passenger car model is coupled with a detailed thermal system model, similar to the one in figure 4, but with only a two-thermal-mass motor block model and simplified models of the cooling system and HVAC. The results are evaluated by referencing the same overall vehicle model, simulated without co-simulation with only one solver. For the NEDC (simulated time 1180 s), the computing time was 1400 s. With approximations and the variable macro step size (see 4.1; H: Ø 0.92 s) the co-simulation computing time was 310 s (speed-up: 4.5 and real-time capable). It lead to an acceptable error e.g. of the fuel consumption value of << 1%. Compared to θ_A = 12.1 (see 2.1) there is more speed-up capability. This can be reached using larger Ø H, however finally leading to inacceptable accuracy. For more complex models of the thermal, exhaust system or the powernet, more speed-up is reached.

Figure 13 shows three different simulations for the same acceleration sequence in this cycle. The simulation results for the engine speed of the reference, a co-simulation with correlating fixed H and the co-simulation with variable H and 0. order extrapolation are compared.

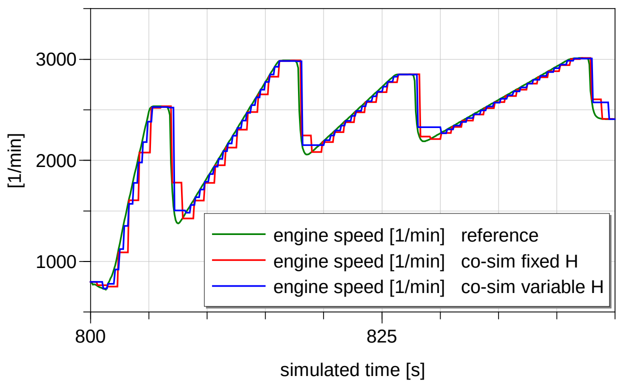


Figure 13: engine speed in an acceleration sequence

The same three simulation results as in the upper figure are taken in figure 14. To compare the behavior of an approximation method in combination with variable macro steps additionally, the warming-up curves of the average oil temperature are taken.

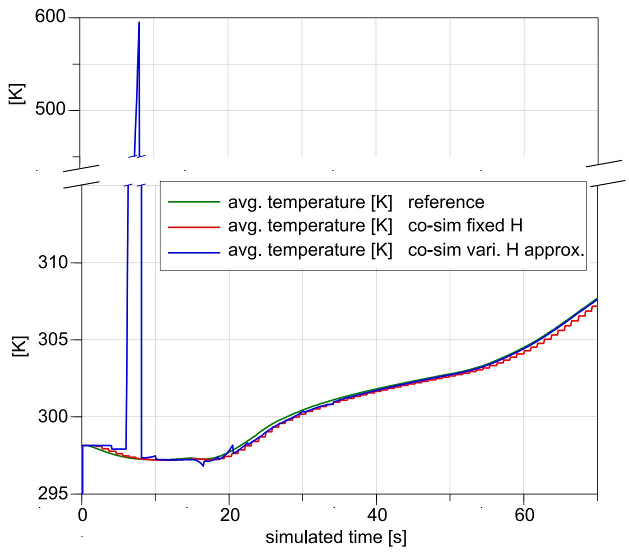


Figure 14: warming-up at cycle start / negative approximation effect.

The approximated curve is mostly more congruent to the reference. However, in the first part, the problems of applying higher order extrapolation together with to large step sizes is obvious.

This is one of the challenges in using a parallel and strictly monotone modular technique. Therefore, as mentioned in 4.1, the control bus has to be adopted for a quasi approximation order control together with further improved step size control methods.

6 Conclusions and Outlook

In this paper we presented a Modelica based modular approach for overall vehicle system simulation. The advantages of using co-simulation in this context are deduced and achieved computational speed-up results are shown. The modular approach consists of two parts: a modular multi-domain vehicle library and the adapted co-simulation framework MDPCosim [11]. The modular library allows configuring complete vehicles by assembling the needed subsystem models, which is also possible as co-simulation slave to be simulated in parallel. Additionally, it provides interface models that can be easily configured by the user to set up a co-simulation run. The implementation of different categorized coupling aspects is shown. In particular, a heuristic method for macro step size control that is used for the overall vehicle simulation is explained. As advantage its parameters were chosen according to the a priori known drive cycle. Though, there are many challenges, which have to be regarded to make modular simulation more applicable.

Thus, there is remaining work to be done. A preferable way is the adoption of FMI [10], once a reliable implementation also for the mentioned large multi-domain models is available (not the case at the beginning of the present work). The FMI standard provides a suitable set-up for the algorithms described above. Consequently an even more common use of the modular library approach will be feasible, also including more different tools.

Acknowledgements

We are grateful for the collaboration with TU München providing access to MDPCosim; Marcus Schulz, Cooperative State University Stuttgart, for fruitful discussions and TLK Thermo GmbH for kindly providing TISC [17]

References

[1] Arnold, M. Multi-Rate Time Integration for Large Scale Multibody System Models. IUTAM Symposium on Multiscale Problems in Multibody System Contacts, Stuttgart, Germany, 2006.

[2] Blochwitz, T. et. al. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. Proc. 8th International Modelica Conference. The Modelica Association. Dresden, Germany, 2011.

[3] Busch, M., Schweizer, B. Numerical Stability and Accuracy of Different Co-Simulation Techniques: Analytical Investigations Based on a 2-DOF Test-Model. The 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, Finland, 2010.

[4] Clauß, C., Arnold, M., Schierz, T., Bastian, J. Master zur Simulatorkopplung via FMI. ASIM-Konferenz STS/GMMS 2012, ISBN 978-3-901608-39-1, Wolfenbüttel, Germany, 2012.

[5] Dassault Systèmes, Dymola 2012, 2011 URL http://www.3ds.com/products/catia/portfolio/dymola

[6] DLR Institute of Robotics and Mechatronics, The Powertrain Library (Version 2.1.0), 2011 URL http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-5312/8907_read-16072/

[7] Dronka, S. Die Simulation gekoppelter Mehrkörper- und Hydraulikmodelle mit Erweiterung für Echtzeitsimulation. Dresden, Germany, PhD thesis, Technische Universität Dresden, 2004

[8] European Commission. NEDC. Consolidated Directive 70/220/EEC, 2006.

[9] Faure, C. et al. Methods for real-time simulation of Cyber-Physical Systems: application to automotive domain. Proc. 1st IEEE Workshop on Design, Modeling and Evaluation of Cyber Physical Systems, Istanbul, Turkey, 2011.

[10] FMI Specification 2.0 Beta 3, available for free from URL http://www.functional-mockup-interface.org/ (2.0 beta)

[11] Friedrich, M. Parallel Co-Simulation for Mechatronic Systems. München, Germany: PhD thesis, Technische Universität München, Institute of Applied Mechanics, 2011.

[12] Friedrich, M., Schneider, M., Ulbrich, H. A Parallel Co-Simulation for Mechatronic Systems. The 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, Finland, 2010.

[13] Johansson, B., Krus, P. Modelica in a Distributed Environment Using Transmission Line Modelling. Proc. Modelica Workshop 2000, Lund, Sweden, 2000.

[14] Kanth, D. Zur steifigkeits- und kopplungsbasierten Partitionierung mechatronischer Systeme. Stuttgart, Germany, PhD thesis, Univerity of Stuttgart, 2010

[15] Petridis, K., Klein, A., Beitelschmidt, M. Asynchronous method for the coupled simulation of mechatronic systems. Proceedings in Applied Mathematics and Mechanics, Bremen, Germany, 2008.

[16] Rumbolz, P., Baumann, G., Reuss, H-C. Messung der fahrzeuginternen Leistungsfluesse im Realverkehr. ATZ 05 2011, Germany, 2011

[17] TLK-Thermo GmbH, TISC, 2012, URL http://www.tlk-thermo.com.

[18] UNECE, WLTC v4, 2012, URL http://www.unece.org/.