Using BCVTB for Co-Simulation between Dymola and MATLAB for Multi-Domain Investigations of Production Plants

Irene Hafner[1], Matthias Rössler[2], Bernhard Heinzl[2], Andreas Körner[1],
Felix Breitenecker[1], Michael Landsiedl[3], Wolfgang Kastner[2]

[1)] Vienna University of Technology, Institute of Analysis and Scientific Computing
Wiedner Hauptstr. 8-10, 1040 Wien
[2)] Vienna University of Technology, Institute of Computer Aided Automation
Treitlstr. 3, 1040 Wien
[3)] dwh Simulation Services
Neustiftgasse 57-59, 1070 Wien

Abstract

This paper discusses the cooperative simulation of models implemented in Modelica, Simscape, Simulink and MATLAB for the aim of energy optimization in cutting factories. To simulate the thermal processes in production halls, the machines and the room itself have to be modelled in varying detail. To achieve a quite accurate comprehensive model, the individual machines and the room are modelled in different software and then simulated with the co-simulation tool BCVTB, which stands for *Building Controls Virtual Test Bed.* The communication between the individual models requires a lot of preparative work and as can be seen at the end of the paper, it works fine for a fixed communication time step but is not possible with a continuous synchronization for all given software. Still, the possibilities of co-simulation with BCVTB can be found sufficient for the needs of thermal processes which react very slowly and not in time steps of hugely differing dimensions respectively, but require a period of time which can easily be approximated small enough for a certain scenario.

Keywords: co-simulation; BCVTB; energy optimization; Dymola/Modelica

1 Motivation

Nowadays it has become more and more important to be able to simulate models with partial models of different complexity and differing requirements regarding solver algorithms, step sizes and other model-specific properties. To meet these requirements, models of such complexity are approached via co-simulation. Co-simulation stands for "Cooperative Simulation". One can tell from the name that the aim is to simulate separate models and let them communicate and synchronize to certain points in time given by an overall simulation which lets all partial models cooperate.

The aspects discussed in this paper are part of the INFO (Interdisziplinäre Forschung zur Energieoptimierung in Fertigungsbetrieben) project which is promoted by the Austrian Research Promotion Agency (FFG). Its aim is to optimize the energy consumption in cutting factories. Therefore it's necessary to simulate the thermal processes in production halls. Since all different machines in one production hall require individual modelling approaches, certain solvers and even different software, this problem is approached with co-simulation.

Via the Ptolemy-based co-simulation tool BCVTB (Building Controls Virtual Test Bed), a room model implemented in Modelica, machines implemented in Modelica, Simscape and Simulink as well as a MATLAB data model of the measured heat emission of a machine are co-simulated. Figure 1 gives an overview of the desired communication between the individual simulators.

2 Building Controls Virtual Test Bed

The Building Controls Virtual Test Bed was designed at the University of Berkely to allow the communication of the simulators Ptolemy, EnergyPlus, Dymola, Matlab, Simulink, Radiance and BACnet. BCVTB

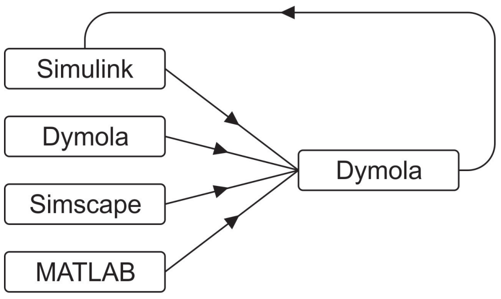
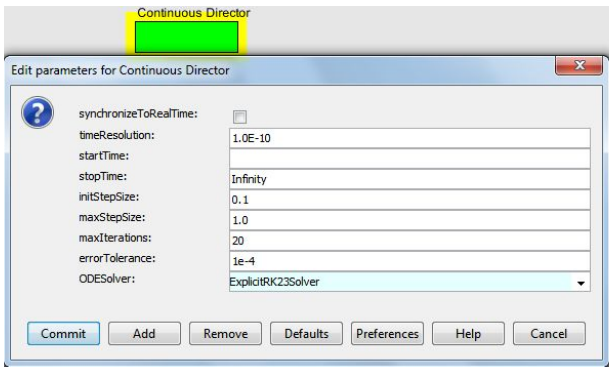


Figure 1: Overview of the Intended Communication between the Individual Simulators

resembles the Ptolemy interface but offers additional blocks (*actors*, as they're referred to in Ptolemy) and on the other hand lacks Ptolemy elements which are not necessary for the use of co-simulation, which BCVTB has been developed for. Though some of the Simulators would be able to interact without the BCVTB interface (like Dymola and Simulink), the use of different step sizes or even solver algorithms is only possible with co-simulation.
To control the synchronization of the individual simulators, BCVTB provides certain so-called *directors*.
The *Continuous Time Director* (CT) allows the user to choose a variable step solver (explicit RK23 or RK45) for the total simulation as well as setting solver options like the maximum step size or the error tolerance (see Fig. 2).

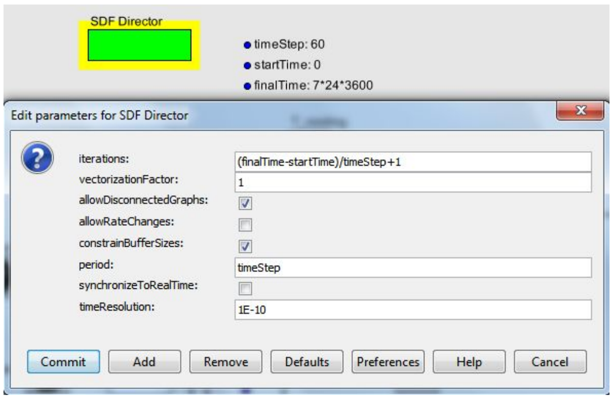


synchronizeToRealTime:	☐
timeResolution:	1.0E-10
startTime:	
stopTime:	Infinity
initStepSize:	0.1
maxStepSize:	1.0
maxIterations:	20
errorTolerance:	1e-4
ODESolver:	ExplicitRK23Solver

Figure 2: Continuous Time Director

If it's sufficient for a model to synchronize all partial models at predefined fixed time steps, the *Synchronous Data Flow* (SDF) director can be used. All properties of the SDF Director can be seen in Fig. 3:

From the BCVTB interface, the different simulators have to be accessed with *Simulator* actors. These actors establish the communication among the individual Simulators via BSD sockets, which also have been developed at the University of Berkeley and are used for inter-process communication (see [3] for

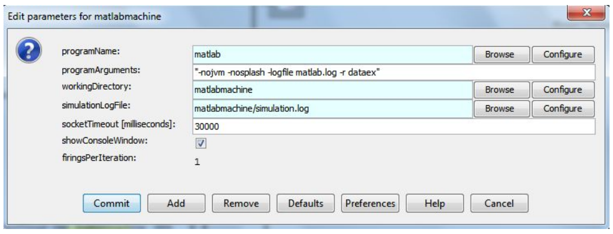


- timeStep: 60
- startTime: 0
- finalTime: 7*24*3600

iterations:	(finalTime-startTime)/timeStep+1
vectorizationFactor:	1
allowDisconnectedGraphs:	✓
allowRateChanges:	☐
constrainBufferSizes:	✓
period:	timeStep
synchronizeToRealTime:	☐
timeResolution:	1E-10

Figure 3: Synchronous Data Flow Director

further information).
All values needed by a simulator have to be connected to the input port, which allows multiple inputs; all values which the simulator returns to BCVTB at each synchronization time step can be accessed from the output port of the simulator actor. The options of the simulator actor (see Fig. 4 for a simulator actor accessing MATLAB) define the simulator to be called as well as options for the simulator, the execution file, the path where it can be found and a parameter *socketTimeout*. This parameter defines how many milliseconds BCVTB has to wait for the simulator to respond before canceling the simulation and returning an error. If a BCVTB model fails due to this *socket time out*-error, there is either an error in the partial model or it simply takes longer than the given socketTimeout to load and thus is not able to respond early enough. Hence it is important to choose an adequate amount of time for complex partial models.

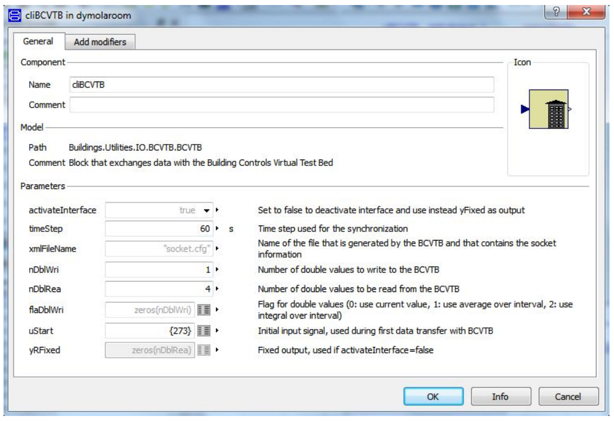


programName:	matlab
programArguments:	"-nojvm -nosplash -logfile matlab.log -r dataex"
workingDirectory:	matlabmachine
simulationLogFile:	matlabmachine/simulation.log
socketTimeout [milliseconds]:	30000
showConsoleWindow:	✓
firingsPerIteration:	1

Figure 4: Simulator Actor Accessing MATLAB

2.1 Communication between Dymola and BCVTB

To enable the communication of BCVTB with Dymola, the developers of BCVTB have implemented the Modelica Buildings Library which provides a BCVTB block (see Fig.5).



cliBCVTB in dymolaroom

Component

Name: cliBCVTB

Comment:

Icon

Model

Path: Buildings.Utilities.IO.BCVTB.BCVTB

Comment: Block that exchanges data with the Building Controls Virtual Test Bed

Parameters

activateInterface	true		Set to false to deactivate interface and use instead yFixed as output
timeStep	60	s	Time step used for the synchronization
xmlFileName	"socket.cfg"		Name of the file that is generated by the BCVTB and that contains the socket information
nDblWri	1		Number of double values to write to the BCVTB
nDblRea	4		Number of double values to be read from the BCVTB
flaDblWri	zeros(nDblWri)		Flag for double values (0: use current value, 1: use average over interval, 2: use integral over interval)
uStart	{273}		Initial input signal, used during first data transfer with BCVTB
yRFixed	zeros(nDblRea)		Fixed output, used if activateInterface=false

Figure 5: BCVTB Modelica Block enabling the Communication between Dymola and BCVTB

Inputs to the block are all values to be transferred from Dymola to BCVTB, outputs are all values needed from the BCVTB. In the block properties the time steps at which Dymola has to synchronize with BCVTB can be defined by setting the parameter *timeStep* to the desired value.
nDblWri defines the number of values Dymola returns to BCVTB and *nDblRea* stands for the number of values Dymola will receive from BCVTB at each synchronization time step. All data received from BCVTB is kept constant between the synchronizations.
The parameter *uStart* stands for the value which is returned to BCVTB at the very first synchronization.

2.2 Communication between MATLAB and BCVTB

In MATLAB, the first step necessary to enable the communication with BCVTB is to create a socket connection via

```
sockfd = establishClientSocket('socket.cfg');
```

Further the following values have to be exchanged with BCVTB at every desired time step by calling

```
[retVal, flaRea, simTimRea, dblValRea ] = ...
exchangeDoublesWithSocket(sockfd, flaWri, ...
length(u), simTimWri, dblValWri);
```

retVal, flaRea, simTimRea and *dblValRea* represent the values obtained from BCVTB which can now be used in the MATLAB function. MATLAB has to submit *sockfd, flaWri, length(u), simTimWri* and *dblValWri* to BCVTB. Before completely exiting Matlab, the socket is closed with

```
closeIPC(sockfd);
```

2.3 Communication between Simulink and BCVTB

For the communication with Simulink, BCVTB also offers a preimplemented block. Inputs are again all values from Simulink to be sent to BCVTB and outputs are the values Simulink needs from BCVTB. The underlying subsystems can be seen in Fig. 6.

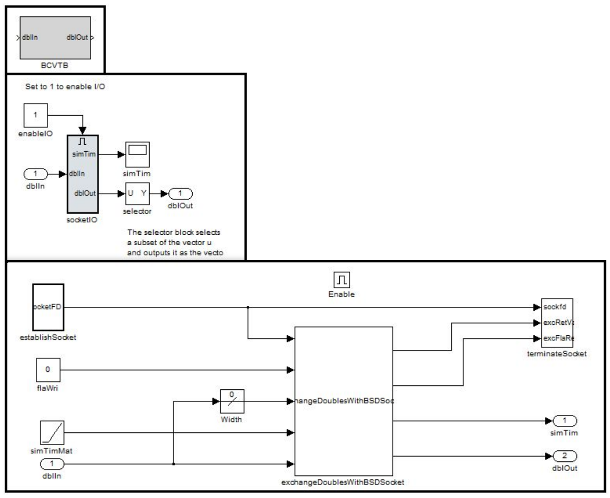


Figure 6: BCVTB Simulink Block enabling the Communication between Simulink and BCVTB

In contrary to the BCVTB block for Dymola, the time step for synchronization cannot simply be defined by a block parameter. For all preimplemented examples BCVTB offers, the time step of the Simulink solver is chosen fixed and equal to the BCVTB time step so there's no problem since the synchronization automatically takes place at the correct time.
To be able to benefit of one of the main advantages of co-simulation - the usage of different solvers and different step times - additional programming work has to be done. To fulfill this purpose, the BCVTB block is put in an *If Action Subsystem* which is activated only if the time step of the BCVTB director is crossed. In case of a SDF director, which means a constant time step, the maximum time step for the solver in Simulink is set to this constant and the time in Simulink modulo the BCVTB time step is compared in every Simulink time step. If the Simulink time crosses the BCVTB time step, the modulo value changes and after zero-crossing detection to evaluate the return value at the

desired time within a certain tolerance, the If Action Subsystem is activated and the exchange takes place (see also section 3.3 and Fig. 11). If a CT director is used in BCVTB, the time step varies and can't be foreseen, so the time in BCVTB is compared to the time in Simulink and at every time step iterated this way the subsystem is activated by sending a discrete impulse at these points in time.

3 Model Description

The model described in this paper uses Dymola/Modelica, MATLAB, Simscape and Simulink apart from the main model in BCVTB. It's purpose is to demonstrate the thermal processes in a production hall. The hall itself is modelled in Modelica. The different machines are implemented in Modelica, Simscape and as simple data model in MATLAB. To obtain a bearable room temperature for human workers which possibly enter the hall, a controller is implemented in Simulink. The waste heat of the machines and the cooling heat from the controller are transferred to the room model at each synchronization time step via the BCVTB interface. The BCVTB model can be seen in Fig. 7:

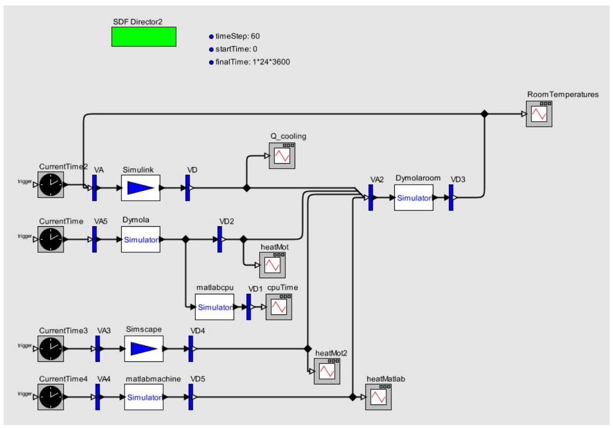
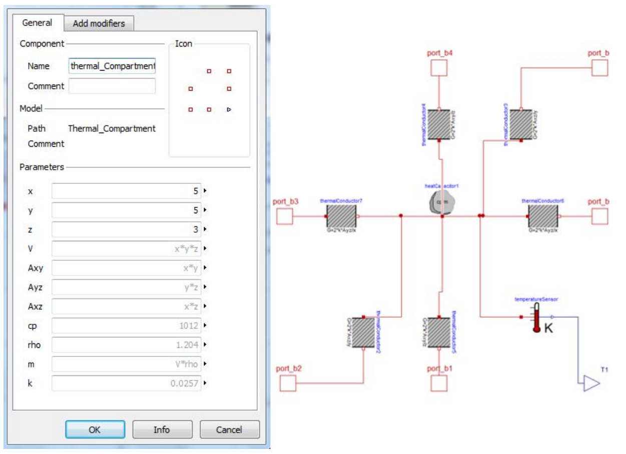


Figure 7: Model for Synchronization in BCVTB

The model is supervised by a SDF Director which demands so-called *firing* of the individual simulators every 60 seconds. The stop time can be defined by the parameter *finalTime* in seconds. Since the machines don't need any values from BCVTB apart from the time, they receive the current simulation time only. The simulator actors *Simscape*, *Dymola* and *Matlab*, which enable the communication with the respective machine models, return the heat outputs which are then sent to the room model called by the *Dymolaroom* simulator actor. The output of the *Dymolaroom* simulator is a temperature measured in one of the compartments of the room (see section 3.1) which is then sent to the controller represented by the *Simulink* simulator actor. The output of the controller is again sent to the room model and treated as a heat source. To obtain a better documentation of the simulation process, the model also communicates with a MATLAB function which stores the elapsed cpu time to an excel-file and additionally sends it to BCVTB for immediate visualization. The cpu time taken by the communication and execution of the m-file realizing the cpu documentation can be regarded negligible in comparison to those of the other partial models, which are way more complex and therefore expensive.

3.1 Room Model in Dymola/Modelica

The model of the production hall is realized as a compartment model. Each thermal compartment basically represents a cuboid with a certain heat capacity and conduction at the surfaces. The graphical model and all parameters of a thermal compartment can be seen in Fig. 8.



Component

Name	thermal_Compartment
Comment	

Model

Path	Thermal_Compartment
Comment	

Parameters

x	5
y	5
z	3
V	x*y*z
Axy	x*y
Ayz	y*z
Axz	x*z
cp	1012
rho	1.204
m	V*rho
k	0.0257

Figure 8: Model for a Thermal Room Compartment Implemented in Modelica - Parameters and Graph

The model of the production hall consists of six thermal compartments at $5 \times 5 \times 3$ m^3 each (see Fig. 9).

The heat emitted by the machines and the regulation heat flow from the controller can be accessed at the output port of the BCVTB block and are transferred as prescribed heat flow to the compartments where the machines are found in the production hall. The temperature measured in one of the compartments is returned to the BCVTB model and further to the controller.

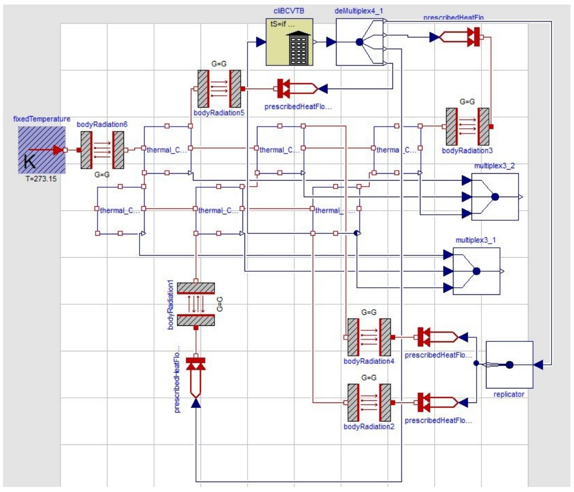


Figure 9: Model Graph for a Machine Hall Implemented in Modelica

3.2 Machine in Dymola/Modelica

Since the main focus lies on coupling the individual models, the machines involved are held rather simple. The machine implemented in Modelica consists basically of a DC motor. Since version 3.2 of the Modelica standard library, the heat dissipated in an electrical circuit can be used in a thermal system by activating an optional heat port at certain components. The electrical energy lost at the resistor of the model is converted into thermal energy, which is measured as heat flow from the resistor heat port to the room represented by a heat capacitor.

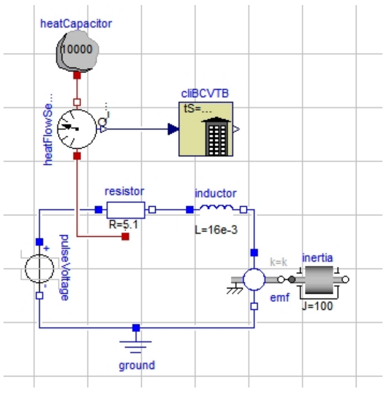


Figure 10: Model Graph of a DC Motor Implemented in Modelica

To simulate different loads by machines which don't run 24 hours a day, the voltage applied to the voltage source is chosen as pulsating with 320V at working hours and 0V at night.

3.3 Machine in Simscape

The machine in Simscape is represented by a motor similar to the one implemented in Modelica. To use the waste heat emitted at the resistor, the rated power is manually calculated from the voltage drop and sent to a thermal system as heat flow. Again, the working hours of the machine are set via the voltage source.

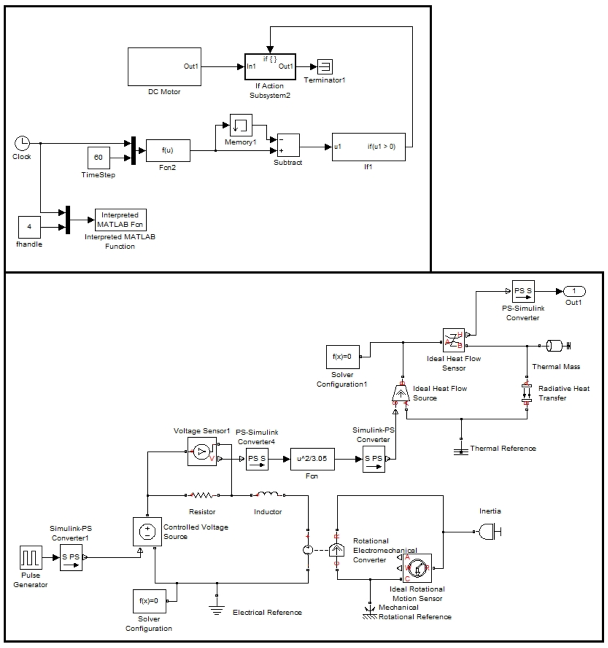


Figure 11: Model of a DC Motor Implemented in Simscape

3.4 Controller in Simulink

The temperature control is realized rather simply. The model gets the temperature measured in one of the *Thermal Compartments* of the Dymola room model and compares it to the desired room temperature. If the room is more than one Kelvin too warm (cold resp.), the control returns minus (plus resp.) 100W heat flow to two room compartments.

3.5 Data Model in MATLAB

The data model in MATLAB is rather simple. The heat emission of a machine over one day is read out of an excel-file and returned to the BCVTB model and further the Dymola room model at each time step.

4 Simulation Results

The model is simulated for one day to show the behaviour of the model for this time span. At 8 a.m.

all machines start working and the room temperature (measured in one compartment for the cooling system) which can be seen in Fig. 12 begins to rise. As soon as the room temperature reaches 294.15 K, the control starts cooling.

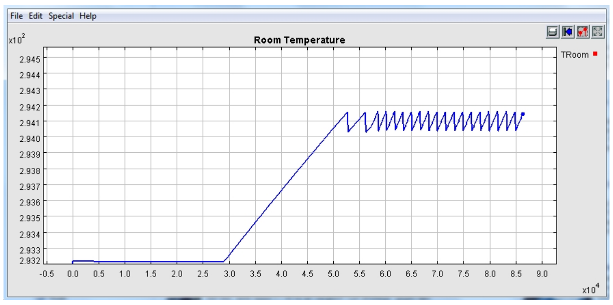


Figure 12: Progress of the Temperature in One Compartment

The temperature graph of all compartments is shown in Fig. 13. The temperature measured in the compartment shown above corresponds to the green one in Fig. 13. One can easily see that the compartments containing machines (blue, red and pink) respond much more quickly than the others.

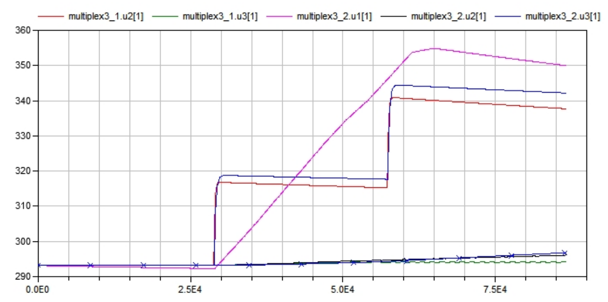


Figure 13: Progress of the Room Temperature in All Compartments

The heat emitted by the individual machines is demonstrated in Fig. 14. Turning down the machines implemented in Simscape and Dymola causes a step response similar to the one caused by switching them on. The measured heat emission transferred to BCVTB with MATLAB shows a rather permanent emission during working hours.

A very important result of the simulation is the documentation of the individual step sizes. Figure 15 shows the solver time steps between two synchronization references of the simulation. For the simulation of the machine and the room model in Dymola, the *Dassl* solver is used. The machine model in Simscape is simulated with ode15s, a variable step solver for stiff systems. Since the control in Simulink only deals with

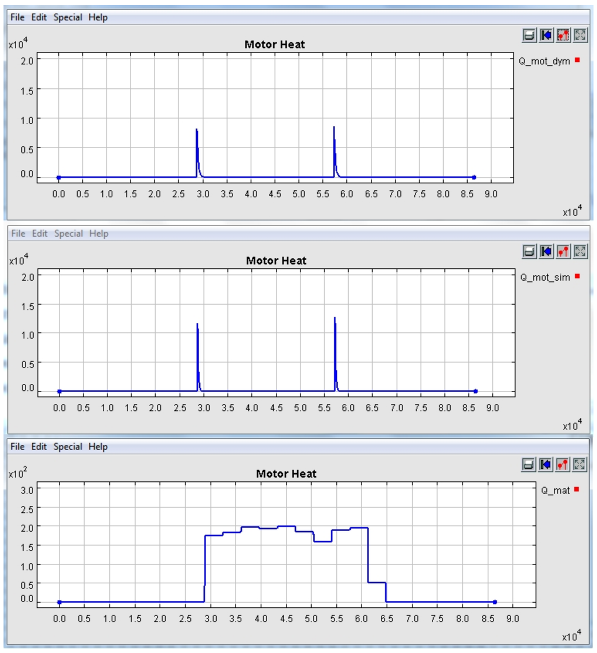


Figure 14: Heat Emitted to the Room by the Machines (Simulated in Matlab, Simscape and Dymola) over One Day

discrete states, *variable step discrete* is chosen for the simulation. The fixed time step for synchronization in the BCVTB model is set to 60 seconds.

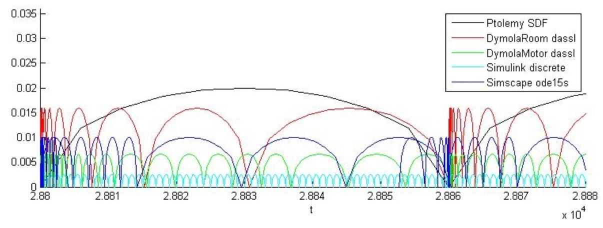


Figure 15: Plot of the Different Solver Time Steps between Two Synchronization References

One of the most important advantages of co-simulation becomes very obvious in this plot: The time steps in the machines, which also differ clearly from each other, are significantly smaller than the time steps of the room in Dymola. This makes perfect sense due to the fact that for the machines systems of equations out of electrical and mechanical circuits have to be solved. Since electrical and mechanical components interact much faster than thermal ones, the underlying systems require accordingly smaller time steps. Fig. 16 shows the different solver steps made in a very small interval around a synchronization reference. This clearly points out the redundant steps made by the Simscape solver to iterate the accurate time to communicate with BCVTB.

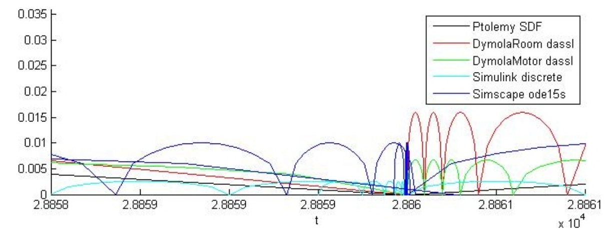


Figure 16: Plot of the Different Time Steps at a Synchronization Reference

Finally, the progression of the room temperature during the simulation of the same model over three days is shown in Fig. 17. Of course the very simple way of cooling can't prevent the temperature from boundless rising in compartments with machines.

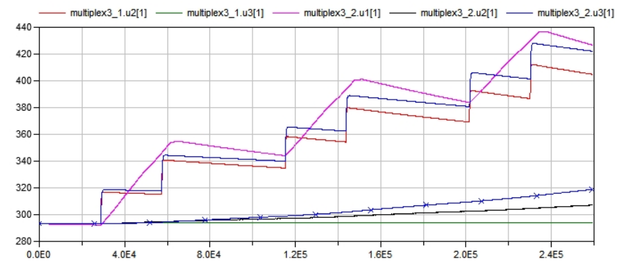


Figure 17: Progress of the Room Temperature in All Compartments over Three Days

5 Conclusion

At the first impression, BCVTB seems like a quite advanced tool to enable cooperative simulation in a rather easy way. It's true that after successfully installing compatible releases of every software required and modifying the given synchronization tools to even allow differing solver time steps, coupling of several partial models in a BCVTB model can be realized without huge modifications.
On the other hand it's not possible to let models communicate with BCVTB at variable time steps with the given BCVTB blocks. In Simulink the communication at time steps which aren't known before can be realized by activating a subsystem containing the BCVTB block. To also achieve this in Dymola, most parts of the given BCVTB block would have to be rewritten.
What's more is that between two synchronization time steps all values from BCVTB are extrapolated uniformly so depending on the actual graph and the synchronization step size, the single errors could sum up to an amount which causes the model to fail any validation. For the described use in thermal systems which react very slowly, co-simulation with BCVTB might be considered sufficiently accurate, but to achieve a valid co-simulation which requires precise or at least reliable approximations with arbitrarily small errors, other possibilities of co-simulation will have to be considered.

6 Outlook

In the course of this project, the limits of co-simulation with BCVTB will be further explored considering the complexity of individual models as well as the amount of partial models involved. Additionally, a room model in the building energy simulation program EnergyPlus will be implemented and further compared to the room model in Dymola to depict the advantages of the different software regarding co-simulation with BCVTB as well as the behaviour as thermal model for a production hall.

Acknowledgement

This work was partially supported by the Klima- und Energiefonds of the Austrian Federal Government within the Neue Energien 2020 program (FFG Project No. 825384).

References

[1] Wetter M. Building Controls Virtual Test Bed User Manual Version 1.1.0. Berkeley, California: Building Technologies Department, Environmental Energy Technologies Division, Lawrence Berkeley National Laboratory, 2012. Available from:
http://simulationresearch.lbl.gov/bcvtb

[2] Heinzl B., Rössler M. et al.. Studies on Multi-Domain Modelling and Thermal Coupling of a Machine Tool. Winterthur, Switzerland: ASIM 21. Symposium Simulationstechnik, 2011 ISBN: 978-3-905745-44-3

[3] Stevens W.R., Fenner B., Rudoff A.M.. Unix Network Programming: The Sockets Networking API, Vol 1. Addison-Wesley Professional, 2004 ISBN: 9780131411555

[4] Modelica Buildings Library V1.1. Available From:
http://simulationresearch.lbl.gov/modelica