A VARIABLE DETAIL MODEL SIMULATION METHODOLOGY FOR CYBER-PHYSICAL SYSTEMS

Ir. T.G. Broenink
Dr. Ir. J.F. Broenink
Robotics and Mechatronics
University of Twente
Enschede, Netherlands
Email: t.g.broenink@utwente.nl

KEYWORDS

cyber-physical systems, model driven design, port-based modeling, variable detail models

ABSTRACT

This publication is about simulating cyber-physical models with varying levels of detail, how to structure the models to make this possible and how to design the sub-models required. A structured method is posed to select which signals of the model to use for the structure and how to determine the different parts of the models to abstract away when varying the detail of sub-models. This method is applied to two examples based on a lab setup available. One example shows hard limits due to encoder processing, the other example shows the effects of simplified dynamic models. The method is effective in both examples and shows a clear trade-off between accuracy and performance.

INTRODUCTION

Design of cyber physical systems can benefit greatly from model based design (Broenink et al. 2016). Not only does model based design allow for solutions to common design conflicts (Ni and Broenink 2014), the current state of the art allows the partial synthesis of software and hardware description (Kuipers et al. 2016). However larger and especially hybrid systems are difficult to model and simulate (Derler et al. 2012), requiring complex simulations and more computations. This is why it is important to make models that are competent enough for the design challenges of cyber-physical systems (Lee 2008). Modeling and simulation of cyber-physical systems requires several decisions during the modeling process. Decisions on what to model, how detailed this should be modeled, and how to exactly implement it. Depending on the specific problem statement, different levels of detail are required to find solutions. However, more detailed models require more computations to solve, increasing the time required to simulate. Thus a model needs only as much details as required for the problem. It is beneficial for modeling and simulation to be able to select the level of detail at which a system is modeled and simulated, assuming that a method exists to simulate these different levels of detail in a model. This method can either be a versatile simulation package, or a way to co-simulate multiple different simulators. (Bastian et al. 2011; Nägele and Hooman 2017) This allows for a trade off between model accuracy and simulation speed. The same trade off that is made by taking a larger time-step when simulating a model. It also allows for a trade off between development time and accuracy. If it is easy to change the detail level of a sub-model it is possible to model a simpler model first and only creating the more complex model when necessary.

A model can be used to answer different questions about the final design. Depending on the question posed, some behavior of the model is more important then others. This means that some parts of the model need to be modeled and simulated in more detail than others. In order to do this, the level of detail of a model should be changeable per part. To be able to change the level of detail on different parts of the model, it is required that the structure of the model is able to accommodate this change, allowing parts of the model to be replaced. Thus it should be possible to create variable detail simulation with correctly structured modeled.

A method to structure these model is proposed here, a way to allow the varying of the implementation of the different parts of the model with a minimal change in structure. Furthermore a guideline is proposed for the sub-models used, a guideline on how to model these sub models and how to easily change the level of detail. This method is illustrated based on two different examples. The first example is based on a signal processing case. The second example is based on a dynamic modeling case.

METHODOLOGY

In order to simulate a system with varying amount of detail, two steps are needed. The first one is to structure the model in such a way that it is possible to change the detail of one element in such a way that the rest of the model is unaffected. The second step is to model the resulting sub models in different levels of detail, combined with analyzing the limits of certain approximations.

Proceedings 32nd European Conference on Modelling and Simulation ©ECMS Lars Nolle, Alexandra Burger, Christoph Tholen, Jens Werner, Jens Wellhausen (Editors)
ISBN: 978-0-9932440-6-3/ ISBN: 978-0-9932440-7-0 (CD)

Model structure

The structure of a model is defined as the collection of sub-models composing the model, and the connections, or interfaces, defined between these sub-models. These interface consist of definitions of uni- and bi-directional signals. Included in this definition are not only direction, but also data-type and possibly update rate. When changing the details of one of the sub-models these interfaces are not allowed to change, thus all levels of detail will have to have the same interface. However it is possible to implement new sub-structures within an sub-model.

This requirement, to keep the interface unchanged, means that the interfaces used for the model structure will have to be based on signals that remain the same, irrelevant of the level of detail. As the encoding of information is very likely to change based on the different levels of detail used, signals that represent unencoded values should be used. These signals usually correspond to physical signals available in the system, instead of the IO signals of system parts. These signals can include:

1. Physical quantities, e.g. velocities, voltages, forces.
2. System states, e.g. started/stopped, switch pressed/released.
3. Representations of Physical quantities, e.g. set-points and sensor values.

This requires a split between input/output models and the actual core of the sub-model. All models should be transformations between physical, or detail-invariant signals. This is in contrast to conventional model boundaries, where physical system boundaries are often used, e.g. output pins, or communication interfaces.

Based on a simple electro-mechanical system, an example is given. Take the following system: A controller controlling an electromotor, applying a voltage to reach a desired angular position. Classically, this system can be split in a controller and a plant, giving the system of Fig.1. In the ideal case the controller can directly apply a voltage V to the motor and is able to directly read the angle ϕ. This is all that is required to determine the control laws and the plant dynamics. However the signals send between the controller and the motor are not directly the voltage V and the angle ϕ, even if these signals are supposed to represent those values. The controller needs some way to convert internal signals into an output voltage for the motor. The same goes for the angle of the motor, it has to be converted in some way to a signal that the controller understands. For the output voltage this is done by some form of motor driver, which has a signal input, and will supply a voltage based on this input, as seen in Fig 3. For the angle of the motor some sort of sensor is required to sense the physical angle ϕ and to turn it into a signal representing this angle in some way.

In the current configuration this is done either inside the Motor model or the Controller model. Depending on the actual encoding used for these signals this might be a complicated process.

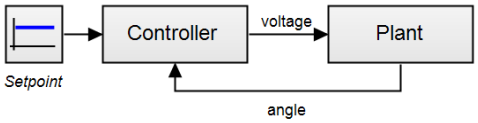


Figure 1: System for controlling a motor

When the detail invariant signals in this setup are identified, 5 signals are found:

1. The voltage on the motor (V)
2. The angle of the motor (ϕ)
3. The voltage signal as represented in the controller.
4. The angle signal as represented in the controller.
5. The set-point of the controller.

The conversion between these signals depends strongly on the implementation of the specific models. It can be as simple as assuming that the angle signal in the controller is the angle of the motor, or can be as complicated as a fully modeled PWM driver for the motor.

Based on these signals, a second model structure is created, using interfaces as defined by these 5 signals. This results in Fig.2. This structure allows the changing of the level of detail of one of the sub-models, without influencing the other sub-models.

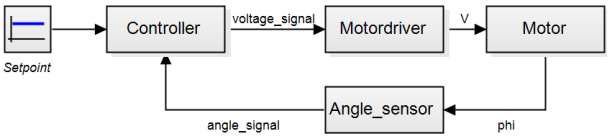


Figure 2: System for controlling a motor, with interfaces based on detail invariant signals

Variable detail

To model the sub-models with different levels of detail, it is important to analyze the impact of different parts of the sub-model on the performance. It is possible to reduce the detail the model using mathematical reduction techniques (Antoulas et al. 2001; Benner et al. 2013), but the focus of this method is on understanding the different parts of the model and deciding which to include in the final design, so that there are understandable effects that can be neglected or included. The interface of the sub-models was already determined in the previous step. Thus it is only

necessary to change the implementation of the relations between these signals. The sub-model should be modeled using a behavioral (Willems 2007) or port-based (Breedveld 2004) modeling technique, i.e. a technique where the different physical phenomena are visible as distinct elements of the model. This includes models like block-diagrams, bond graphs, or sub-structures. Representations in single equations e.g transfer functions or state-space representations are less suited for this purpose.

As the different physical effects are represented separately in the model, it should be possible to identify their individual effects. It is then possible to model what would happen if these effects were left out of the model. These effects can then be classified into three distinct categories:

1. Effects that are essential to a model's functioning.
2. Effects that are essential for a certain region of inputs/states (hard limits).
3. Effects that degrade overall model accuracy when left out (soft limits).

The first effects can never be abstracted away, as this would compromise model functionality completely. The second and third effect can be left out if the model is only used in certain regions. An example of these three effects can be made based on an electromotor, shown in Fig 3. The different effects of this model can be classified as:

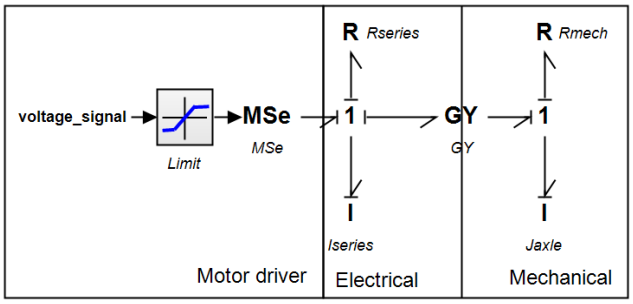


Figure 3: Port-based motor model using bond graphs

- Essential
  - The signal to electrical conversion of the MSe.
  - The transduction of the electromotor, represented as the gyrator (Gy).
- Hard limits
  - The limitation of the motor driver, this is can be left out as long as the input signal never reaches this limit.
- Soft limits
  - The Series inductance ($Iseries$) and moment of inertia ($Jaxle$), change the dynamic behavior of the motor, when left out.
  - The Electrical resistance ($Rseries$) changes the start-up behavior of the motor, and the maximum torque, when left out.
  - The Mechanical resistance ($Rmech$) reduces model accuracy at higher speeds when left out.

These classifications can then be used to create multiple models at different levels of detail. For example:

- A basic model only containing the essential effects.
- A model including the limit.
- A model for the detailed dynamic behavior including all the parasitic elements.

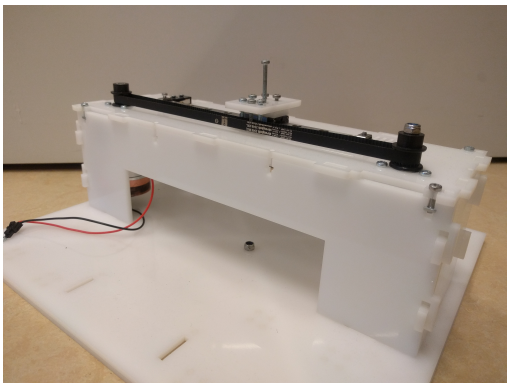
Which can all be used for different problem statements.

Method

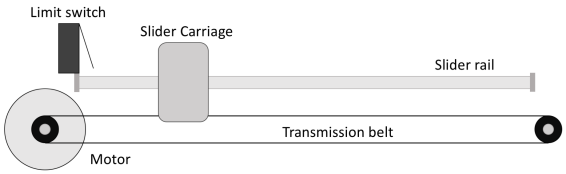
To summarize, to create a structure suitable for these detail variations, the following steps are performed.

1. Identify detail invariant signals, including but not limited to:
   (a) Physical quantities, e.g. velocities, voltages, forces.
   (b) System states, e.g. started/stopped, switch pressed/released.
   (c) Representations of Physical quantities, e.g. setpoints and sensor values.
2. Define interface and model structure based on detail invariant signals
3. Model sub-models based on behavioral or port-based modeling principles
4. Identify region of validity of simplifications of model effects, classify these effects as:
   (a) Essential
   (b) Hard limit
   (c) Soft limit
5. Create models based on required regions of validity.

It is assumed that the created model(s) and sub-model(s) are validated and tested, as part of the creation of these models.



(a)



(b)

Figure 4: The test setup as inspiration for the examples (a), and a schematic overview of the structure (b)

TEST SETUP

To further demonstrate the use of this approach, two examples are provided. Both examples are based on a test setup available in our lab. A linear motion slider setup, shown in Fig 4. This test setup is patterned of a common motion used in mechatronic systems. A motor moves a linear slider via a transmission belt. The angle of the motor is encoded using a magnetic encoder, which gives the absolute motor angle. The motor is driven with a PWM signal. Identifying the detail invariant signals results in the same signals as mentioned en the methodology (Figure 2).

As the motor encoder gives an absolute motor angle from 0 to 2π rad, it is not possible to discern multiple rotations. However with some software processing, the actual rotation can be calculated. The dynamic model of the slider consists of three parts:

1. The dynamics of the electromotor.
2. The dynamics of the transmission belt.
3. The dynamics of the slider carriage.

Modeling this dynamic behavior gives the graph of Fig.8 The total model of this system can then be constructed equivalently to Fig.2.

ENCODER SIMULATION

As mentioned, the magnetic encoder present on the motor measures the absolute angle of the motor axis. The expression for the angle output from the sensor is:

$$\phi_{sensor} = \phi_{motor} \mod 2\pi \tag{1}$$

This value is then sampled by the controller to be processed back into the actual angle. In order to process this sensor angle back into the true angle of the motor the following algorithm is used, presented in pseudo-code:

Listing 1: Processing for angle sensor

```
angle_difference = angle - previous(angle)
if angle_difference < -pi then
        rotations +=1
if angle_difference > pi then
        rotations -=1
output = angle + rotations * 2 * pi
```

With this processing the true angle can be reconstructed, assuming that equation 2 holds for the angular velocity of the motor.

$$|\frac{\dot{\phi}}{f_{sample}}| < \pi \tag{2}$$

Whenever the angular velocity of the motor exceeds this limit, the processing interprets the motion as going in the other direction and with a different magnitude. The effect of this processing can be seen in the graph of Fig 5. It can be seen that after the maximum velocity ($\omega_{max} = 100\pi \mathrm{rad/s}$) is reached that the processing reports a velocity of $\omega = -\omega_{max}$. This effect can be likened to the aliasing that happens with digital sampling.

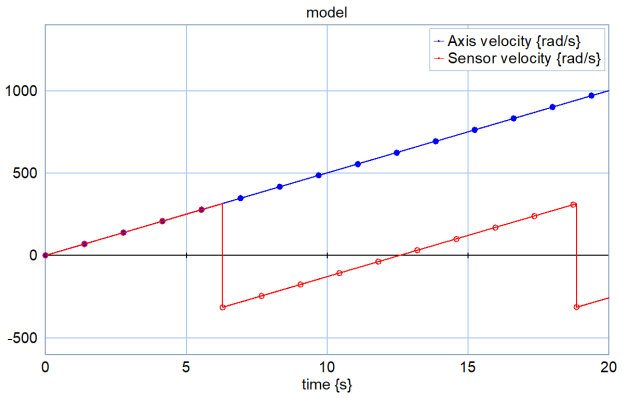
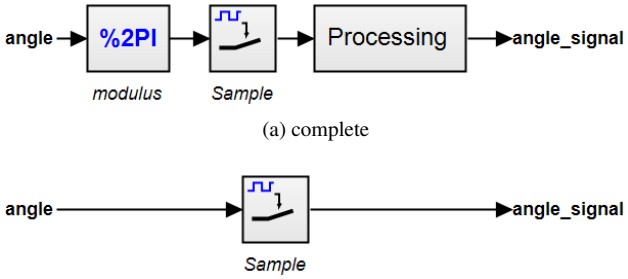


Figure 5: Effect of sensor aliasing when supplied with an ever increasing velocity

As the angle sensor is a sub-model with the actual angle as input and the angle signal as output, different implementations can be made. The first one is the full implementation. Using equation 1 as sensor model, sampling this, and processing it with the pseudo-code given. The second implementation is simplified up to the point of directly



(a) complete

(b) simple

Figure 6: A overview of both sensor models. The processing algorithm is shown in Listing 1

Model	time	speedup
simple	403ms	12%
complex	452ms	0%

Table 1: Simulation time and speed for a 1000 s simulation at 100 Hz and 100 rad/s

sampling the actual angle. An overview of both processing methods is given in Fig 6.

Both implementation are simulated using 20-sim. The system is given a constant velocity set-point of 100 rad/s and a controller-frequency of 100 Hz. The resulting simulations and speeds versus real time are shown in table 1. From Fig 5 and Table 1 it can be concluded that the simple implementation of the sensor can result in a speed increase when the model remains within the maximum angular velocity supported by the processing algorithm. When the model is used outside of this valid region, the simple model is no longer valid. To show the failure mode of the encoder aliasing, a simulation is done of a step to 50 rad. The result of this simulation is shown in Fig 7.

While it is expected that the angle_signal follows the angle exactly, the angular velocity becomes too large in this example, thus the angle processing cannot process correctly. The effect, which was predicted using Fig 5, is that the angle estimation goes in the opposite direction of the actual angle. This failure is especially catastrophic as the controller tries to correct for the deviation in processed angle by accelerating further, thus increasing the problem. This results in the actual angle overshooting the 50 rad.

This problem could be prevented using multiple methods. One of these methods would be to increase the controller frequency, or at least increase the sampling frequency of the angle sensor and sub-sample this signal. Another method of preventing this would be to design the system in such a way that this velocity is never required, either by limiting the controller, or changing the aggressiveness of the control.

Even when a system is not designed to be able to reach these maximum velocities, it is still important to keep in mind that this limit exists. An example of this limit unexpectedly applying was during a student project where a system with a motor loaded with a wheel could never reach these velocities. However, when the system was tested without the wheel to analyze the motor behavior, this limit was exceeded. This resulted in unexpected measurements for the students participating in this project.

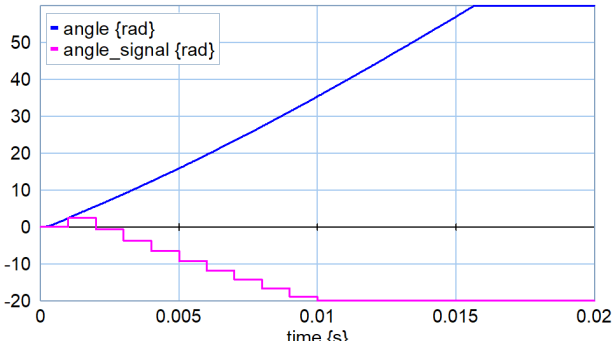


Figure 7: Breakdown of the angle sensor processing, the angle_signal goes negative, while the actual angle keeps increasing. Controlled at 100 Hz

DYNAMIC MODEL

The dynamic behavior of the plant can be modeled in multiple layers of detail. The dynamic behavior can be roughly divided into three segments:

- The motor
- The belt transmission
- The slider

A model containing these three parts is shown in Fig 8. This model is based on bond graphs, each half arrow representing the two conjugate power variables e.g. voltage and current, velocity and force, or torque and angular velocity. The

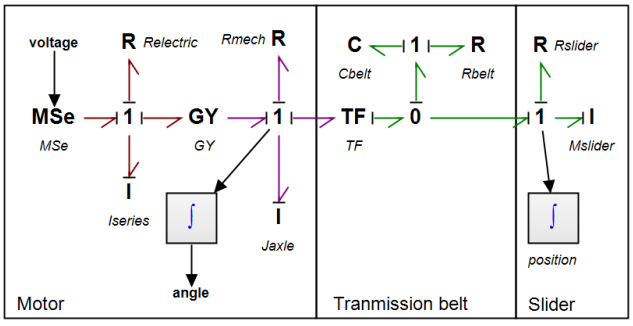


Figure 8: Full dynamic model of slider plant

model of the plant can be simplified. The elasticity of the belt introduces some oscillations during movement. Thus if the use of the model is to only estimate motion and position, instead of describing the full motion accurately it can be simplified by assuming the transmission belt to be ideal.

This is for example the case if a homing procedure of the controller has to be tested. When the transmission belt is assumed to be ideal it is also possible to combine both the rotational inertia of the motor with the mass of the slider, thus resulting in a system that is only second order, instead of fourth order. The resulting simplification is shown in Fig 9.

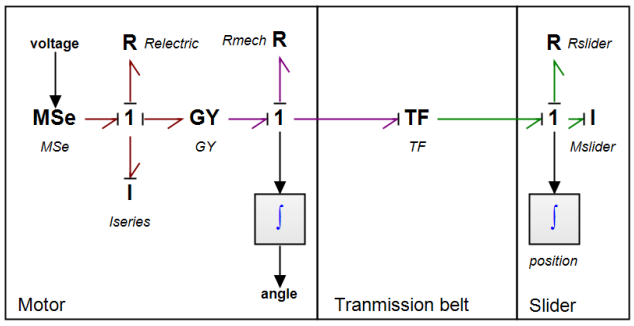


Figure 9: Simplified dynamic model of the slider

The results of these models can be compared. A step is applied to both models to move to 10 cm. The results of these motions are shown in Fig 10. It can be seen here that the final motion of both models is the same. However the oscillations at the beginning of the motion are not present in the simplified model. This same difference can be seen from the bode plots of both systems. These are shown in Fig 11. Here it can be seen that the higher frequencies of this model are not the same. Thus this simplification can only be used for low frequencies.

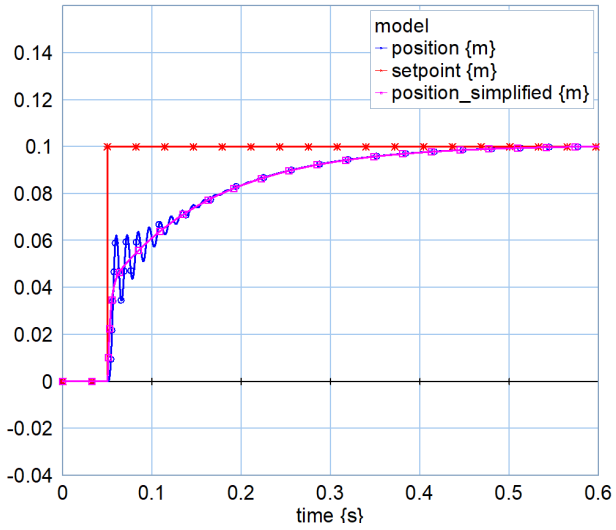
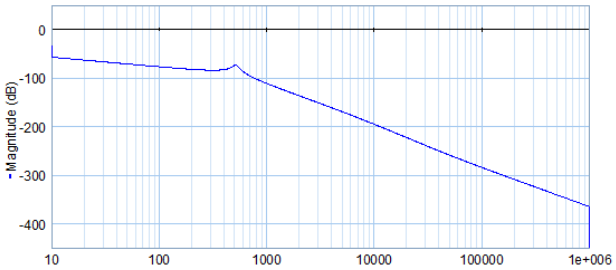
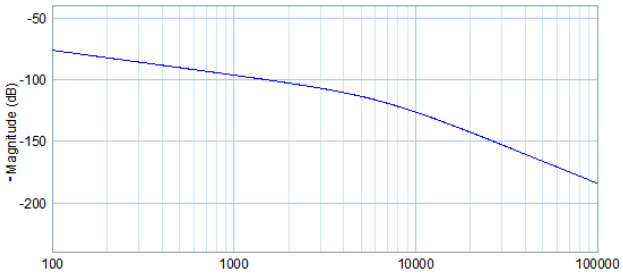


Figure 10: Simulation of (simplified)-plant model, controller frequency 100Hz

This lack of oscillation can be clearly seen in a speed comparison of both models. Both models are simulated for 100 of these steps. A controller frequency of 100Hz is used. The resulting simulations and speeds versus real time are found in table 2. Based on this data it is concluded that the simple model is suitable for either slow movements, or



(a) complete



(b) simple

Figure 11: The bode plot of both plant implementations

Model	time	speedup
simple	4.733s	16%
complex	5.484s	0%

Table 2: Simulation time and speed for a 100 steps of 10 cm. Simulated at a 100 Hzcontroller frequency

situations in which the exact motion of the slider is less relevant. One example of this might be to test sequence control in the control system of the setup. This does not require exact motion, and should thus be modeled with the simpler plant model for increased performance. Faster motions, or situations where the exact position of the slider is relevant must be simulated with the more complex model. One example situation would be to evaluate the performance of the system when trying to reach a certain position.

CONCLUSION

Based on the previous examples it can be concluded that it is possible to do variable detail simulations on models, given that these models are structured in a correct manner. The methodology proposed in section 3 is successfully applied to two examples. This shows that the methodology is both applicable to hard-limits and soft-limits. The varying detail results in a speed increase for the simpler models. This effect could be much more pronounced in larger systems, where the differences in detail are bigger and in which more sub-models are present.

The next step in this research is to validate this methodology for larger and more complex systems. This will give better insight into the impact on larger systems. In order to maximize the efficiency of the multiple methods of detail, it is useful to automatically evaluate simulations based

on the limits of the sub-models. This allows switching to a more complex model if so required. This validation could be done after a simulation, but could also be implemented live, thus allowing an adaptable simulation.

REFERENCES

Antoulas, A. C., Sorensen, D. C., and Gugercin, S. 2001. "A survey of model reduction methods for large-scale systems". *Contemporary mathematics*, 280:193–220.

Bastian, J., Clau, C., Wolf, S., and Schneider, P. 2011. "Master for co-simulation using fmi". In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*, number 63, pages 115–120. Linkping University Electronic Press; Linkpings universitet.

Benner, P., Gugercin, S., and Willcox, K. 2013. "A survey of model reduction methods for parametric systems".

Breedveld, P. C. 2004. "Port-based modeling of mechatronic systems". *Mathematics and Computers in Simulation*, 66(2-3):99–128.

Broenink, J. F., Vos, P.-J. D., Lu, Z., and Bezemer, M. M. 2016. "A co-design approach for embedded control software of cyber-physical systems". In *System of Systems Engineering Conference (SoSE), 2016 11th*, pages 1–5. IEEE.

Derler, P., Lee, E. A., and Vincentelli, A. S. 2012. "Modeling cyber–physical systems". *Proceedings of the IEEE*, 100(1):13–28.

Kuipers, F. P., Wester, R., Kuper, J., and Broenink, J. F. 2016. "Mapping CSP Models for Embedded Control Software to Hardware Using Clash". In *Communicating Process Architectures 2016, Copenhagen, DK*, Concurrent System Engineering Series, pages 133 – 150, Engeland. Open Channel Publishing Ltd. cpa bibtex: kuipers2016.

Lee, E. A. 2008. "Cyber Physical Systems: Design Challenges". pages 363–369. IEEE.

Nägele, T. and Hooman, J. 2017. "Co-simulation of cyber-physical systems using HLA". In *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*, pages 1–6. IEEE.

Ni, Y. and Broenink, J. F. 2014. "A co-modelling method for solving incompatibilities during co-design of mechatronic devices". *Advanced Engineering Informatics*, 28(3):232–240.

Willems, J. 2007. "The Behavioral Approach to Open and Interconnected Systems". *IEEE Control Systems Magazine*, 27(6):46–99.

AUTHOR BIOGRAPHIES

Tim Broenink (MSc 2016) is a PhD student at the Robotics and Mechatronics Lab of EE at the University of Twente. He is working on a project for a NWO-TTW perspective program regarding cyber physical systems. His track within this project relates to robust motion control for cyber-physical systems. His research interest includes behavior driven development of hardware and software, automated testing, and co-simulation.

Jan Broenink (MSc 1984; PhD 1990) is Associate Professor in Embedded Control Systems at the Robotics and Mechatronics Lab of EE at the University of Twente. His current research interests are on cyber-physical systems, embedded control systems (realization of control schemes on mostly networked computers) and software architectures for robotics. For that, he is interested in: model-driven design and meta-modeling of robot software architectures; designing software tools including (co)-simulation; concurrent and systems engineering. Since July 2017 he is chairman of the Robotics and Mechatronics group, together with prof. Stefano Stramigioli, who is Scientific Director.