pre-print version (March 2015)
to appear in in 52nd Design Automation Conference (DAC), June 2015
http://dx.doi.org/10.1145/2744769.2744857

Including Variability of Physical Models into the Design Automation of Cyber-Physical Systems

Hamid Mirzaei Buini, Steffen Peter, Tony Givargis
Center for Embedded and Cyber-physical Systems (CECS), University of California, Irvine, USA
Email: {mirzaeib, st.peter, givargis}@uci.edu

***Abstract*—A good cyber-physical-systems (CPS) design methodology must conduct trade-off analysis of both the physical characteristics of the CPS as well as its cyber sub-system in a holistic manner. This paper presents a design space exploration (DSE) approach for CPSs that emphasizes the variabilities of the physical subsystem and control aspects of the system. We propose the application of parameterizable physical models and automatic recalculation of control algorithm parameters for the explored systems. The resulting parameterizable models can be applied in a systematic simulation-based DSE framework that facilitates the identification of superior system configurations. We applied the proposed design flow to a real non-linear inverted pendulum system with a range of physical and cyber settings. The results show the feasibility and effectiveness of our approach in the design of physical and control parts of CPSs. Our work supplements existing work on cyber system modeling and plays an integral part in the design automation of such systems.**

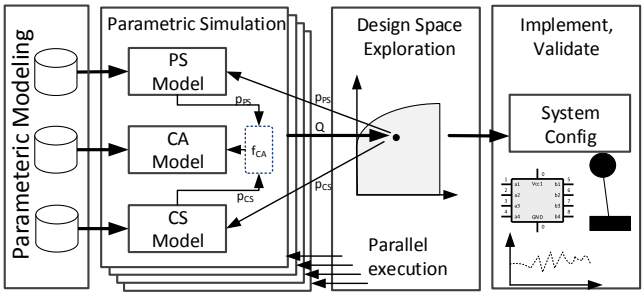


Fig. 1: Proposed CPS design framework: Parametric models are instantiated in a parameterizable simulation. The DSE tool invokes simulations parameterized with properties of the PS and the CS. Aim is to identify superior design points which can be validated and implemented.

I. Introduction

Generally, a cyber-physical-system (CPS) is one that combines computational and physical entities in a unified design effort. The design of CPSs needs good understanding of both subsystems, as small changes in the physical subsystem (PS) or the cyber subsystem (CS) may have significant consequences with respect to the overall system performance. For example, it is well known that the weight and size of physical objects like pendulums [16] directly influence the performance requirements for the CS – but also that scheduling decisions on the CS may affect the timing jitter of an otherwise correct control application so that the control process fails [3]. Therefore, a good CPS design methodology must carefully model and account for physical attributes of a system.

Traditional design tools are well suited in addressing design constraints and optimization objectives of the CS, but often fall short adequately to consider physical and control aspects of a CPS. Recent research has made tremendous advances in the development of frameworks for CPS that support the allocation and automatic evaluation of physical components in their CPS context [5], [14]. However, those frameworks operate on a high level of granularity by selecting pre-configured components but do not support parameterization of the physical components. Parameters of physical components include, for instance, setting the diameter of a pipe, the strength of a spring, or the size of mechanical parts. Another important aspect that has not been addressed in current CPS design automation frameworks is the consideration of the impact of the changes in the control subsystem. The control algorithm (CA) specifies how measurements from the PS are processed and how actuation commands are generated so that the CPS can achieve its objectives. As such, the CA is the logical bridging element between the CS and the PS, and needs to be designed and adapted when subsystems, including physical aspects, undergo changes.

The goal of this work is the development of a design space exploration (DSE) framework that considers the variabilities of the PS and the required adaptations of the CA. Our proposed design flow is shown in Fig. 1. The basic idea is to derive parameterizable models of the PS, the CA and the CS. The parameterizable models then can be instantiated by the DSE tool that, for each design point, invokes an executable simulation model to evaluate the quality of the CPS design. After an evaluation of the designs, a system configuration is proposed for implementation, containing parameters for the physical and the cyber subsystems. Our design flow is facilitated by the following main contributions of this paper:

1) We ensure rigorous parametric description following the equation derivation approach of the physical subsystem,

2) Settings of the CA are determined as a function of the parameters of the PS and the CS for every iteration of the design space exploration, which eliminates time consuming search for suitable configurations,

3) The DSE tool instantiates pre-compiled parameterized executable models of the CPS, which can be parallelized for the search of superior system configurations.

This work was supported in part by the National Science Foundation under NSF grant number 1136146.

DAC '15, June 07 - 11, 2015, San Francisco, CA, USA

http://dx.doi.org/10.1145/2744769.2744857

The encapsulated control knowledge and the efficient simulation enable CPS design analysis, available for system engineers even without strong control background. We show the steps, necessary to prepare the models, and the application of the models in a design space exploration including a non-trivial trade-off between control quality and energy consumption. We evaluate our approach for a rotary inverted pendulum system, for which we explored the design space that included a selectable length of the pendulum and a flexible sampling time in the CS. The results, which we validated in a real-life setup, show that our approach can automatically identify superior cyber-physical configurations which would have been ignored using existing design space exploration techniques.

II. Model-based Design of CPS

In this section we describe todays' model-based design (MBD) flows for CPSs. Specifically, we outline the state-of-the-art in automation of key steps of the design. The input to a CPS design flow is the CPS problem definition, which conventionally consists of the specification of the PS, design objectives like the actual CPS task, and nonfunctional requirements like control quality metrics and power efficiency. In MBD flows [10], generally, first the physical system is modeled, a CA is designed, and a CS is architected. The system can finally be implemented after successful validation. The following steps outline the design flow in greater detail.

1) Physical Modeling: Physical modeling concerns the expression of the physical system consisting of plant and actuators in mathematical or logical terms. The physical model can be used for two main purposes: first to build a simulation model of the CPS, and second to design a control algorithm based on this model. The wide selection of applicable models for this purpose spans from conventional models like transfer function or state space models to complex probabilistic models for large-scale systems. Besides system identification methods, equation derivation has been applied to obtain the model parameter values of the PS. In particular, the derivation of equations that govern the system using laws of physics has been applied in CPS development frameworks in practice [17]. Modeling of PSs even using those frameworks still requires experienced domain experts and, to date, automation tools are unavailable or provide limited benefits. However, blueprints of physical models can be reused from a repository as applied in [5] and [14], but without parameterization.

2) Control Algorithm Design: With the aim to find a correct CA for the CPS, the complex physical model needs to be simplified so that control design methods can be applied. Similar to the physical models, CAs can be selected from a repository to match the system properties and design objectives [5]. CAs have a range of parameters that influence the overall control quality. For example, PID controllers have three gain parameters. While for some specific use cases such as building control, an automatic control selection and parameterization was proposed [12], in general the selection and parameterization of the CA still have to be performed manually, which limits the use for automated design.

3) Cyber System Design: The design of a suitable computation system that implements the designed control algorithm is a non-trivial task, since many CS design decision may affect the quality of the CA. For example, the CA has to be discretized, a sampling time has to be selected, and the delay and jitter of the processing have to be considered. To cope with the gap between control and CS, control engineers might specify the acceptable ranges for some parameters such as delay and jitter in a contract-based approach [7]. Alternatively, the CS and the CA can be co-designed with certain jitter and delay in mind [6], [9]. These approaches are valuable for a good CS/CA co-design but neglect possible variability of the PS. While verification and calculus methods [15] exist to validate the correctness of specific systems, typically, designers rely on the simulation to validate the cyber system design.

4) Simulation: Applying the models of the PS, the CS, and the CA, simulation-based analysis can be executed to evaluate the system under development. The main objective of the simulations, which can be performed by capable off-the-shelf tools such as Simulink [2] and Modelica [1], is to evaluate and validate the performance of the CPS design. If all the requirements are met, the design can advance to the next step, namely implementation. Otherwise, reiteration of earlier steps in the design flow are required, which makes faults identified during simulation very expensive to fix, in particular if domain experts are required to manually refine the CA or the PS. Simulations are integral part of several approaches [5], [13], [14] to systematically analyze CPS design spaces. For instance Mühleis et al. [13] proposed a DSE solution based on co-simulation of control and CS, assessing design objectives such as cost or energy consumption for the whole CPS.

III. Parametric CPS Design Flow

In this section we introduce our parametric design flow, which is based on the standard MBD flow discussed in the previous section. As shown in Figure 1, our flow relies on parameterizable models for the PS, the CA, and the CS, which have to be crafted first, before they can be instantiated in simulation and for the DSE.

1) Parametric Physical Modeling: To craft the parameterizable model of the PS we propose the Equation-Based Modeling (EBM) [4] method. In contrast to system identification or learning techniques, EBM is not focused on finding a fixed set of parameters for a single instance of PS, but facilitates the derivation of the equations symbolically. The result is a parameterized model that describes the physical system behavior as a set of differential equations (E_{ps}), and their parameters P_{ps}, so that the model can be expressed as:

$$M_{ps} = (P_{ps}, E_{ps}) \tag{1}$$

The representation of the PS as M_{ps} is rewarding for two reasons: First, it allows composition of smaller physical subsystem to larger PSs [17], and second, M_{ps} is reusable and can be directly instantiated during the simulation step. However, it should be noted that no general guidelines are available how to craft M_{ps}, and the result becomes complex already for small systems like the example we discuss in Section IV.

2) Parametric Control Algorithm Design: As mentioned in Section II, generally CAs are already designed in parametric form, so that the CA models M_{ca} can be expressed as:

$$M_{ca} = (P_{ca}, E_{ca}) \tag{2}$$

where E_{ca} is the set of equations describing the CA, and P_{ca} is the set of CA parameters. Like the physical models, M_{ca} can be directly applied as parametrizable model in the simulations. A major issue is that the parameters P_{ca} depend on the PS as well as the CS and therefore must reflect changes in those

subsystems. A complex search for fitting parameters is not feasible for complex DSEs, because the search is required for each design point. Therefore, we propose to follow a derivation technique similar to the EBM approach discussed for the PS. The idea is to express p_{ca} as a mapping from properties of the models from PS and CS, and the design objective parameters (p_{obj}). This mapping $\mathbf{f_{ca}}$ is expressed as:

$$\mathbf{p_{ca}} = \mathbf{f_{ca}}(\mathbf{p_{ps}}, \mathbf{p_{cs}}, \mathbf{p_{obj}}). \quad (3)$$

Since $\mathbf{f_{ca}}$ computes the actual controller parameters, $\mathbf{f_{ca}}$ does not need to be symbolic, or in a closed form, but can also be represented numerically, as demonstrated in the example in Section IV. Crafting of $\mathbf{f_{ca}}$ requires the application of control theory methods for each type of controllers manually. The result is a transformation of the general model (4) to

$$M_{ca} = (\mathbf{f_{ca}}(\mathbf{p_{ps}}, \mathbf{p_{cs}}, \mathbf{p_{obj}}), E_{ca}), \quad (4)$$

which can be instantiated in simulations, with all parameters directly based on existing knowledge in the system.

3) Simulation and Design Space Exploration: The simulation tool we introduce next, instantiates the building blocks that were described in the previous steps. Since the idea is to run the simulation for each parameter setting in the design space, we are interested in a short simulation time. A standard approach is to obtain the performance values of interest (Q) by instantiating the parametrized models in a interpreted simulation environment such as Simulink. This approach is technically possible, but requires significant amount of time for each simulation. To accelerate the process we considered compiled simulation models, which is supported by Simulink and Modelica. However, compiling parameterized models still required to recompile the simulation model for each setting. To avoid unnecessary recompilations, we instead expose the parameters from the parametrizable models of the PS (1) and the CA (4), and only compile the unparametrized models, so that the simulation is expressed as:

$$Q = \text{SIM}_{M_{PS},M_{CS},M_{CA}}(\mathbf{p_{ps}}, \mathbf{p_{cs}}, \mathbf{p_{ca}}) \quad (5)$$

Hence, the simulation model is compiled only once for each model allocation, but recompiling is not required for changing model parameters in this tool. In Modelica, for instance, we only need to set the parameter values in an XML file and execute the same binary file for all the parameter settings. The result is a native executable simulation code which can be invoked by design exploration tools which only have to provide the parameters of the PS, the CS and the design objectives.

Algorithm 1 DSE for parametrized physical models

Input:
Simulate( ) ▷ The simulation code called as a procedure
$\mathbf{f_{ca}}$ ▷ The parametric CA design function
$D \subset \mathbb{R}^m$ ▷ Design Space

Output:
$J : D \longmapsto \mathbb{R}^n$ ▷ Evaluated Variables-of-interest

1: **procedure** EVALVOI($\mathbf{p_{ps}}, \mathbf{p_{cs}}, \mathbf{p_{obj}}$)
2: $\quad \mathbf{p_{ca}} \leftarrow \mathbf{f_{ca}}(\mathbf{p_{ps}}, \mathbf{p_{cs}}, \mathbf{p_{obj}})$
3: $\quad \mathbf{Q} \leftarrow$ Simulate($\mathbf{p_{ps}}, \mathbf{p_{cs}}, \mathbf{p_{ca}}$)
4: $\quad$ Put Q in the queue U
5: **for all** $(\mathbf{p_{ps}}, \mathbf{p_{cs}}, \mathbf{p_{obj}}) \in D$ **do in parallel**
6: $\quad$ EvalVOI($\mathbf{p_{ps}}, \mathbf{p_{cs}}, \mathbf{p_{obj}}$)
7: Sort U to find the mapping J

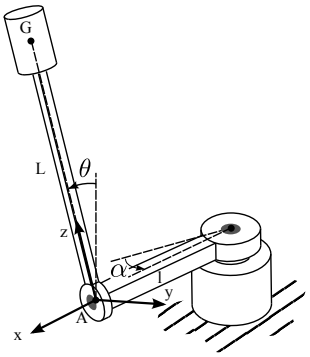


Fig. 2: One degree of freedom inverted pendulum

To study the design space, the pre-compiled simulations can be invoked in generic design space exploration (DSE) algorithms. As one example, Algorithm 1 demonstrates the feasibility of DSE that supports physical parameters, by systematical evaluation of all physical and cyber configurations in the search space (line 5, 6). Subfunction `EVALVOI` computes p_{CA}, invokes the simulation, and stores Q. The result is a set of evaluated variables-of-interest for each design point in the design space. From this space, the system configuration is selected evaluating the stored results. A benefit of the proposed algorithm is that procedure `EVALVOI` is "embarrassingly parallel", so that it can easily utilize parallel multi- and many-core platforms to accelerate the search process. Notably, Algorithm 1 does not utilize design space pruning techniques like they are required for larger design spaces. However, the proposed algorithm can be applied for smaller CPSs, such as the rotary inverted pendulum case study presented next.

IV. Case Study: Rotary Inverted Pendulum

In this section we use the rotary inverted pendulum example [16] as a case study to demonstrate the steps of framework described in the previous sections. The goal of the system, shown in Figure 2, is to produce appropriate actuator angle α to keep the pendulum in upright position, i.e, $\theta \approx 0$. In following subsections we discuss of the design flow including the crafting of the models (A-C) and the execution of the DSE, including a trade-off analysis and practical validation (D-E).

A. Parametric Physical Modeling

As explained in previous section, a parametric model of the PS is fundamental for the subsequent design steps. The block diagram of the complete CPS is shown in Figure 3. The controller should output the motor voltage in order to balance the pendulum. At the same time, the controller must track the reference command for the angle α. Here, we assume that the main output of the system is α and we want to move the weight to the commanded angle α_r. For this purpose, we assume that all the state variables, $\mathbf{x}$, are measured and provided to the controller.

The PS is shown using a dashed box in Figure 3. The PS consists of the actuator and the pendulum dynamics subsystems. First, the model of each subsystem is obtained and next they are combined to build the PS model. In the following paragraphs the EBM of these two subsystems is explained. We use the following symbols through-

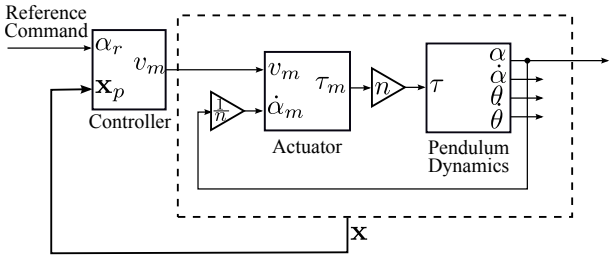


Fig. 3: Block diagram of the inverted pendulum example: A closed loop control system consisting of Controller, Actuator, and Pendulum Dynamics.

out this paper for the physical parameters and variables:

α_m	Motor angle (before gearbox)
B_m	Motor viscous friction coefficient
i_a	Motor winding current
J	Motor, Gearbox and actuator arm moment of inertia
K_b	Motor back-emf constant
K_t	Motor torque constant
l	Actuator arm length
L	Pendulum length
L_a	Motor winding self inductance
m	Mass of the pendulum weight
μ_m	Actuator viscous friction coefficient
μ_p	Pendulum joint friction coefficient
n	Gearbox ratio
R_a	Motor winding resistance
τ	Torque applied by the actuator
τ_m	Torque applied by the motor (before gearbox)
v_m	Voltage applied to the motor terminal

a) Pendulum Dynamic Equations: Dynamics modeling is required because we want to know how the pendulum angle changes when a torque is applied by the actuator. Assuming the mass of pendulum rod is negligible, we write the Newton-Euler equations for the rigid body consisting of pendulum weight and rod in the $\{xyz\}$ coordinate frame as follows [8]:

$$\sum \mathbf{F} = m\mathbf{a}_G \tag{6}$$

$$\sum \mathbf{M}_A = \mathbf{r}_{G/A} \times m\mathbf{a}_G + d\mathbf{H}_A/dt, \tag{7}$$

Expanding these equations we obtain following equation that relates the angular variables and their time derivatives:

$$-\mu_p\dot{\theta} + mgL\sin\theta = mL^2\ddot{\theta} - mL^2\dot{\alpha}^2\sin\theta\cos\theta - mLl\ddot{\alpha}\cos\theta \tag{8}$$

To involve the torque applied by the actuator in a new equation, torque equation around the motor axis can be used. The torque equation results in:

$$J\ddot{\alpha} = \tau - lF_y\cos\theta + lF_x\sin\theta - \mu_m\dot{\alpha}. \tag{9}$$

where $F_x = (-l\dot{\alpha}^2 + L\sin\theta\ddot{\alpha})$ and $F_y = (l\cos\theta\ddot{\alpha} - L\ddot{\theta} + L\sin\theta\cos\theta\dot{\alpha}^2)$. If we solve equations (8) and (9) for $\ddot{\alpha}$ and $\ddot{\theta}$ we will get:

$$\ddot{\alpha} = f_\alpha(\alpha, \dot{\alpha}, \theta, \dot{\theta}, \tau) \tag{10}$$

$$\ddot{\theta} = f_\theta(\alpha, \dot{\alpha}, \theta, \dot{\theta}, \tau) \tag{11}$$

These equations are used to define a nonlinear state space (SS) model for the pendulum as with the state variables $\mathbf{x}_p = (\ \alpha \quad \dot{\alpha} \quad \theta \quad \dot{\theta}\ )^T$ and input variable $u_p = \tau$ as

$$\dot{\mathbf{x}}_p = \mathbf{f}(\mathbf{x}_p, u_p). \tag{12}$$

This SS model is used inside the bigger simulation model of the whole system in the next step of our design flow.

The next step concerns the linearization of (12) with the aim to design the LQR Control method. For this purpose, we find the Jacobean of vector function $\mathbf{f}$ with respect to $\mathbf{x}_p$ and u_p and evaluate it in a reference solution of the (12), $(\mathbf{x_{p0}}, u_{p0})$. One reference solution is $\mathbf{x_{p0}}(t) = \mathbf{0}, u_{p0}(t) = 0$, which means the system is at rest with all the variables set to zero. So, the linearized system can be written as:

$$\dot{\mathbf{x}}_p = A_p\mathbf{x}_p + B_p u_p, \tag{13}$$

$$A_p = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-\mu_m}{J} & \frac{glm}{J} & -\frac{l\mu_p}{JL} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{l\mu_m}{JL} & \frac{g(ml^2+J)}{JL} & -\frac{\mu_p(ml^2+J)}{JL^2m} \end{pmatrix} \tag{14}$$

$$B_p = \begin{pmatrix} 0 & \frac{1}{J} & 0 & \frac{l}{JL} \end{pmatrix}^T \tag{15}$$

These matrices are the first part of the linearized PS model of our example. The model will be concluded with the actuator SS matrices which are explained next.

b) Actuator Modeling: The actuator in our case study is assumed to be a geared DC motor. The linear equation based model is:

$$L_a\dot{i}_a + R_a i_a = v_m - K_b\dot{\alpha}_m \tag{16}$$

$$\tau_m = K_t i_a - B_m\dot{\alpha}_m \tag{17}$$

Defining the actuator state and input as $x_a = i_a$ and $\mathbf{u}_a = (\ v_m \quad \dot{\alpha}_m\ )^T$, the resulting SS model of the actuator is:

$$\dot{x}_a = A_a x_a + B_a\mathbf{u}_a = -\frac{R_a}{L_a}x_a + (\ \tfrac{1}{L} \quad -\tfrac{K_a}{L_a}\ )\mathbf{u}_a \tag{18}$$

$$\tau_m = C_a x_a + D_a\mathbf{u}_a = K_t x_a + (\ 0 \quad -B_m\ )\mathbf{u}_a \tag{19}$$

Also, assuming the gearbox attached to the DC motor is ideal we will have $\tau = \tau_m n$ and $\alpha = \alpha_m/n$.

B. Parametric Control Algorithm Design

In LQR, the design objective is expressed by a quadratic cost function. In our case study this cost function is:

$$J = \int_0^\infty \left(\mathbf{x}^T Q\mathbf{x} + Rv_m^2\right) dt = \int_0^\infty (q||\alpha - \alpha_r||^2 + v_m^2)dt \tag{20}$$

where $\mathbf{x} = (\ \alpha - \alpha_r \quad \dot{\alpha} \quad \theta \quad \dot{\theta} \quad i_a\ )^T$ and we have chosen $Q = q\mathbf{v}\mathbf{v}^\mathbf{T}$ where $\mathbf{v} = (\ 1 \quad 0 \quad 0 \quad 0 \quad 0\ )^T$ and $R = 1$.

The above cost function tries to balance the energy consumption and control performance. The first component in (20) , $\int_0^\infty ||\alpha - \alpha_r||^2 dt$, defines control performance metric as the mean-squared error (MSE) of angular command tracking. The parameter q in (20) is the relative weight of control performance to power consumption which is described by the second component $\int_0^\infty v_m^2 dt$.

In LQR, the control law is the state feedback which can be described using the following equation:

$$v_m = -K\mathbf{x}. \tag{21}$$

The State vector $\mathbf{x}$ is the combination of the pendulum dynamics and actuator SS model state vectors, with one modification, to subtract the reference command α_r from α.

By solving the algebraic Ricatti equation [11], the optimal

Algorithm 2 The Control Algorithm

Input: K, α_r ▷ LQR gain and angular command

1: **for all** sampling time **do**
2: | Measure $\alpha, \dot{\alpha}, \theta, \dot{\theta}$ and i_a
3: | $v_m \leftarrow -K(\ \alpha - \alpha_r \quad \dot{\alpha} \quad \theta \quad \dot{\theta} \quad i_a\ )^T$
4: | Apply voltage v_m to Motor

gain K in (21) is calculated. This gain optimizes the cost function (20) guaranteeing that the closed loop system will be stable. Solving Ricatti equation to obtain optimal gain, K, corresponds to (3) in the proposed design flow. Therefore, the concrete form of (3) is:

$$K = \mathbf{f_{lqr}}(L, q) \tag{22}$$

For the control we only have to choose a sampling time, h, compute and apply the control signal in successive sampling periods. The CA is shown in Algorithm 2.

C. Simulation

We built the executable specification model of Figure 3 in Modelica, instantiating the pendulum equation (12), actuator equations (18) and (19) and the discrete-time state feedback (21). By compiling the Modelica model we have the simulation executable code. The inputs to this code are pendulum length L, sampling time h and controller gain K. The output is the objective variable which is MSE of tracking error $||\alpha - \alpha_r||$. At the end of this step, the general form of simulation procedure explained in Section III is:

$$MSE = \text{SIM}_{\text{Inv\_Pendulum}}(L, h, K), \tag{23}$$

while K is the result of (21) and can be computed externally using the "control" Python package.

D. Design Space Exploration

1) Setup and Execution: The DSE in our case is an implementation of Algorithm 1 in Python, instantiating the simulation executable code of (23). The design objective parameter q is fixed to 10 ($1/rad^2$) in this case study. We studied pendulum lengths from 0.01 (cm) to 60 (cm) and sampling times from 0.01 (s) to 0.14 (s). The parallelism of the DSE and the pre-compiled simulation allowed us to finish the exploration within is 87(sec) on a 48-core Intel Xeon @ 2.7GHz platform. The naive approach without parallelization and using Simulink in normal mode results in 2360 (sec). Using existing DSE flows with no parametric CA design, the running time would be approx. 86 days if we want to try only 5 values for each element of gain parameter K.

2) Control Quality: Figure 4 shows the stability region of the DSE. This plot is generated by comparing the computed MSE with a fixed threshold of 0.035 (rad^2). A system with MSE below the threshold is considered stable. Figure 4 also compares our parametric control approach to static reusable controllers. The stability regions for fixed-controllers designed for 6 (cm) and 34 (cm) pendulums are shown on top of the region for our approach. Evidently, the static controllers only cover small range of the design space, while our approach provides good design points for all physical settings.

3) Power Consumption: The second important metric is power consumption. The total power of the system is the sum of the power for the PS and the CS: $P_{CPS} = P_{PS} + P_{CS}$. P_{PS} is the required power of the actuator ($P_{PS} = v_m i_m$), which is

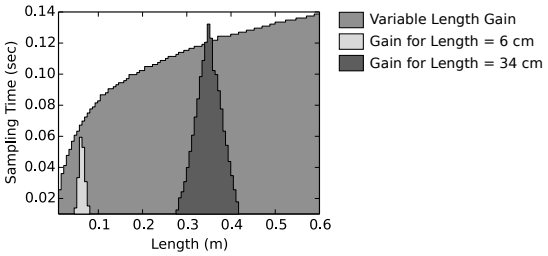


Fig. 4: Stability region for different CA gains.

part of the actuator model (16). P_{CS} can be computed from the processing time (t_{prc}) and the average power consumption of the microcontroller for sleep (P_{slp}) and run (P_{run}):

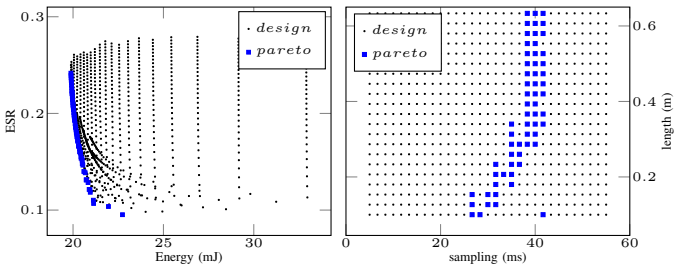
$$\bar{P}_{\text{CS}} = \frac{t_{\text{prc}}}{T} P_{\text{run}} + \left(1 - \frac{t_{\text{prc}}}{T}\right) P_{\text{slp}} \tag{24}$$

The power consumption for a fixed pendulum length is shown in Figure 7. It is visible that smaller h reduces energy due to better control performance, but for very small h the required processing power outweighs the savings. The result is an interesting trade-off between the control and power performance of the whole system, which we study next.

4) Control-to-Energy Trade-Off: Figure 5 (a) shows the scatter graph for power consumption and control quality for the discussed design space. An ideal system would be located in the bottom left. The Pareto front (in blue) shows all potentially beneficial design points. For each system, that is not part of the Pareto front, at least one system exists with better power consumption and better control quality. Figure 5 (b) shows the highlighted points in the design space of sampling rate and length. The results deliver a set of superior design points, and confirm that the design space does not contain a superior pendulum length, sampling rate, or controller setting that would dominate the entire design space, which confirms the importance of the holistic DSE in order to identify superior design points.

E. Practical Evaluation

To validate the results for the modeling and the controller design, we implemented a pendulum system that supports a range of physical configurations. As examples, Figure 6 shows two pendulums with a length of 6(cm) and 34(cm). For each experiment, we could use different pendulum lengths, LQR gains, and sampling rates. The control program was



(a) (b)

Fig. 5: Simulation results for energy and control quality (a), and the highlighted Pareto set in the design space (b).

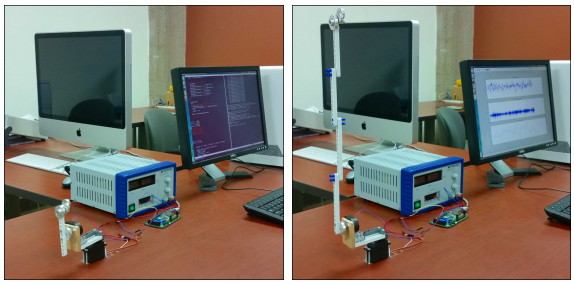


Fig. 6: Pendulum setup using 6cm (left), and 34cm length.

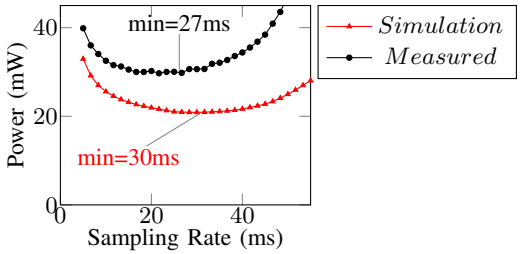


Fig. 7: Measured and simulated energy consumption for the 34cm pendulum.

implemented on an Cortex-M3 ARM development board. Like the DSE in the previous subsection, all the experiments could be performed by a system engineer without manually revisiting the control algorithm. We used our framework to return the energy-optimal system and parametrization for a given set of invariant system parameters (either size or sampling rate). After applying the computed configuration values to the experiment, we expected that each pendulum length has the expected stability. The experiments confirmed the assumption to be correct, since each system ran stable for at least one minute. As cross-validation we applied the controller setting of the small pendulum (6cm) for the long pendulum (34cm) and vice versa. As a result the short and long pendulums tilted after four and one seconds, respectively, which confirmed the simulations results shown in Figure 4.

A comparison of the measured energy values and the simulation results is shown in Figure 7. Notably, the actual power consumption is about 10(mW) above the values obtained in the simulation. We suspect the higher power consumption is caused higher friction in reality than considered in our PS model. However, more important is that the practical measurements validate our assumption of a cyber-physical power consumption minimum. The measurements can identify this minimum as 27(ms), which is well in range of the estimated optimum point of 30(ms) for the system.

V. Conclusions

The results of the experiments presented in the previous section underline the importance of tailored control design in a holistic design process of CPSs. Variabilities of the physical subsystem have to be reflected in the control algorithm in order to find superior design points. We presented a DSE framework that instantiates parameterized models of the physical system, the control algorithm and the cyber system to find those design points. The applied EBM approach for the physical models opened the design space for the physical subsystems, and the derived configuration of the control algorithm parameters helped to reduce the design space to superior designs only. The actual DSE then is based on parameterized executable simulations, generated in Modelica, which facilitate a highly parallel DSE. The DSE in the pendulum use case enabled tool-supported managemant of the trade-off between control quality and energy consumption - even for non control experts.

Evidently, the modeling steps in our design flow are still complex and require knowledgeable control engineers and physical domain experts. However, following our design methodology, resulting models can be packaged as reusable parameterizable components which in turn enable systematic DSE of CPSs as part of design automation tools.

References

[1] *OPENMODELICA*, 2014. http://www.openmodelica.org.

[2] *Simulink - Simulation and Model-Based Design*, 2014. http://www.mathworks.com/products/simulink/.

[3] A. Aminifar, P. Eles, Z. Peng, and A. Cervin. Control-quality driven design of cyber-physical systems with robustness guarantees. In *Design, Automation and Test in Europe (DATE)*, 2013.

[4] D. Broman. *Meta-Languages and Semantics for Equation-Based Modeling and Simulation*. PhD thesis, Department of Computer and Information Science, Linkoping University, Sweden, 2010.

[5] A. Canedo, E. Schwarzenbach, and M. A. Al Faruque. Context-sensitive synthesis of executable functional models of cyber-physical systems. In *ACM/IEEE International Conference on Cyber-Physical Systems*, 2013.

[6] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Systems Magazine*, 23(3):16–30, June 2003.

[7] P. Derler, E. Lee, M. Törngren, and S. Tripakis. Cyber-physical system design contracts. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2013.

[8] J. Ginsberg. *Engineering Dynamics*. Cambridge University Press, 2007.

[9] D. Goswami, R. Schneider, and S. Chakraborty. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASPDAC)*, 2011.

[10] J. Jensen, D. Chang, and E. Lee. A model-based design methodology for cyber-physical systems. In *Wireless Communications and Mobile Computing Conference (IWCMC)*, 2011.

[11] D. Kirk. *Optimal Control Theory: An Introduction*. Dover Books on Electrical Engineering Series. Dover Publications, 2004.

[12] M. Maasoumy, Q. Zhu, C. Li, F. Meggers, and A. S. Vincentelli. Co-design of control algorithm and embedded platform for building hvac systems. In *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on*, 2013.

[13] N. Mühleis, M. Glaß, L. Zhang, and J. Teich. A co-simulation approach for control performance analysis during design space exploration of cyber-physical systems. *SIGBED Rev.*, 8(2):23–26, June 2011.

[14] H. Neema, Z. Lattmann, P. Meijer, J. Klingler, S. Neema, T. Bapty, J. Sztipanovits, and G. Karsai. Design space exploration and manipulation for cyber physical systems. In *Workshop on Design Space Exploration of Cyber-Physical Systems (IDEAL')*, 2014.

[15] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, 2000.

[16] F. Zhang, K. Szwaykowska, W. Wolf, and V. Mooney. Task scheduling for control oriented requirements for cyber-physical systems. In *Real-Time Systems Symposium (RTSS)*, 2008.

[17] Y. Zhu, E. Westbrook, J. Inoue, et al. Mathematical equations as executable models of mechanical systems. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2010.