Embedded Adaptive STC Control Development by a Free Toolchain

Gernot Grabmair, Simon Mayr

FH OÖ Forschungs & Entwicklungs GmbH, Stelzhamerstraße 23, A-4600 Wels, AUSTRIA

KURZFASSUNG/ABSTRACT:
In this study we present an adaptive STC controller design for small embedded systems by a new free tool-chain for model based control design. This is based among other software on the open simulator Scilab-XCos. After a very short introduction of model based design terms this article focuses on the code generator and the other programs of the tool-chain. The design concept is demonstrated by the non-trivial adaptive STC control of the cart system in simulation and on a real laboratory experiment.

1 INTRODUCTION

Model-Based Design (MBD) is a mathematical and visual method of addressing problems associated with designing complex control, signal processing and communication systems. It is used in many motion control, industrial equipment, aerospace, and automotive applications. Common tool-chains are quite expensive commercial solutions due to the origin of MBD in aerospace and automotive industries.

Commercial code generators for Matlab-Simulink (M&S), Dymola, etc. do exist. On the other hand, INRIA and others provide free code generators for the outdated Scilab-Scicos – an open source pendant of M&S, e.g., [2], [3], and some more.

The main idea presented in this paper is the MBD development with a complete free (or low-cost if target hardware is included) tool-chain from the modeling and control design to the hardware realization using an integrated development environment (IDE).

This paper is organized as follows. Starting with a short description of the parts of the whole tool-chain we focus on the details of the code-generator itself. Afterwards, an adaptive control law for the cart system is derived as a non-trivial application. Finally, the model based design process is demonstrated by the implementation of this controller on a low-cost embedded system board.

2 MBD – THE TOOL-CHAIN

For the major development steps of a model-based controller design (plant modeling, control design and simulation, code generation and transfer to the target) the reader is referred to Fig. 1. The plant modeling and control design steps are already possible in the standard Scilab-XCos.

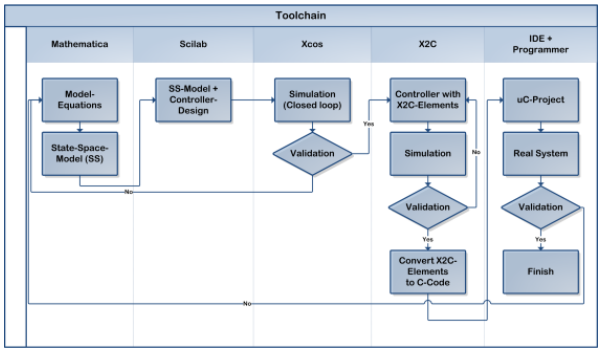

Figure 1. MBD and the tool-chain.

Scilab [1] is a free and open source software for numerical computation maintained by Scilab Enterprises similar to the commercial Matlab from Mathworks. The graphical dynamical system modeler in Scilab is called XCos and the counterpart to Simulink from Mathworks. The range of features can be extended by add-ons, e.g.

Modelica based Coselica toolbox (see [8]). System identification and simulation verification is done in Scilab-XCos too. At least for more dedicated control laws the system has to be described in mathematical terms, i.e., in form of ODEs.

After the controller has been developed and tested in the simulation environment, it must be converted into C-code and transferred to the target. This process should occur as automated as possible. There are several possibilities to generate C-code from an existing XCos schematic. In order to name a few:

- Gene-Auto, [3]: only works with the previous version of Scilab, called Scicos.
- Real-time Linux as target system, [2]: supports the outdated Scicos.
- Scicos-FLEX, [4]: supports the outdated Scicos.

As alternative, we make use of our own generator X2C – see below for a more detailed description.

2.1 The code generator X2C

The code generator X2C, see [5], was originally developed more than ten years ago at the JK-University Linz, Austria as a Simulink extension generating assembler code for TI-DSPs. Later, the system was extended to generate C-code and to largely comply with MISRA (S2C), see [7]. This long history and the simple effective concept of the code generator system stands for stability of its main kernel.

X2C natively includes into XCos and can be simulated in parallel with plant and the controller, see Fig. 2. For the plant blocks of the Coselica library are used. The controller part is modeled using dedicated X2C-XCos blocks. These blocks are full featured XCos blocks extended with an enhanced parameter editor and the connection to the back-end for e.g. code generation. All the glue code needed for these X2C-XCos blocks is fully auto generated from the X2C block's model. Special blocks are used to model and specify the targets in-ports and out-ports.

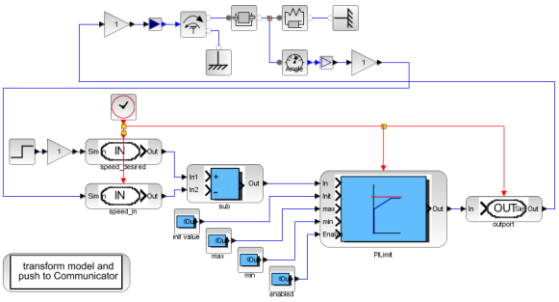

Figure 2. System model with plant and controller.

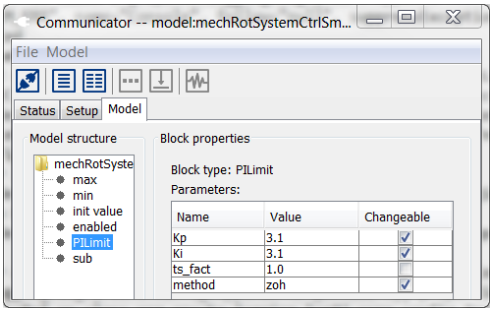

Figure 3. The X2C Communicator.

This system model shall be simulated to provide detailed feedback about the expected performance of the overall system for further optimization. In this simulation the blocks are implementing exactly the code which will run on the target. In simulation the developer has easy access to all signals of the plant and controller to analyse the behaviour or to inject faults for testing.

To move on to the target implementation the model transformation and code generation is executed by a single mouse click. The model transformation will step through the XCos model ignoring all non-X2C-XCos blocks to detect all X2C-XCos blocks and their associated clock domain and hence the sample time. The result is an abstract model implemented in Java holding all relevant information about the blocks, their parameters and their connections. Further on, the code generator is applied on this abstract model using methods from graph theory to check, to partition and order the model and to generate the final code. The generated code is written as substantially MISRA [7] conform ANSI C code in object oriented style.

The central tool for the developer is the so called Communicator, see Fig. 3. The Communicator is the home of the code generator and the interface to the modeling / simulation environment and the target.

For effective development rapid feedback on changes in design is valuable. With X2C it is reality that e.g. parameters can be changed in the model or in the Communicator and the parameters on the target are updated instantly. That means manually tuning controller parameters becomes a task of selecting the block and parameter in the XCos diagram and using the keyboard or mouse wheel to tune the parameter while watching the plant and the automatically updated Scope for feedback. The Scope is featuring functionality like a conventional oscilloscope. Through the Scope the developer has access to online data of block ports, I/O ports, variables and registers of the target in a way like in the simulation environment.

3 APPLICATION: CART SYSTEM

As a reference application the well-known cart system is presented, see Fig. 4. As common to real world applications the vehicle load capacity is not known. We assume the vehicle mass $\tilde{m}_1$, linear friction coefficient $\tilde{d}_1$ and static friction coefficient F_C as unknown but constant. The cart with mass m_1 is driven by a dc drive with external excitation.

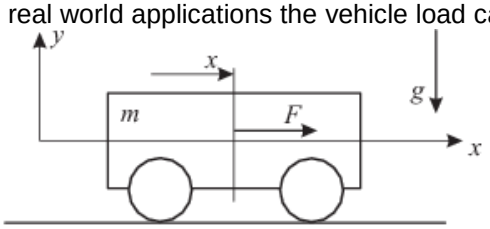

Figure 4. The well-known cart system.

By means of system reduction, we introduce equivalent parameters $(\tilde{m}_1, \tilde{d}_1)$ and the machine constant β integrating the whole drive-train into the mathematical model of the cart.

The model equations can be written in the form $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}u$. We assume that the static friction is ignored because its value will be identified and compensated in all measurements by the well-known approach.

$$\begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -\frac{\tilde{d}_1}{\tilde{m}_1} \end{pmatrix} \cdot \begin{pmatrix} x \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{\beta}{\tilde{m}_1} \end{pmatrix} \cdot u_A \qquad (1)$$

3.1 Continuous system identification and adaptive control design

As already mentioned, the cart parameters are assumed unknown but constant. For full physical insight into the adaption parameters, we do not identify the sampled but the continuous plant itself. In order to get rid of the time derivatives the second equation of the linear system model is filtered by realizable stable filters F_0 and F_1 with free coefficients α_i and results in one data line of the algebraic equation system for identification

$$\begin{pmatrix} \alpha_0 \left(x - \frac{\alpha_1}{\alpha_0} F_1 * x - F_0 * x \right) & F_1 * x & F_0 * sign(v) \end{pmatrix} \cdot \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix} = F_0 * u_A \beta \quad (2)$$

linear in the unknown parameters $\theta_1 = \tilde{m}_1$, $\theta_2 = \tilde{d}_1$ and $\theta_3 = F_C$, whereby * indicates the convolution operator in time-domain. For implementation purpose the filters have to be discretized and $sign(v)$ approximated.

The unknown parameters can be estimated using standard recursive least square algorithm. For the adaptive self-tuning control (STC) approach the filters and a standard RLS algorithm are implemented in combination with a linear state control law parametrized in $\tilde{m}_1$ and $\tilde{d}_1$.

3.2 Experiment – Cart system

In order to demonstrate the physical modeling capabilities the Coselica toolbox is used for plant modeling, see Fig. 5.

3.2.1 Adaptive STC Control

In order to meet the proposed goal, STC control with unknown but constant parameters is applied to the real plant. The first 2 seconds are used for settling the online RLS identificator and the filters as derived above. Then the parametrized control law $u_A = -\boldsymbol{k}(\boldsymbol{\theta})^T \boldsymbol{x}$ is activated, see Fig. 6.

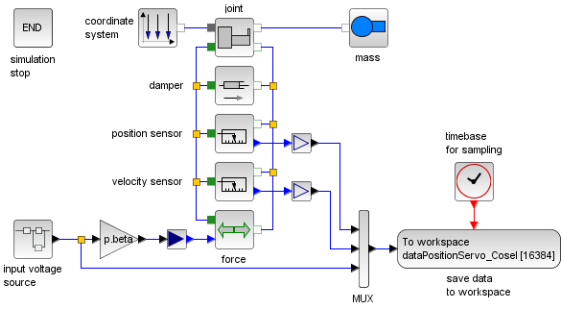

Figure 5. Cart system - Coselica.

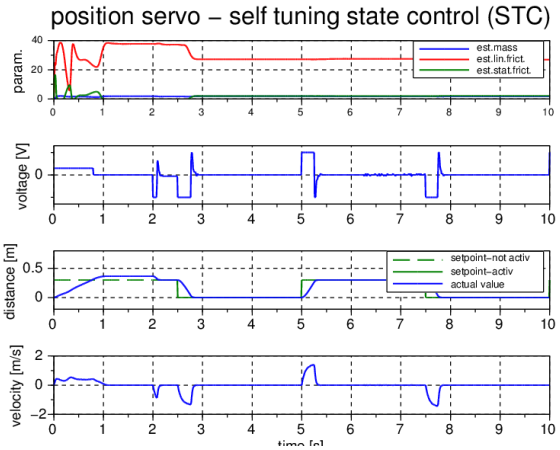

Figure 6. STC control measurements

4 CONCLUSION

The presented work is available for free – to a large extent under a BSD licence, see the references [5], [6].

Ongoing development is targeted towards efficient handling of vectorized signal lines, Scilab-code to C-code conversion, more block libraries, industrial targets, IEC 61131-3 code generation (PLC), adaption of the FMI (functional mockup interface) for model exchange with various commercial simulators, and some state machine concept. For most parts we plan to adapt again existing free tools.

We gratefully acknowledge the support from the Austrian funding agency FFG in the COIN-project ProtoFrame and the internal funding in the project MOdoPS. Part of this work was conducted within the research project “sustainable and resource saving electrical drives through high energy and material efficiency” sponsored within the EU program “(Regio 13)” and within the COMET K2 center ACCM. Further, we are indebted to our master students Höglinger Thomas and Würflinger Florian for testing and measurement preparation.

REFERENCES

[1] Scilab: Free and Open Source software for numerical computation, Scilab Enterprises, 2012, http://www.scilab.org/.
[2] Roberto Bucher, et al., RTAI-Lab tutorial: Scilab, Comedi, and real-time control, 2006.
[3] Ana-Elena Rugina, et al., Gene-Auto: Automatic Software Code Generation for Real-Time Embedded Systems, DASIA, 2008.
[4] Scicos-FLEX code generator, References, http://erika.tuxfamily.org/drupal/scilabscicos.html/.
[5] X2C in Scilab-XCos, 2013, http://www.mechatronic-simulation.org/.
[6] 2013, http://modelbaseddesign.at/.
[7] MISRA Consortium, 2013, http://www.misra.org.uk/.
[8] Reusch, 2013, References, http://www.kybdr.de/software/.