

INEXPENSIVE PROTOTYPING FOR MECHATRONIC SYSTEMS^{*)}

Dusko Jovanovic, Bojan Orlic, Jan Broenink, Job van Amerongen

Drebbel Institute for Mechatronics, Twente Embedded Systems Initiative,
Control Engineering, Faculty of EE-Math-CS
University of Twente
P.O.Box 217, 7500 AE, Enschede, the Netherlands
email: d.jovanovic@utwente.nl

Abstract

This paper provides a description of an approach based on our 20-sim modelling and simulation PC software package in combination with various hardware targets with an Analog Devices Inc. DSP board as an instance. The main virtue of this approach is support for a number of (virtually any decent) hardware platforms for, in this case, a price less than a few percent compared to existing market solutions.

1 Introduction

It is a common statement that nowadays in an engineering cycle of mechatronic products, the time span from a concept to the first prototype needs to be as short as possible. An idea of an immediate real-world validation of a modelled and simulated control system is inarguably attractive. The idea is captured both by academia and commercial solutions providers (Heck *et al.*, 2002), but there are just a few recognized implementations, of which most famous is a hardware extension of MATLAB/Simulink Real-Time Workshop, released by dSPACE GmbH.

Our modelling and simulation tool 20-SIM (CLP, 2002) lately being extended with C-code generation functionality, covers all the necessary facilities to efficiently support mechatronic design. Especially for the mechatronic engineer, because constructing the control computer code is done automatically out of the controller submodels designed in 20-SIM. This alleviates the user of mastering the specialities of the target computer system, necessary to develop the software drivers of the I/O devices on the target. Furthermore, as the step from control law to control code is being automated, one serious source of user's mistakes has been eliminated.

We specialized the general 20-SIM C-code generation facility for the target at hand, the Analog Devices Inc. ADSP-21992 EZ-kit Lite board (EZ-KIT, 2002) containing the ADSP-21992 mixed-signal DSP (ADSP, 2002), especially designed for motion control.

^{*)} This project is a part of a research supported by PROGRESS, the embedded system research program of the Dutch organization for Scientific Research, NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW.

It is a so-called *evaluation* board, implying that it is only suitable in experimental situations. Of course, this is one reason of being lower in price than the sophisticated boards offered by dSPACE.

The board we used here was donated by Analog Devices Inc. in the context of their University donation program. It was for use in our *Mechatronics Project* in the second year of the EE BSc programme where in two weeks time teams of students learn to apply knowledge from lectures on Measurement, System Modelling and Simulation, Transducer Technology and Control Engineering. Application of control engineering and controller realization was new compared to previous years (Broenink *et al.*, 2001). Writing software was absolutely not possible in *this* project. Therefore, the software generation part needed completely be automated. This approach is also applicable in a more research- or industrial-oriented setting. The researcher can concentrate on the control engineering aspects, leaving the software implementation aspects for being solved.

In this paper, first a description will be given of this approach. Section 3 will describe the basic hardware and software required for such a setup. In section 4, a case study developed as demonstrator for the *Mechatronics Project* will be presented. However, the results can also be used for industrial prototyping as well.

2 Approach

The design cycle of a mechatronic system consists of several steps. The first step is the conceptual design stage, where basic ideas are evaluated. Based on simple models it can be evaluated whether it is possible to realise a product that meets the specifications. Simple models can be of great help to get insight in the design and can help to predict e.g. the achievable bandwidth. In a next stage, when one of the concepts has been selected, the models can be further detailed and different control strategies can be evaluated. The final stage is that mechanical and electronic or computer-based parts of the process are realised and built together. The Control Laboratory of the University of Twente has a history of over 30 years in doing research on software that supports this mechatronic design process. In the past this has resulted in the simulation program TUTSIM and more recently in the modelling and simulation program 20-SIM (CLP, 2002).

The leading idea of this approach is to provide for a continuous support in the design stretch of an embedded control implementation when migrating from a proper modelled design to coding for the chosen hardware platform. Using a sophisticated modelling and simulation tool, the user *can* refine a prototype setup from a coarse sketch to the level of fine-tuned plant dynamics and control strategy, by means of simulations. Furthermore, on the course towards the final design, as soon as a simulation of the control part of the prototype is performed, it is *possible* to deploy the calculations directly on the attached hardware platform. The immediate validation gives an opportunity to compare measured signals from the prototype setup with signals predicted by simulation.

In 20-SIM it is possible to start with an elementary model that represents the main physical phenomena in system and add more detail to this model later on. Because of the port-based modelling features of the program, represented by iconic diagrams or bond graphs, the models remain close to the physical reality in all stages of the design. Together with the features for controller design this enables that the influence of changing e.g. a mass, compliance or a motor constant, be directly reflected in the

performance of the closed loop system. Representations are possible, among others, in the form of pole-zero plots, bode plots or time responses (Amerongen, 2002).

In most cases the controller will be realised with the aid of digital computer hardware. Therefore, it is important that effects of discretisation and quantisation of signals be evaluated in simulation.

The next step in the design process is to realise the mechanical construction and to realise the digital controller in a proper control computer. Nowadays solutions become available for automatic code generation of controllers that have been designed on a higher level in terms of block diagrams and transfer functions. The combination of Matlab and dSpace is a well known but expensive example of such an approach. 20-SIM also enables automatic generation of C-code for real-time control purposes, but is not dependent on specific hardware. The C-code generation of 20-sim is based on templates that can be adopted to different hardware platforms, thus enabling to run the controllers on digital hardware ranging from dSPACE hardware to PC's and DSP's.

3 Tools

3.1 20-SIM

20-sim is a modelling and simulation tool developed by ControlLab Product B.V, (CLP, 2002; Broenink and Kleijn, 1999), a spin-off company of the Control Engineering group of the University of Twente. It is a standard MS Windows application consisting of several integrated modules that support modelling and design of mechatronics products in many aspects.

Users' inputs to the modelling module can be performed by means of one or more 20-sim Editors: bond graphs (Breedveld, 1985), block diagrams, iconic diagrams, equations (Broenink, 1999) or by importing Matlab models. Models are usually constructed by combining and adapting submodels from the available submodel library. Nonlinearities can straightforwardly be specified by editing the equations of the selected (or newly created) submodels. Additional editors are available for designing specific linear systems directly addressing the control aspects of mechatronic design: Filter Editor, Linear System Editor, and Controller Design Editor. Entry can be done via transfer functions, zero-poles gains or state space descriptions. All editors facilitate one-click visualisation of the most common representations of linear systems: Step response, Bode, Nyquist, Nichols and Pole Zero plots. In the Controller Design Editor all these diagrams (plus root-locus calculations for Pole Zero plots) are available for all important system transfer functions (Loop Transfer, Sensitivity, Complementary Sensitivity, etc). This 'multiple view' option allows that each user can select the representation most suited in a specific situation and supports a multidisciplinary design team.

The tool complies well with the demand of offering a time-efficient and elaborate feedback to the user on the modelling/design decisions. By means of a flexible simulation module including 3-D animations of the modelled object, the 20-SIM Simulator allows for reliable verification.

The newest functionality of 20-SIM is to generate C-code from a selected submodel. Code specific for the target computing system, like device drivers, can be specified as source code files, called *templates*. The link with the code coming from the submodel

selected in 20-SIM is established via *code generation directives*, indicated as “%TERM%, analogous to the standard pre-processor directives of C/C++-compilers. In the 20-SIM code generation module, first the selected submodel is compiled into C statements and the *code generation directives* are specified: this means substituting the names of the variables, parameters etc. Also code generation directives are specified indicating the amount of variables, parameters, states and derivatives present in the submodel. This is for instance necessary for allocation of data-arrays in the final code. The result is a set of C-files which can be compiled, linked and loaded to the target computer system. For this action, the target-specific software development tools are used.

The 20-SIM user has no need to write software for the target computing system. All controller code is specified in the 20-SIM controller design and testing phase.

3.2 Target computing system

The target computing system can virtually be any computing board with suitable I/O facilities and having proper real-time behaviour. In this case, we used the Analog Devices Inc. ADSP-21992 EZ-kit Lite board (EZ-KIT, 2002), donated to us by Analog Devices Inc. in their framework of academic donations. The ADSP21992 DSP (ADSP, 2002) contains an ADSP-219x 16-bits fixed point CPU running at 160 MHz, 8-channel ADC (20Msamples/s) with 14 bits resolution, one three phase and two mono phase PWM 16 bits outputs, three programmable 32-bit interval timers, a 32-bit incremental encoder interface unit with companion encoder event timer and 16 bits Digital I/O. The DSP also contains 32Kx24bit program memory and 16Kx16 bit data memory. Furthermore, on the evaluation board, an 8-channel 12-bit Digital to Analogue converter, USB, CAN and a serial port are available.

3.3 Code generation for the ADSP 21992

The total system of 20-SIM, DSP-evaluation board and accompanying specific compiler should support rapid prototyping of a closed-loop control system such that control laws designed using 20-sim Editor can be implemented on the DSP without manual coding. This process should be accomplished as much as possible within 20-SIM. Furthermore, as a simulation not only gives a qualitative impression of dynamic behaviour of a system under consideration but also a quantitative characterization of the system responses as well, the toolset should also offer a kind of data logging for every run performed. Comparison of captured real-system signals with responses predicted by simulation by showing them in *one* plot on the PC is possible.

Special *mapper* models need to be designed, to let the 20-SIM code generation module map the variables of the submodels to the signals of the I/O peripherals on the target computing system. At simulation, these *mappers* pass through the value of the simulation variable, whereas at C-code generation, the *mappers* generate calls to the target specific driver functions, thus connecting the variables in the CPU of the DSP to the I/O peripherals. A special *mapper* function has been designed to configure the data logger, which takes care of capturing designated signals for storage and inspection on the PC. The icons of the mappers are shown in Table 1.

The logger itself consists of two parts: one part on the DSP and another part on the PC. The DSP part of the logger, which is specified in the template, reads signals from the running DSP code and sends it to the serial port. Only those variables being specified in an accompanying simulation experiment are captured. Furthermore, a buffer is used to

decouple the hard-real time control loop from the in principle soft-real time data acquisition part. The PC side listens to the serial port, and when data arrives, it stores them on file in 20-SIM format, such that validation using 20-SIM (or any other data processing package) is possible. At this moment, we use the serial port, implying that this data communication is the slowest part of the total software. Therefore, a facility was made to only send variables each n^{th} time step. This can be specified in the logger.

 <i>ADC1</i>	Analogue Input	 <i>ENC1</i>	Encoder input
 <i>DAC1</i>	Analogue Output	 <i>PWM1</i>	PWM output
	Data Logger		

Table 1 Mapper Icons

The resulting code generation facility indeed alleviates the user from writing DSP code. The only DSP-specific work to be done is to download the code to the DSP board, start running it, and also start the data-logger (only when it is specified in the model that signals from the DSP need to be logged on the PC).

4 Case study: a mechatronics project

As a case study the design and realisation of a small mechatronic system that can be used as a training setup for students will be considered. The first step is to start with a rather simple model that describes the basic components of a servo system: a DC motor that drives a mechanical load (Figure 1).

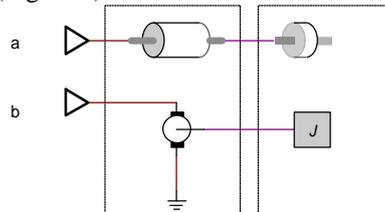


Figure 1 DC-motor and load, a) conceptual model, b) most simple model of physical phenomena

In this simple model the motor only consists of the conversion from electrical energy to mechanical energy and vice versa. The load is only modelled as an inertia. This model can be extended with a (flexible) transmission, the motor can be further detailed with inertia and friction and friction can be added to the load as well. This leads to the following model (Figure 2).

Such a servo system can be well controlled by means of a PD-type controller (Amerongen and Breedveld, 2002) consisting of a position feedback of the load or the motor and a velocity feedback of the motor. In this example we will choose for feedback from the motor angle only use a lead network in the feedback path. The controller can be converted into discrete form and a comparison can be made between the continuous time and discrete time system. Proper AD- and DA-resolutions and sampling rate can be

determined in this phase. Figure 3 shows a comparison between the continuous time and discrete time system and Figure 4 shows the responses.

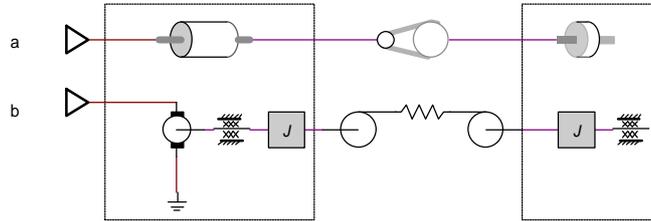


Figure 2 DC-motor and load, a) conceptual model extended with transmission, b) more detailed model of the various components

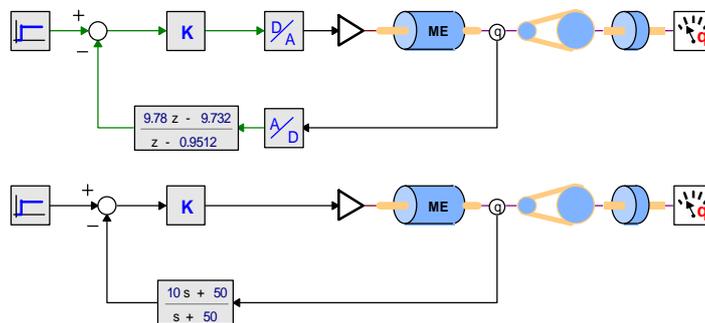


Figure 3 Upper part: discrete-time realisation of the control system
Lower part: continuous-time realisation of the control system

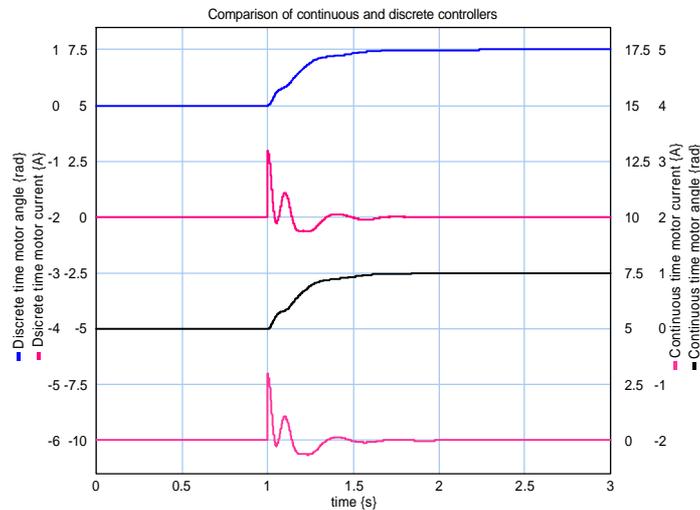


Figure 4 Simulation comparing motor angles and currents for a continuous-time (lower curves) and a discrete-time (upper curves) controller

The responses of these models reveal that the continuous-time system and the discrete-time system have similar responses. At this stage the digital controller should be brought into *one* submodel for which C-code can be generated automatically (Figure 6). The generated C-code can be compiled, linked to the drivers for the specific real-time computer hardware and finally loaded into the real-time computer. Figure 5 shows the setup of the digital controller board and the real servo system.

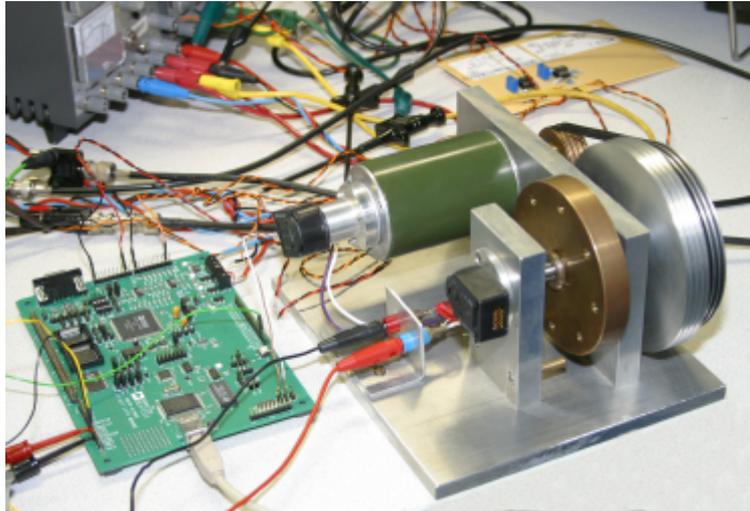


Figure 5 The digital controller board and the real servo system

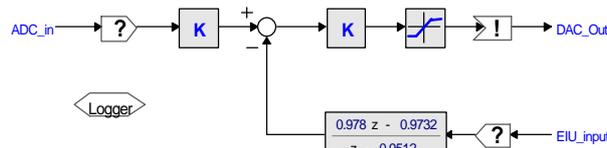


Figure 6 The controller sub model. The controller has been enhanced with the *mappers* for calling the drivers of the interface and with a logging block

After realising and running the controller with the real servo systems, it appeared that the responses of the simulated system and the real system are not completely similar. Closely examining the plots suggests that this could be due to the Coulomb friction that was disregarded in the model. After bringing the Coulomb friction into the simulation model as well, almost similar responses are obtained (Figure 7)

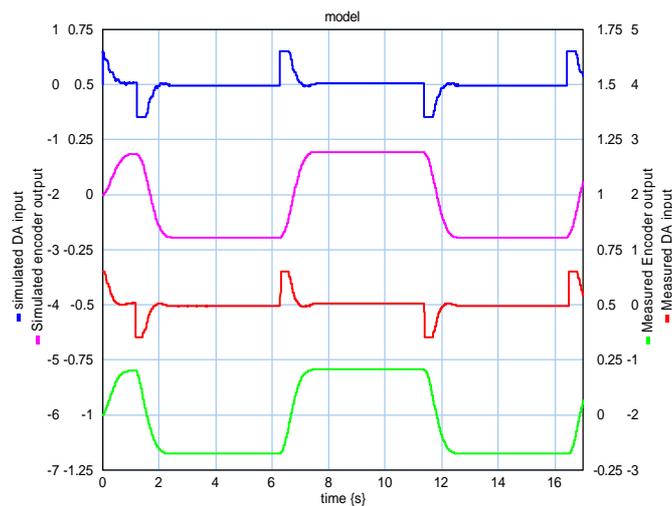


Figure 7 Comparison of simulated (upper two curves) and measured responses (lower two curves). The controller output (motor current) and the motor angle are plotted

5 Conclusions and future work

The modelling and simulation program 20-SIM enables modelling of physical systems such that the simulation model has a close resemblance to the physical reality and with parameters directly related to the physical phenomena. For the design of mechatronic systems this is an attractive feature because it enables to consider the alternatives of changing the construction or the controller. It was also demonstrated that effects like Coulomb friction could easily be added to the model. Furthermore, the new 20-SIM code generation facility allows for graphical programming of real-time control software in C, thus allowing a fast way of prototyping mechatronic control systems.

Future work consists of also constructing C++ templates, exploiting concurrency, and extending the logger to a real-time displayer with also functionality to change controller parameters while the control software runs on the DSP board.

6 Acknowledgements

The authors gratefully acknowledge Gerald Hilderink for work on the software drivers, Marcel Schwitz for hardware support and Controllab Products B.V. for their quick reactions and support while realising the the software for the ADSP board. The donation of Analog Devices is also gratefully acknowledged, which was established via contacts with Carlo Kessels from ACAL NL and Theo Lammerink of the University of Twente.

References

- ADSP (2002), *ADSP 21992 Mixed Signal DSP Data Sheet*,
http://www.analog.com/UploadedFiles/Datasheets/637820587ADSP-21992_pra.pdf,
- Amerongen, J.v. (2002), The Mechatronics Handbook: The Role of Controls in Mechatronics, in:
The Mechatronics Handbook, R. H. Bishop, CRC Press, Boca Raton (FA), USA, 21-1 - 21-17.
- Amerongen, J.v. and P.C. Breedveld (2002), Modelling of Physical Systems for the Design and Control of Mechatronics Systems, *IFAC Professional Briefs, published in relation to the 15th triennial IFAC World Congress* (<http://ifac-control.org>), pp. 1-56.
- Breedveld, P.C. (1985), Multibond-graph elements in physical systems theory, *Journal of the Franklin Institute*, **319**, (1/2), pp. 1-36.
- Broenink, J.F. (1999), 20-Sim software for hierarchical bond-graph/block-diagram models, *Simulation Practice and Theory*, **7**, pp. 481-492.
- Broenink, J.F. and C. Kleijn (1999), Computer-aided design of mechatronic systems using 20-SIM 3.0, *Proc. Proc. 2nd Workshop on European Scientific and Industrial Collaboration WESIC'99*, Newport, United Kingdom, (Ed.), pp. 27-34, ISBN: 1-89927-423-5.
- Broenink, J.F., P.P.L. Regtien and T.S.J. Lammerink (2001), A Mechatronic Design Project Integrating Measurement, Simulation and Transducer Technology, *Proc. Symposium "Virtual and Real Tools for Education in Measurement*, September 17-18, Enschede, Netherlands, P. P. L. Regtien and M. J. Korsten (Ed.), pp. 63-66, ISBN: 90-365-1664-1.
- CLP (2002), *Controllab Products B.V.*, <http://www.20sim.com>,
- EZ-KIT (2002), *EZ-KIT Lite™ for Analog Devices ADSP-2199x DSP Family*,
<http://products.analog.com/products/info.asp?product=2199X-HARDWARE>,
- Heck, B.S., L.M. Wills and G.J. Vachtsevanos (2002), Software tecnology for implementing reusable, distributed control systems, *IEEE Control Systems Magazine*, **23**, (1), pp. 21-35, ISSN: 0272-1708.