View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Online Research Database In Technology

Technical University of Denmark

Using the adaptive blockset for simulation and rapid prototyping

Ravn, Ole

Published in:
Proceedings of IEEE Symposium on Computer Aided Control System Design

Link to article, DOI:
10.1109/CACSD.1999.808657

Publication date:
1999

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Ravn, O. (1999). Using the adaptive blockset for simulation and rapid prototyping. In Proceedings of IEEE Symposium on Computer Aided Control System Design IEEE. DOI: 10.1109/CACSD.1999.808657

DTU Library
Technical Information Center of Denmark

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

MM3-4 3:00

Proceedings of the 1999 IEEE
International Symposium on Computer Aided Control System Design
Kohala Coast-Island of Hawai'i, Hawai'i, USA • August 22-27, 1999

Using The Adaptive Blockset for Simulation and Rapid Prototyping

Ole Ravn, Associate Professor,
Department of Automation, Technical University of Denmark,
Building 326, DK-2800 Lyngby, Denmark,
E-Mail: or@iau.dtu.dk

Abstract

The paper presents the design considerations and implementational aspects of the Adaptive Blockset for Simulink which has been developed in a prototype implementation. The basics of indirect adaptive controllers are summarized. The concept behind the Adaptive Blockset for Simulink is to bridge the gap between simulation and prototype controller implementation. This is done using the code generation capabilities of Real Time Workshop in combination with C s-function blocks for adaptive control in Simulink. In the paper the design of each group of blocks normally fund in adaptive controllers is outlined. The block types are, identification, controller design, controller and state variable filter.

The use of the Adaptive Blockset is demonstrated using a simple laboratory setup. Both the use of the blockset for simulation and for rapid prototyping of a real-time controller are shown.

Keywords: Adaptive Control, Simulation, CACSD, Controller Implementation.

1 Introduction

Control engineers today are faced with the issue of reuse of code, specifically controller code. Rapid prototyping tools are emerging for the commonly used simulation packages, such as Real Time Workshop for Simulink and AutoCode for MatrixX. However, some barrier for the designer still exists in doing a prototype implementation of an advanced controller like an adaptive controller. This barrier is partly due to the fact that no implementation of the standard blocks of an adaptive controller in a version feasible for simulation and automatic code generation exists. In the approach taken here an added benefit is the fact that it is same controller source code used both in simulation and real-time control, which gives an enhanced security.

This research is a continuation of the work reported in (Torp et al., 1994) and (Nørgaard et al., 1995) and complements and extends the ideas of that work. The Integrated Real-Time Control and Simulation Tool (IRCST) which was developed in that context has been used in teaching and research. In the present work the possibilities of using a standard graphical interface, contrary to the textual one that was used in IRCST, are explored. The textual interface of the IRCST system has made the system less intuitive to new users and makes the initial training investment substantial. The main core of algorithms used are the same in IRCST and the present system.

Unlike the situation for system identification where good tools exists (i.e. (Ljung, 1992)) the field of adaptive control is less fortunate. The Adaptive Blockset for Simulink is an attempt to remedy the situation and the basic design of the blockset is not limited to adaptive control enabling its use as a general framework for other types of controllers.

The outline of the paper is as follows, in section 2 the design of the elements of the adaptive blockset is described and in section 3 a laboratory setup is used to demonstrate the use of the blockset both in simulation and real-time control. Section 4 gives conclusion and section 5 references.

2 Adaptive Blockset for Simulink

The intuitiveness of the graphical representation of the standard indirect adaptive controller shown in figure 1 is normally not matched by the equivalent C or MATLAB implementation. (Åström and Wittenmark, 1995), (Åström, 1983)

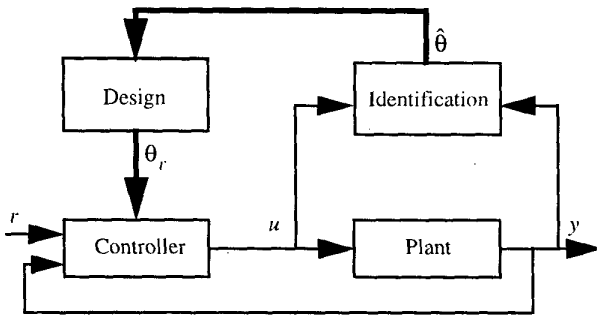

Figure 1: Block diagram of an explicit or indirect adaptive control system.

This fact makes the implementation of adaptive controllers more difficult as a number of programming and real-time issues has to be resolved by the designer.

The aim of the Adaptive Blockset is to transfer this intuitive-

ness to the graphical specification of an adaptive controller in the Simulink environment. This first step makes simulation and experiments simpler as this can be done in a well-known standard environment with easy access to the analysis and design facilities of MATLAB. The second step is to transfer the verified Simulink block diagram of the adaptive controller to a real-time adaptive controller by the use of the code generation tool Real Time Workshop for Simulink (RTW, 1998). In this way the necessary real-time programming expertise is embedded in RTW and made available to the adaptive control designer. Another advantage is the close relationship between the real-time controller code and the simulation code. The automatic code generation makes the task of version control much simpler as just one version of the code has to be maintained.

An example of an adaptive controller built using The Adaptive Blockset is shown in figure 2. Notice the similarity to the schematic diagram show in figure 1.

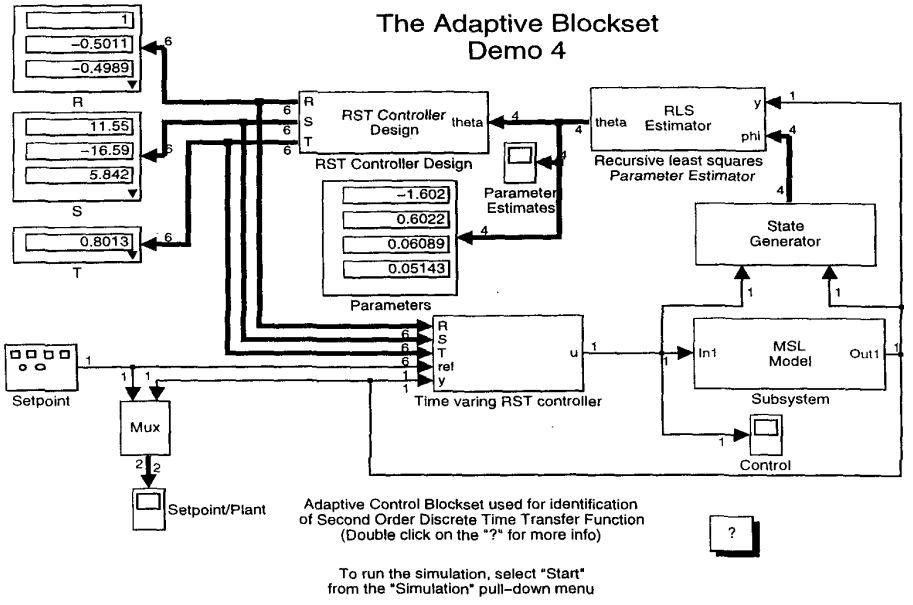

Figure 2: Diagram from the Adaptive Blockset for Simulink.

Most of the blocks provided in The Adaptive Blockset are written as C s-functions (Simulink, 1998). This is due to fact that RTW does not support m-file s-functions and that drawing them using standard Simulink blocks is not feasible. Furthermore the core C code for the adaptive controller blocks were already available in the IRCST project. The blocks are therefore not so easily modifiable by the user as normally seen with Simulink Blocksets. However the use of C s-functions give much flexibility in designing the blocks for instance in taking care of input/output vector width propagation. Furthermore the experienced user can add new blocks to the Blockset based on the underlying C matrix library or other existing C code. (RTW, 1998).

The main block types in the prototype implementation of The Adaptive Blockset are described below.

A. Identification block

In figure 3 the parameter estimation block is shown. It has two inputs, the output signal and the regressor vector. The regressor vector is normally generated by the State Generator block described below. The output is the parameter estimates. The parameters in the dialogue box include a choice of different versions of the recursive least squares algorithm i.e. plain RLS shown below, RLS with UDU factorization and the EFRA algorithm (Salgado et al., 1988).

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \phi^T(t)\hat{\theta}(t-1)) \quad (1)$$

$$K(t) = P(t)\phi(t) = \frac{P(t-1)\phi(t)}{1 + \phi^T(t)P(t-1)\phi(t)} \quad (2)$$

$$P(t) = P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{1 + \phi^T(t)P(t-1)\phi(t)} \quad (3)$$

Initial values for the parameter estimates and the covarians matrix P can be given in several formats. If an empty vector is provided for the initial parameter estimate, parameters for an integrator is used as suggested by (Wellstead and Zarrop, 1991).

$$\frac{B(q)}{A(q)} = \frac{q^{-1}T_s}{1 - q^{-1}} \quad (4)$$

The sample time is also a parameter and is for convenience by default given as the global variable `Ts`. The number of parameters is determined on the basis of the width of the regressor vector used as input. In this way the addition of an element of one to the regressor vector enables the implicit estimation of a load disturbance without having to make changes to the parameter estimation block itself. (Isermann et al., 1992).

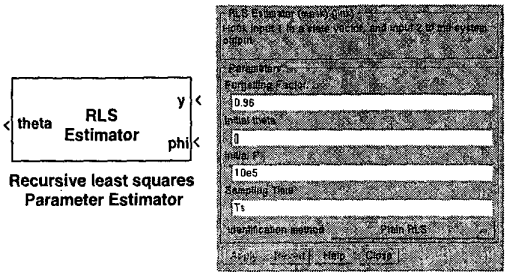

Figure 3: Diagram and dialogue box from the parameter identification block.

B. Controller Design block

Based on the parameter vector as input the controller parameters are computed. The output is the coefficients of the R, S and T polynomials as separate ports. Different designs can be achieved through appropriate choice of the design polynomials.

$$\frac{B_m}{A_m} = \frac{BT}{AR + q^{-d}BS} \tag{5}$$

$$A_m B = AR + q^{-d}BS \tag{6}$$

The choice of designs include but is not limited to pole placement, model following, dead-beat and different flavours of minimum variance. The design polynomials are the desired closed loop denominator `Am`, the observer polynomial `Ao`, a pre-specified part of R called `Rpre`, a pre-specified part of S called `Spre` and the desired closed loop numerator `Bm`. There is a choice of cancellation of the zeros of the system. For instance integral action in the controller can be included by specifying `Rpre` as `[1 -1]`. See (Åström and Wittenmark, 1995) for a more detailed description on controller design.

The solution of the general Diophantine equation at each sampling instance is though flexible quite time consuming for high model orders. This problem can if needed be solve by symbolically solving the appropriate Diophantine equation and just evaluating the resulting expressions. The symbolical manipulation can be automated using for instance packages like Maple. The symbolical solution can even be output as C code from Maple for easy inclusion into the design.

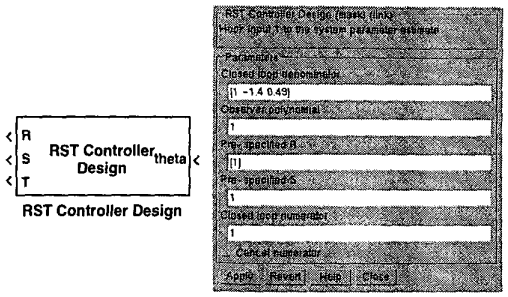

Figure 4: Diagram and dialogue box for the controller design block.

In the examples given below the general Diophantine equation solver has been used.

C. Controller

In the current version of The Adaptive Blockset the well-known 3 term controller is included This controller is sometimes denoted the RST controller because of the names of the polynomials. The inputs are the coefficients of the 3 polynomials and the reference set-point as well as the output signal of the system. The output of the system is the control signal.

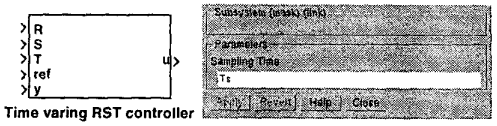

Figure 5: Diagram and dialogue box for the time varying 3-term controller.

D. State Generator

The State Generator block generates a regressor vector as output based on the input u and the output y of the system. The parameters include the model order given as the number of delayed input and output signals to be included in the regressor. The total number of parameters is thus twice the model order. Special versions of the State Generator block enables the user to perform regressor filtering and to output differenced regressors to be used for explicit load estimation as described in (Isermann et al., 1992). See section 3.3.

E. System Block

System block facilitating easy change of system model, from discrete time, over continuous time to non-linear MSL (Mechatronic Simulink Library, (Ravn et al., 1996)) model and finally AD/DA converter blocks for RTW controller generation.

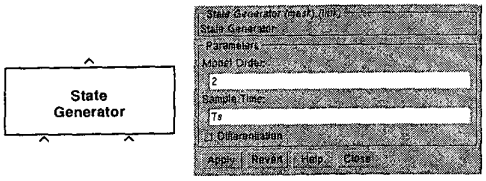

Figure 6: Diagram and dialogue box for the state generator block.

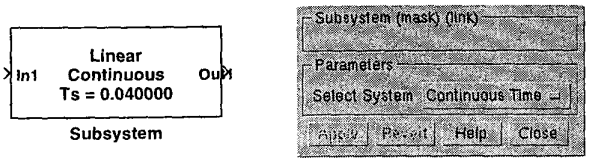

Figure 7: Diagram and dialogue box for the system block.

F. Other blocks in TAB

There are a number of blocks included with TAB for load estimation, fixed parameter three term controller, evaluation of parameters and other performance measures.

3 Application example

To demonstrate the applicability of The Adaptive Blockset, design of an adaptive controller for a laboratory setup has been done. The setup is used as a position servo mechanism having optional process and measurement noise sources and furthermore a load disturbance can be added. The system has a significant dead-band due to friction. This complicates the control of the system because of the non-linearity introduced. A picture of the servo system is shown in figure 8.

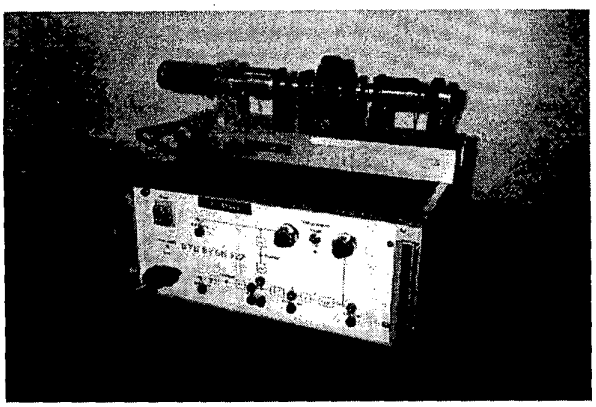

Figure 8: Simple servo mechanism

The control of the system is done using a Motorola MVME-162 system running the OS-9 real-time operating system.

Initially a simulation study is conducted using The Adaptive Blockset and for simulating the servo system a MSL (Mechatronic Simulink Library, (Ravn et al., 1996)) model is used. The MSL system is shown in figure 9.

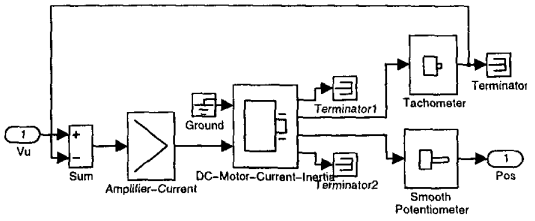

Figure 9: Diagram for the simple servo in MSL.

Two examples has been developed, a parameter estimation experiment and an adaptive control of the servo mechanism. Both simulation and real-time experimental results are shown for each of the two examples. The parameters for the different algorithms are chosen following the standard defaults.

A. Using The Adaptive Blockset for parameter estimation

The adaptive blockset has been used for recursive parameter identification. The setup uses just a few blocks from the blockset. The results of the simulations is shown in figure 10 and of the real-time experiments with the RTW generated code in figure 11. Notice the strong correspondence between the plots. The parameters converges after a few samples and the parameters found in the simulation and the real-time experiments are very close. The sampling time is 40 ms.

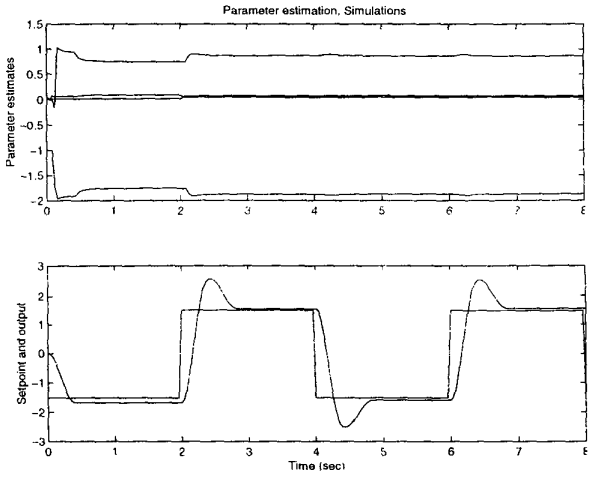

Figure 10: Parameter estimates and reference and system output for the simulated MSL system.

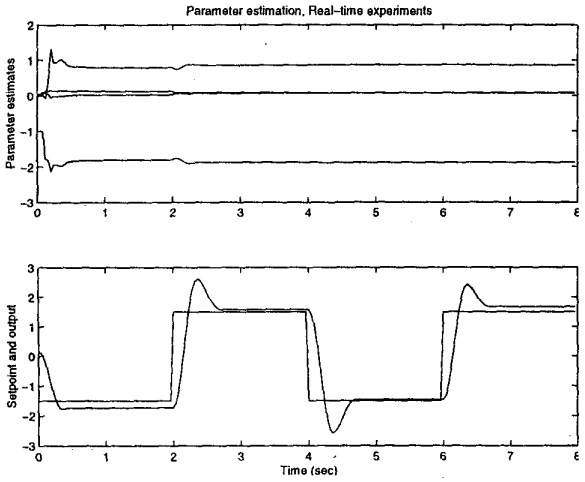

Figure 11: Parameter estimates and reference and system output for the real system controlled with RTW generated controller

B. Using The Adaptive Blockset for adaptive control

The adaptive controller from figure 1 is used here. In figure 12 the simulation results are shown and in figure 13 the results of the real-time experiments done with the RTW generated code are displayed. Again the correspondence is satisfactory after some difference in the initial transitent phase due to different initial conditions.

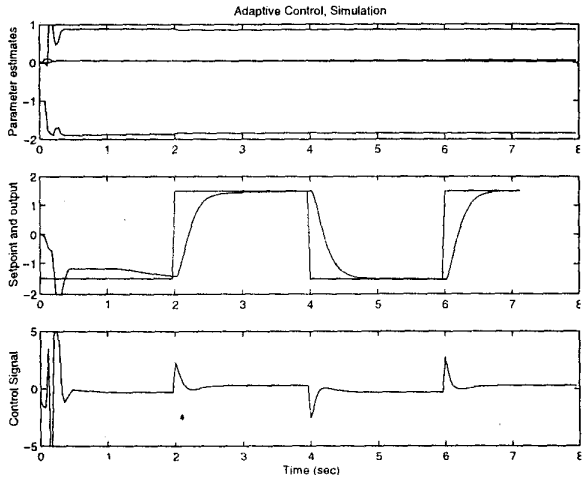

Figure 12: Parameter estimates, reference and system output and control signal for the simulated MSL system.

C. Using TAB for load estimation

To show the simple inclusion of additional features in the simulation in figure 14 the diagram for parameter estimation in the presence of load disturbances on the system. The system can be described as:

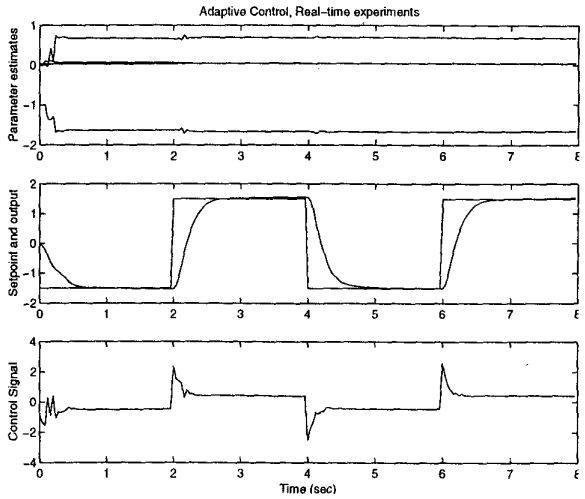

Figure 13: Parameter estimates, reference and system output and control signal for the real system controlled with RTW generated controller.

$$A(q^{-1})y(t) = B(q^{-1})u(t) + d + e(t) \qquad (7)$$

Two methods are implemented (described in greater detail in (Isermann et al., 1992)) implicit load estimation where the regressor vector is argmented by an element fixed to 1. The corresponding parameter estimate is the load estimate d.

$$\phi(t) = \left[-y(t-1) \ \ldots \ -y(t-n) \ u(t-1) \ \ldots \ u(t-m) \ 1\right]^T \qquad (8)$$

$$\hat{\theta} = \left[\hat{a}_1 \ \ldots \ \hat{a}_n \ \hat{b}_1 \ \ldots \ \hat{b}_m \ \hat{d}\right]^T \qquad (9)$$

The disadvantage of this method is that the dynamics of the load disturbance need to match that of system, otherwise the RLS forgetting may shut down the estimation of the load after some time. The estimate of the load can be used in a feed forward or the controller can be of integral type.

The other method considered here is the so called explicit load estimation. The idea is that equation (7) is multiplied by the differens operator $\Delta = 1 - q^{-1}$ cancelling d. That means, Du and Dy are used in the regressor vector and the prediction error is replaced by the error: $\tilde{e}(t) = \Delta y(t) - \Delta\phi^T(t)\theta$. The DC-load is then estimated by a weighted sum:

$$\hat{d} = w\hat{d} + (1-w)(y(t) - \phi^T(t)\theta) \qquad (10)$$

w being a weight factor chosen as for instance 0.8-0.9. The resulting TAB diagram is shown in figure 14

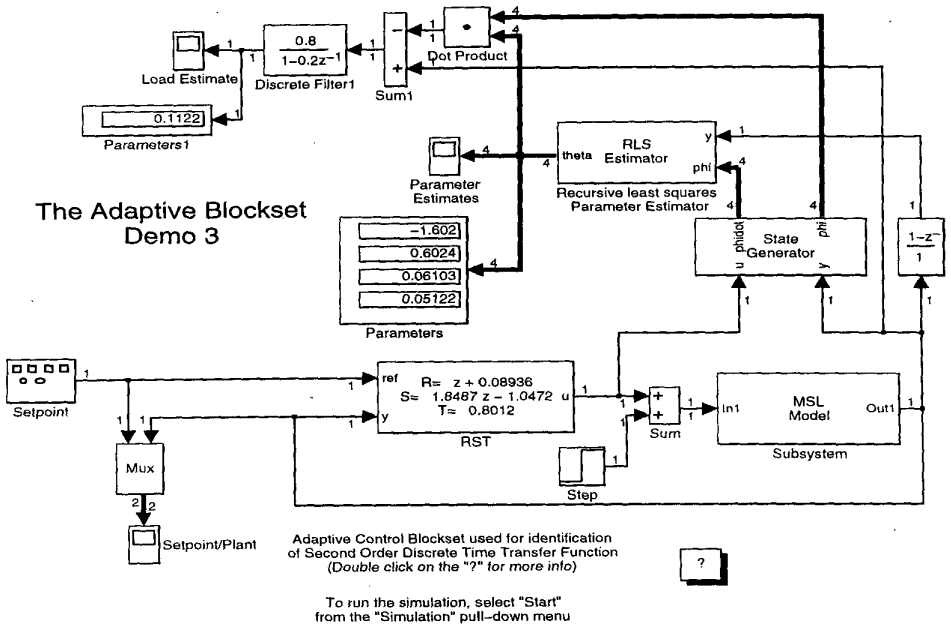

Figure 14: Diagram from The Adaptive Blockset for Simulink. Explicit load estimation

4 Conclusion

The paper has presented the design consideration for The Adaptive Blockset for use with Simulink. An application example has shown the use of the blockset and demonstrated the usefulness of generating the prototype controller code from the simulation model of the controller.

TAB has been used in the practical work as well as for teaching adaptive control and has performed very well. Especially the ability to automatically generate real time code using RTW has been greatly appreciated. The performance of the RTW generated real time code for an the adaptive controller described above was sufficient, even though no special measures has been taken in order to optimize the C-code for the TAB blocks.

Future developments include more blocks supporting for instance direct adaptive control, GPC and delta domain adaptive controllers. Future versions of Simulink will hopefully include better support for vector width propagation and variable mask layout as this will make the future versions of TAB even more streamlined.

A prototype evaluation version of The Adaptive Blockset (TAB) is made available for download from the World Wide Web:

```
http://www.iau.dtu.dk/~or/tab.html
```

References

Åström, K. J. (1983). Theory and application of adaptive control - a survey. *Automatica*, 19(5):471–486.

Åström, K. J. and Wittenmark, B. (1995). *Adaptive Control*. Addison Wesley, second edition.

Isermann, R., Lachmann, K., and Matko, D. (1992). *Adaptive Control Systems*. Systems and Control Engineering. Prentice Hall.

Ljung, L. (1992). *System Identification Toolbox*. The Mathworks Inc.

Nørgaard, P. M., Torp, S., Christensen, A., and Ravn, O. (1995). A CACE tool for analysis and design of adaptive control systems. In *Proceedings of Symposium on Adaptive Systems in Control and Signal Processing , ACASP95*, Budapest, Hungary.

Ravn, O., Szymkat, M., Uhl, T., Betemps, M., Pjetursson, A., and Rod, J. (1996). Mechatronic blockset for simulink, - concept and implementation. In *Proceedings of the 1996 IEEE Symposium on Computer Aided Control System Design CACSD'96*, Dearborn, MI, USA.

RTW (1998). *Real-Time Workshop, User's Guide*. The Mathworks Inc. Version 2.2.

Salgado, M. E., Goodwin, G. C., and Middleton, R. H. (1988). Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control*, 47(2):477–491.

Simulink (1998). *SIMULINK, Using Simulink*. The MathWorks Inc. Version 2.2.

Torp, S., Nørgaard, P. M., Christensen, A. C., and Ravn, O. (1994). Implementational issues in CACSD. In *Proceedings of the 1994 IEEE Symposium on Computer Aided Control System Design CACSD'94*, Tucson, Arizona, USA.

Wellstead, P. and Zarrop, M. B. (1991). *Self-tuning Systems, Control and Signal Processing*. Wiley.