DIRECT IMPLEMENTATION OF SELF-TUNING LQ CONTROLLER FROM SIMULINK INTO B&R PLC

Martin Dvořáček
Doctoral Degree Programme (1), FEEC BUT
E-mail: xdvora73@stud.feec.vutbr.cz

Supervised by: Petr Pivoňka
E-mail: pivonka@feec.vutbr.cz

ABSTRACT

In this article, automatic build ANSI C source code from simulink model is realized through the Real-Time Workshop Embedded Coder toolbox for Simulink. Created source code is implemented to PLC and tested in control of physical model. Automation runtime target for Matlab and Automation Studio 3.0 from B&R was used for programming PLC.

1. INTRODUCTION

Control algorithm implementation is time consuming. Development and testing can be separated into several phases. SIL (software in the loop) and PIL (processor in the loop) phase are widely known. Correctness of algorithm is verified using SIL. PIL verify implementation on target HW platform. For implementation in target processor must be algorithm re-write into target source code. For portability between simulation and development environment is good idea implement control algorithm in ANSI C. Make hand-written mistake-free ANSI C source code is difficult and time-consuming.

2. ALGORITHM DEVELOPMENT

In Simulink C language is supported in S-function block. MATLAB/Simulink can be used for create and debug the algorithm. In first step of development are the algorithms and numerical methods implemented using MATLAB M-files or Simulink M-files S-Function. Firstly these are verify in simulation and then in real-time simulation with real plane control. After checkout M-files and MATLAB functions are rewrite to the C source code. This must by repeatedly verify in simulation and real-time control. After debagging may by implemented into target PLC for PIL test and then can be used for control of real plane. Time for algorithm development and implementation can by rapidly reduced if automatic generating of source code may by used.

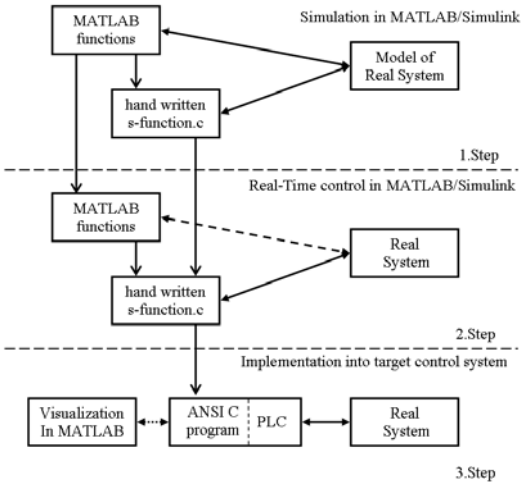

Figure 1: Scheme for hand written develop algorithm in MATLAB/Simulink

2.1. DEVELOPMENT WITH REAL-TIME WORKSHOP EMBEDDED CODER

Using Real-Time Workshop Embedded Coder ANSI C source code in production quality can by created automatically. Source code can by automatically created from simulink model or its parts (Embedded MATLAB function block typically). Embedded MATLAB Function block in simulink model must by use instead M-file s-function or MATLAB function block. Syntax and majority of MATLAB functions can by used in this block.

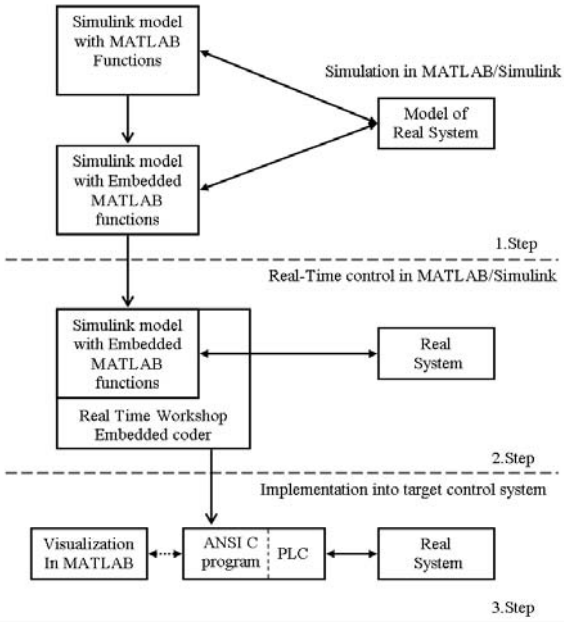

Figure 2: Algorithm develop using Real Time Workshop Embedded coder

2.2. B&R AUTOMATION RUNTIME TARGET FOR MATLAB

This SW is MATLAB assistance which cooperates with Real-Time Workshop Embedded Coder and development environment of B&R PLC (Automation Studio 3.0). Implements automatically transfer generated C source code into requested project in Automation Studio 3.

3. SELF-ADAPTING LQ CONTROLLER

The purpose of adaptive controllers is to adapt parameters of control law to changes of the controlled system. Controller is separated into two main parts: identification and controller. Recursive least mean squares (RLS) and identification based on neural network(NN) are used. And the linear quadratic optimal controller is used as the control algorithm. Figure 3 shows the architecture of the controller where *w* denotes desired value, *u* action value and *y* output of the controlled system.

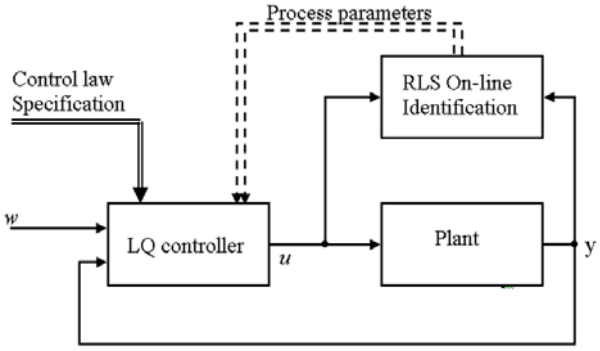

Figure 3: Architecture of self-tuning LQ controller

3.1. ON-LINE IDENTIFICATION

The model 3. order is used in this case.

$$F_M\left(z^{-1}\right)=\frac{b_1 z^{-1}+b_2 z^{-1}+b_3 z^{-1}}{1+a_1 z^{-1}+a_2 z^{-1}+a_3 z^{-1}} \tag{1}$$

The model can be written in vector forms as follows

$$\hat{y}(k)=\varphi^T(k)\theta(k) \tag{2}$$

where

$$\varphi(k)=\left[u(k-1)\ u(k-2)\ u(k-3)\ -y(k-1)\ -y(k-2)\ -y(k-3)\right]^T \tag{3}$$

is the vector of measured inputs and outputs and

$$\theta(k)=\left[b_1(k)\ b_2(k)\ b_3(k)\ a_1(k)\ a_2(k)\ a_3(k)\right]^T \tag{4}$$

is vector of estimated system parameters.

3.2. RLS ALGORITHM

The method is widely used and is easy to implement. Vector of estimated parameters is computed at every step by equation

$$\theta(k)=\theta(k-1)+K(k-1)\left[y(k)-\varphi(k)^T\theta(k-1)\right] \tag{5}$$

where *y(k)* is process output. *K* is computed as follows

$$K(k)=\frac{P(k-1)\varphi(k)}{\lambda+\varphi(k)^T P(k-1)\varphi(k)} \tag{6}$$

where λ is forgetting factor and *P(k)* is covariance matrix

$$P(k)\frac{P(k-1)-K(k)\varphi(k)^T P(k-1)}{\lambda} \tag{7}$$

3.3. IDENTIFICATION BASE ON NN WITH LEVENBERG-MARQUART LEARNING

This method is numerical solution of minimization sum of squares generally nonlinear function. Inputs and outputs from process are accumulated to matrix

$$X(k)=\left[\varphi(k) \quad \varphi(k-1) \quad \ldots \quad \varphi(k-2)\right] \tag{8}$$

Iterative algorithm is given by equation

$$\theta(i\mid k)=\theta(i\mid k-1)-\left[J^T(i\mid k)J(i\mid k)+\lambda I\right]^{-1}J(i\mid k)E(i\mid k) \tag{9}$$

Where *k* is step, *i* is iteration in step, λ is optional parameter which determine evolution of prediction error and *E(i|k)* is error vector

$$E(i\mid k)=T^T(k)-X^T(k)\theta(k) \tag{10}$$

T(k) is vector of last values of process outputs

$$T(k)=\left[y(k) \quad y(k-1) \quad y(k-2) \quad y(k-3)\right] \tag{11}$$

Jacobian matrix *J* is evaluated in each iteration [5]

$$J(i\mid k)=\frac{\partial E(i\mid k)}{\partial\theta(i\mid k)}=\frac{\partial\left(T^T(k)-X^T(k)\theta(k)\right)}{\partial\theta(i\mid k)}=-X^T(k) \tag{12}$$

3.4. LQ CONTROLLER

Linear quadratic controller is state controller with feedback proportional gains from process states. Quadratic criterion without end state is

$$J=\sum_{k=0}^{\infty}\left(x^T(k)Qx(k)+u^T(k)Ru(k)\right) \tag{13}$$

where matrix *R* scale control energy and matrix *Q* scales error of system states. Input output description is get from identification. State model observer is computed from them [5]. Control signal is given by

$$u(k)=-K_{LQ}x(k) \tag{14}$$

K_{LQ} vector is solved as follow

$$K_{LQ}=\left[R+B^T P_{LQ}B\right]^{-1}B^T P_{LQ}A \tag{15}$$

$$P_{LQ}=Q+K_{LQ}^T RK_{LQ}+\left(A-BK_{LQ}\right)^T P_{LQ}\left(A-BK_{LQ}\right) \tag{16}$$

3.5. FINAL IMPLEMENTATION

Validation of controller was realized using SIL, PIL and real-time control physical plan from Simulink. Real Time Workshop Embedded coder was used for building C source

code from simulink model. Only this generated source code was compiled and transferred into control PLC. Physical model was in control. Results are on Figure 4.

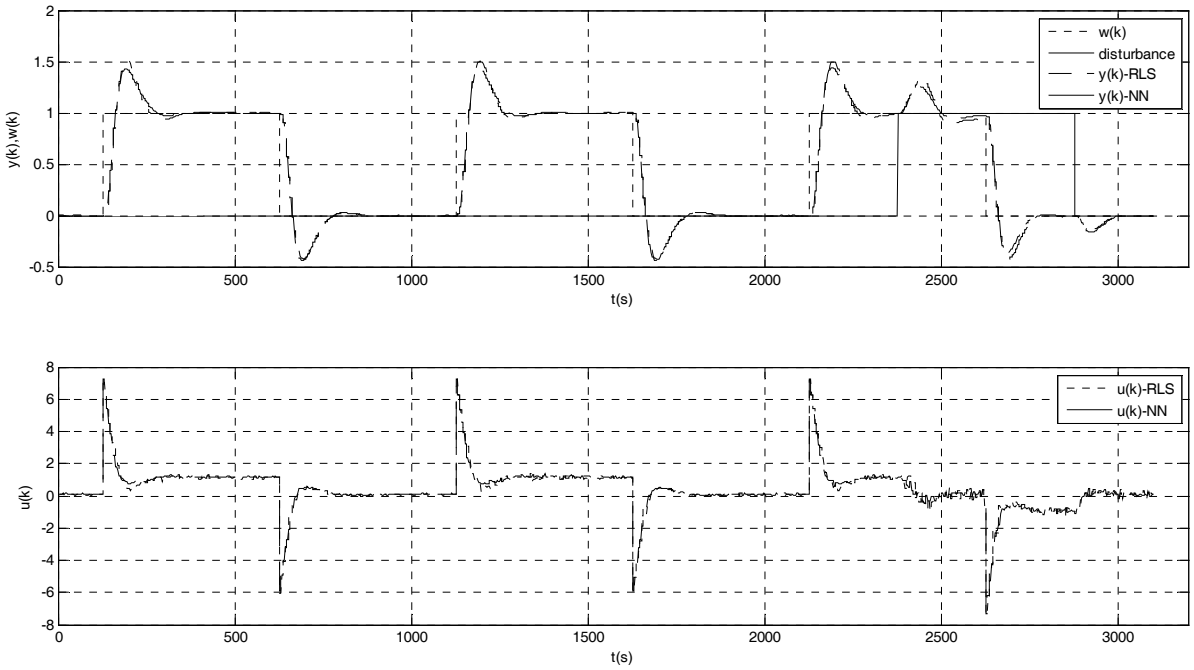

Figure 4: Results of implemented self-tuning LQ controller with RLS and NN identifications based. At the top desired value and output of controlled system. Bellow action value.

4. CONCLUSION

This paper verified possibility use Real Time Workshop Embedded coder for automatically created ANSI C source code from simulink model. Self-tuning LQ controller was created in simulink. Validate in simulation and real-time control with physical model. Consequently the C source code was built using Real Time Workshop Embedded coder and implement to the target control system (B&R PLC).

ACKNOWLEDGEMENT

The paper has been prepared as the solution by the Czech Ministry of Education in the frame of MSM MSM0021630529 Intelligent Systems in Automation.

REFERENCES

[1] The MathWorks: Real-Time Workshop Embedded Coder 5, User's Guide

[2] The MathWorks: On-line documentation, http://www.mathworks.com

[3] B&R Industrie-Elektronic: B&R Automation Runtime Target for Simulink

[4] Veleba, V. Pivoňka, P.: Adaptive controller with identification Based on Neural Networks for Systems with Rapid Sampling Rates. WSEAS Transactions on Systems, 2005, vol. 4. issue 4, pp.385-388, ISSN 1109-2777

[5] Dvořáček, M.: Estimátor v systému regulace s proměnnou strukturou, Diploma thesis in Czech, FEEC BUT, Brno 2008