Real Time & Power Efficient Adaptive - Robust Control

Lavinius Ioan Gliga, Cosmin Constantin Mihai, Ciprian Lupu, Dumitru Popescu

gliga.l.ioan@hotmail.com, mihai.cosmin2007@hotmail.com,
ciprian.lupu@acse.pub.ro, dumitru.popescu@acse.pub.ro

gliga.l.ioan@hotmail.com

Abstract. A design procedure for a control system suited for dynamic variable processes is presented in this paper. The proposed adaptive - robust control strategy considers both adaptive control advantages and robust control benefits. It estimates the degradation of the system’s performances due to the dynamic variation in the process and it then utilizes it to determine when the system must be adapted with a redesign of the robust controller. A single integral criterion is used for the identification of the process, and for the design of the control algorithm, which is expressed in direct form, through a cost function defined in the space of the parameters of both the process and the controller. For the minimization of this nonlinear function, an adequate mathematical programming minimization method is used. The theoretical approach presented in this paper was validated for a closed loop control system, simulated in an application developed in C. Because of the reduced number of operations, this method is suitable for implementation on fast processes. Due to its effectiveness, it increases the idle time of the CPU, thereby saving electrical energy.

1. Introduction

When dealing with processes with dynamic variables, classical control designs are no longer suitable. If a process evolves rapidly, or is subject to structural disturbances, it may quickly go into states that were not accounted for when designing the controller, thus making it impossible to obtain the desired performance or even stability. Several design strategies exist to deal with these kind of systems, the most important ones being adaptive and robust control. Both involve the presence of a high-level supervisor in the control system, which will handle the evolutions of the process. In the case of adaptive control, the supervisor computes a new algorithm at every sampling moment, the strategy being very potent when used on processes that evolve quickly over time. Robust control is used for processes with structural disturbances: a single control algorithm is computed in the design phase, which only works if the disturbances do not change the process’ behaviour too much. Obviously, robust control does not work for highly dynamic processes, but it is cost effective when implemented on predictable systems (like train controls).

To design control system, one must first identify the dynamic model $M$ of the process $P$, and then calculate the control algorithm $C$. The pair $(C, M)$ defines the nominal system in fig. 1.

![Figure 1. The nominal System](image)

The mathematical model $M$ is identified by solving the optimization problem [1][2]

$$
\min_{C} J_f(C_{\text{fixed}}, M)
$$

where $J_f$ is the identification criterion built using the steady state error and the constant controller.

The calculation of the control algorithm is based on the identified model and is obtained as the solution to the minimization problem.
\[ \min C(M, M_{\text{fixed}}) \tag{2} \]

with \( M \) fixed and \( J_C \) being the optimal criterion built using the steady state error of the close loop system. The control algorithm design for the nominal system implies the successive solving of these two different optimization problems.

However, if the integral criterion would be identical for the two problems, they could be solved in a unified manner, using the same method, as will be detailed in subsequent sections. Compared with other system identification and controller design methodologies, where two different procedures are used: one for identification and another one for computing the controller, this new method has multiple advantages: easier implementation (only one function can achieve multiple ends), easier code maintenance and lower learning curve for system and/or design engineers who need to maintain and/or design such a system.

The main objective is to ensure that the real system represented by the pair \((C, P)\), from fig. 2, has optimal performance at any given time.

![Figure 2. The real System](image)

For the real process \( P \), with dynamic variable parameters, either the adaptive strategy or the robust one are recommended for implementation. Adaptive control [3][4] implies the (re)identification of the process’ model and the (re)design of the control algorithm at every sampling moment: for a new model, only a new controller can preserve the system’s performances. From the \( k \) to the \( k+1 \) sampling moments, adaptive control strategy can be summarized as

\[ (C, M)_k \rightarrow (C, M)_{k+1} \tag{3} \]

Robust control [5] is recommended for processes with parametric or structural disturbances. It requires a single controller to ensure the desired performances. It is suitable for a class of models that must reside near the process [6], which has the same parameters values as the ones used for designing the controller. The performance degradation of the resulting closed loop system is formally expressed by equation (4), with distance \( \delta \) (being the performance degradation threshold) small enough

\[ \left| \frac{CP}{1 - CP} - \frac{CM}{1 - CM} \right| < \delta \tag{4} \]

If the degradation becomes too great, the controller will no longer ensure the needed performance, making robust control inappropriate for system with highly dynamic variable.

We propose a combined adaptive – robust [7] algorithm design methodology, built on the advantages of the two previous strategies, which also eliminates their disadvantages. Compared to classical adaptive strategy, the new procedure preserves nominal performance for the physical system by (re)identifying only in case of significant performance degradation, not at each sampling time. This clearly reduces the numerical calculations, thus increasing the idle time of the processor, saving electrical energy. The robust strategy is used to compute a single control algorithm, tolerant to disturbances. This controller is suited for a whole range of process models. The control algorithm remains unchanged if the degradation condition – inequality (4) remains satisfied. The controller is...
then recalculated after the identification of the dynamic model. This eliminates the computation of the control algorithm at every sampling moment, by imposing a reserve of robustness.

2. Adaptive – Robust Control

An optimal criterion [8] (integral criterion for optimal control) is considered and we propose an iterative design procedure, where the two problems of identification and control are intertwined (playing the main role, one after another):

- Using the current available controller, the next model \( M_{i+1} \) is determined, if it is necessary
- Using the obtained model, the subsequent controller \( C_{i+1} \) is computed

The connection between the two problems, identification and control, is illustrated by the recursive mechanism expressed in (1) and (2).

The integral optimality criterion \( J \) and its absolute numerical value are considered. For process \( P \), model \( M \) and controller \( C \), (4) it can be written

\[
|J(C, P) - J(C, M)| < d
\]

where

- \( J(C, P) \) is the measurement of the real system’s performance
- \( J(C, M) \) is the measurement of the nominal system’s performance
- \(|J(C, P) - J(C, M)| \) is the measurement of the performances’ degradation, due to changes in the process
- \( d \) is the performance degradation threshold, but scaled for the difference in integral criterions

Considering (5), the following necessary condition must be satisfied for the nominal performance is achieved to be achieved by the real system

\[
||J(C, P) - J(C, M)| \leq |J(C, M)|
\]

where \(|J(C, M)| < \delta \). \(|J(C, M)| \) has larger values when a change occurs in the process, therefore behaving like a classical robustness margin, but converges to 0 when the parameters are constant, thus providing more sensitivity to future unexpected variations.

The restriction (6) expresses the robust behaviour of the algorithm, since the degradation is bordered by the low values provided by \(|J(C, M)|\). In addition, if inequality (8) is unsatisfied, the performance of the closed loop system has degraded so much that the process needs to be identified again and a new controller must be computed.

The \( C_i \) is considered, available for the current model \( M_i \), and the next replica is computed [9]

\[
M_{i+1} = \arg\min_M [J(C_i, M) + J(C, P)]
\]

For model \( M_{i+1} \) the following controller is computed

\[
C_{i+1} = \arg\min_C J(C, M_{i+1})
\]

(7) and (8) are solved if (6), which is being checked at each sampling moment, is unsatisfied.

The system needs to be adapted (i.e. the controller must be recalculated) when (4) is no longer satisfied, which means that the parameters of the process have changed so much, that
the closed loop performance of the real system is too different from the nominal one’s. Thus, a new model of the process is identified and used to compute a robust controller. This regulator should be able to passively accommodate changes in the process, in the vicinity of the current one for which it was designed [6], thus minimizing the number of necessary adaptations.

3. The adaptive – robust control approach
The proposed approach is presented for a first order process controlled by a PI regulator. While a multitude of different processes can be described by such transfer functions, there are many more which cannot. However, the design methodology remains the same, just the number of mathematical calculations increases with the grade of the process’ transfer function and of the controller’s. The authors did not find including long mathematical calculations and formulas necessary or useful for understanding the proposed approach. While they may help others integrate the solution more easily into their different application, it also makes the paper more difficult to understand. The authors are always ready to help anyone with using the proposed methodology on more complicated systems with different controllers. The mathematical formulations for the general case may be presented in a future paper.

A process expressed using a first order transfer function is considered [10][11]

\[
P = \frac{K}{T_s + 1}
\]  \hspace{1cm} (9)

and the PI control algorithm which will be used for the process

\[
C = K_p (1 + \frac{1}{T_i s})
\]  \hspace{1cm} (10)

The integral criterion \( J \) used in (7) and (8) is

\[
J = \int_0^{\infty} \varepsilon^2(t) + T_e \varepsilon^2 \dot{\varepsilon}(t) dt
\]  \hspace{1cm} (11)

where the first part of the criterion is used to minimize the response time, while the second one is needed to control the overshoot – these two performance indices being very important when designing a control system. While others, such as frequency specifications or command signal constraints, may appear, they are not usually as important as the ones considered by this criterion. The criterion will also be minimized, thus imposing a zero value of the steady state error, the most important parameter in performance specifications.

The criterion is, however, useless in its current form, as it is impossible to implement in an effective way on actual microcontrollers, so it has to be rewritten using algorithm from [10] to its direct form, expressed in the space of the model and controller parameters.

First, the terms of the integral are separated, and each will be computed separately.

\[
J = \int_0^{\infty} \varepsilon^2(t) + T_e \varepsilon^2 \dot{\varepsilon}(t) dt = \int_0^{\infty} \varepsilon^2(t) dt + T_e \int_0^{\infty} \dot{\varepsilon}^2(t) dt = J_1 + T_e J_2
\]  \hspace{1cm} (12)

The Laplace transform is applied
\[ J_1 = \int_0^\infty \varepsilon^2(t)dt = \int_0^\infty \varepsilon(t)\varepsilon(t)dt = \int_0^\infty \varepsilon(t)[L^{-1}[\varepsilon(s)]]dt \]
\[ J_1 = \int_0^\infty \varepsilon(t) \left[ \frac{1}{2\pi j} \int_{-j\infty}^{j\infty} \varepsilon(s)e^{st}ds \right] dt \]
\[ J_1 = \frac{1}{2\pi j} \int_{-j\infty}^{j\infty} \varepsilon(s) \left[ \int_0^\infty \varepsilon(t)e^{st}dt \right] ds = \frac{1}{2\pi j} \int_{-j\infty}^{j\infty} \varepsilon(s)\varepsilon(-s)ds \]

The steady state error is written as
\[ \varepsilon(s) = \frac{1}{1 + H_d(s)} R(s) = [1 - H_o(s)]R(s) \]  (14)

Therefore, the first part of the criterion can be written as
\[ J_1 = \frac{1}{2\pi j} \int_{-j\infty}^{j\infty} [1 - H_o(s)][1 - H_o(-s)]R(s)R(-s)ds \]  (15)

Before formulating the direct expression of \( J_1 \), the following condition must be met: the difference between the grades of the denominator and the numerator of the steady state error’s transfer function must be at least 1. This condition will be tested for a first order process controlled with a PI algorithm. The open loop transfer function of the resulting system is
\[ H_d = \frac{K_r(T_i s + 1)}{s T_i} \]
\[ K \frac{K K_r(T_i s + 1)}{s T_i(T s + 1)} = \frac{M(s)}{s N(s)} \]  (16)

and the steady state error is
\[ \varepsilon(s) = \frac{N(s)}{s N(s) + M(s)} = \frac{T_i(T s + 1)}{s T_i(T s + 1) + K K r(T_i s + 1)} = \frac{T_i T s + T_i}{T_i T s^2 + (1 + K K r)T_i s + K K r} \]  (17)
\[ \varepsilon(s) = \frac{b_2 s + b_0}{a_2 s^2 + a_1 s + a_0} \]

The direct form of the criterion’s first term is, per [10],
\[ J_1 = \frac{T_i + K_r K T}{2 K_r K (1 + K r K)} \]  (18)

The second part of the criterion follows roughly the same steps, but the condition applied to the transfer function of the steady state error is checked upon the error’s derivative
\[ L[\dot{\varepsilon}(t)] = \frac{-M(s)}{s N(s) + M(s)} = \frac{-K K_r(T_i s + 1)}{s T_i(T s + 1) + K K r(T_i s + 1)} = \frac{-K K_r T_i s - K K_r}{T_i T s^2 + (1 + K K r)T_i s + K K r} \]  (19)
\[ L[\dot{\varepsilon}(t)] = \frac{c_1 s + c_0}{a_2 s^2 + a_1 s + a_0} \]
The direct expression for the second part of the integral criterion, according to [10], is

\[ J_2 = \frac{K_r K (1 + K_r K T_i)}{2T_i (1 + K_r K)} \]  

(20)

Overall, the \( J \) integral criterion is equal to

\[ J = J_1 + J_2 = \frac{T_i + K_r KT}{2K_r K (1 + K_r K)} + \frac{T^2 e^2}{2T_i (1 + K_r K)} \]  

(21)

The minimization of the above non-linear criterion is accomplished using an appropriate nonlinear direct method [12][13]. The Nelder – Mead [14] procedure was chosen because of its good performance for this class of problems. It is based on the SIMPLEX [15] algorithm.

As it can be seen the controller is designed in continuous time. For an implementation, it must be discretized, using one of the known methods: Euler, Tustin, etc.

The robustness of the controller refers to the different changes in the process that it can accommodate, while first maintaining the stability of the real system and then the attained performances. The same principle which is described in [6] will be used here. It will be detailed in the next section.

By knowing the parameters of the process, this nonlinear optimization method can be used to minimize the criterion \( J \) to compute the parameters of the controller, and the value of the integral criterion.

The opposite is also applicable: by knowing the parameters of the controller and the integral criterion’s value, the parameters of the process can be obtained. It is impossible to know the exact value of the integral criterion, when considering a real application for a real process; therefore, it has to be estimated. The same estimation will be used for the degradation condition. Using a Riemann sum [16], the integral criterion will be rewritten as

\[ J = \int_0^\infty \varepsilon(t) + T^2 e \dot{\varepsilon}(t) dt \equiv \hat{J} = T^2 e^2 \cdot \varepsilon(N) \cdot h + \sum_{k=1}^\infty \varepsilon(k) \cdot h \]  

(22)

where \( N \) is the current sampling moment, translating the criterion from the continuous time domain to the discrete one. As the discrete criterion depends only on the steady state error, its value can be computed fast and easy, at each sampling moment.

To sum it all up, the proposed control algorithm is:

1. At each sampling moment, compute the integral criterion’s value

\[ \hat{J} = T^2 e^2 \cdot \varepsilon(N) \cdot h + \sum_{k=a}^b \varepsilon(k) \cdot h \]  

(23)

where “\( a \)” is the sampling moment when the last identification of the process was performed, and “\( b \)” is the current sampling moment.

2. If the degradation condition

\[ |\hat{J} - \hat{J}^*| < \hat{J}^* \]  

(24)
where $\hat{f}^*$ is the integral criterion’s value computed using the last identified parameters of the process and computed parameters of the controller, is unsatisfied, then estimate the current model of the process

$$M_{\text{new}} = \arg \min_M [J(C_{\text{old}}, M) + \hat{f}]$$

(25)

doing the Nelder-Mead method for minimizing the criterion.

At every sampling moment, the performance of the real system is estimated using the discretized version of the integral criterion. Also, the performance degradation (the difference between the estimated performance of the real system and known one of the nominal system) is checked. If it is too great, then a new model of the process is reidentified using the current parameters of the controller and the degradation’s value, by minimizing the integral criterion. After the model is obtained, its parameters are utilized to compute the new parameters of the controller, that is suited for that model. The performance value of the nominal system is remembered (it being computed together with the new regulator) and this will be used to check the performance degradation of the real system. Thus, the number of adaptations is reduced, a re-computation being performed only when necessary.

4. Implementation

To test the proposed control system, we developed a software process simulator. The application, developed in C [17] using the LabWindows/CVI [18] IDE and the Nlopt library [19], validated the theoretical approach proposed in this paper. The name of the program is CAR (“Control Adaptive – Robust” = “Adaptive – Robust Control”) and it simulates the response of a closed loop system, being composed of a first order process and a PI controller, at a given sampling period.

The user can modify any of the following parameters whenever he/she desires:

- The parameters ($K, T$) of the process $P$ (they are the real parameters of the real process). Whenever this pair of parameters is changed, the real value of the integral criterion “J4” is (re)evaluated, using the current parameters of the controller. The parameters correspond to the gain and time constant of a first order transfer function. The process’ discrete equivalent is computed automatically, as is for the controller.
- The sampling period
- The weighting of the overshoot
- The robustness (degradation) coefficient
- The reference

The UI informs the operator about the estimated parameters of the process (“K_e” and “T_e”), the integral criterion’s values estimated for the process (“J4”) and computed for the model (“J4_e”), the current steady state error and the evolution of the process in the last 100 sampling moments, via the graph in the upper right side of the UI.

The workflow of the program is the following:

1. The user sets the initial values, then starts the simulation by pressing the button located in the lower middle part of the UI.
2. The process is assumed to be properly identified, so the estimated parameters \((K_e, T_e)\) are set to be equal with the real parameters of the process. The PI controller is computed \((K_r, T_i)\) using the estimated values of the parameters of the process, and the simulation starts, using the sampling period provided by the user.

3. At each sampling moment, the algorithm estimates the value of the integral criterion and checks the degradation condition. If it is not satisfied:
   a. Use the estimated value of the integral criterion, and the current controller to compute the estimated parameters of the process
   b. Calculate a new controller, using the previously estimated parameters, and record the resulting value of the integral criterion (“J4_e” in the UI)

4. Compute the command to the process and the output of the process. Display the steady state error. The desired (red line) and the real (yellow line) output of the process are displayed on the graph, in the upper right side of the UI.

![Figure 3. Application UI](image)

Multiple tests were performed on multiple processes. The user can input any parameters for a first order transfer function, subject to certain reasonable constraints (which in no way diminish the generality of the proposed approach), which will be presented later.

The user can input the parameters of a continuous first order transfer function, and the indicators will display the estimated parameters of the model and the controller’s parameters in continuous time. However, the processes and controller are discretized for the implementation. After the user modifies a parameter of the process, the discrete time equivalent is computed. When a new controller is recomputed, it is immediately discretized. These are not time consuming operations, because the discrete equivalents are already hard coded, and the new parameters are directly used in them, as constants.

Many real-world processes can be described using a first order transfer function. Among these are the evacuation of liquid from a reservoir, the flow of a liquid through short and long pipes, etc. In the case of Figure 4, we chose, for testing, an air heater. It’s transfer function is
\[ H(s) = \frac{5}{10s + 1} \]  

(27)

In the left most image of Figure 4, the air heater is turned on, and the temperature of the air exiting it, after a time, stabilizes at 20° C, the desired value. However, at some point (the centre image in Figure 4) more electrical power is fed into the heater, thus modifying its heating capabilities, the new transfer function being

\[ H(s) = \frac{30}{10s + 1} \]  

(28)

The change in the output error is detected and the process’ performance is estimated using the discretized integral criterion. As its value becomes too different from the one of the nominal system, a new controller is computed. Although it is computed very fast after the change in the process has taken place (just a few sampling moments later), the heating rods have a certain inertia and it also takes a while for them to cool off, although the power to them has been lowered. When their temperature has been sufficiently lowered, the exiting air again has the desired temperature (the right-most image from Figure 4).

The sampling period of the system was 0.05 seconds, to test how low the sampling time can be. As the code implementation was non-optimal, lower sampling moments can still be achieved.

The robustness of this controller, the range of changes in the parameters of the process which can be accommodated, is described as in [6]. If the process’ parameters change, however both still reside inside the circle of robustness which can be defined in the space of the parameters of the process, a circle centred on the model which was used to compute the current controller, the system does not need to be adapted. The zone of robustness for the exemplified process is shown in Figure 5, for the initial transfer function.

After thoroughly testing the simulator for different pairs of parameters representing different processes, from the intervals \( K = [0.25, 100] \) and \( T = [0.25, 100] \), the following results were obtained:

- The identification procedure’s precision is less than \( 10^{-2} \), in the absence of noises. It is not perfect, due to the estimation of the integral criterion.
- If the sampling period is properly chosen for the given process, the stability is maintained, and the desired performance is always reached.
Any change of the processes’ amplification can be accommodated. The changes in the time constant can negatively influence the performances and even the stability, if the sampling period is not corrected accordingly.

This method is not suitable for unstable processes, or when the amplification is negative.

The lowest sampling time attained is 50 ms, without using any parallel programming techniques [20][21].

5. Conclusion
An adaptive – robust method for designing a control system was presented. It combines the advantages of both robust and adaptive control, while minimizing their disadvantages.

The process identification problem is treated implicitly, using an indicator to measure the degradation of the system’s performances, in case of dynamically changing parameters. When this estimation is greater than a threshold, the robust controller needs to be adapted. The adaptive - robust strategy computes the controller using the direct expression of an integral criterion, which depends only on the parameters of the process and the controller. It minimizes this criterion using the Nelder – Mead method. This approach can also be used for slow processes or processes with little to no variance in parameters. In such cases, it will behave like a classical robust control algorithm, with the bonus of being very sensitive to unexpected sudden changes in the system behaviour.

The theoretical approach presented in this work is validated after implementing an adaptive – robust algorithm for a closed loop control system, simulated in an application developed in C. The minimum sampling period attained is 50 ms. Thus, not only is it practical to use for fast processes, but it also increases the CPU’s idle time when being used for slower ones. Therefore, electrical energy is being saved, making the microcontroller/PLC more power efficient, or saving computational load which can be used elsewhere.

The proposed adaptive – robust approach has the significant advantage of eliminating the traditional identification procedure and assuring robust qualities for the numerical control.

As future work, the proposed design methodology can be extended for:

- Processes which have such a dynamic behaviour, that the order of the transfer function changes. Therefore, different forms of the integral criterion have to be used.
Testing the identification method in real conditions, where noises are present. The estimation works well in the absence of perturbations, but in a real environment, noises are present.

Development of an automatic procedure which changes the sample time of the adaptive–robust system accordingly to the change detected in the process' time constant.

References
[1] P. Borne, F. Gh. Filip, D. Popescu, M. Benrejeb, “Automatique Avancée et Informatique Appliquée”, Editura Academiei Romane, București, România, 2009
[2] F. L. Lewis, D. Vrabie, V. L. Syrmos, “Optimal Control”, John Wiley and Sons, 2012
[3] K. J. Åström, B. Wittenmark, “Adaptive Control”, Courier Corporation, 2008
[4] N. Doelman, R. Fraanje, I. Houtzager, M. Verhaegen, “Adaptive and real-time optimal control for adaptive optics systems”, European Control Conference Proceedings, 2009
[5] M. Green, D. J. N. Limebeer, “Linear Robust Control”, Courier Corporation, 2012
[6] L. I. Gliga, C. C. Mihai, C. Lupu, D. Popescu, “Adaptive – Robust control a computational efficient time implementation”, 13th International Conference on Engineering of Modern Electric Systems, 2015
[7] P. Ioannou, J. Sun, “Robust Adaptive Control”, Courier Corporation, 2013
[8] S. S. Ge, C. Wang, Y. Li, T. H. Lee, M. H. Ang, “Adaptive optimal control for linear discrete time-varying systems”, IEEE Conference on Cybernetics and Intelligent Systems Proceedings, 2013
[9] A. Locatelli, “Optimal Control: An Introduction”, Springer Science & Business Media, 2001
[10] S. Călin, M. Tertiașco, I. Dumitrache, C. Popea, D. Popescu, “Optimizări în automatizări industriale”, Editura Tehnică, 1979
[11] F. Li, “Robust Control Design: An Optimal Control Approach”, John Wiley & Sons, 2007
[12] P. Borne, D. Popescu, F. Gh. Filip, D. Ștefănoiu, “Optimization in Engineering Sciences – Exact Methods”, ISTE – Wiley, 2012
[13] P. Borne, D. Popescu, F. Gh. Filip, D. Ștefănoiu, “Optimization in Engineering Sciences – Metaheuristics, Stochastic Methods and Decision Support”, ISTE – Wiley, 2012
[14] V. Sinlapakun, W. Assawinchaichote, “Optimized PID controller design for electric furnace temperature systems with Nelder Mead Algorithm”, 12th Internation Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) Proceedings, 2015
[15] I. Maros, “Computational Techniques for the Simplex Method”, Springer Science and Business Media, 2012
[16] S. Waner, S. Costenoble, “Applied Calculus”, Cengage Learning, 2013
[17] K. Jamsa, L. Klander, “Totul despre C și C++ - Manualul fundamental de programare în C și C++”, Teora, 2000
[18] National Instruments, “Getting Started with LabWindows/CVI”, 2015
[19] S. G. Johnson, “The NLOpt nonlinear-optimization package”, http://ab-initio.mit.edu/nlopt, 2014
[20] M. McCool, J. Reinders, A. Robison, “Structured Parallel Programming: Patterns for Efficient Computation”, Elsevier, 2012
[21] Microsoft Corporation, “Multithreading with C and Win32”, https://msdn.microsoft.com/en-us/library/y6h8hye8.aspx, 2016
[22] D. Popescu, C. Lupu, C. Petrescu, A. Monica, M. Mihaela, “Sisteme de Conducere a Proceselor Industriale”, Printech, 2004
[23] D. Ștefănoiu, J. Culița, P. Stoica, “Fundamentele identificării și modelării sistemelor”, Printech 2005
[24] O. Nelles, “Nonlinear Systems Identification: From Classical Approaches to Neural Networks and Fuzzy Models”, Springer Science and Business Media, 2013