U.P.B. Sci. Bull., Series C, Vol. 70, No. 1, 2008 ISSN 1454-234x

MULTIPLE-MODELS CONTROL SYSTEMS – SWITCHING SOLUTION

C. LUPU[1], C. PETRESCU[2]

Sistemele multimodel sau multicontroller reprezintă una dintre abordările performante utilizate în controlul în timp real a proceselor neliniare. Utilizarea acestor structuri presupune rezolvarea unor probleme specifice, şi anume alegerea celui mai bun algoritm de reglare precum şi comutarea acestora. Lucrarea prezintă o metodă pentru comutarea algoritmilor de reglare în sisteme multimodel, bazată pe principiul comutării fără şocuri între regimul de funcţionare manuală şi automată. Verificarea metodei este realizată pe o structură de reglare numerică în timp real de tip RST. În final este prezentată şi o variantă de implementare software a acestei metode.

Systems with multiple models or multi-controller structure represent one of the successful solutions for the real-time control of the nonlinear processes. The use of these structures imposes solving some specific problems, like best algorithm selection and control algorithm switching. The paper proposes a method for switching the algorithms of the multiple-models structure, based on the principles of manual to automatic bumpless transfer. The applicability of the method is proved using a real-time structure with an RST control algorithm. In the end, its software implementation is also shown.

Keywords: control systems, switching algorithm, manual-automatic bumpless transfer, real-time systems

1. Introduction

The essential condition for the real-time function of a control system is preserving the closed-loop performances in case of non-linearity, structural disturbances or process uncertainties. A valuable way to solve these problems is the multiple-models or multicontroller structure. The first papers mentioning the "multiple-models" structure/system have been reported in the 90s. Balakrishnan and Narendra are among the first authors addressing problems of stability, robustness, switching and designing of this type of structures in their papers [1].

Research refinement in this field have brought extensions to multiple-model control concept. Parametric adaptation procedures – Closed-Loop Output

[1] Lecturer, Dept. of Automatics and Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: cip@indinf.pub.ro
[2] Lecturer, Dept. of Automatics and Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: catalinp@indinf.pub.ro

Error [2], use of Kalman filter representation [3], the use of neural networks [4] or of the fuzzy systems are some of the important developments.

Related to classical control loops, multiple-model systems need addressing some supplementary specific aspects:

- Dimension of multiple-model configuration;
- Selection of the best algorithm;
- Control law switching.

From the multiple-models control systems viewpoint, two application oriented problems can be highlighted:

- Class of systems with nonlinear characteristic, which can not be controlled by a single algorithm;
- Class of systems with different operating regimes, where different function regime doesn't allow used of a unique algorithm or imposes usage of very complex one with special problems on implementation.

As function of the process particularity, several multiple-models structures are proposed [1]. One of the most general architectures is presented on Fig. 1.

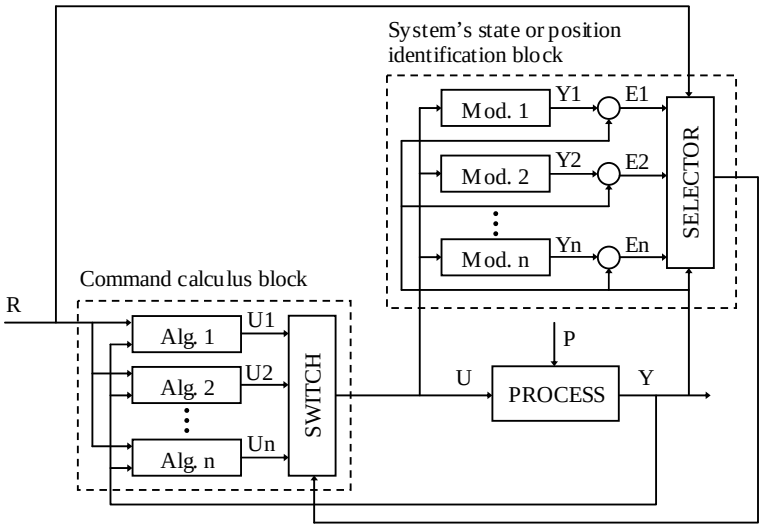

Fig. 1. General scheme for multimodel structure

In this figure the blocks and variables are as follows:

- Process – physical system to be controlled;
- Command calculus – unit that computes the process control law;

- System's state or position identification– component that provide information about the model–control algorithm "best" matching for the actual system's state;
- Mod. 1, Mod. 2, …, Mod. N - previously identified models of different regimes or operating points;
- Alg. 1, Alg. 2, …, Alg. N – control algorithms designed for the n models mentioned above;
- SWITCH – mixing or switching between the control laws;
- SELECTOR – based on adequate criteria evaluations, provides information about the most appropriate model for the system's current state;
- *Y* and *Y1*, *Y2*,…, *Yn* – output of the process and outputs of the models, respectively;
- *U* and *U1*, *U2*, …,*Un* – output of the Command calculus block and outputs of the n control algorithms, respectively;
- *R* – system's set point or reference trajectory;
- *P* – disturbances of physical process.

As noted above, function of the process particularities and the approach used to solve the "control algorithms switching" and/or "the best model choice" problems, the scheme can be adapted on the situation by adding/eliminating some specific blocks. This paper focuses on the "switching" problem.

2. Control algorithms switching

Corresponding to multi-model structure's function logic, after finding the best algorithm for the current process's functioning point, the next step consists on switching the control algorithm. Two essential conditions must be verified with respect to this operation:

- To be designed so that no bumps in the applications of the control law are encountered;
- To be (very) fast.

Shocks determined by the switching operation cause non-efficient and/or dangerous behaviors. Moreover, slow switching determines boiling down the control algorithm's action zone, which involves only the system's performances' alteration.

These are the main problems to be solved when designing the algorithms' switching block. Firstly, from structurally point of view, this block may contain all algorithms' implementation, or, secondly, at least the algorithms' coefficients.

The switching operation is done based on the information provided by the system's state or position identification block. This information consists of a trigger-signal to start the operation and the number of the algorithm that will become active.

Classical solutions

Present solutions solve more or less this problem and they are based on maintaining in active state all the control algorithms, also called "warm state". This supposes that every algorithm receive information about the process output *y(k)* and set the point value (eventually filtered) *r(k)*, but only the control law $u_i(k)$ is applied on the real process, the one chosen by the switching block. This solution does not impose supplementary function logic for the system's architecture and, for these reasons, it gives the possibility of switching very fast the algorithms. The drawback of this approach is that when designing the multi-model structure several supplementary steps are necessary.

These supplementary conditions demand the matching of the control algorithm outputs' in the neighborhood switching zones. The superposition of models identification zones accomplishes this aspect. That can be seen on Fig. 2.

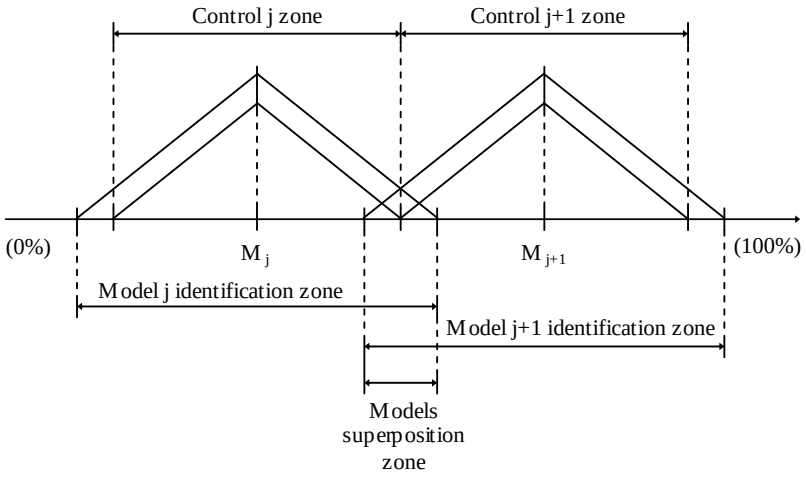

Fig. 2. Superposition of identification zones for two neighbor-models and their corresponding control actions

As a result of this superposition, the multi-model structure will have an increased number of models.

Another approaches [6], [7] propose mixing two or more algorithms' outputs. The "weighting" of each control law depends on the distance from the

current process's operating point and the action zone of each algorithm. Based on this, the switching from an algorithm to another one is done using weighting functions with a continuous evolution in [0–1] intervals. This technique can be easily implemented using fuzzy approach. An example is presented on Fig. 3.

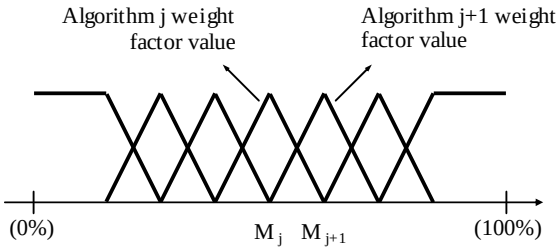

Fig. 3. Algorithms weighting functions for a specified operating position

This solution involves solving control gain problems, determinated by the mixing of the algorithms' outputs.

Proposed solution

In this paper, it is presented a solution that provides very good results for fast processes with nonlinear characteristics. The main idea is that, during the current functioning of multiple-models control systems with N model-algorithm pairs, it is supposed that just one single algorithm is to be maintained active, the good one, and all the other N-1 algorithms rest inactive. The active and inactive states represent automatic, respectively manual, regimes of a control law. The output value of the active algorithm coresponds to the manual control for all the other N-1 inactive algorithms. In the switching situation, when a "better" *Aj* algorithm is found, the actual *Ai* active algorithm is commuted in inactive state, and *Aj* in active state, respectively. For a bumpless commutation, it must be solved the manual–automatic transfer problems, and the solution to this it is proposed in section 2.

The system can be implemented in two variants – first - with all inactive algorithms holding on manual regime, or – second - just a single operating algorithm (the active one) and activation of the "new" one after the computation of the currently corresponding manual regime and switching on automatic regime. Both variants have advantages and disadvantages. Choosing one of them necessitates knowledge about the hardware performances of the structure. After a general view, the first variant seems to be more reasonable.

In all situations, it is considered that the active algorithm's output values represent manual commands for the "new" selected one.

3. Manual–Automatic bumpless transfer

The implementation practice highlights important problems like manual-to-automatic/automatic-to-manual regime commutations, respectively turning out in/from the control saturation states. Of course, these problems exist in analogical systems and have specific counteracting procedures, which are not applicable on numerical systems.

The process operation begins on "manual" regime, this procedure being used as long as the process did not reach the nominal functioning zone. When comutation, it is recommended having a very good matching between the set point and process's output values. This strategy releases the system of the shocks sent to the actuators.

In the following, these facts will be illustrated using an RST control algorithm.

Practical considerations about the real-time algorithm implementation

Consider the process's discrete model:

$$A(q^{-1})y(k) = B(q^{-1})u(k) \tag{1}$$

where $A(q^{-1})$ and $B(q^{-1})$ polynomials are:

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_{nA} q^{-nA} \\ B(q^{-1}) &= 1 + b_1 q^{-1} + \ldots + b_{nB} q^{-nB} \end{aligned} \tag{2}$$

with $nA \le nB$. For this model, an RST algorithm is used:

$$S(q^{-1})u(k) + R(q^{-1})y(k) = T(q^{-1})y^*(k) \tag{3}$$

where: $u(k)$ - algorithm output, $y(k)$ - process output, $y^*(k)$ - trajectory or filtered set point. The corresponding polynomials are:

$$\begin{aligned} S(q^{-1}) &= 1 + s_1 q^{-1} + \ldots + s_{nS} q^{-nS} \\ R(q^{-1}) &= 1 + r_1 q^{-1} + \ldots + r_{nR} q^{-nR} \\ T(q^{-1}) &= 1 + t_1 q^{-1} + \ldots + t_{nT} q^{-nT} \end{aligned} \tag{4}$$

The closed-loop control representation is given on Fig. 4.

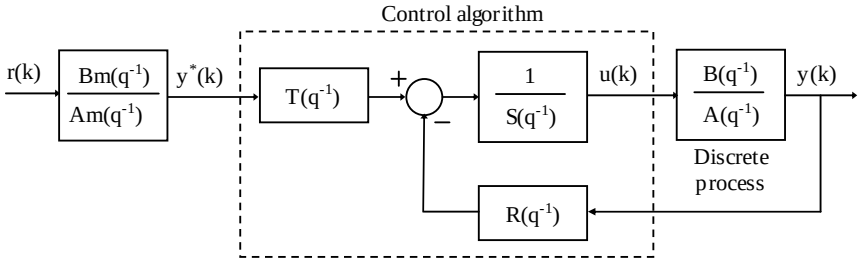

Fig. 4. Two freedom-degrees closed-loop canonical form

The control algorithm in (3) can be rewritten as follows:

$$u(k)=\frac{1}{s_0}\left[-\sum_{i=1}^{nS}s_i u(k-1)-\sum_{i=1}^{nR}r_i y(k-1)+\sum_{i=1}^{nT}t_i y^*(k-1)\right] \tag{5}$$

nS, *nR* , *nT* express the corresponding polynomials degrees and also the memory dimension for the software implementation of the algorithm. For example, if *nR*=2, then it should be reserved three memory locations for the process's output: *y(k)*, *y(k-1)*, *y(k-2)*. Respectively, the same rule applies for *u(k)* and $y^*(k)$.

When necessary, an imposed trajectory can be generated using a trajectory model generator:

$$y^*(k)=\frac{Bm(q^{-1})}{Am(q^{-1})}r(k) \tag{6}$$

with *Am* and *Bm* like:

$$\begin{aligned} Am(q^{-1}) &= 1+am_1q^{-1}+\ldots+am_{nAm}q^{-nAm} \\ Bm(q^{-1}) &= 1+bm_1q^{-1}+\ldots+bm_{nBm}q^{-nBm} \end{aligned} \tag{7}$$

For the practical implementation, one must be interested on the control algorithm and, eventually, the trajectory's model generator. One single iteration of the continuous monitoring program the implies following steps:

- Process's data acquisition;
- Trajectory computation (if necessary);
- Control law computation;
- Sending the controls to the actuators;

- Process evolution graphical display;
- Actualization of the algorithm's memory for the new iteration.

For example, the control law computation, when $nR = nS = nT = 1$ and without trajectory generator ($y^*(k)=r(k)$), is like in the following:

$$u(k) = \frac{1}{s_0}\left[-s_1 u(k-1) - r_0 y(k) - r_1 y(k-1) + t_0 y^*(k) + t_1 y^*(k-1)\right] \qquad (8)$$

and (9) gives the algorithm's memory actualization for the next iteration:

$$u(k-1) = u(k); \quad y(k-1) = y(k); \quad y^*(k-1) = y^*(k); \qquad (9)$$

Manual/automate transfer

In real functioning, M→A transfer is preceded by "driving" the process in nominal the action zone. To avoid command's switching "bumps", one must respect the following two conditions:

- Process's output must be perfectly matched with the set point value;
- Accordingly with the algorithm's complexity (function of the degrees of controller polynomials), the complete algorithm's memory actualization must be waited of.

Neglecting these conditions lead to "bumps" in the transfer because the control algorithm's output value is computed using the actual, but also the past, values of the command, process and set point, respectively.

At the same time, there are situations when the perfect "matching" between process's output and set point value is very difficult to be obtained and/or needs very long time. Hence, the application of this procedure becomes impossible in the presence of important disturbances.

In this context, since the algorithm's output is the manual command set by operator and the process's output depend on command, the set point remains the only "free" variable in the control algorithm's computation. Therefore, the proposed solution consists in the modification of the set point value, accordingly with the existent control algorithm, manual command and process's output.

Algorithm's memory actualization is done similarily as in the automatic regime. For practically implementation it is necessary a supplementary memory location for the set point value. From (6), it results the expression for the set point's value:

$$y^*(k) = \frac{1}{t_0}\left[\sum_{i=0}^{nS} s_i u(k-i) + \sum_{i=0}^{nR} r_i y(k-i) + \sum_{i=0}^{nT} t_i y^*(k-i)\right] \tag{10}$$

When the set point (trajectory) generator (6) exists, keeping all the data in correct chronology must be with respect to the following relation:

$$r(k) = \frac{Am(q^{-1})}{Bm(q^{-1})} y^*(k) \tag{11}$$

System's functioning scheme is presented on Fig. 5.

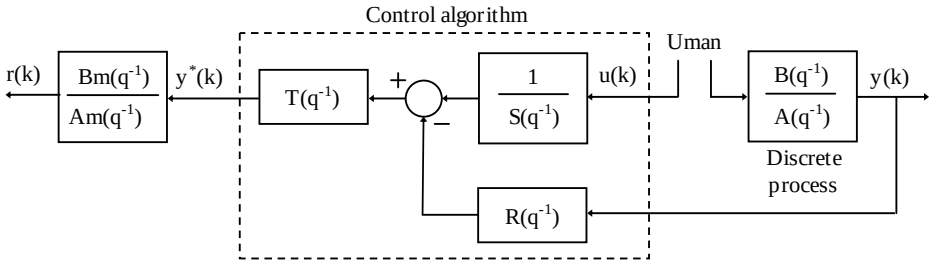

Fig. 5. Computation of the set point value for imposed manual command

Concluding, this solution proposes the computation of that set point value that determines, accordingly to the algorithm's history and process's output, a control equal to manual command applied by the operator. At the time instant of the M→A switching, there are no gaps in the control algorithm's memory that could determine bumps. An eventually mismatching between the set point and process's output is considered as a simple change of set point's value. Moreover, this solution can be successfully used in cases of command limitation.

The only inconvenient of this solution is represented by the necessary big computation power when approaching high order systems, which is not, however, a problem nowadays.

4. Experimental results

We have evaluated the achieved performances of the multi-model control structure using a process simulator software application, developed on National Instruments's LabWindows/CVI. On Fig. 6, one can see a positioning control system, its operation medium having variable viscosity. The main goal is to control the piston's position.

The nonlinear relation between the position Y (%) and actuator command U (%) is presented on Fig. 7.

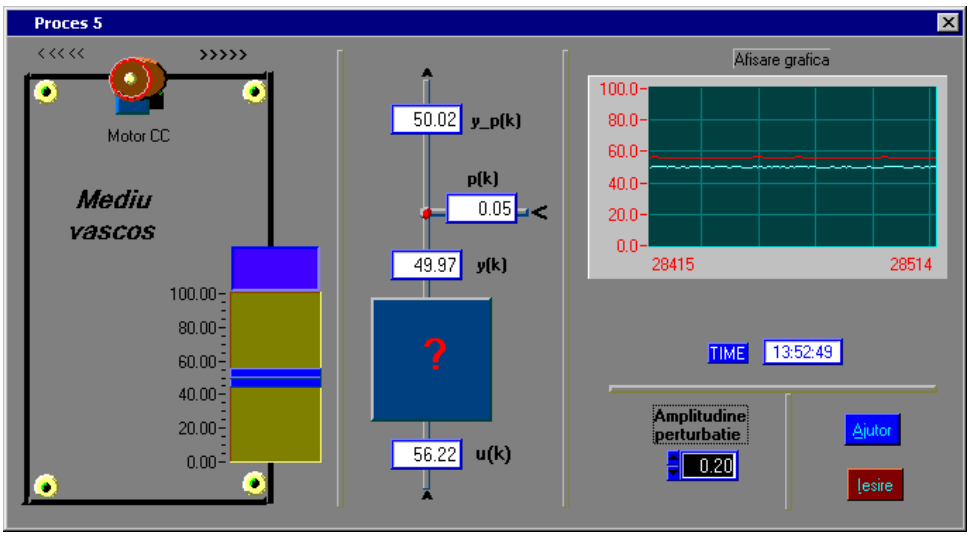

Fig. 6. Process simulator software application

One considers three operating points P_1, P_2, and $P3$ on the plant's nonlinear diagram (Fig. 7). Three different models are identified like: M_1 (0-30%), M_2 (30-70%) and M_3 (70-100%). These will be the zones for corresponding algorithms.

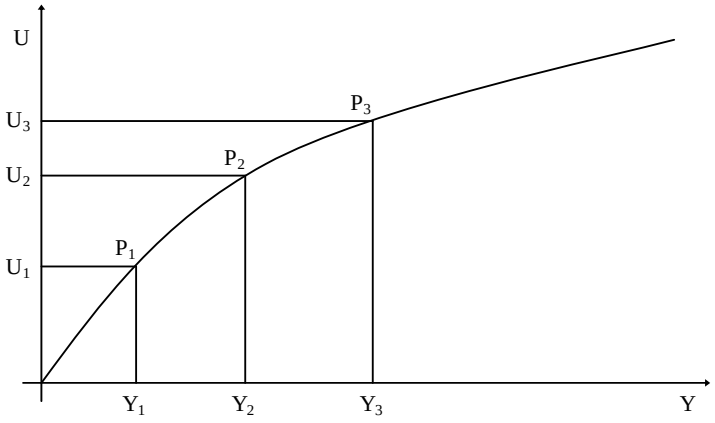

Fig. 7. Nonlinear diagram of the process

Accordingly to the models-algorithms matching zones (Fig. 2), we have identified the models M_1, M_2 and M_3, as being appropriated to the following intervals (0-40%), (20-80%) (60-100%), respectively. For a sampling period T_e=0.2 sec, the least-squares identification method from Adaptech/WinPIM platform identifies the next models:

$$M_1 = \frac{0.35620 - 0.05973q^{-1}}{1 - 0.45401q^{-1} - 0.09607q^{-2}}$$
$$M_2 = \frac{1.23779 - 0.33982q^{-1}}{1 - 0.98066q^{-1} - 0.17887q^{-2}} \quad (12)$$
$$M_3 = \frac{2.30953 - 0.08959q^{-1}}{1 - 0.82743q^{-1} - 0.00659q^{-2}}$$

In this case, we have computed three corresponding RST algorithms using a pole placement procedure from Adaptech/WinREG platform. The same nominal performances are imposed to all systems, through a second order system, defined by the dynamics ω_0 = 3.0, ξ = 2.5 (tracking performances) and ω_0 = 7.5, ξ = 0.8 (disturbance rejection performances) respectively, keeping the same sampling period as for identification.

All of these algorithms control the process in only their corresponding zones.

$$R_1(q^{-1}) = 1.670380 - 0.407140q^{-1} - 0.208017q^{-2}$$
$$S_1(q^{-1}) = 1.000000 - 1.129331q^{-1} + 0.129331q^{-2}$$
$$T_1(q^{-1}) = 3.373023 - 3.333734q^{-1} + 1.015934q^{-2}$$

$$R_2(q^{-1}) = 0.434167 - 0.153665q^{-1} - 0.239444q^{-2}$$
$$S_2(q^{-1}) = 1.000000 - 0.545100q^{-1} - 0.454900q^{-2} \quad (13)$$
$$T_2(q^{-1}) = 1.113623 - 1.100651q^{-1} + 0.335417q^{-2}$$

$$R_3(q^{-1}) = 0.231527 - 0.160386q^{-1} - 0.000879q^{-2}$$
$$S_3(q^{-1}) = 1.000000 - 0.988050q^{-1} - 0.011950q^{-2}$$
$$T_3(q^{-1}) = 0.416820 - 0.533847q^{-1} + 0.187289q^{-2}$$

To verify the proposed switching algorithm, it was designed and implemented a multi-model controller real-time software application, which can be connected with the process simulator. The user interface is presented on Fig. 8.

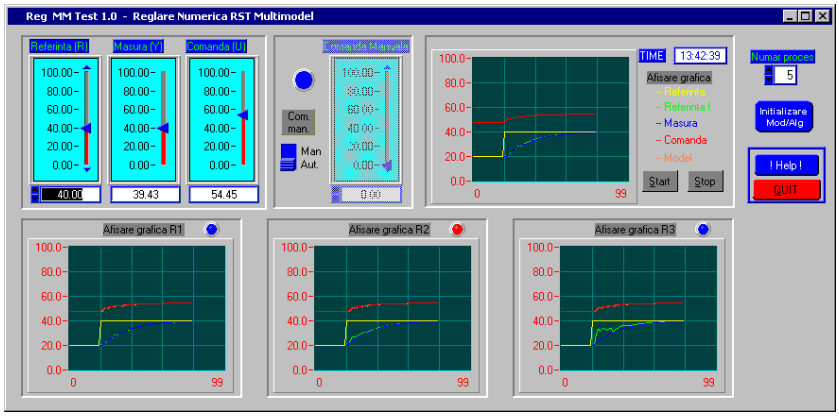

Fig. 8. Multi-model controller real-time software application

On the top of Fig. 8, there are respectively the set point, the output and control values, manual-automatic general switch, general manual command and graphical system evolution display. On the bottom of Fig. 8, one can see three graphical evolution displays corresponding to the three controllers (R_i, S_i, T_i, i=1...3). The colors are as follows: yellow – set point value, red – command value, blue – process output value and green – filtered set point value.

Using this application, few tests were effectuated to verify the switching between two algorithms. The switching procedure is determinated by the change of the set point value. These tests are:

- from 20% (where algorithm 1 is active) to 40% (where algorithm 2 is active). The effective switching operation is done when the filtered set point (and process output) becomes greater than 30%. Fig. 9(a) presents the evolutions.
- from 60% (where algorithm 2 is active) to 80% (where algorithm 3 is active). The effective switching operation is done when the filtered set point (and process output) becomes greater then 70%. Fig. 9(b) presents the evolutions.

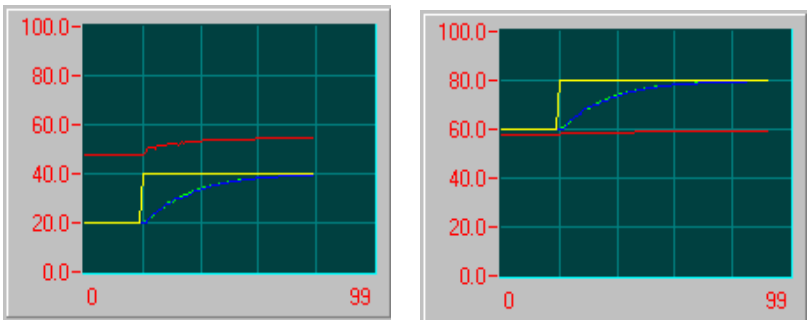

Fig. 9. a) switching test 1 b) switching test 2

In both tests, one can see that there are no shocks or there are very small oscillations in the control evolution by applying this approach. Increasing the number of models-algorithms to 4 or 5 could eliminate the small oscillations.

5. Conclusions

The method was successfully tested on a process simulator software application with nonlinear characteristic, using a 3 multi-model/controller real-time software application. Moreover, the first variant (with all algorithms active) of the approach was implemented, ensuring the fast switching (one step) between algorithms.

With regards to the results obtained in the paper, the switching method can be successfully recommended in multi-model real-time control structures for fast processes.

R E F E R E N C E S

[1] *Narendra, K. S. and J. Balakrishnan,* Adaptive Control using multiple models, IEEE Transactions on Automatic Control, vol. 42, no. 2, February, page. 171 – 187, 1997

[2] *Landau, I.D. and A. Karimi,* Recursive algorithm for identification in closed loop: a unified approach and evaluation, Automatica, vol. 33, no. 8, pp. 1499-1523, 1997

[3] *Lainiotis, D.G. and Magill D.T*, Recursive algorithm for the calculation of the adaptive Kalman filter weighting oefficients. IEEE Transactions on Automatic Control, 14(2):215–218, April 1969.

[4] *Balakrishnan., J.,* Control System Design Using Multiple Models, Switching and Tuning, Ph. D. Dissertation, University of Yale, USA, 1996

[5*] Landau, I. D., R. Lozano and M. M'Saad*, Adaptive Control, Springer Verlag, London, ISBN 3-540-76187-X, 1997

[6] *Dussud, M., S. Galichet and L. Foulloy*, Fuzzy supervision for continuous casting mold level control, IFAC, 2000.

[7] *Pages, O., P. Mouille and B. Caron*, Multi-model control by applying a symbolic fuzzy switches, IFAC, 2000.