Memorias del Congreso Internacional de Ciencias Básicas e Ingeniería - CICI 2016
ISBN: 978-958-8927-23-7

Evaluating the specification requirements for (N)EPSAC-MPC implementation on a Programmable Logic Controller (PLC)

J. Camila Espitia Duarte
University of Ibague, Colombia
Faculty of Engineering - Research Area
camila.espitia@gmail.com

Andres Hernandez and Robin De Keyser
Ghent University, Belgium
Department of Electrical Energy, Systems and Automation (EeSA)
Andres.Hernandez@ugent.be, robain.dekeyser@ugent.be

***Abstract*—Programmable logic controllers (PLC) are by far the most common hardware used in industry for process automation. As observed in many electronic devices, in the recent years, processing capabilities have increased, thus making possible to implement complex algorithms in such embedded devices. Among the different control methodologies, Model Predictive Control (MPC), stands out due to its ability to deal with constrained control problems. MPC is becoming the standard control strategy in process control as it meets the current industry demands of quality, production, resource optimization, low hardware cost, among others. The aim of this work is to develop a methodology to implement in a straightforward manner advanced control algorithms on a PLC, while providing a guideline on the minimum specifications required to properly choose a PLC reference, thus reducing the gap between the theoretical contributions and industrial practice. An experimental validation is achieved by implementing and comparing the performance of a classical PID, a linear (EPSAC) and nonlinear (NEPSAC) approach to constrained MPC on the Festo MPS PA workstation.**

I. Introduction

The PID Family controllers (P, PI, PD, PID) until a few years ago were the most control strategy used in the industry, because of its undoubtedly simplicity, low implementation cost and acceptable response. As a result, they have several advantages over more robust control strategies [11] [6]. However, industry requirements are constantly increasing [18]. Currently, several works have presented the implementation modern strategies and robust control, such as, MPC (Model Predict Control) which exceeds the performance of traditional PID, applied to different types of industrial processes [1] [4] [12] [16] [2] [13]. In order to implement modern control strategies in real form, certain characteristics are required for the hardware of the controller where the implementation is performed. The controller is usually a PLC - controller standard of the industry [19] [3], PLCs manufacturers are introducing devices with better characteristics related to the memory capacity, programming languages and the processing speed instruction [14] [21]. On the other hand, the software specialized in control like MatLab include interesting tools that represent a link to migrate from theoretical developments to real implementations [15] [17] [22].
Work starts of the fact that is possible, with available tools, to implement advanced control strategies on PLCs, but its focuses are the needs of to provide a clear-simple procedure and shown the minimum PLC hardware requirements for it. The detail of the implementation of predictive control strategies: EPSAC and NEPSAC on Siemens PLC 1615-3 is presented. The experimental validation is performed using a pilot FESTO MPS PA with pressure system - SISO linear system, and level system - SISO nonliear system. Matlab was used for data processing and as an interface for programming the PLC, with tool PLC Coder and the traditional PID was used like benchmark to results.

II. Methodology Description

The methodology used as an interface for PLC programming and experimental validation of their performance under implementation of robust control strategies, it begins with the formulation of the control strategy within a block function in Matlab-Simulink (Stage 1) and the translation of this using appropriate language through PLC Coder tool (Stage 2), the file result is loaded, compiled and downloaded along with other data acquisition subroutines to the PLC (Stage 3) using a IDE (Integrated Development Environments). Finally the scheduled will run a control strategy to maintain a plant process in desired states (Stage 4). The instructions for data acquisition are executed each sampling time, in order to allow the collecting, storing in a global data base. (Stage 5) The data is displayed data into a HMI (Human Machine Interface), from which they are exported as a text file (*.txt) for later analysis (Figure 1).

III. Elements of Methodology

A. Control Strategy - (N)EPSAC

The Extended Prediction Self-Adaptive Control (EPSAC) (Keyser and Cuawenberghe, 1985) belongs to the large family of Model Predictive Control (MPC) algorithms. MPC is a set of control strategies, which most attractive feature is its ability to handle constraints, because MPC has detailed knowledge of the dynamic behavior of the system (including critical units) represented in a mathematical model. As it is stated above MPC seems to be a good option to face the industry

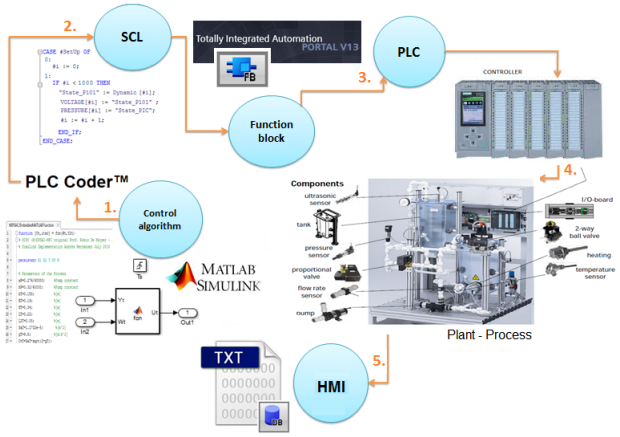


Fig. 1. Interface Matlab/TIA Portal

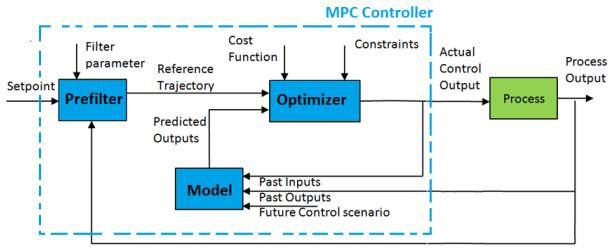


Fig. 2. Model Predictive Control System.

challenges. Elements: (1) Model: Used to get information that helps to predict its evolution. (2) Prefilter: Responsible for calculating a reference trajectory (Setpoint-new sequence to current sample time) along a control horizon finite and slider. (3) Optimizer: Algorithm that includes minimizing a cost function for get an control action to apply and ensure the process will remain as close as possible to the reference trajectory. EPSAC formulation uses input/output models and its representation is as presented in Figure 2. The type of model used to capture the dynamics of the system is a discrete transfer function. The optimizer used is an algorithm or control sequence that minimizes a cost function of the form:

$$J(U_{opt}) = \sum_{k=N1}^{N2} [Error]^2 = [r(t+k/t) - y(t+k/t)]^2 \quad (1)$$

The function is applied to k steps forward from the current moment t. $N1$ and $N2$ are the minimum value and the maximum value of the control horizon. The reference trajectory is given by $r(t+k/t)$ and $y(t+k/t)$ is the prediction of the system response, which can be considered as cumulative result such as shown in the next equation:

$$y(t+k/t) = y_{base}(t+k/t) + y_{opt}(t+k/t) \quad (2)$$

Where $y_{base}(t+k/t)$ is the prediction of the system response using the model and the data known so far $x(t+k/t)$ and including the estimation of disturbances $n(t+k/t)$ and $y_{opt}(t+k/t)$ corresponds to the response of the system when a vector of future optimal actions $U_{opt}(t+k/t)$ is applied.

$$U_{opt}(t+k/t) = (G'G)^{-1}G'[r(t+k/t) - Y_{base}(t+k/t)] \quad (3)$$

G: Coefficient vector for step response - These values correspond to cumulative constant, so that only calculated once. Restrictions that normally act on a process can be expressed as: (1) Manipulated variable limit $Uopt(t)min \leqslant Uopt(t) \leqslant Uopt(t)max$. (2) Limit rate of change of the manipulated variable $\triangle Uopt(t)min \leqslant (Uopt(t) - Uopt(t-1)) \leqslant \triangle Uopt(t)max$. (3) Manipulated variable limit $Y(t)_{min} \leqslant Y(t) \leqslant Y(t)_{max}$. It is usual that the constrains are only taken into account after of calculated the control signal, but it is no the best option when control signal breaks the constrains because the process can be carried to its limits and operate in a dangerous manner and not optimal. Another option is to consider the constraints inside cost function, in the EPSAC approach, it is adopted the Lagrange multiplier implementation to express the cost function and the QP - Hildreths quadratic programming procedure (Luenberger, 1969, Wismer and Chattergy, 1978) - to solve it.

$$J(Uopt) = \frac{1}{2} * Uopt' * H * Uopt + b * Uopt + F \quad (4)$$

Subject to: $A * Uopt \leq b$. $H = G'G$, $F = G' * Error$, A and b are matrices which describe constraints acting on the process and depends on the process parameters and values limits of process variables. With Lagrange multiplier implementation a problem of n variables is reduced to one of $n+m$ variables, where n is the dimension of U_{opt} vector and m is the dimension of A, to get:

$$\lambda = -[A * H^{-1} * A']^{-1}(b + A * H^{-1} * F) \quad (5)$$

$$Uopt = -H^{-1} * F - H^{-1}A'\lambda \quad (6)$$

First term is the general solution of EPSAC without constrains, and the second term is a correction term due to constraints. To find the Lagrange multipliers the quadratic programming procedure is applied (equations 7 and 8), it is search element by element iteratively using the matrices $P = A * H^{-1} * A'$ and $d = H^{-1} * A * F + b$, because this part is also determined as quadratic programming problem with λ_i as the decision variable. The search is stops when the values found converge (to zero for inactive constraints / to one positive value for the active constraints), or the maximum number of iterations is exceeded.

$$(\lambda_i)^{m+1} = max(0, w_i^{m+1}) \quad (7)$$

$$w_i^{m+1} = -\frac{1}{h_{ii}}[k_i + \sum_{j=1}^{i-1}(h_{ij} * (\lambda_j)^{m+1}) + \sum_{j=i+1}^{n}(h_{ij} * (\lambda_j)^m)] \quad (8)$$

$m+1$ indicates update, h_{ij} is the ij^{th} element in the P and k_i is the ith element in the d. In the Non-linear Extended Prediction Self-adaptive control (NEPSAC) approach the non-linear model is used directly. It makes use of the developments established for the linear version starting from the equation

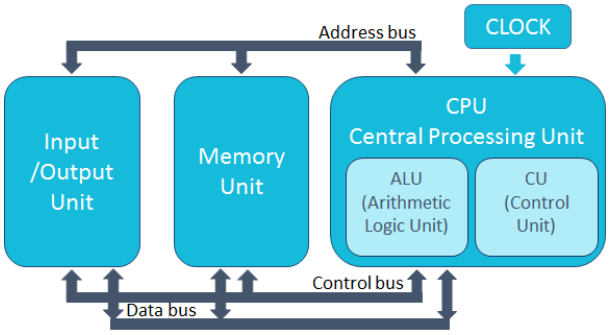


Fig. 3. Architecture of a PLC.

(2), with the difference that it uses an iterative procedure to make Y_{opt} as close as possible to zero, thus avoiding the super position principle and allowing the equation to be valid for non-linear systems. The gradual reduction of Y_{opt} involves careful selection of a control scenario to ensure rapid convergence, satisfied by using the previous computed control action, it also involves re-calculating the G matrix at each sampling time.

B. Matlab Simulink-PLC Coder

The simulink-PLC coder is a hardware-independent Matlab Toolbox, where Structured Text files are generated under the IEC 61131-3[1]. Structured text can be generated in formats compatible with various IDEs (Integrated Development Environments) including Siemens TIA Portal, which, result is a file with extension *.scl[2].

C. The Programmable Logic Controller (PLC)

1) PLC general description: PLCs are sequential machines where the instructions indicated in the user program, stored in its memory, are executed consecutively. It generates some commands or control signals starting from the input signals readed from the plant. If signal changes are detected, the controller reacts according to the program performing the required orders. PLCs are designed to control tasks and the industrial environment. hence, PLCs should have: Reliable hardware (e.g., to work inside environments with high humidity and temperature, etc), facilities for extend and include other control loops and user-friendly interface. The basic architecture of PLC has a central processing unit (CPU), memory unit (that although the figure 3 is shown separated from the CPU, part of it is integrated into the CPU), and input/output unit. CPU is supplied with a clock with a frequency that determines the operating speed of the PLC and provides the timing and synchronisation for all elements in the system. The information travels in digital format by paths called buses. The Figure 4 shows the organization of memory in a PLC and the area that corresponds to each of the elements that are part of a program user (set of code blocks and data blocks). Program user is first save into load memory (non-volatile type) along with the hardware configuration, into the work memory (volatile type) are stored the parts of the user program, relevant to the current execution. The system memory is available to store retentive data at power-off, such as bit memories[3], timers and counters, temporary local data of a block for the duration of processing, tags from global data blocks and the values from the input and output modules (process images input/output). The cycle time (figure 5) represents the time that CPU needs

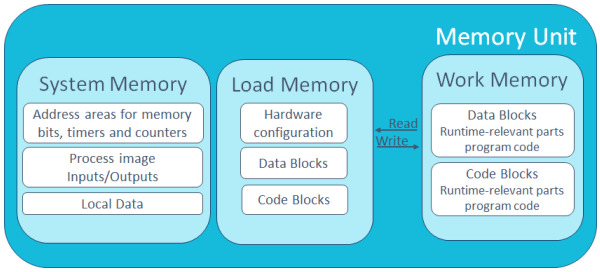


Fig. 4. Memory areas PLC.

to execute a program, including all user program sections, update of the process image of the inputs/outputs and system activities interruptions. At the start the signal states of the input modules are transmitted to the process image of the inputs, then CPU processes the user program and executes the instructions specified on it, at the end the process image of the outputs is transmitted as signal state to the output modules. The cycle time could increases due to software interrupt processing, hardware interrupt processing, diagnostics and error handling, communications, special functions such as control and monitoring of tags or program status, transfer and clearance of program parts (compression of the user program memory) and internal memory test.

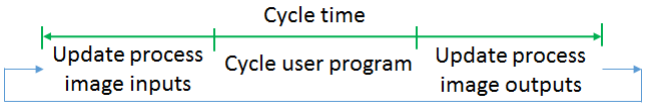


Fig. 5. Cycle time PLC.

2) Classic Selection Criteria to PLC:

- Brand (cost, technical support).
- Communication System.
- Programming languages. The choice of programming language could depend of the training and experience of the programmer, the problem (particular application), the structure of the control system, or the interface for other people or departments.
- Number of Inputs/outputs digitals and analogs.
- Space Memory.

[1]IEC 61131-3 (2013). It specifies the syntax and semantics of a unified suite of programming languages for edition of a PLC user program. It consists of two textual languages, Instruction List (IL) and Structured Text (ST), and two graphical languages, Ladder Diagram (LAD) and Function Block Diagram (FBD).

[2]SCL (Structured Control Language) corresponds to the textual high-level language ST (Structured Text).

[3]Bit memory is a memory area of the CPU, which can be addressed from any code block, it area can be used to store temporary results, for example.

3) PLC Siemens 1516-3 PN/DP: Controller for high-end applications, CPU 1516-3 PN/DP sends and receives data and signals from the connected IO devices within a PROFINET IO system, it supports SCL and it has enough inputs and outputs to Festo MPS PA Workstation (32 digital outputs, 32 digital outputs, 8 analog inputs, 4 analog outputs). The PLC 1516-3 has a resolution of 16 bits (1 word) including sign, it means that a of 10VDC data will be represented by an integer value of 27648. Memory characteristics: 1MB to work memory for program, 5MB to work memory for data, 24MB to load memory into SIMATIC memory card. CPU processing time is between 10ns (Bit operations) and 64ns (Floating-point arithmetic). The minimum value of the update time of the process image input/output is 39μs/word, and the basic time expenditure for an interrupt is 80μs.

D. IDE - Tia Portal V13

Totally Integrated Automation is an IDE designed by Siemens to program its PLCs of last generation, TIA PORTAL can be programmed in any of the languages of the IEC 61131-3 standard always that the PLC supports it. Tia Portal, it also allows programming of SCADA (Supervisory Control And Data Acquisition) or HMI systems through WINCC. A SCADA system enables access to the data of a process for monitoring, management and control of it, due to the graphical user interface communicating with the system. The IDE to use depends on the choice of PLC.

E. Plant - Festo MPS PA workstation

Festo MPS PA workstation is a training system developed by Festo, it combines a set of analogue and digital sensors and actuators with which can be constructed different closed loops, to try n control strategies.

IV. Development of Methodology

A. Stage 1: Design and simulation of control strategy

The control strategies EPSAC and NEPSAC are designed within an Embedded MATLAB Function block (Figure 6) with two inputs (desired value - setpoint, value measured - sensor value) and one output (control action - pump voltage). In order

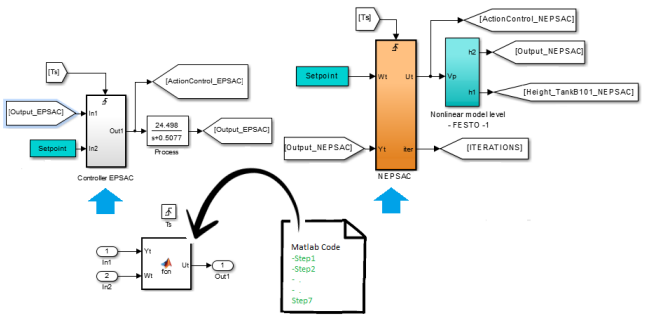


Fig. 6. Simulink scheme for EPSAC and NEPSAC implementation.

to simulate the sampling time, the function was introduced within a subsystem to which was added another input (Trigger-

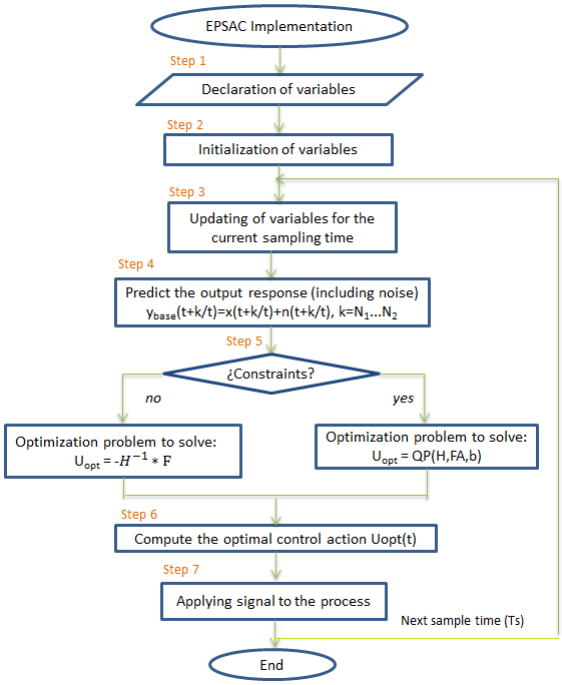


Fig. 7. EPSAC Method

pulse train with period equal to Ts). The performance of each control strategy is verified through simulation. In the case of level is added one more output with which is possible to observe the number of iterations to achieve a desired value. A flow-chart for the implementation of EPSAC with and without constrains is depicted in Figure 7. The calculation of the control action starts with the declaration and initialization of variables and constants that will be used, then with the upgrade of variables for the current sampling time and the prediction of the system output from the model considering this noise, then a number of future optimal actions are determined and finally the control signal to apply is calculated. After applying the control signal, the steps from the updating of variables are repeated for a new sampling time. To NEPSAC (Figure 8) the significant difference is that from step 4 to step 6 are repeated a finite number of times to progressively reduces the value of Y_{opt}.

B. Stage 2: Coding algorithm to appropriate language

Translation of the control algorithm within the embedded MATLAB function to *.SCL language, using PLC Coder. With right click on the embedded MATLAB function to initiate the generation of the file *.SCL compatible with TIA Portal. The figure 9 shows a portion of the program written in Matlab and encoded in SCL. The part a. corresponding to step 3 - update variables: in Matlab shift vector is used to discard the oldest value, in SCL this process is not direct, so a helper vector (tmp) is used to make the shift, additionally vectors are worked with the control structure FOR. The part b. corresponding to step 5 - optimization problem to solve (EPSAC unconstrained) one important difference is with the translate the previous steps

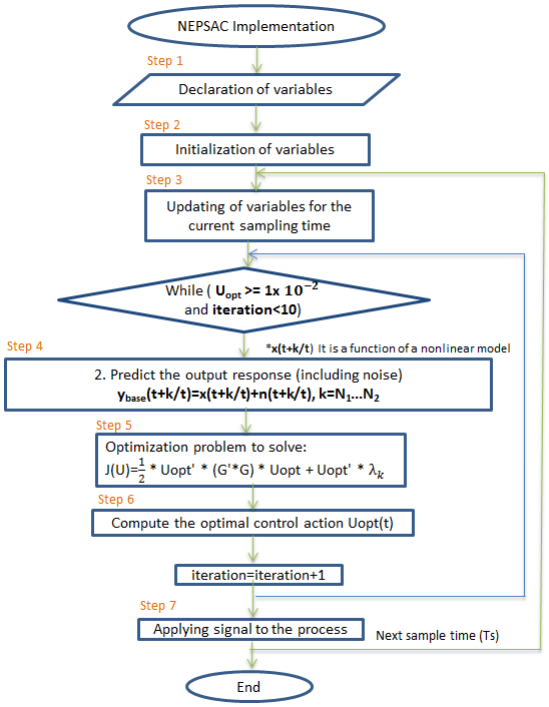
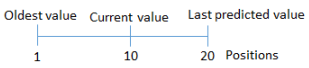


Fig. 8. NEPSAC Method

are omitted, $Ybase$ is calculate only when it is going to be used and not before.

a) **Step 3:** Updating of variables for the current sampling time:

$$[..Future(t+k), Present(t), Past(t-k)..]_{1..20}$$




```
X=[0; X(1:length(X)-1)];
```

```
#tmp[0] := 0.0;
FOR #k := 0 TO 18 DO
  #tmp[#k + 1]:= #X[#k];
END_FOR;
FOR #k := 0 TO 19 DO
  #X[#k] := #tmp[#k];
END_FOR;
```

b) **Step 5:** Optimization problem to solve:

```
Setpt=ones(10,1)*Wt;
Err=Setpt-Ybase;
Uopt=(G'*G)\(G'*Err);
```

```
#y := 0.0;
#c_y := 0.0;
FOR #k := 0 TO 9 DO
  #y := (#G[#k]*#G[#k])+ #y;
  #c_y:=((#Setpt- (#X[9 - #k]+
    #b_N[9-#k]))*#G[#k])+ #c_y;
END_FOR;
#U[10] := (#c_y / #y);
```

Fig. 9. Program part: Matlab(left) Vs SCL(right)

C. Stage 3: PLC programming

Import from TIA Portal V13 the code generated in the previous step and then generate a function block to it. The generated file will enter to TIA Portal as an external source, and it will be the content of a function block. The import process involves the creation of a database to manage all the variables of the new function block, it is necessary because the control code works with previous values, to calculate the current control action (t), the values of (t-1) are used. The downloaded program on the PLC (Figure 10) allows the execution of a control algorithm (independent of the control strategy) and storage of results in a database on-line. Once started the program from scratch, OB100 is executed to give initial values, the outputs are updated, the inputs are updated and it starts with the execution of the OB1 content. Each sampling time Ts the OB1 execution is paused to make way for the execution of the OB35 content, it gives way to compute of the control action and the data acquisition for input and output. Then the execution returns to the OB1 and once the process starts again it ends. Once all instructions end the process is repeated again.

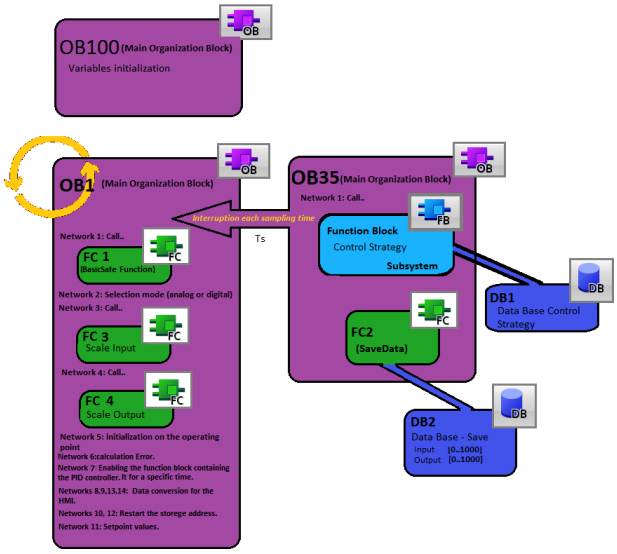


Fig. 10. Program Structure - PLC

PLC Implementation/Troubles: There are several and typical problems that should be consider such as:

- It is critically dependent on the quality of the system model, so the first step for successful implementation of MPC on PLC is the modelling and identification of the process.
- There is a gap between MPC theoretical research and practical application, It is necessary to design low cost application with simple configuration and easy to understand, like PID.

On the other hand, there is a difficulty specifically related with the execution of optimization algorithms in each sampling time as it presupposes a computational cost. Feasible solutions adopted in this work:

- Increase the data loaded off-line.
- Using lighter algorithms.
- Select PLC with appropriate memory capacity and processing speed.

PLC Coder makes two important contributions to mitigate the problems of implementation mentioned before: (1) Decreases the computational load: Avoid making calculations within the program that can be done off-line, the equations are written explicitly and the coefficients are loaded directly. (2) Optimizes memory data: Reduce the use of internal variables, variables are reused for temporary storage of data that do not necessarily have relationship. For this reason there are no direct correspondence between all the variables that are used in Matlab (used to store specific data) and the variables that are used in the resulting code.

D. Stage 4: Experimental validation

For experiments validation are used the following SISO two systems:

1) Pressure System - case linear: System pressure (Figure 11) is an energy storage system with self-regulation. The pump P101 delivers a fluid via a piping system into a pressure tank partly filled with gas (air), the pressure of the gas (air) in the pressure tank is detected by a piezoresistive relative pressure sensor. The manipulated value is the voltage of the pump, which sets the revolution speed. The step response

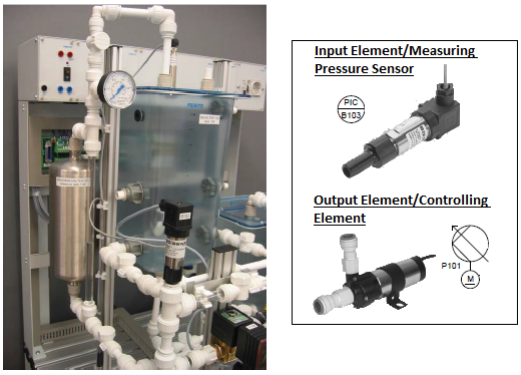


Fig. 11. Reference photo Festo MPS PA Compact Workstation Manual. Pressure System.

(Figure 12) shows a constant time $tao = 3.5s$, then the sampling time should be $Ts = tao/10 = 0.35s$. However, anticipating the computational burden that will have the PLC $Ts = 0.5s$ is chosen, it does not present any risk of error for the identification process because the system not presented a response significantly faster. The transfer function (9) is determined using parametric identification with ARX and PEM models for operating point $OP = 7VDC$.

$$F_T(S) = \frac{24.498}{S + 0.5077} \Longrightarrow F_T(Z) = \frac{10.82}{Z - 0.7758} \quad (9)$$

2) Level System - case non-linear: The pump P101 delivers a fluid from a storage tank B101 to a reservoir tank B102 via a piping system. The level of the fluid inside tank B102 is monitored with a analogue ultrasonic sensor B101. Manipulated value is the voltage of the pump, which sets the fluid quantity (Figure 13). Based on the mass balance for B101 and B102 is obtained the nonlinear model (10,11) to the system, under the

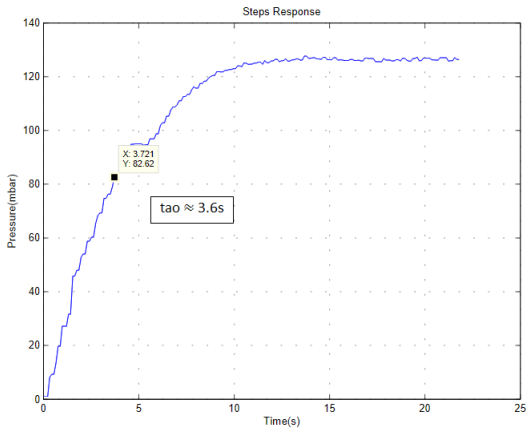


Fig. 12. Step Response Pressure System for OP

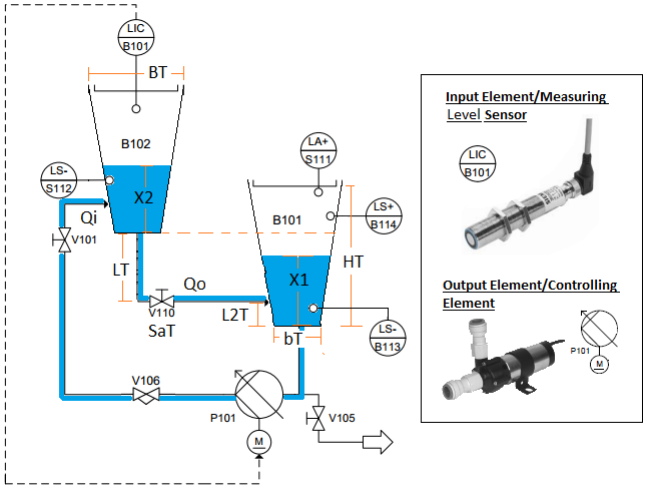


Fig. 13. P&ID System Level.

assumption that fluid density will not change. Where $Qi = aP * U_{control}(t-1) + BP$, $Qo = Cv\sqrt{X2 - X1} = \sqrt{2 * gT} * Sat\sqrt{[X2(t-1) + LT] - [X1(t-1) - L2T]}$, $A2t = [bT + 2 * \frac{BT - bT}{HT} * X2(t-1)]^2$, $A1t = [bT + 2 * \frac{BT - bT}{HT} * X1(t-1)]^2$.

$$X2 = X2(t-1) + [(Qi - Qo) * \frac{Ts}{A2t}] \quad (10)$$

$$X1 = X1(t-1) + [(Qi - Qo) * \frac{Ts}{A1t}] \quad (11)$$

For data acquisition the sampling time will be $Ts = 16s$, the same used in the reference thesis [20].

*E. Stage 5: Export the data as *.txt file*

Data acquisition is exported to a *.txt file from HMI (Figure 14) using the Visual Basic Script. The graphical interface is created with WinCC Runtime Advanced panel which contain all essential functions for operator control and monitoring of machines or plants from PC. Address the storage of data in the *.txt file from PLC database is used the instruction FieldRead.

A1T	Area-depends on the height of fluid in the tank B101
A2T	Area-depends on the height of fluid in the tank B102
X1	Height of fluid to tank B101 for the current Ts
X2	Height of fluid to tank B102 for the current Ts
Qi	Input flow for tank B102
Qo	Output flow for tank B102
aP,BP	Constants Pump relation flow/voltage (equivalent to the constant m, b of the linear equation y=mx+b
bT	Base length of the tanks
BT	Upper length of the tank (cover)
HT	Surface length of the tanks
LT,L2T	Initial values for the calculations heights
Sat,CVT	Constant to quantify the resistance of the valve V110
gT	Gravity constant

The database includes the runtime of OB35 which is calculated with the function RT INFO (Figure 15).

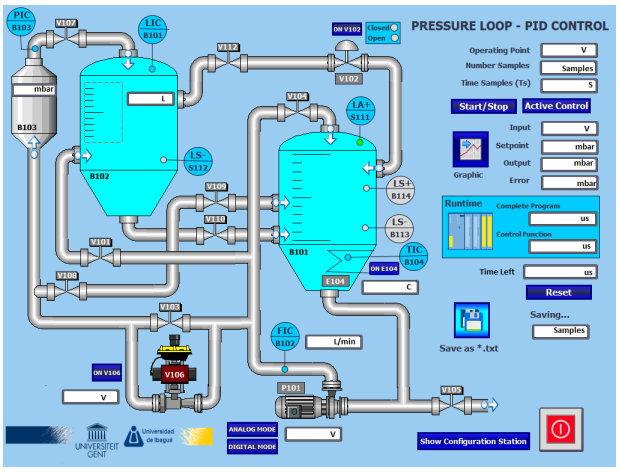


Fig. 14. HMI for managing FESTO MPS PA Compact Workstation

*Code description of VB function to save data from a PLC database to a *.txt file:* (1) Name of the VB function Save. (2) Variables declaration. (3) Use the CreateObject method to create a FileSystemObject object. The File System Object (FSO) model provides an object-based tool for working with folders and files. (4) Address of the text file where will be stored the data. If the file does not exist one is created in the specified direction. (5) Address at database data to read: SmartTags("INDEX1 SAVE")=index datas. (6) Copy the state of the global variables into the local variable: VOLTAGE P101=SmartTags("VOLTAGE P101 SAVE HMI"). (7) Write in a line of the text file: ts.WriteLine(VOLTAGE P101). (8) Finish and close.

V. Experimental Results

The experiments correspond to the application of a setpoint type ladder to evaluate the performance of the controllers when is necessary to follow references.

A. Pressure Control System

1) Performance EPSAC Controllers: PI controller will be used as reference for being the control solution more demand at industrial level implemented on PLCs. Thanks to that

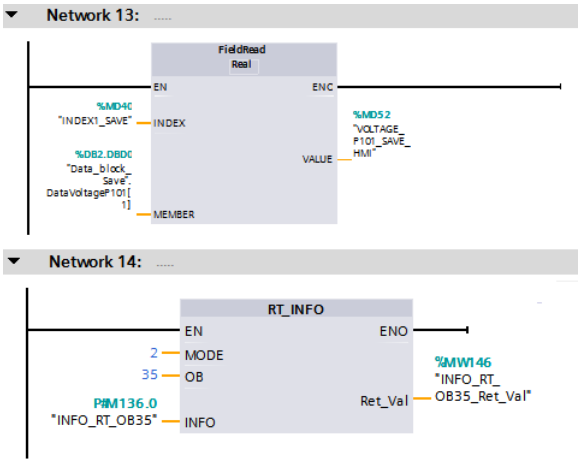


Fig. 15. Programming for HMI.

EPSAC knows the system model and its step response is possible to predict the best value to reach the setpoint, which is reflected in improving three aspects to this case (Figure 16): (1)Improved speed of response: to the same time (t) EPSAC is closer to the reference value that PI. (2)Elimination of delay or dead time, the PI response has delay that is overcome via the EPSAC (Figure 17). (3)Disappears the undesirable effect of initial conditions[4], the first EPSAC values are very close at the end optimal value. The price to pay for the

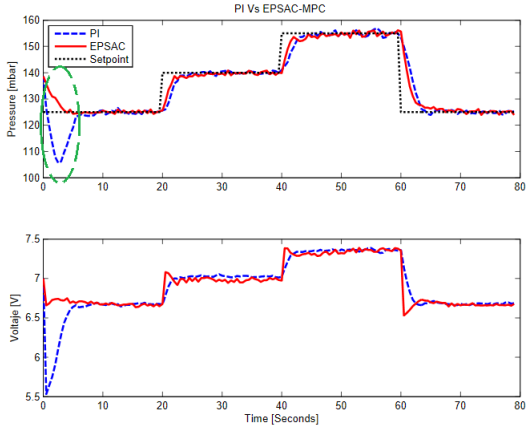


Fig. 16. PI Vs EPSAC - Linear Case

advantage of speed of response that has the EPSAC controller is the consequent increase in the control effort, however, it is possible to improve this situation by restricting the maximum change between a control signal of last time $U_t(t-1)$ and the control signal for the current time U_t(t), through the inclusion

[4]For this experiment began with the plant on the operating point 7v, which provides an pressure of 139mbar , this initial value explains the first value of PI highlighted by the green circle. In the simulation, the controllers does not has initial conditions, thus the values 7v and 139mbar are added to the final values (input and output) in order to compare it with the actual response of the system.

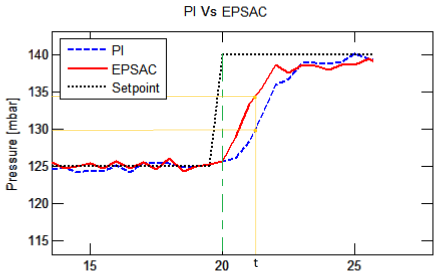


Fig. 17. Delay PI Vs EPSAC - Linear Case

of the constraints (Figure 18). The output signal does not

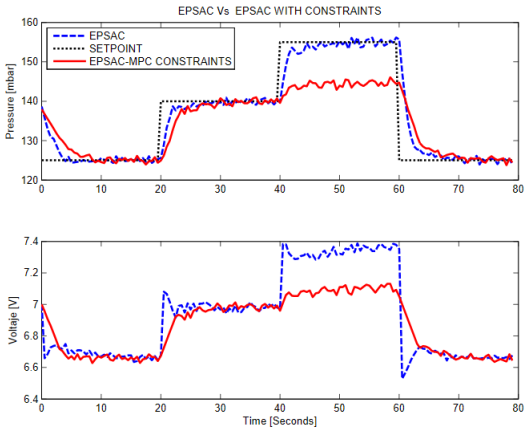


Fig. 18. EPSAC Vs EPSAC CONSTRAINTS: (1)$145mbar > Y_t > 0mbar$, (2)$10VDC > U_t > 0VDC$, (3)$0.05v > U_t(t-1) - U_t(t) > -0.05v$ - Linear Case

exceed the value specified as the maximum value Ymax - Constraint 1. The output signal has a control effort minor that the control effort observed for the EPSAC unconstrained case; the peaks are removed and slope changes are much softer, this thanks to the constraints 3 which limits the changes in control signal $\pm 0.05v$. For maximum setpoint value of a signal control required is not greater than 10v, it means that constraint 2 is no violated for the case studied.

2) Runtime EPSAC Controllers: The running time is directly proportional to the complexity of the control strategy algorithm, which represents a reference to the computational burden that will have to support the controller hardware and represent a selection criterion related to the processing speed of it. Note that the runtime for EPSAC controller with constraints is 65 times larger than for EPSAC without constraints and 574 times greater than the runtime for the PI controller. As expected the controller EPSAC With restrictions have the highest runtime, Which is justified by the number of iterations required to find multipliers Lagrange in order to solve the optimization problem (QPH algorithm). When Y_{max} restriction is active the runtime increases considerably.

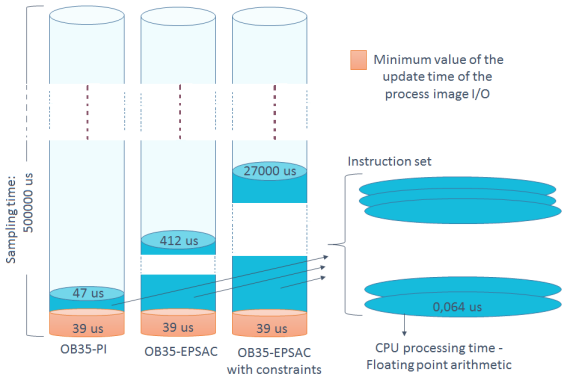


Fig. 19. Runtimes OB35 - Linear Case

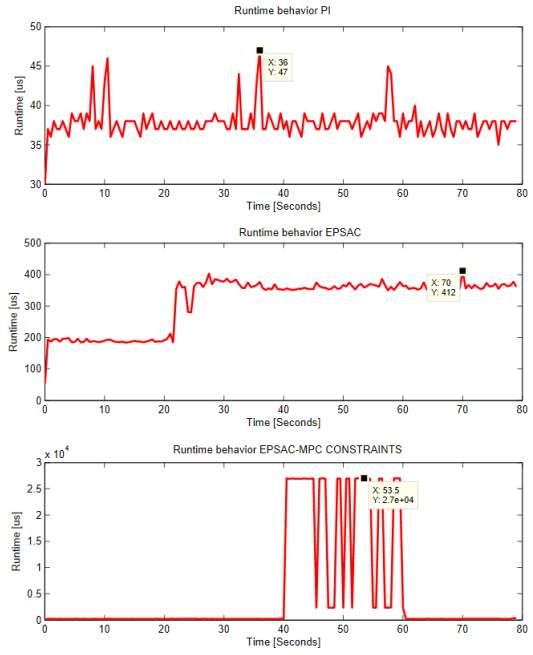


Fig. 20. Runtime behavior - Linear Case

3) Memory Occupation EPSAC Controllers: The EPSAC code is considerably longer compared to the code of a PI algorithm, the EPSAC approach uses large amount of data (is organized into vectors) consulted and updated every sampling time.

Note that the EPSAC controller with constraints uses about 3 times more load memory resources that EPSAC controller without constraints, and about 6 times more that PI controller. As for work memory note the large increase in the resources used by EPSAC controller with constraints regarding the EPSAC controller without constraints (about 5 times) and with respect to PI controller (about 15 times).

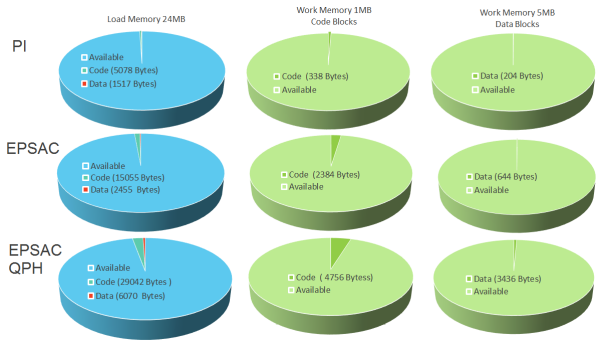


Fig. 21. Memory Occupation - Linear Case

B. Level Control System

1) Performance NEPSAC Controllers: The performance of a NEPSAC controller for the nonlinear level system is compare EPSAC controller which is formulated with the function transfer obtained from the linearization of the model around a operating point $h2o = 0.1991 + L$ and $h1o = 0.1454 - L2$. NEPSAC presents benefits in two relevant aspects: better following of the reference and soft control actions (Figure 22).

$$F_T(S) = \frac{0.06008}{385.7S + 1} \Longrightarrow F_T(Z) = \frac{0.0024}{Z - 0.9594} \qquad (12)$$

Figure 23 shows the number of iterations in relation with the time of the experiment, note that the number of iterations increases when the setpoint value is changed and decreases to one iteration to maintain this value.

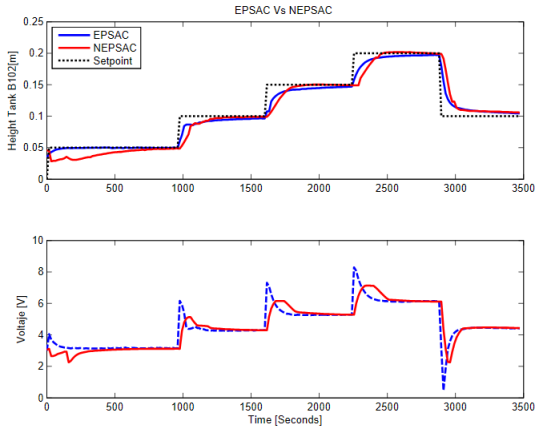


Fig. 22. EPSAC Vs NEPSAC - Non linear Case

Note: that between the graph of real data and the graph obtained by simulation there are some differences, it is due to the difference between the experimental value and real value of the pump constant (relation flow/voltage) and the value for valve opening V110 (This also applies to the simulation of the NEPSAC controller).

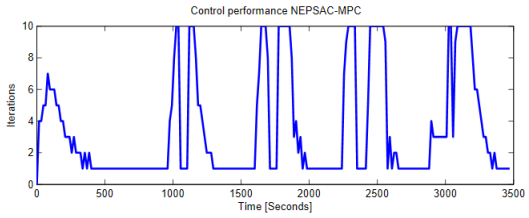


Fig. 23. Iterations NEPSAC - Non linear Case

2) Runtime NEPSAC Controllers: Note that during the change of value in the setpoint the execution time for the NEPSAC becomes up to 9 times major that the runtime for EPSAC controller, where are necessary a greater number of iterations. This result is comparable to the result obtained for the EPSAC controller with constraints for pressure system control, note that the runtime here is much lower because in this case the restrictions are not violated.

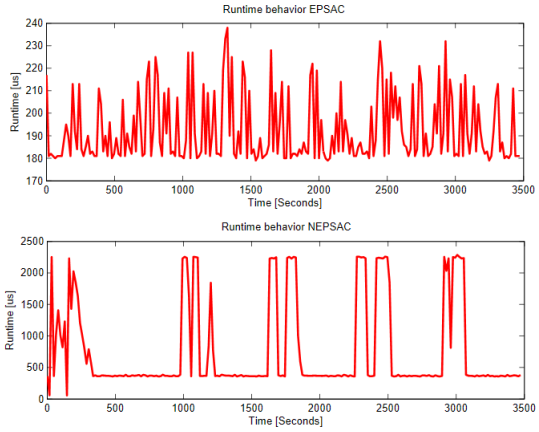


Fig. 24. Runtime behavior - Non linear Case

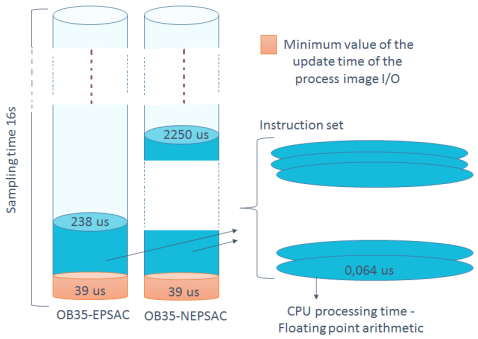


Fig. 25. Runtimes OB35 - Non linear Case

3) Memory Occupation NEPSAC Controllers: The figure 26 shown the occupation memory to non linear case, noten that the NEPSAC approach uses about 2 times more resources that EPSAC approach.

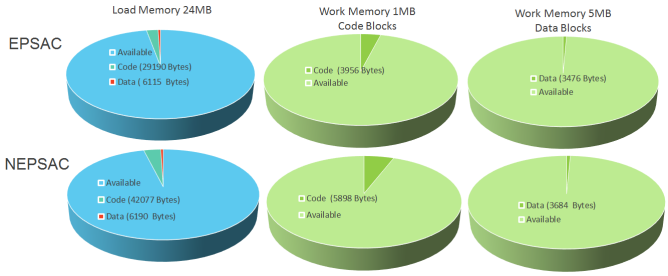


Fig. 26. Memory Occupation - Non linear Case

VI. Conclusion

It has successfully shown the possibility of implementing control strategies predictive: EPSAC. EPSAC with constrains and NEPSAC for SISO Systems on a Siemens PLC 1516-3 and in consequence on devices with similar Technical Characteristics.

A methodology to implement advanced controllers on PLC has been explained step by step, providing the following advantages:

- Mode to use the PLC as data acquisition card for identification of SISO linear and non-linear systems.
- Straightforward methodology to generate SCL code from PLC Coder tool of Matlab.
- Experimental validation on a pilot plant for PI(D), constrained MPC controllers and non-linear MPC controllers.
- Evaluation of memory and CPU requirements to implement advanced controllers.

Future work includes the assessment of computation time in the QP solver and extension of the analysis for multivarible and more complex systems than those evaluated in this study.

References

[1] VALENCIA-PALOMO, G.; ROSSITER, J. A. *Efficient suboptimal parametric solutions to predictive control for PLC applications*, Control Engineering Practice, 2011, vol. 19, no 7, p. 732-743.

[2] GRAICHEN, B. Kpernick K. *PLC Implementation of a Nonlinear Model Predictive Controller*, 2014.

[3] HUYCK, Bart, et al. *Online model predictive control of industrial processes using low level control hardware: A pilot-scale distillation column case study*, Control Engineering Practice, 2014, vol. 28, p. 34-48.

[4] HUYCK, Bart, et al. *Implementation and experimental validation of classic MPC on programmable logic controllers*, Control and Automation (MED), 2012, 20th Mediterranean Conference IEEE, 2012. p. 679-684.

[5] BOLTON, William. *Programmable logic controllers*, Newnes, 2015.

[6] CAMACHO, Eduardo F.; ALBA, Carlos Bordons,*Model predictive control*, Springer Science and Business Media, 2013.

[7] DE KEYSER, Robin.*Model based predictive control for linear systems*, Invited Paper UNESCO Encyclopaedia of Life Support Systems, 2003.

[8] DE KEYSER, R. M. C.; VAN CAUWENBERGHE, A. R.*Typical application possibilities for self-tuning predictive control*, IFAC Symp. Ident. and Syst. Par. Est. 1982.

[9] WANG, Liuping.*Model predictive control system design and implementation using MATLAB*, Springer Science and Business Media, 2009.

[10] ANDERSSON, Lennart, et al.*A manual for system identification. Laboratory Exercises in System Identification*, KF Sigma i Lund AB. Department of Automatic Control, Lund Institute of Technology, Box, 1998, vol. 118.

[11] ISAKSSON, Alf; HAGGLUND, Tore.*Editorial- PID control*, IEE Proceedings - Control Theory and Applications, 2002, vol. 149, no 1, p. 1-2.

[12] YU-GENG, X. I.; DE-WEI, L. I.; SHU, Lin.*Model predictive control status and challenges*, Acta Automatica Sinica, 2013, vol. 39, no 3, p. 222-23.

[13] BINDER, B. J. T.; KUFOALOR, D. K. M.; JOHANSEN, T. A. *Scalability of QP solvers for embedded model predictive control applied to a subsea petroleum production system*, IEEE Conference on Control Applications (CCA), 2015. p. 1173-1178.

[14] MISGELD, Berno JE; POMPRAPA, Anake; LEONHARDT, Steffen.*Didactic approach to multivariable control using IEC 61131 model based design and programmable logic controllers*, IFAC Proceedings Volumes, 2013, vol. 46, no 17, p. 220-225.

[15] GREGA, Wojciech; KOLEK, Krzysztof.*Simulation and real-time control: from Simulink to industrial applications*, Computer Aided Control System Design, IEEE International Symposium, 2002. p. 104-109.

[16] LIMA, Daniel M.; DA COSTA, Marcus V. Americano; NORMEY-RICO, Julio E. *A flexible low cost embedded system for Model Predictive Control of industrial processes*, Control Conference (ECC), 2013 European. IEEE, 2013. p. 1571-1576.

[17] ZUPANI, Borut. *Extension software for real-time control system design and implementation with MATLAB-SIMULINK*, Simulation Practice and Theory, 1998, vol. 6, no 8, p. 703-719

[18] BAUER, Margret; CRAIG, Ian K. *Economic assessment of advanced process control survey and framework*. Journal of Process Control, 2008, vol. 18, no 1, p. 2-18.

[19] DARBY, Mark L.; NIKOLAOU, Michael.*MPC: Current practice and challenges*, Control Engineering Practice, 2012, vol. 20, no 4, p. 328-342.

[20] Rincon Fernandez Pacheco, Alejandro. *Identification and Control of a MIMO System with Slow and Fast Dynamics*, Ghent University, 2014.

[21] (2016) Siemens website. Siemens Supporting Documentation to TIA Portal and PLC1516-3 [Online]. Available: http://www.industry.siemens.com/topics/global/en/tia-portal/pages/default.aspx/

[22] (2016) Mathworks website. Simulink PLC Coder. Generate IEC 61131-3 Structured Text for PLCs and PACs [Online]. Available: http://www.mathworks.com/products/sl-plc-coder/