Auto-tuned Predictive Control Based on Minimal Plant Information

G. Valencia-Palomo * **J.A. Rossiter** *

* *Department of Automatic Control and Systems Engineering, University of Sheffield, South Yorkshire, U.K. S1 3JD.*
(e-mail: g.valencia-palomo@shef.ac.uk, j.a.rossiter@shef.ac.uk).

Abstract: This paper makes two key contributions. First there is a definition and implementation of a novel auto-tuned predictive controller. The key novelty is that the modelling is based on relatively crude but pragmatic plant information. Secondly, the paper tackles the issue of availability of predictive control for low level control loops. Hence the paper describes how the controller is embedded in an industrial Programmable Logic Controller (PLC) using the IEC 1131.1 programming standard. Laboratory experiment tests were carried out in two bench-scale laboratory systems to prove the effectiveness of the combined algorithm and hardware solution. For completeness, the results are compared with a commercial PID controller (also embedded in the PLC) using the most up to date auto-tuning rules.

Keywords: Predictive control, auto-tuning, programmable logic controller, IEC-1131.1.

1. INTRODUCTION

Control design methods based on the predictive control concept have found wide acceptance in industry and in academia, mainly because of the open formulation that allows the incorporation of different types of models of prediction and the capability of constraint handling in the signals of the system.

Model predictive control (MPC) has had a peculiar evolution. It was initially developed in industry where the need to operate systems at the limit to improve production requires controllers with capabilities beyond PID. Early predictive controllers were based in heuristic algorithms using simple models. Small improvements in performance led to large gains in profit. The research community has striven to give a theoretical support to the practical results achieved and thus the economic argument, predictive control has merited large expenditure on complex algorithms and the associated architecture and set up times. However, with the perhaps notable exception of Predictive Functional Control (PFC) (Richalet, 1993), there has been relatively little penetration into markets where PID strategies dominate, and this despite the fact that predictive control still has a lot to offer in the SISO domain because of its enhanced constraint handling abilities and the controller format being more flexible than PID. The major obstacles cost, complexity and the algorithm not being available in the off the shelf hardware most likely used for local loop control.

Some authors have improved the user-friendliness (complexity) of MPC software packages available for high level control purposes (Froisy, 2006; Zhu *et al.*, 2008). Nevertheless, they have the same implementation drawback in that the development platform is a stand-alone computer running under Windows® OS. Furthermore, these packages involve complex identification procedures which thus requires the control commissioning to be in the hands of a few skilled control engineers; ownership by non control experts is an impediment for more widespread utilization.

Some early industrial work (Richalet, 2007) has demonstrated that with the right promotion and support, technical staff are confident users of PFC where these are an alternative to PID on a standard PLC unit. Technical staff relate easily to the tuning parameters which are primarily the desired time constant and secondly a coincidence point which can be selected by a simple global search over horizons choices. Because PFC is based on a model, the controller structure can take systematic account of dead-times and other characteristics, which are not so straightforward with PID. Also constraint handling can be included to some extent by using predicted violations to trigger a temporary switch to a less aggressive strategy.

The vendors conjecture is that PFC was successfully adopted because of two key factors: first there is effective support in technician training programmes (get it on the syllabus) and second the algorithm is embedded in standard PLC hardware they encounter on the job, thus making it easily accessible (and cheap). However, despite its obvious success academia has shied away from the PFC algorithm because its mathematical foundations are not as systematic or rigorous as other approaches; the performance/stability analysis is primarily an a *posteriori* approach as opposed to the a *priori* one more popular in modern literature. So there is a challenge for the academic community to propose more rigorous but nevertheless intuitive and simple algorithms which could equally be embedded in cheap control units.

On the other hand, in recent specialized conferences authors are often focussing on the level of rigor required in the modelling and tuning procedure for different cases (Morari *et al.*, 2008). However, accessibility and useability in such a mass market may require different assumptions from those typically adopted in the literature; specifically much less rigor and more automation in the modelling will be essential.

Hence, the first objective of this paper is to develop an auto-tuned MPC controller based on minimal plant information which would be available from staff at technician level only who may be responsible for maintaining and tuning local loops. Secondly, the paper aims to demonstrate how an MPC algorithm, using this model information, can be embedded in a commercial PLC (Valencia-Palomo and Rossiter, 2008); this paper gives some extensions to that developments in (Valencia-Palomo *et al.*, 2008) and of particular interest to readers will be the incorporation of systematic constraint handling within the PLC unit. A final objective is to contrast the auto-tuned MPC with a commercial PID controller in order to show that the MPC is a practical (available and same cost) alternative to PID for local loops.

The paper is organized as follows: Section 2 outlines the controllers and the auto-tuning rules, Section 3 describes the implementation of the controllers in the target hardware, Section 4 presents the simulation results on real hardware and finally in Section 5 are the conclusions and future work.

2. THE CONTROLLERS

This section outlines the auto-tuning rules and modelling assumptions for the MPC and PID strategies adopted. We note that the auto-tuning rules are only applicable to stable systems so discussion of unstable systems is deferred for future work.

2.1 Modelling assumptions

If anything, this paper is more generous with the auto-tuned PID than the MPC because it allows the PID algorithm a large quantity of measurement data and the ability to dither the input substantially during tuning to extract the required information. Moreover, the complexity of this algorithm means that the modelling is done offline. This decision was taken to give a stiff test for the auto-modelled/tuned MPC algorithms.

For MPC we provide crude modelling information only, for instance as could be provided by a technician or plant operator but specifically avoiding the use of a rigorous least squares model estimator which could be expensive if required for large numbers of loops and impractical to put on the PLC unit. The technician should provide estimates of behaviour as compared to standard second order characteristics: rise-time, settling time, overshoot, steady-state gain and dead-time. From this data an approximate second order model with dead-time is determined[1].

2.2 Design point, auto-tuning and constraint handling for PID

A novel auto-tuned PID controller as described in (Clarke, 2006; Gyöngy and Clarke, 2006) is used. A schematic

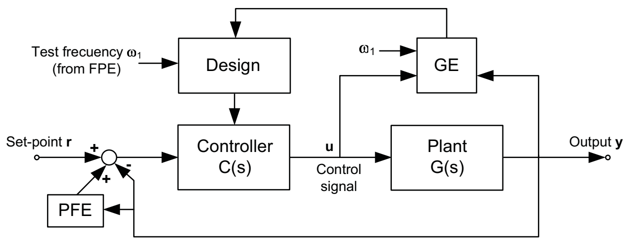


Fig. 1. Schematic diagram of the auto-tuning PID.

diagram of the system is shown in Fig. 1. The objective is to adapt the controller so as to achieve a carefully chosen design point on the Nyquist diagram.

The key components are phase/frequency and plant gain estimators (PFE, GE), described in detail in (Clarke, 2002). In essence a PFE injects a test sinewave into a system and continuously adapts its frequency ω_1 until its phase shift attains a desired value θ_d (in this case the design point). Also forming important part of the tuner, but not shown in Fig. 1, are variable band-pass filters (VBPF) at the inputs of the PFE and GE. These are second-order filters centered on the current value of the test frequency. They are used to isolate the probing signal from the other signals circulating on the loop (such as noise, set-point changes and load disturbances).

The algorithm is initialized using a first-order/dead-time (FODT) approximation $G_a(s)$ for the plant, obtained from a simple step test. The initialization involves the computation of suitable values for the parameters associated with the GE, PFE and the controller.

The controller is based on a design point in the Nyquist diagram. This design point is chosen to obtain the desired closed loop behavior, i.e. rise time, damping value, settling time. In this case, the desired damping value of 0.5 for all the systems is chosen. From this desired damping value, the variables for all the auto-tuning process are obtained as is shown in (Clarke, 2006; Gyöngy and Clarke, 2006).

The PID design does not take explicit account of constraints and thus ad hoc mechanisms are required. Typically input saturation with some form of anti-windup will be used but state constraints are not considered; this is a weakness.

2.3 Basic assumptions for MPC

For the purpose of this paper almost any conventional MPC algorithm can be deployed as the main distinguishing characteristic, **with sensible tuning**, is the model. Hence, assume that the MPC law can be reduced to minimising a GPC[2] cost function of the form:

$$J = \sum_{j=1}^{H_P} \|\hat{\mathbf{y}}(k+j|k) - \mathbf{w}(k+j|k)\|^2 + \sum_{j=1}^{H_C} \|\Delta\mathbf{u}(k+j|k)\|_\lambda^2 \qquad (1)$$

where the second term in the eq. (1) is the control effort and λ is the weighting sequence factor. The reference trajectory $\mathbf{w}(k)$, is the desired output in closed loop of the system and is given by:

[1] We accept that for more complex dynamics a slightly more involved procedure may be required.

[2] To simplify some algebra compared to dual-mode approaches, e.g. (Rossiter *et al.*, 1998).

$$\mathbf{w}(k+i|k) = \mathbf{s}(k+i) - \alpha^i [\mathbf{s}(k) - \mathbf{y}(k)]\,;\ 1 \leq i \leq H_P \tag{2}$$

where $\mathbf{s}(k)$ is the set-point and α determines the smoothness of the approach from the output to $\mathbf{s}(k)$. The objective (1) can be expressed in more compact form in terms of the predicted output:

$$\min_{\Delta \mathbf{u}} \mathbf{J}(\Delta\mathbf{u}) = \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{f}^T\Delta\mathbf{u} + \mathbf{b} \tag{3}$$

$$s.t. \qquad \mathbf{R}\Delta\mathbf{u} \leq \mathbf{c} \tag{4}$$

where $\Delta\mathbf{u}$ is the vector of future inputs increments and the other matrix details are omitted for brevity but available in standard references, e.g. (Maciejowski, 2002; Rossiter, 2003; Camacho and Bordons, 2004).

The tuning parameters are usually taken to be the horizons H_P, H_C and weights λ. However, more recent thinking suggests that H_P should be larger than the settling time, H_C is typically 2 or 3 (for practical reasons rather than optimality which requires higher values) and λ becomes the major tuning parameter, albeit some may argue a poor mechanism for tuning. The parameter α will also have a substantial impact but is rarely discussed except in PFC approaches.

2.4 Constraint handling for MPC

The systems considered in this paper are stable, therefore in the absence of output constraints, for a reachable set point the system will only violate the constraints in presence of disturbances or overshoots derived from set point changes. In practice, one may not be able to program a complete QP solver, so a sensible way of handling constraints is to interpolate two control laws (Rossiter and Grieder, 2005), one with good performance (e.g. $\Delta\mathbf{u}_{fast}$) and one with good feasibility (e.g. $\Delta\mathbf{u}_{slow}$) , using:

$$\Delta\mathbf{u} = (1-\beta)\Delta\mathbf{u}_{fast} + \beta\Delta\mathbf{u}_{slow}; \quad 0 \leq \beta \leq 1 \tag{5}$$

The variable β is used to form the mix of fast and slow according to the predicted situation (if feasible $\beta = 0$). Hence, the optimization procedure reduces to simple linear program in one variable that is a set of inequality checks of the form:

$$\min \beta \;\; s.t. \;\; R_i\beta - c_i \leq 0, \quad i = 1, ..., H_C \tag{6}$$

Remark 1. If $\beta = 1$ and the constraints are still being violated, the inputs are saturated. Essentially this means the state is outside the maximal admissible set for the unconstrained control law designed for good feasibility. Such scenarios need more complex strategies not covered in this paper.

2.5 Simple auto-tuning rules for MPC

There are many alternatives for auto-tuning, some with better properties then used here are possible, but the authors felt this paper should initiate discussion with an industrial standard. Thus, the predictive control design and tuning procedure is described next.

For the MPC the prediction horizon H_P is chosen equal to the settling time plus H_C, with $H_C \ll H_P$. Assuming normalisation of input/output signals, $0.1 \leq \lambda \leq 10$, a form of global search can be used to settle on the 'best' parameters against some criteria, however, if we take the criteria to be the cost J of eq. (1) with $\lambda = 1$, then this fixes λ and H_C is chosen to be as large as possible[3]. The design response speed α_{fast} (for $\Delta\mathbf{u}_{fast}$) will be taken as half the open-loop time constant α_0 so that the controller has to deliver some extra speed of response as well as stability and offset free tracking; and α_{slow} (for $\Delta\mathbf{u}_{slow}$) will be taken as 0.95 to have a smooth close loop response and avoid overshooting in set point changes. Thus, the auto-tuning is fixed precisely by the model parameters and the technician role is only to provide best estimates of these parameters (in practice some iteration should take place).

Fig. 2. Allen Bradley PLC – SCL500 processor family.

3. IMPLEMENTATION OF THE ALGORITHMS ON A PROGRAMABLE LOGIC CONTROLLER

This section briefly introduces the PLC and the corresponding implementation of the controllers described in the previous section.

3.1 Allen Bradley — Rockwell Automation Inc. PLC

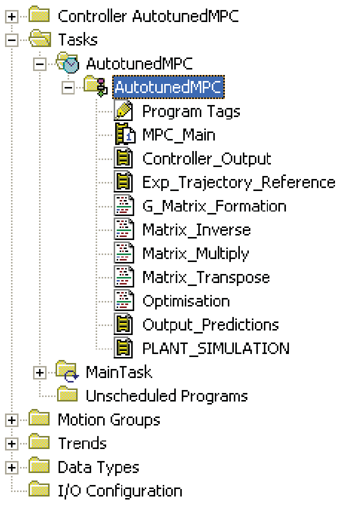
PLCs are by far the most accepted computers in industry which offer a reliable, safe and robust system; we will not revisit the reasons here. Nevertheless, normally their use is only to implement control sequences in open loop and/or different structures of PID controllers. For the purposes of this paper, the implementation is based on the family of SLC500 processors belonging to the Allen Bradley PLC systems, e.g. see Fig. 2.

The Allen Bradley set of PLC includes the facilities to be programmed in 3 of 5 languages in agreement with the IEC 1131.3 standard using Control Logix 5000^{TM} software programming package. Each of these allows for any combination of programming languages to be used for a single project. These three languages are: (i) *Ladder Diagram*, (ii) *Function Block Diagram* and (iii) *Structured Text.*

3.2 PID

The Control Logix 5000^{TM} software programming package also includes a function block to implement a PID controller. The PID function block is a professional development from Rockwell Inc. used in industry (with a PLC) to control a variety range of processes.

[3] For sensible sample times, a choice of $H_C \geq 5$ implies that this is approximately equivalent in behaviour to a dual-mode approach.



```
Controller AutotunedMPC
Tasks
    AutotunedMPC
        AutotunedMPC
            Program Tags
            MPC_Main
            Controller_Output
            Exp_Trajectory_Reference
            G_Matrix_Formation
            Matrix_Inverse
            Matrix_Multiply
            Matrix_Transpose
            Optimisation
            Output_Predictions
            PLANT_SIMULATION
    MainTask
    Unscheduled Programs
Motion Groups
Trends
Data Types
I/O Configuration
```

Fig. 3. Structure of the MPC algorithm in the target PLC.

The tuning of the PID is done off-line by the algorithm described in subsection 2.2. The obtained parameters are passed to the PID block before downloading the program to the PLC. As noted earlier, the PID has been unfairly favoured here in that this off-line procedure requires a certain amount and type of experimental data for the model identification of the process.

This controller is going to be used to compare the results obtained with the auto-tuned MPC.

3.3 MPC

For the implementation, is worth mentioning that it is advisable to use a a graphical language such as *ladder logic* or *function block* because technical staff are much more familiar with this than structured text; also it is easier to maintain and debug for the changing nature of bits, timers, counters etc. while being monitored. However, a significant barrier for MPC implementation is that operations between vectors and matrices are not defined in any of the supported IEC 1131.1 languages, thus, all of these operations have to be programmed from scratch.

The complete structure of the proposed MPC program (based on subsections 2.3–2.5) is shown in Fig. 3. The algorithm has been programmed in the High Priority Periodic Execution Group[4] called `AutotunedMPC` which contains the routines summarised in Table 1. The software design, matrix formations, sequence of execution and the computation of the calculated output is described in detail in (Valencia-Palomo and Rossiter, 2008); with the exception of the routine `Optimisation` which is new to this paper and developed to include constraint handling, as described in subsection 2.4.

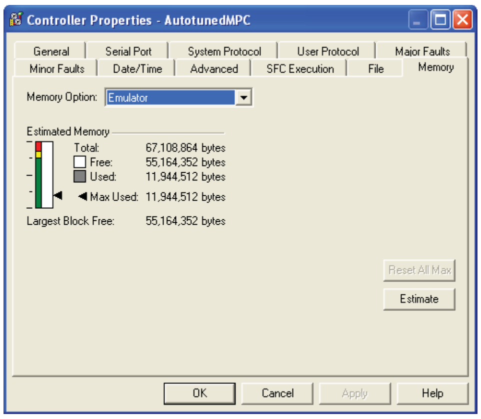
It can be seen from the properties of the controller with the RSLogix programming tool (Fig. 4) that the programm uses 17% of the available storage of the PLC including memory requirements for I/O, running cache and other necessary subroutines.

Finally, the input parameters for the program are only those who are related with de model, i.e. dead time t_d, rise time t_r, settling time t_s, overshoot M_P, gain K and sampling time Ts. The tuning is done online in the first scan of the program and is not repeated after, however, some mechanisms to update the model of the plant and tuning parameters can be embedded.

[4] This periodicity is set up with the chosen sample time.

Table 1. Routines and programming languages.

	Routine name	Programming language
1	`MPC_Main`	Ladder logic
2	`Controller_Output`	Ladder logic
3	`Exp_Trajectory_Reference`	Ladder logic
4	`G_Matrix_Formation`	Structured text
5	`Matrix_Inverse`	Structured text
6	`Matrix_Multiply`	Structured text
7	`Matrix_Transpose`	Structured text
8	`Output_Predictions`	Ladder logic
9	`Plant_Simulation`	Ladder logic
10	`Optimisation`	Structured text



Controller Properties - AutotunedMPC

Memory Option: Emulator

Estimated Memory

```
Total:                67,108,864 bytes
Free:                 55,164,352 bytes
Used:                 11,944,512 bytes
Max Used:             11,944,512 bytes
Largest Block Free:   55,164,352 bytes
```

Fig. 4. MPC memory usage in the target PLC.

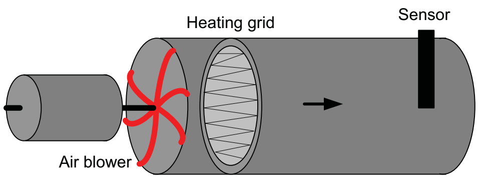


Fig. 5. Heating process.

4. EXPERIMENTAL LABORATORY TESTS

This section shows the experimental results from applying the MPC law via a PLC on a first and a second order plant. For both processes the interest is tracking of step references, which is the most common situation in industry. The PID/MPC experiments ran under the same conditions, in so far as this can be guaranteed.

4.1 First order plant – Temperature control

The first experiment is a heating process consisting of a centrifugal blower, a heating grid, a tube and a temperature sensor, see Fig. 5. The objective is to control the temperature at the end of the tube by manipulating speed of the blower (the input voltage of the D.C. motor).

The model of the plant for the PID off-line tuning is done by a least square estimator assuming a first order plant.

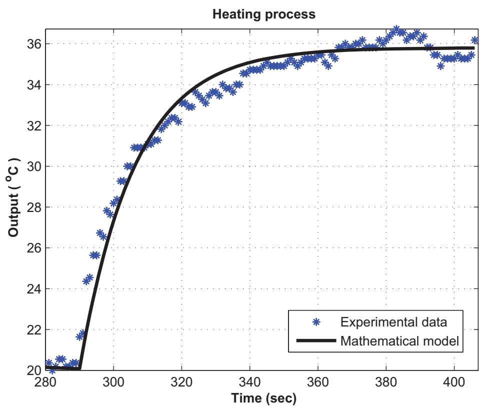


Fig. 6. Model validation of the heating process.

Table 2. Tuning parameters for the PID controller and input parameters for the auto-tuned MPC.

		Heating Proc.	**Speed Proc.**	
Cont.	Par.	Value	Value	Units
PID	P	1.590	0.325	–
	I	0.486	0.799	–
	D	0.0	0.0	–
MPC	Ts	10.0	0.50	Sec
	t_d	2.0	1.00	Samples
	t_r	38.0	4.90	Sec
	t_s	64.0	2.71	Sec
	K	16.0	190.0	–
	M_p	–	–	%

The resulting discrete model, with a sampling time of 10 sec. (which is also roughly the dead-time), is:

$$x(k+1) = 0.94x(k) + u(k)$$
$$y(k) = 0.94x(k)$$

The experimental validation of the mathematical model is shown in Fig. 6.

The tuning parameters for the PID controller and the input parameters for the Auto-tuned MPC are shown in Table 2. The second order approximate model for the predictions is built using standard analysis of the transient response of the plant i.e. nonparametric identification.

The simulation comparisons deploy a step change at $t = 50\,secs$ of the set point from 20 $°C$ to 30 $°C$; after the output reaches a new steady state, the process is disturbed at $t \approx 120\,secs$ by partially blocking the end of the tube. A new change in the set point to 20 $°C$ is required at $t = 180\,secs$ without taking out the disturbance.

It can be seen in Fig. 7 that the plant output is successfully tracking the reference signal for both controllers. Of course this is a simple first order model and thus good control is to be expected. MPC has clearly given better control of overshoot and settling.

4.2 A second order plant – Speed control

This process consists of a motor fitted with a speed sensor, the control objective is to regulate the speed of the motor by manipulation of the input voltage. The same procedure as in the first experiment is applied. The mathematical model of the system with a sampling time of 0.5 sec is:

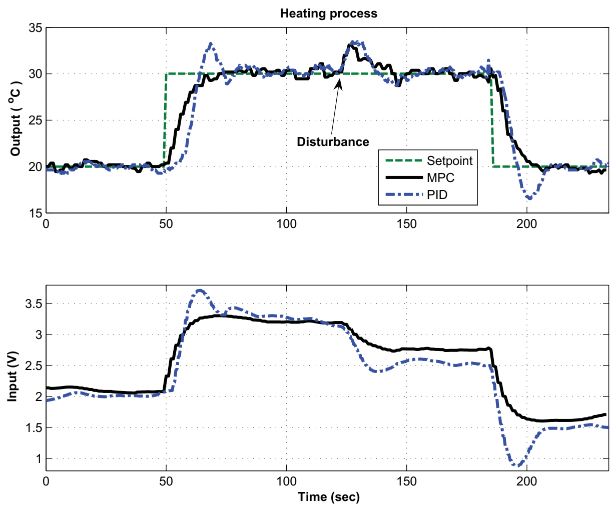


Fig. 7. Experimental test for the heating process.

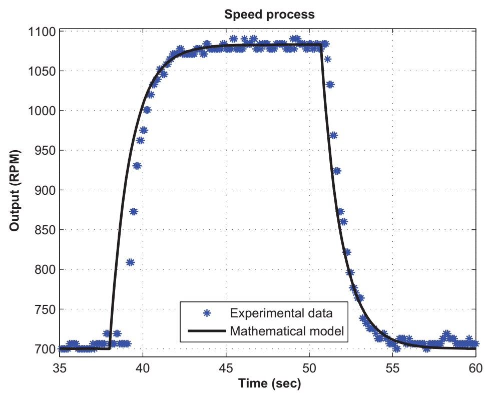


Fig. 8. Model validation of the speed process.

$$x(k+1) = \begin{bmatrix} 0.93 & -0.01 \\ 0.04752 & 0.9964 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} -0.01 & 3.71 \end{bmatrix} x(k)$$

The experimental validation is shown in Fig. 8 and the tuning parameters are in Table 2. Two set point step changes are demanded; once again, the results in Fig. 9 show that MPC and PID are tracking the set point accurately.

4.3 Performance indexes of the algorithms

The numerical performance indexes of the systems with the two different controller strategies are summarised in Table 3. Specifically the table shows the measures of performance are given by the cost function (J), the settling time (τ_s) and the overshoot (M_p). These numbers show that MPC performs similar to the standard PID controller but,in this case, with a much simpler auto-tuning procedure.

The constraint handling was not tested in a rigorous manner to let the PID controller act in the best possible scenario. Despite that, this simple control task finds its optimal value in saturation (Rojas and Goodwin, 2002).

To complete the analysis of the implemented program, the diagnostics tool from the hardware (shown in Fig. 10)

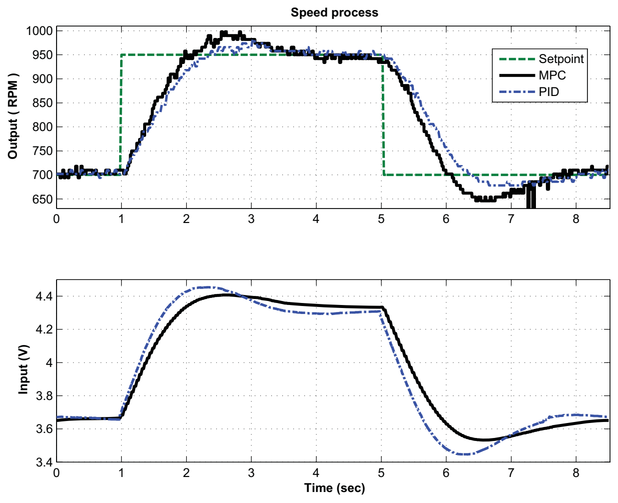
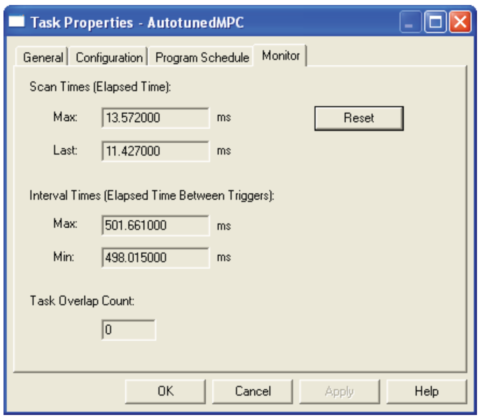


Fig. 9. Experimental test for the speed process.



Scan Times (Elapsed Time):

Max: 13.572000 ms

Last: 11.427000 ms

Interval Times (Elapsed Time Between Triggers):

Max: 501.661000 ms

Min: 498.015000 ms

Task Overlap Count:

0

Fig. 10. Execution time and sampling jittering for the speed process.

Table 3. Performance indices for the systems.

	Heating process			Speed process		
	τ_s	M_p	J	τ_s	M_p	J
PI	31 sec	33.45 %	1645	3.3 s	16%	2,615
MPC	20 sec	0.0 %	1629	3.3 s	8%	2,816

displays that the time for scanning the program each sample time oscillates between 11.42 ms and 13.57 ms while the elapsed time between triggers (sampling instants) for the speed process oscillates between 498.01 ms and 501.66 ms. The significance of this is the potential to apply the algorithm on much faster processes.

5. CONCLUSIONS

This paper has made three contributions. First it has demonstrated that an MPC algorithm with systematic, albeit simplistic, constraint handling can be coded in an industrial standard PLC unit and with sample times of milliseconds. Secondly it has demonstrated that such an algorithm can make use of simplistic modelling information in conjunction with basic auto-tuning rules and still outperform an advanced auto-tuned PID whose design relied on far more information. Moreover the MPC includes constraint handling. Thus, thirdly the paper has demonstrated that MPC is a realistic industrial alternative to PID in loops primarily controlled with PLC units. This final contribution opens up the potential for much improved control of loops where PID may be a poor choice.

These results demonstrate the potential for implementing auto-tuned MPC within a PLC. Some issues the authors intend to pursue are: (i) demonstrate the algorithm in more challenging test rigs such as those with non-minimum phase behaviour and/or significant dead-times; (ii) consider extensions for unstable systems; and (iii) implement more advanced dual-mode type MPC algorithms and more advanced constraint handling facilities into the PLC.

REFERENCES

Camacho, E. and C. Bordons (2004). *Model predictive control*. 2nd ed.. Springer Verlag.

Clarke, D.W. (2002). Designing phase-locked loops for instrumentation applications. *Measurement* **32**, 205–227.

Clarke, D.W. (2006). PI auto-tuning during a single transient. IEE *Proceedings Control Theory and Applications* **153**(6), 671–683.

Froisy, J.B. (2006). Model predictive control — building a bridge between theory and practice. *Computer & Chemical Engineering*. **30**, 1426–1435.

Gyöngy, I.J. and D.W. Clarke (2006). On the automatic tuning and adaptation of PID controllers. *Control Engineering Practice* **14**, 149–163.

Maciejowski, J.M. (2002). *Predictive control with constraints*. Prentice Hall.

Morari, M., C. Jones and M. Zeilinger (2008). Low complexity MPC. In: *International Workshop on assessment and future directions in NMPC*. Pavia, Italy.

Richalet, J. (1993). *Practique de la commande predictive*. Hermes, France.

Richalet, J. (2007). Industrial application of predictive functional control. In: *Nonlinear Model Predictive Control, Software and Applications*. Loughborough, U.K.

Rojas, O.J. and G.C. Goodwin (2002). A simple antiwindup strategy for state constrained linear control. In: *IFAC World Congress*. Barcelona, Spain.

Rossiter, J.A. (2003). *Model predictive control: a practical approach*. CRC Press.

Rossiter, J.A. and P. Grieder (2005). Using interpolation to improve efficiency of multiparametric predictive control. *Automatica* **41**, 637–643.

Rossiter, J.A., B. Kouvaritakis and M.J. Price (1998). A numerically robust state-space approach to stable-predictive control strategies. *Automatica* **34**(1), 65–73.

Valencia-Palomo, G. and J.A. Rossiter (2008). The potencial of auto-tuned MPC based on minimal plant information. In: *Assessment and Future Directions of Nonlinear Model Predictive Control*. Pavia, Italy.

Valencia-Palomo, G., K.R. Hilton and J.A. Rossiter (2008). Predictive control implementation in a PLC using the IEC 1131.3 programming standard. *In review for the European Control Conference 2009.*

Zhu, Y., Z. Xu, J. Zhao, K. Han, J. Qian and W. Li (2008). Development and application of an integrated MPC technology. In: *Proceedings of the 17th IFAC Wolrd Congress*. Seoul, Korea.