Alexandria Engineering Journal (2016) xxx, xxx–xxx

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

Performance comparison of next generation controller and MPC in real time for a SISO process with low cost DAQ unit

V. Bagyaveereswaran [a,*], Tushar D. Mathur [a], Sukrit Gupta [a], P. Arulmozhivarman [b]

[a] School of Electrical Engineering, VIT University, Vellore 632014, India
[b] School of Electronics Engineering, VIT University, Vellore 632014, India

Received 3 September 2015; revised 25 July 2016; accepted 28 July 2016

KEYWORDS

RTDA;
MPC;
Hardware in Loop;
Arduino;
Data Acquisition Unit;
jMPC tool

Abstract In this paper, a brief overview of real time implementation of next generation Robust, Tracking, Disturbance rejecting, Aggressive (RTDA) controller and Model Predictive Control (MPC) is provided. The control algorithm is implemented through MATLAB. The plant model used in controller design is obtained using system identification tool and integral response method. The controller model is developed in Simulink using jMPC tool, which will be executed in real time. The outputs obtained are tested for various constraint values to obtain the desirable results. The implementation of Hardware in Loop is done by interfacing it with MATLAB using Arduino as data acquisition unit. The performance of RTDA is compared with those of MPC and Proportional Integral controller.

1. Introduction

Model Predictive Control (MPC) is a control algorithm that uses a model of the process to predict the future response of a plant. Model Predictive Control is a method that emphasizes on making controllers that can adjust the control action before a change in the output set point actually occurs [1]. MPC algorithm requires the knowledge of input/output constraints and weights along with Prediction Horizon and Control Horizons to be provided by the user for the control action. To work effectively and provide a good response, the model estimated should be an accurate fit for the plant. MPC can be digitally implemented through jMPC tool which is provided as a freeware add-on for MATLAB and is an initiative of Industrial Information and Control Centre (I^2C^2), a joint venture between AUT University and The University of Auckland, New Zealand [3].

It is a well-known fact that PID controller is the most widely used controller in industrial applications [4]. Tunable parameters in PID controllers are not directly related to set point tracking, disturbance rejection and robustness. So designing a PID controller to achieve desired performance is a tedious task. It is difficult to tune a PID controller which satisfies set point tracking and disturbance rejection simultaneously [7]. According to a study conducted by Ender on control loops in industries, 30% of the control loops operate in manual mode while more than 60% are not tuned properly

* Corresponding author.
E-mail address: vbagyaveereswaran@vit.ac.in (V. Bagyaveereswaran).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

http://dx.doi.org/10.1016/j.aej.2016.07.028

[5]. The classical linear proportional-derivative (PD) controller and the proportional-integral-derivative (PID) controller cannot provide good enough performance in controlling highly complex, nonlinear, and uncertain processes [11]. The performance of PID controller is poor for dead time dominant and nonlinear processes. In these scenarios we have to use additional components such as Smith predictor for which implementation is complex [6]. Fuzzy controller can be an alternative with better performance, but implementation is more complex. So we need to design a controller that is as simple as a PID controller with better performance [4].

RTDA controller possesses the characteristics of MPC and PID controller. Its structure is as simple as that of a PID controller. The tuning parameters θ_R, θ_T, θ_D and θ_A are directly related to performance parameters-set point tracking, disturbance rejection, robustness and an additional parameter aggressiveness respectively. The values of these parameters are fixed to lie between 0 and 1 [4]. A block diagram representation for a RTDA controller and alternate tuning rule for a Ogunnaike's tuning rule was proposed by Kariwala. He demonstrated the efficiency of the modified tuning rules using numerical examples of FOPTD processes [14]. The RTDA control law for second order process with dead time (SOPTD) was developed by Anbarasan and it is tested on process with minimum or non-minimum industrial nonlinear CSTR process [12]. The RTDA controller tuning parameters can be tuned using fuzzy scheduling for a standard nonlinear process. The performance of the fuzzy scheduled RTDA compared with the other established controllers such as MPC, PID in MATLAB simulation. The proposed idea possesses adaptive capability [15].

The main purpose of this paper was to design RTDA controller, MPC and PID controller for a linear SISO pressure process and implement it in real time using low cost DAQ unit. Finally the performance of the next generation controller is compared with other controllers.

Section 2 describes the process modeling using system identification tool and integral response method. Design of RTDA controller and MPC is discussed in Section 3. In Section 4 hardware setup of the process is described and design of data acquisition unit is presented. The real time implementation of controllers on a linear single input single output pressure process is presented in Section 5. The performance of the controller is analyzed and compared in Section 6 and followed by conclusion in final section.

2. System identification

The schematic diagram of the pressure process to be controlled is shown in Fig. 1.

where

PT – Pressure Transmitter
PC – Pressure Controller
E/P – Electro Pneumatic Converter
SP – Set point

2.1. Process model in FOPDT form

The first step is finding the First order process with dead time (FOPDT) model of the pressure process. Integral equation method was used which requires the open loop step response of the system [9]. Step input was applied to the system and it was allowed to settle. The input and pressure variable values were recorded. Using these values FOPDT model was developed and is given by

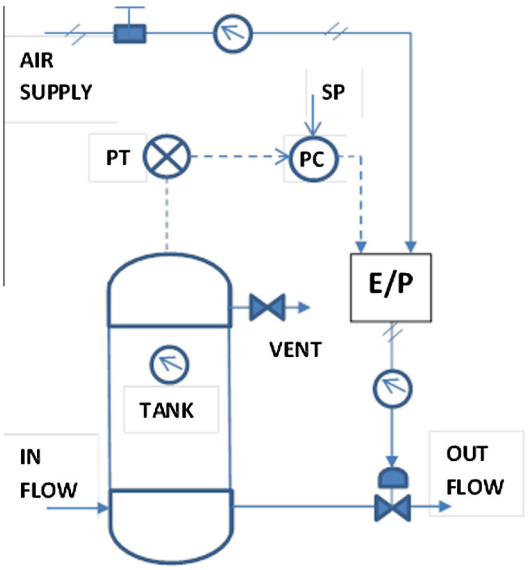

Figure 1 Pressure control system.

$$y(s) = e^{-\alpha s}\frac{K}{\tau s+1}u(s) = e^{-0.5s}\frac{1.01}{18.42s+1}u(s) \quad (1)$$

where K, α and τ are Process gain, dead time and time constant respectively. The above model is used in RTDA controller design. The accuracy of the FOPTD model is shown in Fig. 2.

2.2. Process model in state space form

The process to be controlled is presented in state space model form in MPC design. System Identification (SI) Toolbox is provided by MATLAB as add-on to estimate the models in state-space form, transfer functions, polynomial models. The data are imported to the tool in either time domain or frequency domain. The black box modeling is used to identify the system model. An open loop test was carried out to gather data from the system, which was then used in SI Tool in MATLAB [2]. Then the model is estimated in state space form of order 4 using N4SID (subspace) estimation. To check for the best fit of the model, we use the *Model Output* function which compares the estimated state space model with the open loop test model. The usual best fit for the model should be above 90 as shown in Fig. 3. The model is then imported to the workspace using the *To Workspace* function in the toolbox where it is converted to discrete time model with sampling period of 0.1 s.

Identified model is defined by the following matrices in state space Eqs. (2) and (3):

$$x_{k+1} = Ax_k + Bu_k; \quad (2)$$

$$y_k = Cx_k + Du_k \quad (3)$$

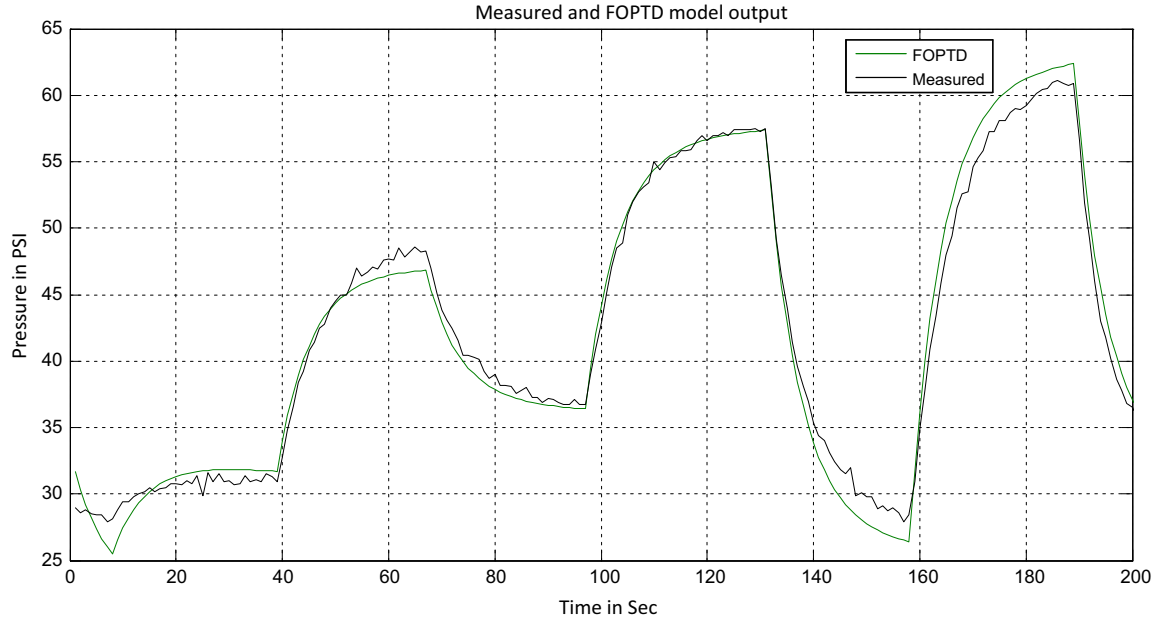

Figure 2 Open Loop Response of Process and FOPTD model.

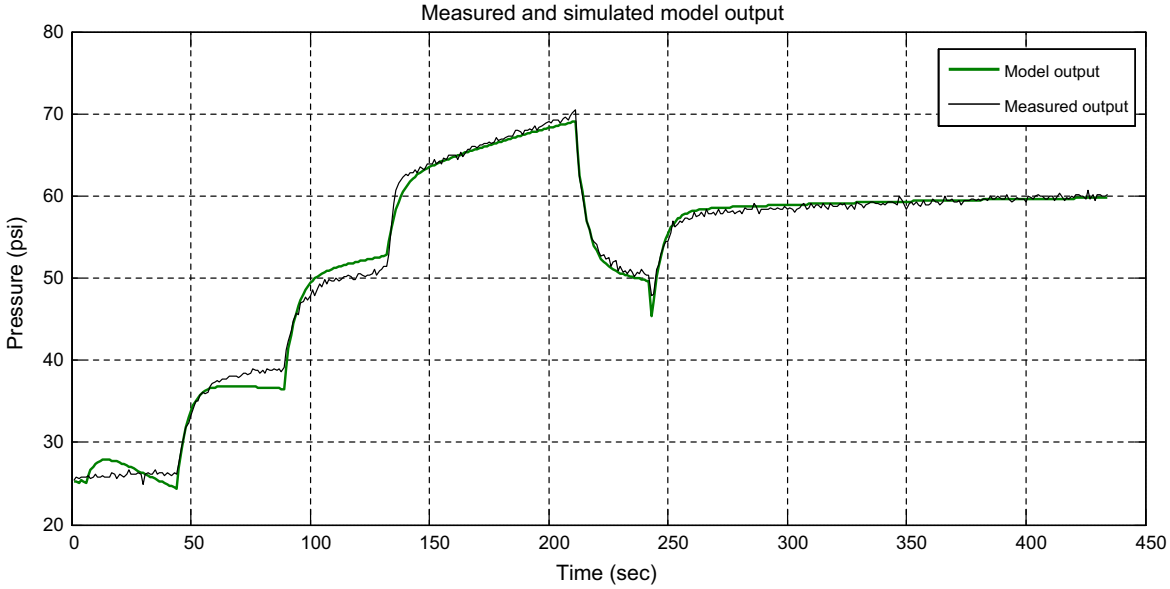

Figure 3 Open Loop Response of Process and SS model.

$$A = \begin{bmatrix} -0.04048 & 0.05959 & 0.02174 & 0.01957 \\ 0.08651 & -0.1663 & -0.1547 & -0.1428 \\ -0.08021 & 0.1214 & -0.06931 & -2.35 \\ 0.04702 & -0.1408 & 1.994 & -0.2926 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0006206 \\ -0.001383 \\ 0.003693 \\ -0.00213 \end{bmatrix}$$

$$C = [\,147.3 \quad -1.206 \quad -1.43 \quad 0.1961\,] \quad D = [0]$$

3. Controller design

3.1. RTDA controller design for FOPTD model

The tuning parameters of RTDA controller are directly linked to desired performance parameters. The following are the steps in controller design [8];

1. Process model: develop a First Order Plus Delay Time model for the process to be controlled. The continuous time model is given by

$$y(s) = \frac{Ke^{-\alpha s}}{\tau s + 1} u(s) \tag{4}$$

where K, α and τ represent gain, dead time and time constant respectively.

2. Discrete model: convert the developed continuous model to discrete form and is given by

$$\hat{y}(k+1) = a\hat{y}(k) + bu(k-m) \tag{5}$$

where m denotes dead time in discrete domain.

3. Uncorrected model prediction: a future prediction for the process output m time steps can be obtained from the following equation

$$\hat{y}(k+m+1) = a^{m+1}\hat{y}(k) + b\mu(k,m) + b\eta_i u(k) \tag{6}$$

where $\mu(k,m) = \sum_{i=1}^{m} a^i u(k-i)$ and $\eta_i = \frac{1-a^i}{1-a}$ for $1 < i < N$

4. Model prediction update: there will be a mismatch between a FOPDT model and true process. The modeling error is given by

$$e(k) = y(k) - \hat{y}(k) \tag{7}$$

Error $e(k)$ can be decomposed into two parts: first one due to model process mismatch and the other due to unmeasured disturbances.

$$e(k) = e_m(k) + e_D(k) \tag{8}$$

$$e_D(k) = \theta_R e_D(k-1) + (1-\theta_R)e(k) \tag{9}$$

5. Future disturbance prediction and updated model prediction: future disturbance effect is given by the equation

$$\widehat{e_D}(k+j) = e_D(k) + \frac{(1-\theta_D)}{\theta_D}[1-(1-\theta_D)^j]\Delta e_D(k) \tag{10}$$

where

$$\Delta e_D(k) = e_D(k) - e_D(k-1) \tag{11}$$

with this prediction updated model prediction is given by

$$\tilde{y}(k+m+i) = \hat{y}(k+m+i) + \widehat{e_D}(k+m+i) \tag{12}$$

6. Control action formulation: at each point of time the control input is calculated such that the difference between predicted process output and desired trajectory is minimum. The desired trajectory is given by

$$y^*(k+j) = \theta_T^j y^*(k) + \left(1-\theta_T^j\right) y_d(k) \tag{13}$$

We will get the control input by solving the least square optimization problem.

$$u(k) = \frac{\sum_{i=1}^{N} \eta_i \psi_i(k)}{b\sum_{i=1}^{N} \psi_i^2} \tag{14}$$

where

$$\psi_i(k) = y^*(k+i) - a^{m+i}\hat{y}(k) - \widehat{e_D}(k+m+i) \tag{15}$$

7. Tuning Parameters: generally we give $\theta_R = 0.5$. Other parameters depend on process.

$$\theta_D = 1 - \theta_R \tag{16}$$

$$\theta_T = 10^{\frac{-\Delta t}{\tau}} \tag{17}$$

$$\theta_A = 1 - e^{-\frac{(N-1)\Delta t}{\tau}} \tag{18}$$

The closed loop block diagram of RTDA controller is shown in Fig. 4 [14].

3.2. MPC controller design

The Model Predictive Control is a type of control algorithm that uses the process model and takes into consideration the constraints on inputs/outputs unlike the conventional controllers such as P, PI (Proportional, Proportional + Integral). This type of control consists of the following:

- Model of the process
- Constraints
- Cost Function

The optimization problem of MPC involves the minimization of the Cost Function which is outside the scope of this paper to discuss. The block diagram of MPC is shown in Fig. 5.

The major requirement of MPC is that a model for the process, a model which describes the input to output behavior of the process, is needed. Mechanistic models derived from conservation laws can be used. Usually, in practice simply data-driven linear models are used. In MPC it is assumed that the model is a discrete state-space model of the form as shown in Eqs. (2) and (3).

All physical systems have constraints. There are physical constraints such as actuator limits and safety constraints such as temperature and pressure limits. Finally, there are performance constraints such as overshoot. Eqs. (3) and (4) define the constraints on output and input respectively:

$$y_{min} \leqslant y \leqslant y_{max} \tag{19}$$

$$U_{min} \leqslant u \leqslant u_{max} \tag{20}$$

The main idea with MPC is that the MPC controller calculates a sequence of future control actions such that a cost function Eq. (5) is minimized [1].

$$J = \sum_{k=0}^{Np} (\hat{y} - r)^T Q(\hat{y} - r) + \sum_{k=0}^{Np} \Delta u^T R \Delta u \tag{21}$$

where

Np – Prediction Horizon
r – Set point
$\hat{y}$ – Predicted Process Output
Δu – Predicted Change in control value, $\Delta u_k = u_k - u_{k-1}$
Q – Output error weight Matrix
R – Control Weight Matrix

4. Data acquisition unit and hardware setup

4.1. Process hardware setup

Hardware setup consists of pressure process station, Arduino Uno and PC. The pressure system in Fig. 6 is available in Process control laboratory in VIT University. The pressure station specifications are given in Table 1.

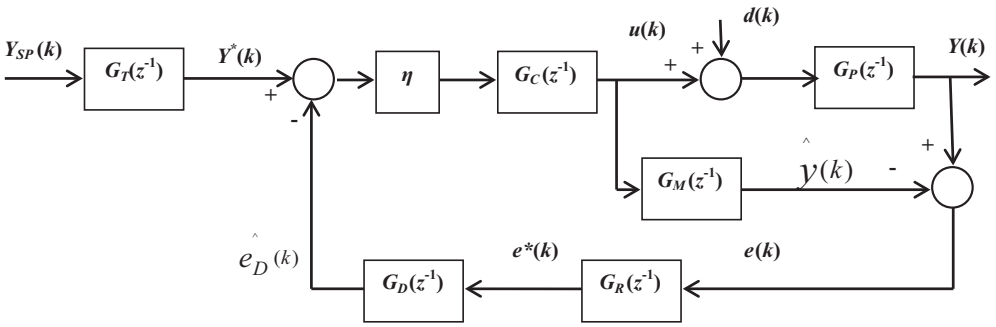

Figure 4 Block diagram of RTDA controller.

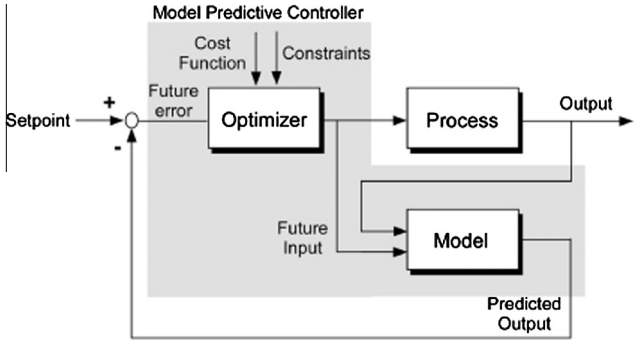

Figure 5 Block diagram of MPC scheme.

The main component is a metallic tank with inlet and outlet ports. We set inlet hand valve at a fixed position. There is a pneumatic actuated control valve at outlet side. Here the tank pressure is the controlled variable and outlet flow rate is the manipulated variable. The pressure inside the tank is measured using a pressure transmitter. It converts the pressure in 0–75 psi range to 4–20 mA current signals. It consists of a primary sensing element that converts pressure to displacement and a secondary element that uses this displacement change for changing resistance. A 12–24 V fixed DC power supply is needed for transmitter to work properly. The 4–20 mA current signals generated by transmitter act as an input for the controller. The controller output would be 4–20 mA current signal. The current to pressure (E/P (or) I/P) converter gives pneumatic signal in 3–15 psi range corresponding to controller generated current signal. The pneumatic signal from I/P converter operates the control valve depending on the control signal. Hence the pressure is controlled inside the tank.

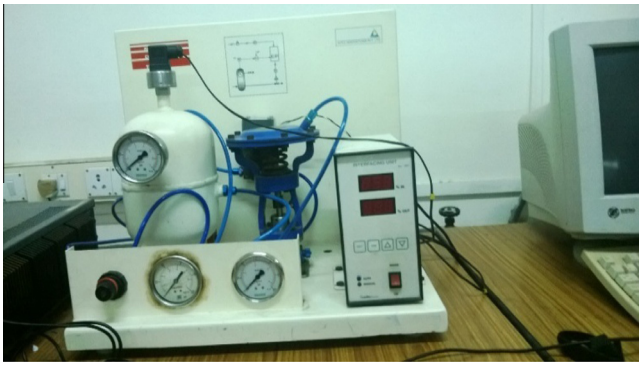

Figure 6 Pressure process station.

Table 1 Pressure process station specifications.

Part name	Function details
Process tank (range)	MS, 0–100 psi
Pressure gauge	0–35 psi
I/P converter	4–20 mA input 3–15 psi output
Air regulator	Size 1/4″ BSP range 0–2 kg/cm^2
Control valve	Size 1/4 pneumatic actuated, input 3–15 psig
Pressure transmitter	Type two wire, input 0–75 psi, output 4–20 mA DC, operating voltage 24 V DC

4.2. Building a DAQ unit

The platform used for data acquisition unit is an Arduino Mega 2560 board. It is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack and a reset button. It can be connected to a computer with a USB cable or can be powered with an AC-to-DC adapter or a 5 V battery. Simulink can be used as a platform to code Arduino for acquiring and transmitting data. Support package for Arduino is available as freeware on MATLAB Web site, hence making Arduino as a low cost DAQ. The Simulink block diagram for data acquisition and calibration is shown in Fig. 7.

The pressure transmitter works on 4–20 mA current loop. To acquire the signal, it should be converted to a voltage range to 1–5 V using a simple current to voltage converter circuit, consisting of a 250 Ω resistance. Arduino then reads the signal in the form of 10 bit samples; the voltage in range of 1–5 V is mapped to numerical values 0–1023. These values should be calibrated to the pressure range. A low pass filter is added to smooth the response.

The transmitter takes current in the range of 4–20 mA. Arduino boards can transmit the output only in the digital form or Pulse Width Modulation (PWM) form. Hence we use PWM. The duty cycle of PWM will depend on the output which ranges from 0 to 255 (8 bit samples) for Arduino PWM output pin. In order to get better sensitivity a multiplication factor with a bias can be added before it. The voltage from the pin will be average of the duty cycle of PWM. This voltage is then converted to 4–20 mA, by using an ideal voltage to

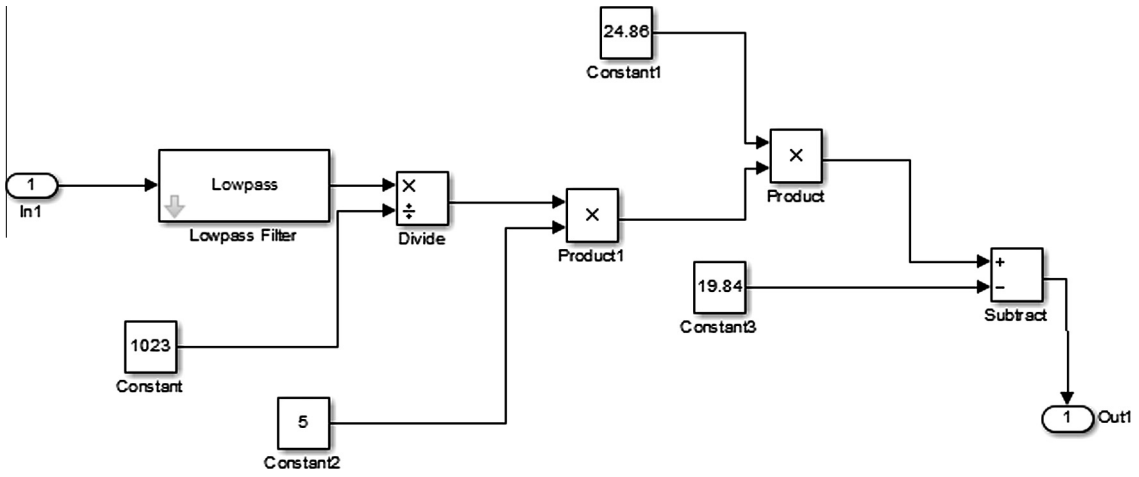

Figure 7 Data acquisition and calibration.

current converter and then transmitted back to the E/P for control action.

5. Real time implementation

5.1. RTDA controller implementation

By using MATLAB-Arduino Support package Arduino can communicate with MATLAB program [13]. MATLAB Program can access both input and output ports of Arduino Uno using interface commands. The whole setup for the experimental validation is shown in Fig. 8. The RTDA controller was developed for the pressure process using the model given in Eq. (1). The tuning parameters are given by $\theta_R = 0.5$, $\theta_D = 0.5$, $\theta_T = 0.955$, $\theta_A = 0.00006$.

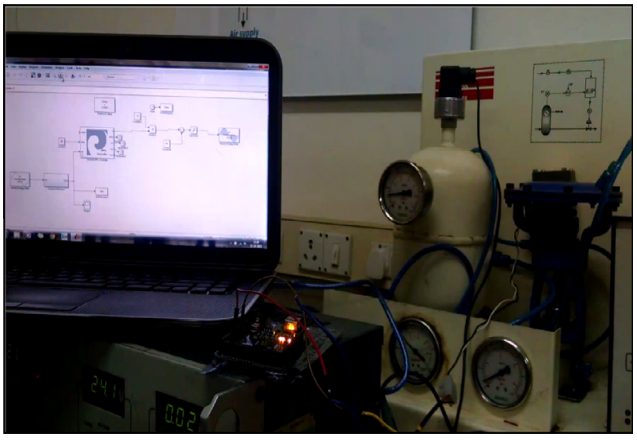

Figure 8 Complete setup of the pressure control station.

5.2. MPC implementation

The jMPC toolbox is used for the real-time implementation of MPC controller, which utilizes the discrete model of the plant. The main problem with MPC toolbox provided in MATLAB is that, it doesn't support real time implementations due to lack of proper solvers required to solve the Quadratic Problem of the MPC controller. The advantage of this tool is that, various solvers can be used to solve the MPC problem (control algorithm) and also, it supports real time implementation. The jMPC Simulink block also requires an object to be loaded into the discrete MPC controller which defines all the parameters and constraints. The object can be simply created from the command window of MATLAB. It is shown in Appendix A. The MPC controller is created as a jMPC object while the plant can be a jSS object for linear simulations.

The model given in Eqs. (1) and (2) is used in object creation. The prediction horizon (N_p) and control horizon (N_c) are determined by a hit and trial method. The input constraints are defined in accordance with the technical specifications of the pressure station.

The block diagram of jMPC for real-time control of the system is shown in Fig. 9. The model is designed in Simulink. A real time pacer is used to match the time of running of the system with real time. This allows us to visualize the performance of the controller with actual time and not the sample instances, which can be beneficial with the existing real time performance results of different controllers.

The MPC algorithm used within the jMPC Toolbox is a linear implementation, where a linear state space model is used for the system prediction. This enables the controller to formulate the control problem as a Quadratic Program (QP), which when set up correctly results in a convex QP, ensuring an optimal input (global minimum).

The Simulink model shown in Fig. 9 was connected to the real time system using Arduino as DAQ is shown in Fig. 7. A scaling factor has been used in the model to increase the sensitivity of the controller output.

The following factors must be taken into account while choosing the prediction horizon and control horizon for MPC [1].

Prediction horizon (N_P):

- A short prediction horizon reduces the length of time during which the MPC controller predicts the plant outputs. When the prediction horizon is short the MPC controller works more like a traditional feedback controller.

Please cite this article in press as: V. Bagyaveereswaran et al., Performance comparison of next generation controller and MPC in real time for a SISO process with low cost DAQ unit, Alexandria Eng. J. (2016), http://dx.doi.org/10.1016/j.aej.2016.07.028

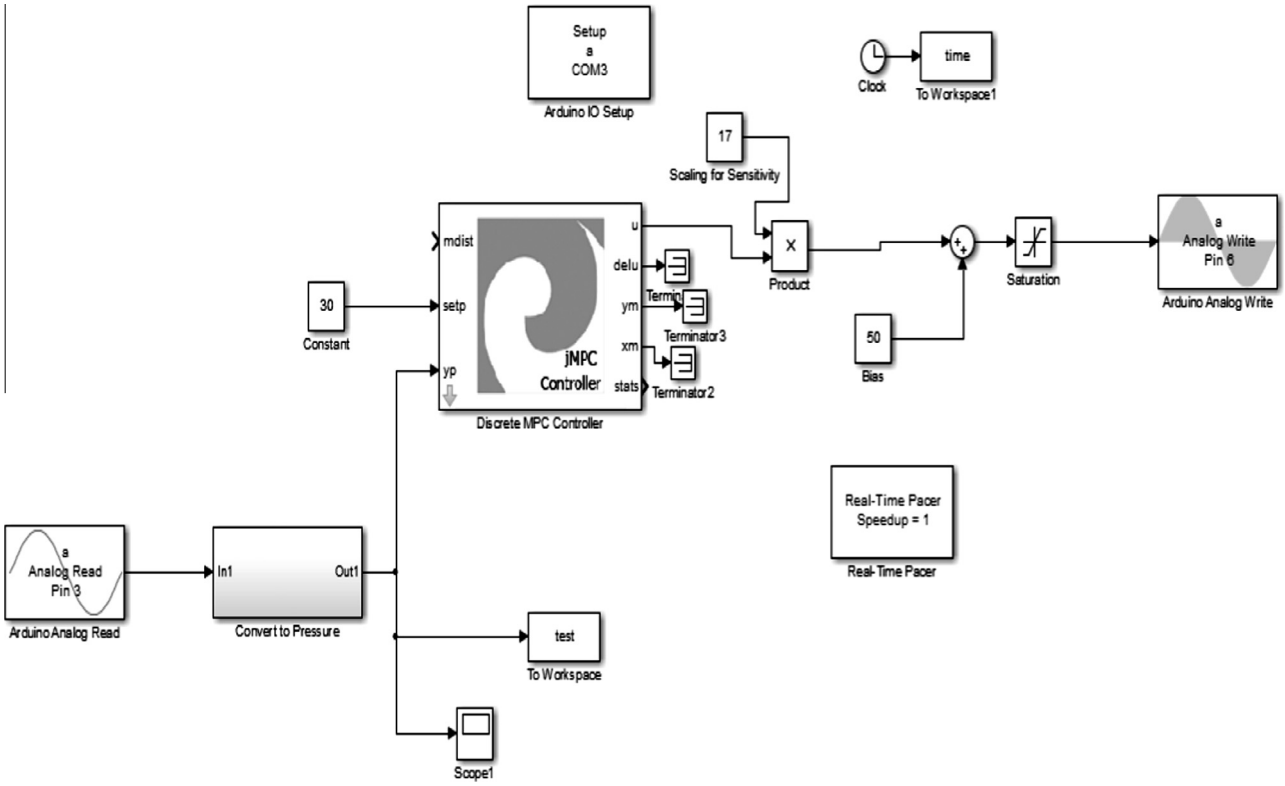

Figure 9 Simulink model for real time implementation of MPC.

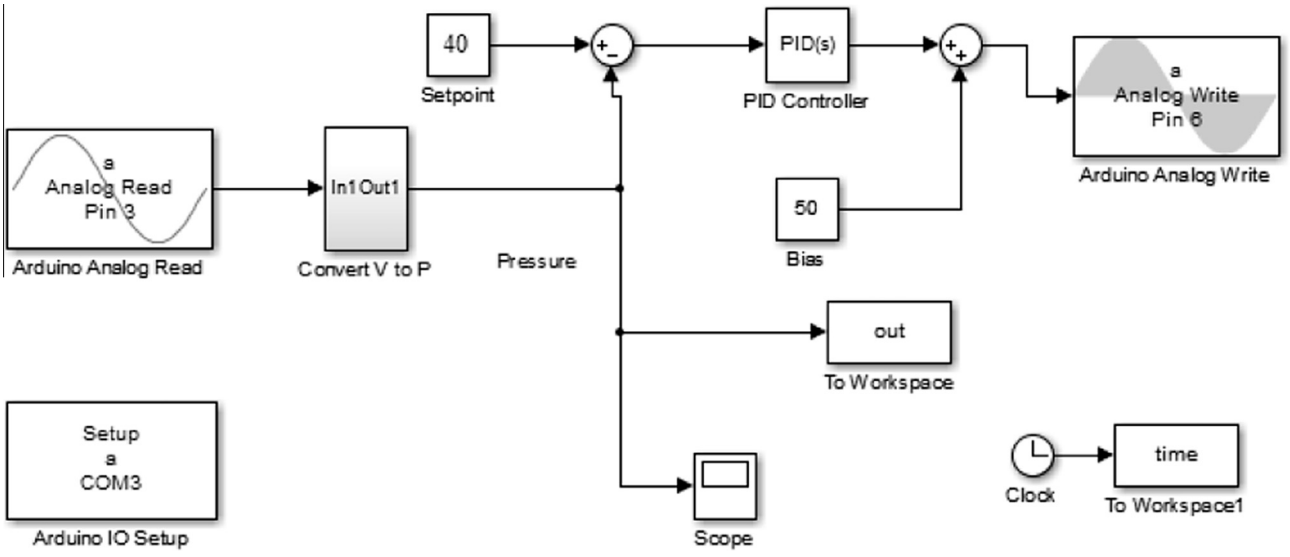

Figure 10 Simulink model for real time implementation of PI controller.

- A long prediction horizon increases the predictive ability of the MPC controller, but the controller performances poorer due to extra calculations.

Control horizon (N_C):

- A short control horizon means more careful changes in the control action.
- A long control horizon means more aggressive changes in the control action.

5.3. PID controller implementation

The tuning parameters for the PI controller i.e. K_P (Proportional gain) and K_I (Integral gain) were calculated using Ziegler-Nichols tuning method. The Voltage to Pressure (V/P) block is used to convert the input Arduino voltage to pressure. This is done by use of a calibration equation as shown in Fig. 7. The same block has been used for jMPC

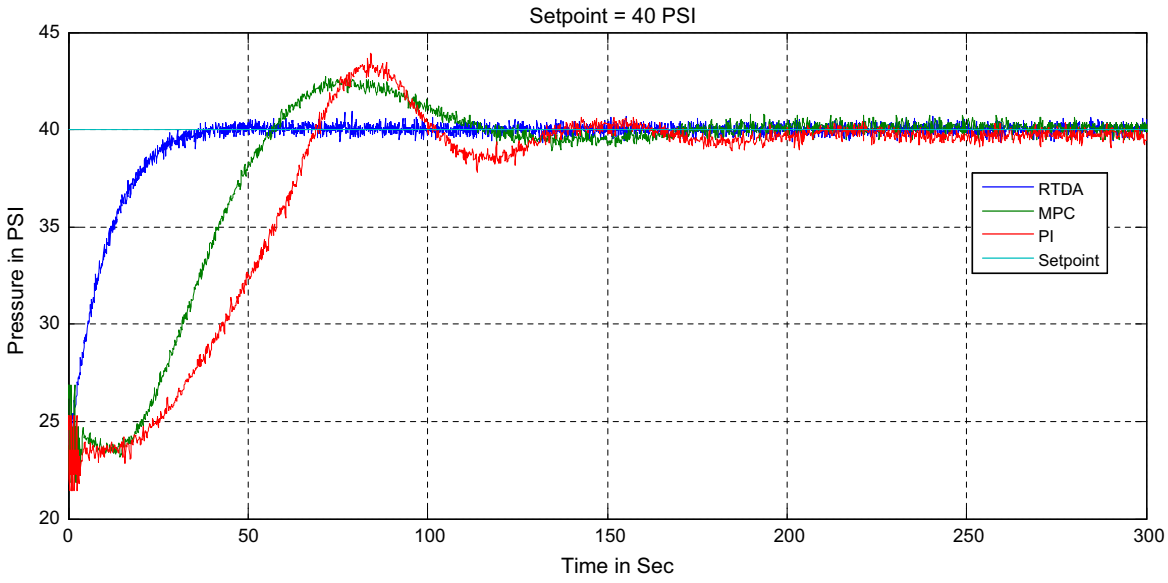

Figure 11 Servo performance with RTDA, MPC and PI controller.

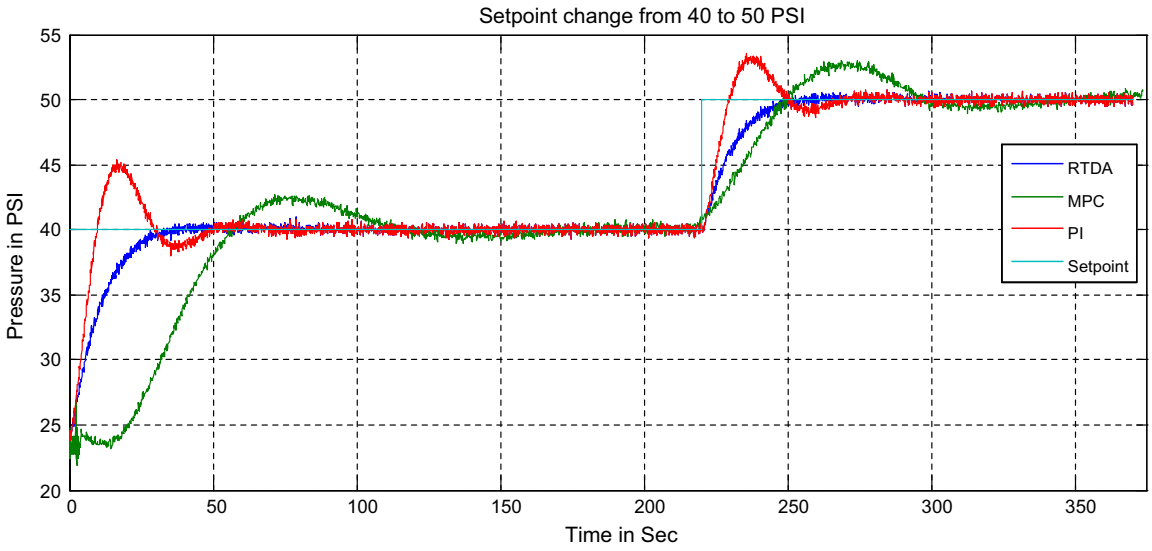

Figure 12 Set point tracking characteristics with RTDA, MPC and PI controller.

model implementation also. The Simulink model for implementation of PI controller on the real time system is shown in Fig. 10.

6. Results and discussions

6.1. Servo performance with controllers

The initial pressure inside the tank is set as 25 psi. A set point of 40 psi was given to the control algorithms. The set point tracking (or) servo response of pressure process with three controllers is shown in Figs. 11 and 12. The pressure settled at 40 psi without any overshoot for RTDA than other two controllers.

The above figure shows the pressure output with MPC for a set point of 40 psi and with the given parameters and constraints. The object for jMPC was created with the same parameters as shown in Fig. 7. The Servo Performance with MPC is obtained when $N_p = 26$; $N_c = 2$; Input weight = 100; Output weight = 2; Input constraints – [0 100]; Output constraints – [−inf, inf]. The input and output weights defined are a measure of importance MPC gives to the plant inputs and outputs. When input weight increased it smoothen the response of the system. Output weights carry more importance in multiple output systems where one of the outputs has a higher priority. For example, when controlling direction and speed of a helicopter, either speed or direction can be given more weight so as to give more preference.

The most commonly used PID controller was considered for pressure process [10]. A SIMULINK model was developed for controlling pressure process as shown in Fig. 9. Using the

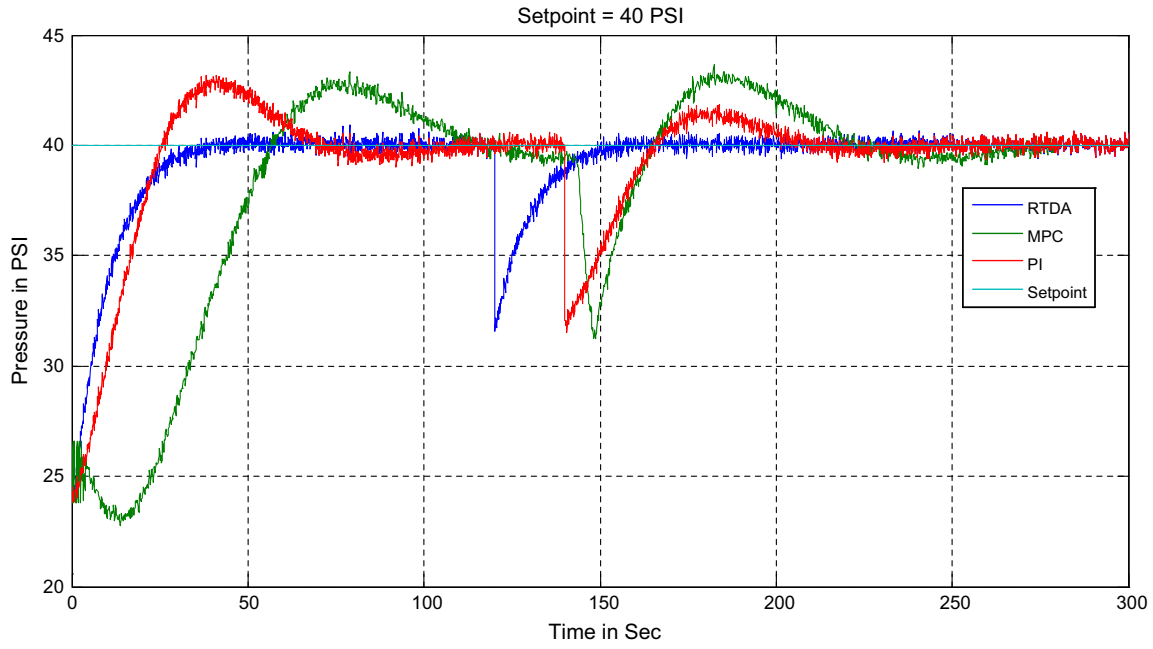

Figure 13 Disturbance rejection ability with RTDA, MPC and PI controller.

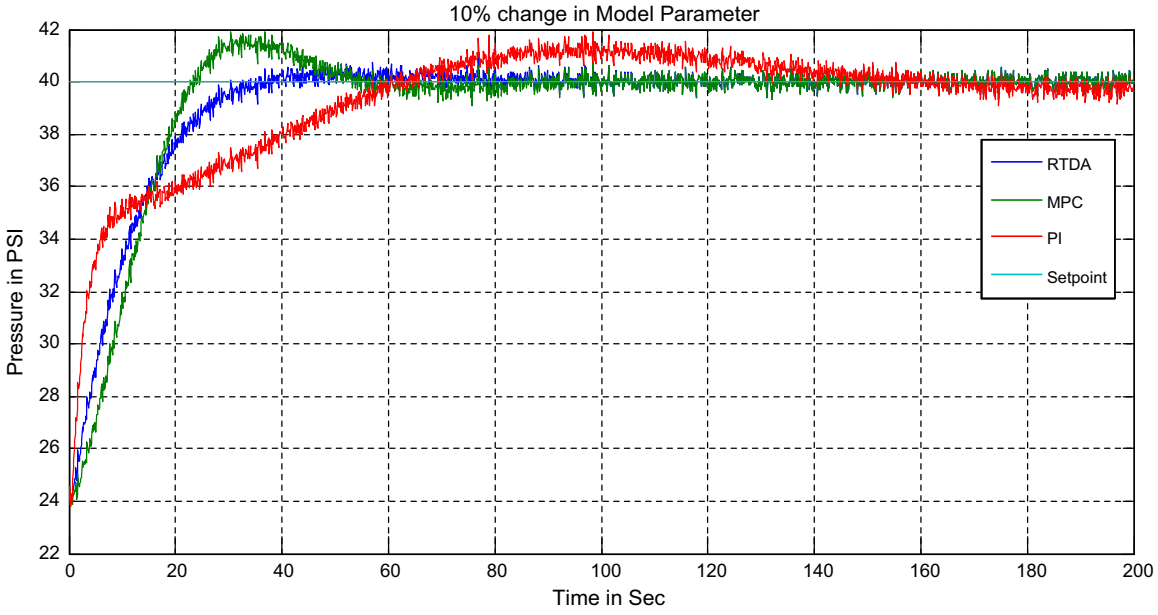

Figure 14 Robustness with RTDA, MPC and PI controller.

identified model given in Eq. (1) controller parameters were tuned. The tuning values are given by $K_P = 0.473191$, $K_I = 0.02216$, $K_D = 0$.

6.2. Regulatory performance with controllers

Similar to the first step pressure was set at 25 psi initially and 40 psi set point was given. The pressure inside the tank settled at 40 psi. Then a disturbance of 10% of current pressure value was introduced at around 120 s for RTDA, at 150 s for MPC and at 140 s for PI controller on the vent side of the process tank which caused pressure to deviate from the steady value. But the RTDA controller eliminated its aftereffects and pressure regained and settled at set point value in short time than MPC and PI controlled response. The regulatory response of process with three controllers is shown in Fig. 13.

6.3. Robustness analysis of controllers

An ideal controller should nullify any effects of model-process mismatch which shows its robustness property. In order to check the robustness property of controllers, instead of taking

Table 2 Performance comparison of RTDA with other controllers.

Specifications	RTDA	MPC	PI
Settling time (s)	50	150	210
Overshoot (%)	0	6.9	9.1
Rise time (s)	30	38	60
ISE	0.0878	0.098	0.213

18 as time constant, its value was modified to be 20 (10% process model mismatch) in a model given in Eq. (1). In state space model shown in Eqs. (2) and (3), the 10% parameter change is introduced and observed that the MPC is able to provide stable response irrespective of model mismatch. The process response with RTDA controller didn't change drastically but there is a huge variation in response to PI. The PI response takes longer time to settle. The robust characteristic of the controllers is shown in Fig. 14.

In Table 2, the performance of the implemented controllers is compared in terms of time domain specifications. It is comprehend that new next generation control strategy exhibits a better performance like other most popular controllers.

7. Conclusions

A new method of RTDA control strategy was used for controlling the pressure process with simple interface unit. Its performance was tested for different characteristics such as servo performance, disturbance rejection ability and robustness in real time. The jMPC tool along with MATLAB provides a simple interface to implement model predictive controller in real time. The performance of the MPC controller was desirable. The data acquisition system and signal conditioning circuit were implemented on a low cost basis on the real time system and proved to work efficiently. Finally the RTDA controller also gives the better performance like MPC scheme. But one advantage that MPC has over RTDA and PID is that process variables can be constrained in MPC but not in other two controllers.

The major contribution in this work is the successful implementation of next generation controller and MPC in real time using a low cost DAQ unit for controlling the pressure process and the success of the attempt to show the better performance in terms of robustness, servo and regulatory characteristics of RTDA controller.

Acknowledgments

Support to this project was from the School of Electrical Engineering, VIT University, India.

Appendix A

The jMPC object for MPC controller design while the process in jSS (state space) object for linear simulations can be created as shown in Fig. A1.

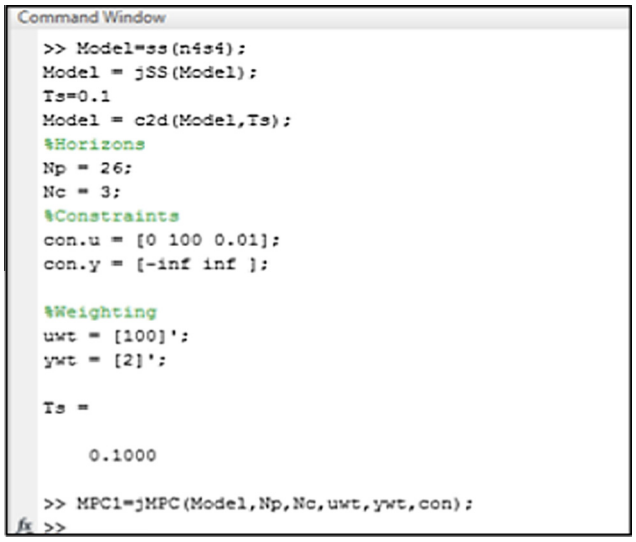

Figure A1 Object creation for jMPC tool.

References

[1] Hans-Petter Halvorsen, MPC Tutorial for LabVIEW, Telemark University College Tutorials, 2011.
[2] System Identification Toolbox™ User's Guide by The MathWorks Inc., 1988–2015.
[3] J. Currie, D.I. Wilson, A Model Predictive Control Toolbox Intended for Rapid Prototyping, 2009.
[4] Kapil Mukati, Babatunde Ogunnaike, Stability analysis and tuning strategies for a novel next generation regulatory controller, in: Proceeding of the American Control ConferenceBoston, Massachusetts, June 30–July 2 2004, 2004.
[5] D.B. Ender, Control Engineering, September 1993, pp. 180–190.
[6] K.J. Astrom, T. Hagglund, Automatic tuning of simple regulators with specifications on phase and amplitude margins, Automuticu 20 (5) (1984).
[7] K.J. Astrom, T. Hagglund, The future of PID control, Control Eng. Pract. (2001) 1163–1175.
[8] B.A. Ogunnaike, K. Mukati, Development, design and implementation of an alternative structure for next generation regulatory controllers, in: Preprints of the Annual AIChE Meeting, San Francisco, CA, 2003.
[9] V. Bayaveereswaran, K.J. Nidhil Wilfred, S. Sreeraj, B. Vijay, System identification from step input using integral equation, Int. J. Appl. Eng. Res. 8 (17) (2013) 2165–2169.
[10] D. Chen, D.E. Seborg, PI/PID controller design based on direct synthesis and disturbance rejection, Ind. Eng. Chem. Res. 41 (2002) 4807–4822.
[11] Ahmad M. El-Nagar, Mohammad El-Bardini, Nabila M. El-Rabaie, Intelligent control for nonlinear inverted pendulum based on interval type-2 fuzzy PD controller, Alexandria Eng. J. 53 (2014) 23–32.
[12] K. Anbarasan, K. Srinivasan, Design of RTDA controller for industrial process using SOPDT model with minimum or non-minimum zero, ISA Trans. (2015).
[13] Kapil Mukati, An alternative structure for next generation regulatory controllers and Scale-up of Cu(InGa)Se2 thin film Co-evaporative physical vapor deposition Process Thesis, 2007.
[14] Antonius Yudi Sendjaja, Zhen Fu Ng, Si Si How, Vinay Kariwala, Analysis and tuning of RTD-A controllers, Ind. Eng. Chem. Res. (2011) 3415–3425.
[15] K. Anbarasan, K. Srinivasan, Fuzzy scheduled RTDA controller design, ISA Trans. 52 (2013) 252–267.