Applied Mathematical Modelling 40 (2016) 100–114

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

A comparison of software-based approaches to identifying FOPDT and SOPDT model parameters from process step response data

Chris Cox *, John Tindle, Kevin Burn

Department of Computing, Engineering and Technology, Faculty of Applied Sciences, University of Sunderland, St. Peter's Campus, Sunderland SR6 0DD, UK

ARTICLE INFO

Article history:
Received 21 November 2011
Received in revised form 2 April 2015
Accepted 8 May 2015
Available online 14 June 2015

Keywords:
Identification
Step response
FOPDT model
SOPDT model
Least-squares algorithm
PSO

ABSTRACT

System identification is the experimental approach to deriving process models, which can take many forms depending upon their intended use. In the work described in this paper, the ultimate aim is to use them in the design of controllers for regulating engineering processes. Modelling always involves approximations since all real systems are to some extent non-linear, time-varying, and distributed. Thus, it is highly improbable that any set of models will contain the 'true' system structure. A more realistic aim is therefore to identify a model that provides an acceptable approximation, in the context of the application in which it is used. In controller design, a first step is often to determine the model using step and frequency response data. This paper compares different modern software approaches that exploit step response data, where the aim is to determine either a first- or second-order-plus-dead-time (FOPDT or SOPDT) transfer function. They include an integral equation method, an algorithm available in the MATLAB Optimization Toolbox, and recently developed in-house software that uses a particle swarm optimisation (PSO) approach.

1. Introduction

Process models can take many forms depending upon their intended application and many controller design algorithms require such models. When the controller is of the proportional plus integral (PI) or proportional plus integral plus derivative (PID) form the model is often assumed to be a continuous-time transfer function of the first-order-plus-dead-time [FOPDT] or second-order-plus-dead-time [SOPDT] structure. This is because the response of such models is a good approximation to the monotonic, non-overshooting continuous-time step response of many of the processes and sub-processes found in industrial process control. Consequently, a sensible approach to control system design begins with the determination of a FOPDT or SOPDT model.

Methods to estimate the parameters of a continuous-time transfer function using step response data are still popular, especially in the process industries [1]. The early approaches relied on graphical techniques, which are epitomised by the methods of Ziegler and Nichols [2], Oldenbourg and Satorious [3], Sten [4], and Rake [5]. For monotonic step responses, the collection of approaches that are based on specific area calculations are well represented in the paper of Nishikawa et al. [6] and the book by Astrom and Hagglund [7]. A limitation of all of all these methods is that the system must be in steady-state before the step is applied and data must be collected until the new steady-state is achieved. In addition, their accuracy can be compromised when data is noise-corrupted.

* Corresponding author.
E-mail address: chris.cox@sunderland.ac.uk (C. Cox).

http://dx.doi.org/10.1016/j.apm.2015.05.007
0307-904X/

More recently, a family of new methods have been published that are more computationally involved, but allow the parameters of the transfer function model, including the time delay, to be estimated simultaneously. The original ideas of Bi et al. [8] and Wang and Zhang [9] have been extended to allow identification under transient initial conditions [10]. Other developments have also been reported. In Liu et al. [11] and Ahmed [12], new methods to handle 'piecewise step tests' have been presented, whilst [13] describes an 'indirect method' where the captured discrete-time data is first used to produce a discrete-time model from which the required continuous-time model is determined. All of the developments covered in [8–12] come under the heading of integral equation (IE) approaches. Whilst IE techniques are not new, the earlier methods were derived in different ways to those highlighted above.

In this paper, the results of employing the IE method plus two software-based identification applications will be compared with those published in a recent paper by Das et al. [14]. To do this, four benchmark transfer functions suggested by Das are employed and modelled as FOPDT and SOPDT systems with both zero and non-zero initial conditions. Standard performance indices are used for the comparison, which provide a measure of the closeness of fit between the original transfer functions and the models. Finally, aspects relating to the design of PI controllers for a subset of the modelled transfer functions are considered, using stability boundary loci in the K_C–K_I plane.

The paper is structured as follows. Section 2 briefly describes the theory behind the IE method, which is developed to fit a FOPDT and a SOPDT model to the process output to a step input. These results are used in the comparison studies that follow. In Section 3, the parameter estimation problem is defined as a nonlinear least-squares optimisation problem the solution of which uses the MATLAB Optimization Toolbox. Being an optimal strategy the user has to select initial or starting values for the model parameters. When this is a problem, Section 4 suggests an alternative solution based on particle swarm optimisation (PSO). The simulation results that demonstrate the accuracy of the individual methods are presented in Section 5. Since the models are to be used for PI and PID controller design, a stabilization study is carried out in Section 6 to ascertain which model is the preferred option. Discussions and conclusions are in Section 7.

2. The integral equation estimation method

2.1. The FOPDT model

The transfer function of the FOPDT model to be identified is given by:

$$G(s) = \frac{K\exp(-sL)}{1+sT} = \frac{Y(s)}{U(s)}, \tag{1}$$

where $u(t)$ and $y(t)$ represent the process input and output, respectively. The three parameters that define the process and are to be identified are the process gain K, the process time constant T, and the process dead time L.

Equation (1) can also be written in the form of the following differential equation:

$$y(t) + T\frac{dy(t)}{dt} = Ku(t-L) + e(t). \tag{2}$$

In Eq. (2), $e(t)$ represents any noise present and is usually considered to be zero mean and white. Since we intend to consider the case when the output has not reached steady-state when the input is applied, Eq. (1) is inadequate. Taking the Laplace transforms of Eq. (2) reveals that:

$$Y(s) + T[sY(s) - y(0)] = K\exp(-sL)U(s) + E(s). \tag{3a}$$

The initial condition on $y(t)$ is represented by $y(0)$. Equation (3a) can be re-written in the following form:

$$Y(s) = -TsY(s) + K\exp(-sL)U(s) + E(s) + Ty(0). \tag{3b}$$

Integrating Eq. (3b) once eliminates the derivative term, a good first step when dealing with noisy data. However, it can be shown that for step inputs, the resulting parameter vector in the estimation vector can be decomposed into only three terms but there are four unknowns. If a second integration is performed the problem is overcome. It follows that integrating Eq. (3b) twice and re-arranging the result, yields the following equation:

$$\frac{Y(s)}{s^2} = -\frac{TY(s)}{s} + K\exp(-sL)\frac{U(s)}{s^2} + \frac{E(s)}{s^2} + \frac{Ty(0)}{s^2}. \tag{4}$$

Taking inverse Laplace transform of Eq. (4) reveals the 'integral estimation' equation:

$$y^{[2]}(t) = -Ty^{[1]}(t) + Ku^{[2]}(t-L) + e^{[2]}(t) + Ty(0)t. \tag{5}$$

In Eq. (5), the number in the square brackets of $y^{[2]}(t)$, $y^{[1]}(t)$, $u^{[2]}(t)$ and $e^{[2]}(t)$ in (5) refers to the number of integrations performed on the relevant variable.

Equation (5) is valid for any bounded input signal $u(t)$. However, Ahmed et al. [10]point out that if the input is restricted to be a step function of size h applied at $t = 0$, the following integral holds for $t \geqslant L$.

$$u^{[i]}(t-L) = \frac{h(t-L)^i}{i!}. \tag{6}$$

Using Eq. (6) in (5) results in the following:

$$y^{[2]}(t) = -Ty^{[1]}(t) + \frac{Kht^2}{2} - KhLt + \frac{KhL^2}{2} + Ty0(t) + e^{[2]}(t) \quad \text{for} \quad t \geqslant L. \tag{7}$$

In least squares form Eq. (7) can be written as:

$$y(t) = \phi(t)\theta + \zeta(t). \tag{8}$$

In Eq. (8):

$$\gamma(t) = y^{[2]}, (t) \quad \phi(t) = [-y^{[1]}(t)\ t^2/2\ t\ 1], \quad \zeta(t) = e^{[2]}(t),$$

$$\theta = \begin{bmatrix} L \\ Kh \\ [KhL - Ty(0)] \\ KhL^2/2 \end{bmatrix}. \tag{9}$$

Equation (9) can be written for $t = t_{d+1}, t_{d+2}, \ldots t_N$ where d is the time delay L expressed in numbers of samples and N is the total number of samples available. Using this data we can write that

$$\Gamma = \Phi\theta + Z. \tag{10}$$

In Eq. (10)

$$\Gamma = \begin{bmatrix} \gamma_{d+1} \\ \gamma_{d+2} \\ \cdots \\ \cdots \\ \gamma_N \end{bmatrix}, \quad \Phi = \begin{bmatrix} \phi_{d+1} \\ \phi_{d+2} \\ \cdot\cdot \\ \cdots \\ \phi_N \end{bmatrix}, \quad Z = \begin{bmatrix} e^{[2]}_{d+1} \\ e^{[2]}_{d+2} \\ \cdots \\ \cdots \\ e^{[2]}_N \end{bmatrix}. \tag{11}$$

The least squares solution for the estimation equation is given by:

$$\theta = [\Phi^T\Phi]^{-1}\Phi^T\Gamma. \tag{12}$$

Initially, it appears there is a problem since the estimation equation is valid only for $t \geqslant L$ and L is one of the parameters to be determined. The problem is solved by using an initial guess for L to start the estimation procedure, then to use the new value for L on the next pass and so on until two successive values for L satisfy some stopping criteria. Even for a bad guess it usually takes no more than four or five iterations. Using $y^{[1]}(t)$ and $y^{[2]}(t)$ in the estimation equation is also beneficial since any noise present in $y(t)$ will be filtered.

2.2. The SOPDT model

In this case the transfer function of the SOPDT model to be identified is given by:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K\omega_0^2 \exp(-sL)}{s^2 + 2\zeta\omega_0 s + \omega_0^2}. \tag{13}$$

In common with the FOPDT case, when the input is applied at $t = 0$ the system may not be in steady-state. To recognise this fact the formulation also includes initial conditions $y(0)$ and $y'(0) = dy(0)/dt$. The estimation equations, derived in a way similar to Eq. (9) and is given by:

$$\gamma(t) = y^{[1]}(t), \quad \phi(t) = \left[-y^{[2]}(t) - y^{[3]}(t)\ t^3/6\ t^2/2\ t\ 1\right],$$

$$\theta = \begin{bmatrix} 2\zeta\omega_0 \\ \omega_0^2 \\ K\omega_0^2 h \\ [C_0 - K\omega_0^2 hL] \\ [C_1 - K\omega_0^2 hL^2/2] \\ -K\omega_0^2 hL^3/6 \end{bmatrix}. \tag{14}$$

In Eq. (14), $C_1 = y(0)$ and $C_0 = y'(0) + 2\zeta\omega_0 y(0)$.

3. The LSQCURVEFIT estimation method

The ***LSQCURVEFIT*** algorithm available in the MATLAB Optimization Toolbox [15] solves nonlinear curve fitting problems in a least-squares sense. In our case it helps find the k parameter values '$a(k)$' of a 'desired' transfer function that solves the problem:

$$min\sum_i [F(a(k), Xdata_i) - Ydata_i]^2, \tag{15}$$

given an input data vector *Xdata* and an observed data vector *Ydata*, both of length N. In addition, $F(a(k), Xdata)$ is a vector valued function that, in this case, captures the step response associated with the chosen model. The formulation makes use of the 'anonymous function' concept, which provides a way of creating a simple function without having to store it to file each time.

3.1. FOPDT model

Consider the FOPDT model with the transfer function given by Eq. (1). If the input $u(t)$ is a unit step function the response $y(t)$ of this system, under non-zero initial conditions, is given by:

$$y(t) = Kh[1 - \exp(-(t-L)/T]H(t-L) + y(0)\exp(-t/T). \tag{16}$$

The syntax for creating the anonymous function is:

$$fhandle = @(\arg list)\exp r. \tag{17}$$

For the FOPDT system:

$$\arg list = [a, Xdata] \text{ and } \exp r = \text{equation (16)}$$

The symbol @ is a required part of the anonymous function definition. The algorithm written in the form of an MATLAB m-file is shown in Algorithm 1

Algorithm 1. m-file optlsqFOPDTpIC

```
function [a,resnorm]=optlsqFOPDTpIC(Working_Data)
  t=Working_Data(:,1);
  y=Working_Data(:,2);
  f=@(a,t)a(1)*(1-exp((-t+a(3))/a(2)))...
                .*heaviside(t-a(3))+a(4)*exp(-t/a(2));
  [a,resnorm] = lsqcurvefit(f,[a1 a2 a3 a4],t,y)
end
```

$[a, resnorm]$ returns the value of the squared 2-norm of the residual corresponding to the solution a. The four terms in the square brackets are the initial conditions. Comparing Eq. (16) with function f in Table 1 it is seen that $a1 = K; a2 = T; a3 = Landa4 = y(0)$.

3.2. SOPDT model

For some PI and PID designs and other more sophisticated model-based controllers a SOPDT model may be more appropriate. SOPDT models can include under-damped, critically damped and over-damped dynamics. In this sub-section we will limit consideration to over-damped systems and in particular to those governed by the transfer function:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K\exp(-sL)}{(1+sT_1)(1+sT_2)}. \tag{18}$$

Table 1
Results when using IE method.

Lguess	*N*	*K*	*T*	*L*
0.01	999	4.395	6.702	1.092
1.1	890	3.354	3.312	1.769
1.77	823	3.072	2.291	2.122
2.13	787	3.005	2.022	2.239
2.24	776	3.000	2.000	2.250

As with the FOPDT system the intention is to account for the influence of the initial conditions. Using exactly the same approach as detailed in the previous sub-section results in the m-file shown in Algorithm 2.

Algorithm 2. m-file optlsqSOPDTpIC

```
function [a,resnorm]=optlsqSOPDTpIC(Working_Data)
  t=Working_Data(:,1);
  y=Working_Data(:,2);
  f = @(a,t)a(1)*(1-a(2)*exp(-a(3)*(t-a(4)))/ ...
       (a(2)-a(3))+a(3)*exp(-a(2)*(t-a(4)))/ ...
       (a(2)-(3))).*heaviside(t-a(4))+ a(5)*(a(3)* ...
       exp(-a(3)*t)-a(2)*exp(-a(2)*t))/(a(3)-a(2))+ ...
       (a(6)+a(5)*(a(2)+a(3)))*(exp(-a(3)*t)- ...
       exp(-a(2)*t))/(a(2)-a(3))
  [a,resnorm] = lsqcurvefit(f,[a1 a2 a3 a4 a5 a6],t,y)
end
```

In Algorithm 2, $K = a(1)$, $T_1 = 1/a(2)$, $T_2 = 1/a(3)$; $L = a(4)$, $y(0) = a(5)$ and $y'(0) = a(6)$.

One of the problems of an iterative approach is how best to choose the starting values since a poor selection may not produce a convergent solution. To check the sensitivity of the final solution to the starting conditions a PSO approach, based on the equations employed by the LSQCURVEFIT method, is utilised. The developments supporting this method are considered in Section 4.

4. The particle swarm optimisation [PSO] estimation method

The original idea was to use PSO to simply check some of the LSQCURVEFIT results. Later it was decided to include the developments as a separate approach since it proved as effective as the methods discussed in Sections 2 and 3. PSO is a population-based evolutionary algorithm [16], in which a 'swarm' of particles continually move through the search space. Each particle represents a possible solution to the optimisation problem. The direction and size of step that a particle moves in the solution space during iteration is determined by a vector with three components.

(i) A contribution from the previous step vector (momentum) – *V*.
(ii) A contribution from the previous best solution found by the current particle (memory) – *Pbest*.
(iii) A contribution from the best nearest neighbour (local mode) – *Lbest* or alternatively.

A contribution from current best particle (elite) in the population (global mode) – *Gbest*.

In this work the experiments have been carried out using only the PSO local search strategy. The performance of each particle is determined using some form of fitness function. The main contribution of [16] was the derivation of simple velocity and position algorithms that in effect mimicked the natural swarm behaviour. The velocity and position of each particle is reshaped according to the equations given in Appendix A and a pseudocode representation of the algorithm shown in Appendix B.

The PSO algorithm was implemented using floating-point numbers to represent the variables since several authors have observed that for population based searches a real valued numerical representation produces faster, more accurate results than binary encoding, for real valued numerical problems [17].

4.1. Steps in implementing the PSO algorithm

A flow chart representing the PSO local model is shown in Fig. 1.

PSO is population-based, stochastic search method suited to continuous variable problems. It relies upon a system of communicating agents or particles and may be considered to be a parallel search for a solution. As already stated, evaluation of the performance of candidate solutions is found using an objective function (payoff information). This method ensures that there is a clear division between search domain and the search algorithm thereby freeing PSO from the constraint of having to use auxiliary or derivative information. PSO is normally implemented as a serial algorithm on a standard computer (PC).

Whilst the PSO method is more complex to implement it does have some significant advantages. The PSO algorithm can virtually always escape from local solutions, producing results that are robust, precise and repeatable.

To obtain a solution the PSO algorithm usually involves a large number of operations but for the most part these calculations just involve simple vector addition and subtraction. As a result solutions are normally found within a reasonable period of time. Although PSO is computationally expensive, the authors decided to investigate it in this application area for all of the reasons outlined.

Matlab was selected for this implementation because its code is concise and functions are available to directly manipulate vector arrays. The version of the PSO algorithm that has been implemented largely follows the standard design advocated in [16]. Refer to Appendices A and B for more details.

The value of the step variables are adjusted in a linear manner during the iteration process. Initially the step size is set to relatively high value so that the search rapidly moves into regions of interest. However when the algorithm is converging on a solution near the end of the run time the step size is gradually reduced to avoid oscillation around the near-optimal solution.

The core of the PSO algorithm is given in Matlab M file code in Algorithm 3.

Algorithm 3. MATLAB code for PSO algorithm

```
% randomise the step size
randglo(stepglo); randloc(steploc); randbst(stepbst);
% update step new, core of the PSO in local model A3=0
stepnew = W*stepold + Al*stepbst + A2*steploc + A3*stepglo;
popnew = popold + stepnew; % update new population (pop)
stepold = stepnew; % update step old vector array
popold=popnew; % update popold ready for next iteration
cnt = cnt + l;
```

5. Results

It is conventional practice to represent high order process models by approximations using FOPDT and SOPDT structures [18,14]. An important element of paper [14] is a new, highly sophisticated, optimal 'model compression' strategy based on frequency domain performance as captured by evaluating the deviation of the H_2 norm between the original and reduced order system. The idea was first suggested by Xue et al. [19] for integer order transfer functions but [14] have extended the method to include fractional order transfer functions. However, they also present the results obtained when using their methodology for determining FOPDT and SOPDT models of the processes $P_2(s), P_3(s)$ and $P_4(s)$, defined as follows:

$$P_1 = \frac{3\exp(-1.25s)}{1+2s}, \tag{19a}$$

$$P_2(s) = \frac{1}{(s+1)(0.2s+1)(0.04s+1)(0.008s+1)}, \tag{19b}$$

$$P_3(s) = \frac{1}{(1+s)^3}, \tag{19c}$$

$$P_4(s) = \frac{9}{(s+1)(s^2+2s+9)}. \tag{19d}$$

The intention here is to compare the results obtained in [14] with those obtained using the three approaches covered in Sections 2–4. The inclusion of system $P_1(s)$ is to confirm that when modelling a FOPDT system with a FOPDT model, all of the procedures realise an exact match.

5.1. Transfer function $P_1(s)$, step input, zero initial condition

The first approach applied is the IE method. This is an iterative method where convergence is normally quite rapid and Table 1 presents results of the iterations for an initial guess of $L = 0.01$ second. Data is generated using a MATLAB Simulink model of the process.1000 data points are collected from $t = 0$ to $t = 10$ at a sample time of 0.01 s.

The initial guess was intentionally a very poor one and an improved choice will reduce the number of repeats. N represents the number of data points used in the estimation counted back from the final value. The reason why the final value for L is 2.25 and not 1.25 is that the step was applied one second after data collection was initiated. This is necessary when the actual time delay is small. Thus, the true value is (L – step time), which in this case is 1.25 s.

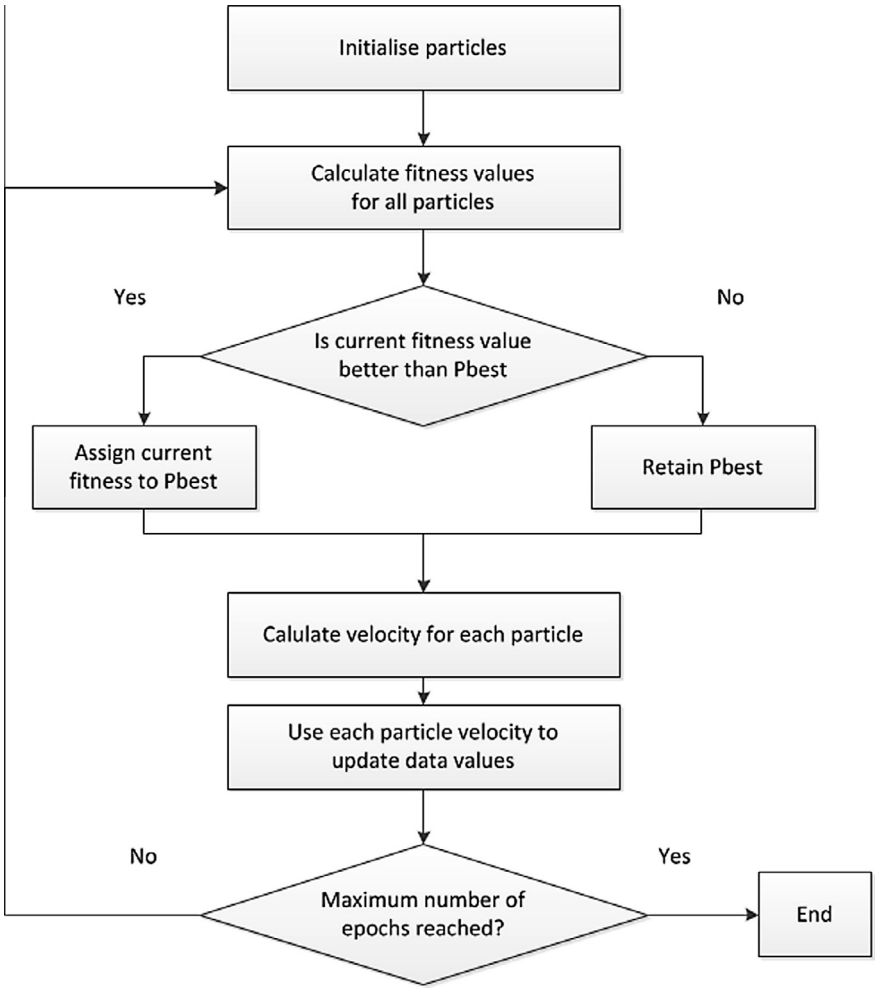

Fig. 1. Flow diagram illustrating the local model PSO algorithm.

The second approach used the least squares algorithm presented in Algorithm 1. The results shown in Table 2 were for starting values [0.50.50.5]. However, a number of different starting conditions were tested, which produced identical results. Note the very small value of the residual norm.

Preliminary experiments were also undertaken to evaluate the performance of the PSO method. In this case, all of the experiments were repeated with a population size of 30, with 1000 epochs and 4 runs. The results obtained are always the average of the four runs. The results obtained using PSO matched those given in Table 1 for the variables K, T and L.

5.2. Transfer functions $P_2(s)$, $P_3(s)$ and $P_4(s)$, step input, zero initial conditions

Tables 3, 5 and 7 show the FOPDT model parameters and Tables 4, 6 and 8 show the SOPDT model parameters for processes $P_2(s)$, $P_3(s)$ and $P_4(s)$, respectively. To compare the 'fit' of the estimation, the tables also include two measures of performance: the integral squared error (ISE) and the integral of absolute error (IAE) where the error in each case is the difference between the 'true' process and model output.

Some initial general observations are summarised as follows. For processes $P_2(s)$ and $P_3(s)$ there is little difference in the parameter values estimated by the four methods. In addition, irrespective of the method used, the SOPDT model fit is better

Table 2
Results when using the LSQCURVEFIT method.

K	T	L	ResNorm
3.0000	2.0000	1.2499	5.62e−007

than the FOPDT model fit, as demonstrated by the measured ISE and IAE values. Only the method employed by Das et al. [14] consistently obtained the correct value for K. Their paper does not give any detailed information about the method employed. However, it seems reasonable to assume that the optimisation criteria used has a final value constraint. Such a constraint obviously affects the values obtained for the other model parameters. The next section explains how it is the combinational effect of all the parameters (rather than one individual parameter) that decides which model is better when designing PI and PID controllers.

The linear systems for which the FOPDT and SOPDT models are most accurate have continuous-time, monotonic, non-overshooting step responses. This implies that the slope of the step response to a positive step should always be non-negative. Fig. 2a and b shows the unit step responses (black) and slope profiles (magenta) for transfer functions $P_3(s)$ and $P_4(s)$ respectively.

Clearly, the step response of $P_4(s)$ is not monotonic and FOPDT and SOPDT models will not be good fits. Associated with the poor fits are the disappointing values for ISE and IAE in Tables 9 and 10 when compared with the values achieved in Tables 5–8.

5.3. Transfer functions $P_2(s)$ and $P_3(s)$, step input, non-zero initial conditions

The values shown in Tables 9–14 are typical of the results obtained. The FOPDT model has only one state and one initial condition, whilst the SOPDT model has two states and two initial conditions. $P_2(s)$ and $P_3(s)$ have four and three states respectively. Neither the FOPDT nor the SOPDT models will be able to capture 'exactly' the transient response of the test transfer functions under a variety of different starting conditions. There is little interest in calculating values for the initial conditions since they rarely have any practical use. The main question is how close are the parameters of the FOPDT and SOPDT models compared to those determined in Section 5.2. Again, SOPDT models fit the data better than FOPDT models.

In Tables 9–12, $P_4(s)$ is not included because its step response is not monotonic. The results of determining average values of the parameters of the FOPDT and SOPDT models, with and without initial conditions, are shown in Tables 13 and 14.

6. Design of all stabilizing PI controllers for processes $P_3(s)$ and $P_4(s)$

In the design of any controller, closed loop stability is an essential prerequisite. A review paper by Saadaoui et al. [20] outlines the present situation regarding the various methods available for determining the set of all stabilizing PID controllers for FOPDT and SOPDT processes. This is often referred to as the 'stabilization problem'. However, in this section the aim is to develop an alternative approach and then present two illustrative examples using $P_3(s)$ and $P_4(s)$.

Consider a system controlled by a PI controller with transfer function:

$$C(s) = K_C + \frac{K_i}{s}. \tag{20}$$

The aim is to find the set of all controller parameters yielding a stable closed-loop system. The conditions that define stability are often developed using a Nyquist diagram. This in turn can be used to develop stability domains in the $K_c \sim K_i$ plane or controller parameter space. This is the strategy to be followed here.

6.1. Stability boundary locus in the $K_C \sim K_i$ parameter plane

Consider an open loop system comprising a process with transfer function $P(s)$ in series with a controller $C(s)$ such that the open-loop transfer function $C_{OL}(s)$ is given by:

$$C_{OL}(s) = C(s)P(s). \tag{21}$$

The open loop frequency response can be obtained by replacing $sbyj\omega$ in (21) to yield:

$$C_{OL}(j\omega) = C(j\omega)P(j\omega). \tag{22}$$

For the PI controller of (20):

$$C(j\omega) = K_C + \frac{K_i}{j\omega} = \frac{j\omega K_C + K_i}{j\omega}. \tag{23}$$

Table 3
FOPDT results for $P_2(s)$.

Method	K	T	L	ISE	IAE
Das	1.0000	1.0565	0.2097	3.7e−004	0.03272
IE	1.0028	1.0530	0.2100	3.4e−004	0.02639
LSQ	1.0035	1.0625	0.2094	3.3e−004	0.02856
PSO	1.0002	1.0121	0.2405	5.3e−004	0.02169

Table 4
SOPDT results for $P_2(s)$.

Method	K	W0	Zeta	L	ISE	IAE
Das	1.000	2.164	1.310	0.042	1.18e−05	0.00501
IE	1.000	2.226	1.337	0.046	9.27e−07	0.00145
LSQ	0.999	2.045	1.248	0.020	2.29e−05	0.01017
PSO	1.000	2.151	1.305	0.024	4.484-05	0.00968

Table 5
FOPDT results for $P_3(s)$.

Method	K	T	L	ISE	IAE
Das	1.000	2.029	1.134	0.00779	0.2180
IE	1.060	2.433	0.987	0.00485	0.1594
LSQ	1.061	2.448	1.005	0.00431	0.1581
PSO	1.037	2.139	1.159	0.00575	0.1430

Table 6
SOPDT results for $P_3(s)$.

Method	K	$W0$	Zeta	L	ISE	IAE
Das	1.0000	0.7120	0.9257	0.4010	1.55e−04	0.02626
IE	0.9910	0.7050	0.8934	0.4000	1.54e−04	0.02345
LSQ	1.0138	0.7540	1.0010	0.4460	3.83e−04	0.04820
PSO	1.0095	0.7764	1.0030	0.4927	4.64e−04	0.04573

Table 7
FOPDT results for $P_4(s)$.

Method	K	T	L	ISE	IAE
Das	1.000	0.8889	0.415	0.00805	0.1295
IE	0.974	0.6633	0.457	0.00410	0.1161
LSQ	0.985	0.7091	0.451	0.00395	0.1184
PSO	0.999	0.8257	0.400	0.00500	0.1185

Table 8
SOPDT results for $P_4(s)$.

Method	K	$W0$	Zeta	L	ISE	IAE
Das	1.000	2.3210	1.0300	0.2650	0.00513	0.1319
IE	0.977	1.6570	0.9040	0.300	0.00762	0.1663
LSQ	0.980	2.7600	1.1500	0.2960	0.00309	0.1132
PSO	0.9976	2.4507	1.1223	0.2443	0.00401	0.1162

It follows that the open-loop phase shift is given by:

$$\angle C_{OL}(j\omega) = \angle C(j\omega)\angle P(j\omega) = -\frac{\pi}{2} + tan^{-1}\left(\frac{\omega K_C}{K_i}\right) + \angle P(j\omega). \tag{24}$$

The open loop gain is given by:

$$|C_{OL}(j\omega)| = |C(j\omega)||P(j\omega)| = \frac{\sqrt{\omega^2 K_C^2 + K_i^2}|P(j\omega)|}{\omega}. \tag{25}$$

Consider Fig. 3, which show a typical Nyquist diagram, where the vertical axis is $Imag[C_{OL}(j\omega)]$ and the horizontal axis is $Real[C_{OL}(j\omega)]$. The problem, as ω increases from 0 to ∞, is to determine the values of K_C and K_i that force the frequency response $C_{OL}(j\omega)$ to pass through the $-1 + j0$ point. This implies that:

$$\angle C_{OL}(j\omega_{-180}) = -\pi \quad \text{and} \quad |C_{OL}(j\omega_{-180})| = 1. \tag{26}$$

In (26), ω_{-180} is the frequency when the phase shift of the open-loop frequency response is -180^0.
It is straightforward to show from (24) and (25) that:

$$T_i = \frac{K_C}{K_i} = \frac{\tan(\phi)}{\omega},$$

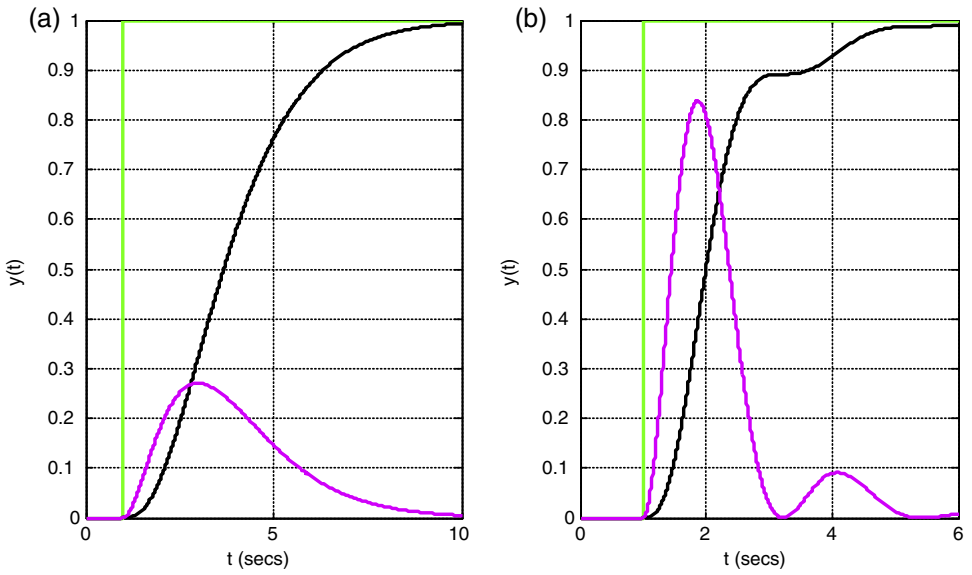

Fig. 2. (a) Step response $P_3(s)$. (b) Step response $P_4(s)$.

Table 9
FOPDT results for $P_2(s)$.

Method	K	T	L	$Y(0)$	ISE	IAE
IE	1.001	1.019	0.239	0.398	7.27e−04	0.0.289
LSQ	1.004	1.078	0.210	0.385	3.83e−04	0.0375
PSO	1.000	1.020	0.233	0.394	6.41e−04	0.0308

Table 10
SOPDT results for $P_2(s)$.

Method	K	$W0$	Zeta	L	$dY(0)/dt$	$Y(0)$	ISE	IAE
IE	1.000	2.234	1.341	0.049	−0.061	0.343	2.6e−6	0.0021
LSQ	0.998	2.008	1.226	0.012	−0.146	0.352	2.7e−5	0.0107
PSO	1.000	2.228	1.338	0.038	−0.105	0.347	4.1e−5	0.0130

Table 11
FOPDT results for $P_3(s)$.

Method	K	T	L	$Y(0)$	ISE	IAE
IE	1.033	2.040	1.445	0.3008	8.60e−03	0.1883
LSQ	1.037	2.190	1.331	0.2826	7.37e−03	0.1919
PSO	1.029	2.382	1.398	0.2750	1.57e−02	0.3440

Table 12
SOPDT results for $P_3(s)$.

Method	K	$W0$	Zeta	L	$dY(0)/dt$	$Y(0)$	ISE	IAE
IE	0.995	0.745	0.921	0.524	−0.275	0.353	6.7e−4	0.039
LSQ	1.012	0.766	1.001	0.490	−0.389	0.391	3.2e−4	0.048
PSO	1.009	0.782	1.000	0.509	−0.408	0.393	4.6e−4	0.046

Table 13
Averaged FOPDT model parameters.

Model	Initial conditions	Kaverage	Taverage	Laverage
$P_2(s)$	No	1.0021	1.046	0.2174
$P_2(s)$	Yes	1.0017	1.039	0.2270
$P_3(s)$	No	1.0395	2.2622	1.0713
$P_3(s)$	Yes	1.0330	2.204	1.3913

Table 14
Averaged SOPDT model parameters.

Model	Initial conditions	Kaverage	ω_o average	ζ average	Laverage
$P_2(s)$	No	0.9998	2.1465	1.3000	0.0330
$P_2(s)$	Yes	0.9993	2.1553	1.3017	0.0330
$P_3(s)$	No	1.0036	0.7369	0.9558	0.4349
$P_3(s)$	Yes	1.0053	0.7640	0.9740	0.5007

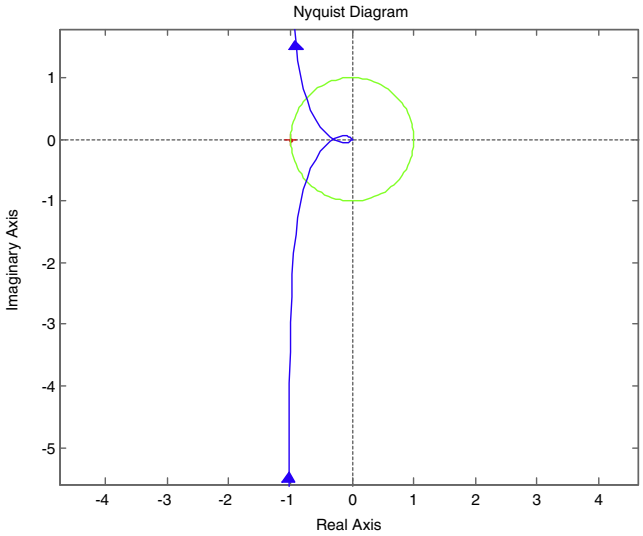

Fig. 3. Typical open loop frequency response.

$$K_C = \frac{\sin(\phi)}{|P(j\omega)|} \Rightarrow K_i = \frac{K_C}{T_i} = \frac{\omega\cos(\phi)}{|P(j\omega)|}. \tag{27}$$

In (27):

$$\phi = -\frac{\pi}{2} - \angle P(j\omega). \tag{28}$$

The next step is to use (27) to plot the stability boundary in the $K_C \sim K_i$ parameter plane. The results using $P_2(j\omega)$ and $P_3(j\omega)$ are very similar consequently the illustrative examples will use $P_3(j\omega)$ and $P_4(j\omega)$.

6.2. Illustrative example using $P_3(s)$

The stability boundary loci for process $P_3(s)$ plus the FOPDT and SOPDT models are shown in Fig. 4a and b, respectively. Each individual curve is plotted from $\omega = 0$ to ω_{-180}, the frequency when the phase shift of the process or model is -180^0. The curves originate at the origin of the parameter plane. The portions of the curves that have negative values of K_C correspond to ω values from $0 to \omega_{-90}$, the frequency when the phase shift of the process or model is -90^0.

The graphs are colour-coded as follows: black: $P_3(j\omega)$; green: PSO model; blue: IE model; red: LSQ model; and magenta: Das model. In both graphs all of the models are low frequency accurate. In all cases the SOPDT models provide are more

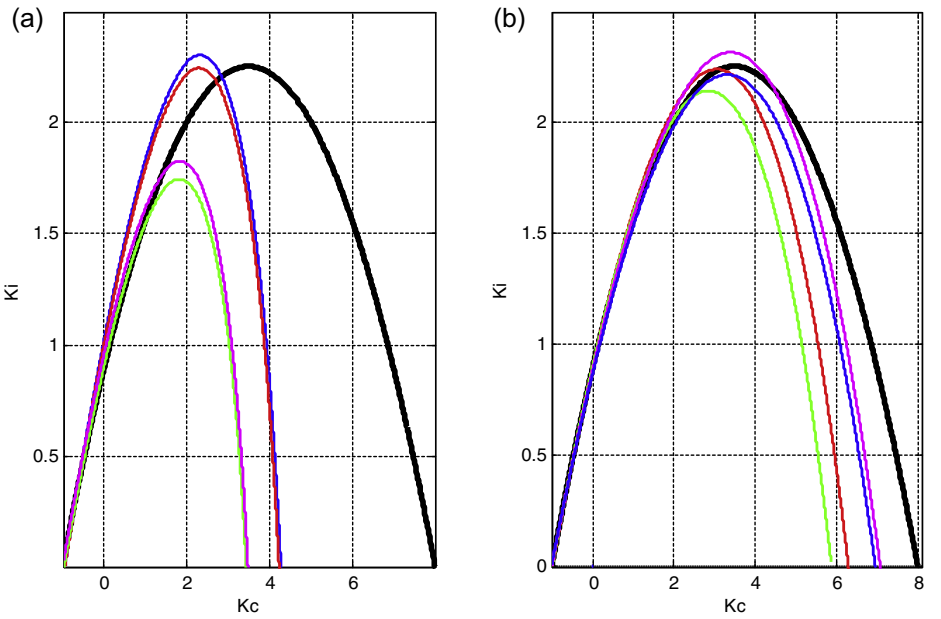

Fig. 4. (a) Stability boundary loci for $P_3(s)$ and FOPDT models. (b) Stability boundary loci for $P_3(s)$ and SOPDT models.

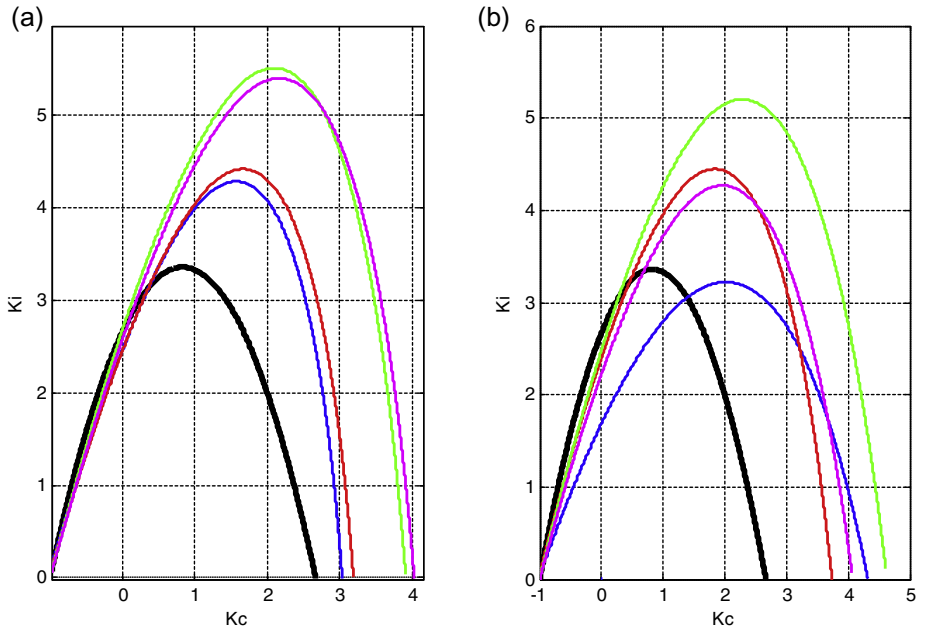

Fig. 5. (a) Stability boundary loci for $P_4(s)$ and FOPDT models. (b) Stability boundary loci for $P_4(s)$ and SOPDT models.

accurate stability prediction than the FOPDT models. The LSQ and IE model results are similar for both cases. The PSO results are the most conservative, but they are the only ones that do not predict unstable behaviour when $P_3(j\omega)$ predicts a stable solution. The Das results are disappointing for the FOPDT case but comparable to the IE result for the SOPDT case.

6.3. Illustrative example using $P_4(s)$

In this case, the results are disastrous and confirm that both FOPDT and SOPDT models should not be used to predict the stability boundary of a system when the step response is not monotonic. Obviously, the SOPDT results provide a better estimate but are still inadequate.

7. Discussion and conclusions

Model-based PI and PID control strategies have been increasingly explored in recent years [21]. The main reason for this is their superior performance in set-point tracking and disturbance rejection. Linked with these advances have been a number of parallel studies on the identification of FOPDT and SOPDT models. This is because the response of such models is generally a good approximation to the monotonic, non-overshooting continuous-time step response of many of the processes and sub-processes found in industrial process control.

Two separate experimental conditions are considered. The first is when the input step is applied once steady-state or near steady-state conditions are achieved. In the second, the step is intentionally applied before steady-state is achieved. This latter situation is particularly useful when the process time constants are of the order of tens of minutes or longer.

For all four methods [Das, IE, LSQ and PSO], when using ideal noise free data, perfect models are possible for those situations when a FOPDT model is used to identify a FOPDT process and a SOPDT model is used to identify a SOPDT process, provided rudimentary rules relating to sample time, number of data points and end of time data collection are adhered to [10]. However, the main aim is to develop FOPDT and SOPDT models of processes that have an order higher than two.

The set of test models are those defined by Eqs. (19b)–(19d) and were used recently in the paper by Das et al. [14]. An inspection of Tables 3–6 shows that for systems $P_2(s)$ and $P_3(s)$, the SOPDT models produce a closer fit to the real data than the FOPDT models as judged by the ISE and IAE criteria, irrespective of method. However, a more quantifiable measure of performance is derived in Section 6, where closed loop stability boundaries are determined based on the various models and compared with that obtained using the actual system equation.

The results for the FOPDT and SOPDT cases are shown in Fig. 4a and b, respectively. These results confirm that the SOPDT models are better than the FOPDT models when predicting the stability boundary as part of a PI controller design study. Unfortunately, there is no clear single best approach. For the FOPDT situation, IE, LSQ and Das all have excursions into the unstable region. Only PSO remains exclusively within the stable domain. For the SOPDT models, the IE and PSO remain within the true system boundary with the IE result slightly out-performing the PSO result. It follows that best results are for the SOPDT models and are, in order, IE, PSO, LSQ, and Das.

Tables 9–12 present the results derived for the parameters of the FOPDT and SOPDT models when using noise-free step response data with non-zero initial conditions. Tables 13 and 14 show the average parameters obtained for the FOPDT and SOPDT models when using zero and non-zero initial conditions. The sets of results turn out to be satisfactorily close.

System $P_4(s)$ is shown not to have monotonic step response. The FOPDT and SOPDT models derived are shown in Tables 7 and 8, respectively. Examination of Fig. 5a and b demonstrates why they should never be used as part of a PI controller design study.

Finally, the performance of the LSQ and PSO approaches, when used to estimate the individual parameter values of FOPDT and SOPDT models, have been found to be satisfactory. The simplicity of the LSQ identification procedure when compared with the IE method makes it particularly useful for undergraduates and those working in industry. A restriction is one must have MATLAB and access to the Optimisation Toolbox. However, a solution to this impasse could be the EzyFit Toolbox [22], which is a free add-on for MATLAB. EzyFit offers a simple and efficient solution by posing the identification problem as a curve fitting routine (EzyFit, 2014). A major advantage of PSO was shown to be its ability to escape local minima associated with suboptimal solutions. In all cases the PSO found a good solution in less than one minute.

Acknowledgements

The authors of this paper wish to acknowledge that the PSO algorithm in this paper is based upon original code created by the late Dr. Sonia Janet Tindle.

Appendix A. The particle swarm optimisation algorithm

Local:

$$V_i^{k+1} = W_i * V_i^k + A_1 * rand() * [Pbest_i - S_i^k] + A_2 * rand() * [Lbest_i - S_i^k]. \tag{A1}$$

Global:

$$V_i^{k+1} = W_i * V_i^k + A_3 * rand() * [Pbest_i - S_i^k] + A_4 * rand() * [Gbest_i - S_i^k], \tag{A2}$$

$$S_i^{k+1} = S_i^k + V_i^{k+1}. \tag{A3}$$

In addition, the inertia weight, W_i, is set according to the following equation:

$$W_i = W_i^{\max} - \frac{W_i^{\max} - W_i^{\min}}{iter}.ctn. \tag{A4}$$

Typical values for $W_i^{\max}$ and $W_i^{\min}$ are 0.9 and 0.4 respectively while *iter* defines the maximum number of iterations chosen and *ctn* represents the current iteration number.

In Eqs. (A1)–(A3), the notation used corresponds to the following brief definitions:

V_i^k = current velocity of particle i at iteration k.
V_i^{k+1} = new position of particle i at iteration $k+1$.
A_1 = adjustable positive factor ◁1.0.
A_2 = adjustable positive factor ◁1.0.
A_3 = adjustable positive factor ◁1.0.
A_4 = adjustable positive factor ◁1.0.
rand() returns a floating point random number between 0 and 1.
S_i^k = current position of particle i at iteration k.
S_i^{k+1} = new position of particle i at iteration $k+1$.
$Pbest_i$ = previous best 'remembered' position of the i_{th} particle.
$Lbest_i$ = local best position of the i_{th} particles nearest neighbour.
$Gbest_i$ = global best particle position over the entire population.

A pseudo algorithm for the PSO local model is listed in Appendix B.

Appendix B

Pseudo algorithm for PSO local model

```
For each particle
(
  Initialise particle
)
Repeat until maximum iteration count reached
(
  For each particle
  (
    Compare desired target and actual curves and
    Evaluate fitness value
  )
  If fitness is better than Pbest
  (
    Set Pbest to current fitness value
  )
For each particle
  (
    Calculate particle velocity
    Use Pbest and velocity to update particle Data
  )
)
```

References

[1] S. Ahmed, B. Huang, S.L. Shah, Novel identification method from step response, Control Eng. Pract. 15 (5) (2007) 545–556.
[2] J.G. Ziegler, N.B. Nichols, Optimum settings for automatic controllers, Trans. ASME 65 (1942) 433–444.
[3] R.C. Oldenbourg, H. Sartorius, The Dynamics of Automatic Control, ASME, New York, 1948.
[4] J.W. Sten, Evaluating second-order parameters, Instrum. Technol. 17 (1970) 39–41.
[5] H. Rake, Step response and frequency response methods, Automatica 16 (1980) 519–526.
[6] Y. Nishikawa, N. Sannomiya, T. Ohata, H. Tanka, A method for auto-tuning of PID control parameters, Automatica 20 (1984) 321–332.
[7] K.J. Astrom, T. Hagglund, PID Controllers: Theory, Design and Tuning, Instrument Society of America, 1995.
[8] Q. Bi, W. Cai, E. Lee, Q.G. Wang, C.C. Hang, Y.X. Zhang, Robust identification of first-order plus dead-time model from step response, Control Eng. Pract. 7 (1) (1999) 71–77.
[9] Q.G. Wang, Y. Zhang, Robust identification of continuous systems with dead-time from step response, Automatica 37 (3) (2001) 377–390.
[10] S. Ahmed, B. Huang, S.L. Shah, Identification from step responses with transient initial conditions, J. Process Control 18 (2) (2008) 121–130.
[11] M. Liu, Q.G. Wang, B. Huang, C.C. Hang, Improved identification of continuous-time delay processes from piecewise step test, J. Process Control 17 (1) (2007) 51–57.
[12] S. Ahmed, Process identification using non-ideal steps, in: 9th Int. Symp. On Dynamics and Control of Process Systems (DYCOPS, 2010), Leuven, Belgium, July 5–7, 2010.
[13] Y.Y. Du, J.S.H. Tsai, H. Patil, L.S. Sheih, Y. Chen, Indirect identification of continuous-time delay systems from step responses, Appl. Math. Model. 35 (2011) 594–611.

[14] S. Das, S. Saha, S. Das, A. Gupta, On the selection of tuning methodology of FOPID controllers for the control of higher order processes, ISA Trans. 50 (2011) 376–388.
[15] MATLAB optimization toolbox: User's guide. Mathworks Inc., 2009.
[16] R.C. Eberhart, J. Kennedy, Swarm Intelligence, Academic Press, USA, 2001.
[17] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, second ed., IEEE Press, USA, 2000.
[18] G. Fedele, A new method to estimate a first-order plus time-delay mode from step response, J. Franklin Inst. 346 (2009) 1–9.
[19] Xue, Y.Q. Chen, A comparative introduction of four fractional order controllers, in: Proc. 4th World Congress on Intelligent Control and Automation, 4, 2002, pp. 3228–35.
[20] Karim Saadaoui, Sami Elmadssia, Mohamed Benrejeb, Stabilizing PID controllers for a class of time delay systems, in: Dr. Marialena Vagia (Ed.), PID Controller Design Approaches – Theory, Tuning and Application to Frontier Areas, InTech, 2012. ISBN: 978-953-51-0405-6. DOI: http://dx.doi.org/10.5772/32293. Available from: <http://www.intechopen.com/books/pid-controller-design-approaches-theory-tuning-and-application-to-frontier-areas/stabilizing-pid-controllers-for-a-class-of-time-delay-systems>.
[21] T. Liu, Q.-G. Wang, H.-P. Huang, A tutorial review on process identification from step or relay feedback test, J. Process Control 23 (2013) 1597–1623.
[22] Ezyfit: a free curve fitting toolbox for MATLAB. Available <http://www.fast.u-psud.fr/ezyfit>.