A hands-on approach to teaching system identification using first order plus dead time (FOPDT) modelling of step response data

Kevin Burn, Chris Cox

Faculty of Technology, University of Sunderland, UK

Corresponding author:
Kevin Burn, Faculty of Engineering and Advanced Manufacturing, University of Sunderland, SR6 0DD, UK. Email: kevin.burn@sunderland.ac.uk

Abstract

This paper describes three step response-based system identification methods of increasing complexity, together with a range of exercises that will enhance student understanding of this area in an engaging and practical way. For illustration purposes and practicality, it is assumed that the model to be identified is of the first order plus dead time (FOPDT) type. The first method uses a popular graphical technique, which is easy to understand and apply, but inaccurate when the response data is not ideal. The second uses the Nelder-Mead simplex method, which is a more powerful technique and has the added benefit of introducing undergraduate students to the concepts of numerical optimisation. The third uses an integral equation (IE) algorithm. The latter two methods, which can be readily extended to other model structures and input types, are also demonstrated using experimental data obtained from a tank level control system.

Keywords

Control systems education, system identification, first order plus dead time (FOPDT), parameter estimation; step response

Introduction

System identification (SI) is the area of mathematical modelling that uses experimentally collected input-output data to identify the dynamic characteristics of a process.[1] Often, the models evaluated are then used in the design of controllers of the proportional-integral-derivative (PID) form, although others forms can also benefit. For many years, system identification has been the subject of a great deal of research, and a wide variety of methods have been developed. They include those based on data acquired during step or pulse response tests, which form the background to the work described in this paper.[2]

SI is of particular importance within the process industries, which include a variety of major sectors such as petroleum, plastics, paper, power generation, food, and water treatment. Despite their obvious diversity, the process conditions that need to be controlled are often similar and include variables such as pressure, level, temperature, flow and pH. The process models of many of these applications are often assumed to be continuous-time transfer functions of a first order plus dead time (FOPDT) or second order plus dead time (SOPDT) structure. This is because the

responses of such models are good approximations to the continuous-time responses of many of the processes and sub-processes found in industry. Other processes, however, require more complex models, such as those with non-minimum phase zeros and/or free integrators in the open loop transfer function, or where the system is not in steady-state at the start of the data acquisition period.

Methods to estimate the parameters of a continuous-time transfer function using step response data feature regularly in the literature. The early approaches were aimed primarily at FOPDT and SOPDT structures and relied mainly upon graphical techniques.[3-6] For monotonic step responses, the methods based on specific area calculations are well represented in the literature.[7,8] For a second order, non-minimum phase system, a graphical method has been proposed that avoids complex experimental design and exploits the step response.[9] A limitation of all of 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. Modelling techniques that require the process to be at steady-state are not generally applicable to processes that contain integration and a graphical technique has been described that uses a FOPDT plus integrator model.[10] An alternative approach is to use an auto-tuning procedure based on relay feedback.[11]

More recently, a family of 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. A common feature of these new methods is that the data used in the evaluations is obtained in discrete-time. In this context, the original ideas have been extended to allow identification under transient initial conditions.[12-14] Other developments have also been reported, such as new methods to handle 'piecewise step tests', which include pulse inputs.[15,16] Finally, the methods have been extended to include integrating processes with time delay.[17] All of these developments come under the heading of integral equation (IE) approaches. In a recent paper,[18] the IE method has been employed with four benchmark transfer functions,[19] to solve the parameter estimation problem and compare the results with an alternative solution based on particle swarm optimisation (PSO).

A plant model is required for numerous controller design algorithms, many of which assume a first order plus dead time (FOPDT) structure. This is partly because a FOPDT response is a good approximation to many industrial processes, but also because PID controller gains can often be initiated using formulae that specify their values in terms of the FOPDT parameters.[20]

The FOPDT model is represented by the following transfer function:

$$Gp(s) = Y(s)/U(s) = \mathrm{K}\, exp(-Ls)/(Ts + 1) \qquad (1)$$

where K is the process gain, T the time constant, and L the time delay, or dead time.

Consider the process response $y(t)$ to a step input $u(t)$ occurring at some time $t \geq 0$. If the system is initially at rest, the response will be of the form shown in Figure 1, where h is the step size (in this case unity).

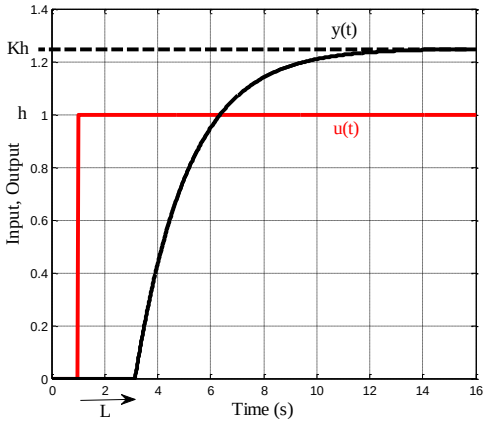


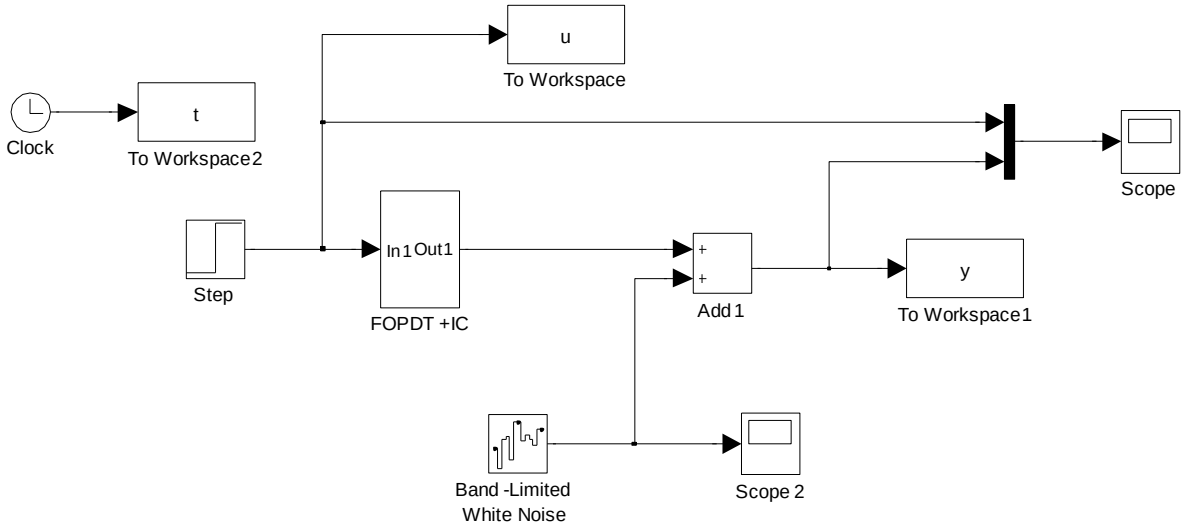
Figure 1. Step response of FOPDT model

The aim of this paper is to describe a small sample of the methods described above and present a range of exercises that can be used to teach FOPDT identification within an introductory control course. Three methods of varying sophistication are discussed, beginning with a graphical technique. This is followed by two more advanced solutions, one of which uses an optimisation method, and the other an IE method. The application of the latter two methods to an experimental system is also described.

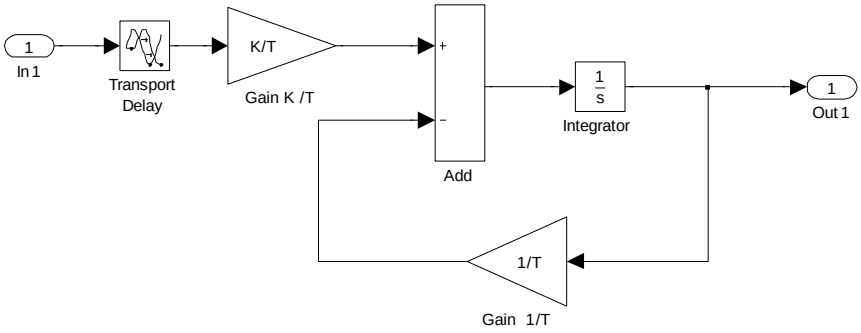
The exercises require access to MATLAB/Simulink, where it is assumed that parameters *K*, *T* and *L* are unknown to the student. It is thus necessary for a third party (e.g. a fellow student or tutor) to generate the response data. The student should also be familiar with the terminology and mathematics of the FOPDT response. The work is primarily aimed at courses with a more practical bias, which often include students with a wide range of mathematical abilities. The exercises are designed to promote inclusivity, by enabling individuals of all abilities to explore methods that some would find difficult in a traditional lecture room setting. All of the MATLAB files used in producing the paper can be obtained by emailing the corresponding author.

Simulink model

The Simulink model shown in Figure 2 can be used to generate FOPDT data with and without (a) added noise, and (b) a non-zero initial condition (IC). The latter is assigned to the integrator block of Figure 2(b).



(a) FOPDT model



(b) FOPDT + IC block

Figure 2. Simulink model to generate FOPDT data

Two-point method

The first exercise requires simulated, noise-free FOPDT step response data of a form similar to that shown in Figure 1.

Exercise 1

- Estimate *K*, *T* and *L* from ideal step response data, obtained under simulation, for a range of simulated FOPDT models, using a combination of measurement and inspection.
- Quantify the accuracy of the results using an appropriate measure e.g. sum of squared error (SSE).

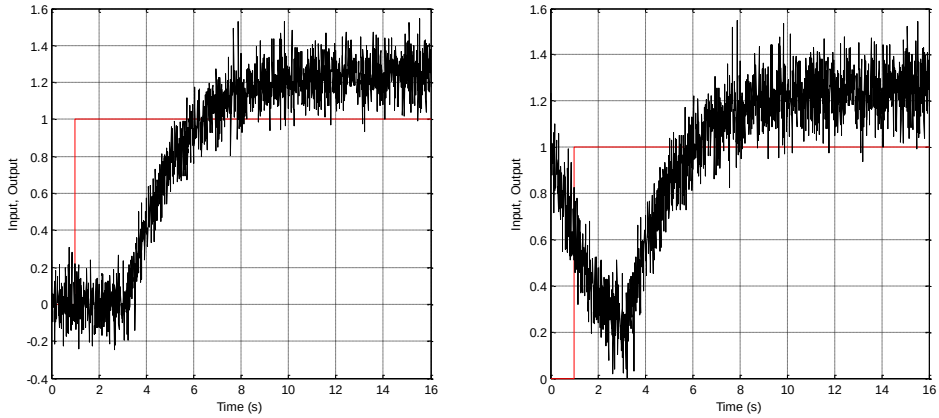
The most basic technique is to visually estimate *K* and *L*, and calculate *T* knowing that if *t* = (step time)+*T*+*L*, the response is at 63% of its maximum value. A more sophisticated method is to use the two-point algorithm.[2] Firstly, the process gain *K* is determined by dividing the steady state output by the input set-point. By estimating the time taken for the response to reach 28.3% and 63.2% of its final value, *T* and *L* can then be calculated as follows:

$$T = 1.5(t_{63\%} - t_{28\%})\ ; \qquad L = t_{63\%} - T \tag{2}$$

The method works well with ideal, noise-free data. However its limitations become apparent when analysing data with significant measurement noise and/or an IC. For example, consider the data shown in Figure 3(a), where noise has been added to the output, and in Figure 3(b) where there is also an IC.

Exercise 2

- Estimate parameters *K*, *T* and *L* from step response data for a range of simulated FOPDT models, with various combinations of noise and ICs.
- Quantify the accuracy of the results and compare them with the results of Exercise 1.



(a) IC = 0 (b) IC = 1

Figure 3. FOPDT data, with band limited white noise of power 10^{-4}. Red = step input applied at t = 1s; Black = process output

The estimated values of *K*, *T* and *L* in Exercise 2 will be at best less consistent, and at worst highly inaccurate, compared to those obtained during Exercise 1. Having established this, it is convenient to introduce a more sophisticated method of identification.

Nelder-Mead simplex method

The Nelder-Mead (N-M) simplex method is an optimization algorithm for minimizing a function of *n*-variables.[21,22] In *n*-dimensional space, a simplex is a set of (*n*+1) points, and the algorithm works by updating the simplex vertices until the minimum value of a function is found within some pre-defined tolerance. This is achieved through a sequence of reflections, expansions, contractions and shrinkages of the vertices until the minimum is found. Also, it is a direct search method, meaning that function values only are used during the search, not gradients. It has been successfully applied in many science and engineering applications over the years, although the algorithm can fail, either due to convergence to a local minimum, or due to stagnation.[23]

In this work, the function to be minimised is given by the MATLAB expression:

```
f=sum((y - Kh*(1-exp(-(t-L)/T)).*heaviside(t-L)).^2);
```
(3)

where h is the step size, `heaviside(x-L)` is the Heaviside step function, and t and y are vectors containing discrete time and response output data, respectively.

The N-M algorithm is available in MATLAB via the function *fminsearch.m*.[24] It can be used directly in this form, or more conveniently using a free-to-download toolbox called EzyFit. This was written by Frédéric Moisy of Paris-Sud University,[25] and was intended to offer an efficient way of curve fitting for nonlinear functions.

To maximise the algorithm's chance of success, a sensible initial estimate of the parameters to be identified is desirable. In this application, this could be from a user's experiential knowledge of the process dynamics, or through a preliminary identification using a less accurate technique, such as the area method. [4] For the work described here, it is usually sufficient to use the default estimates of unity, although the performance can deteriorate with increasing levels of noise. This is an area that can be explored in the following exercise:

Exercise 3

- (Optional) Download and install the EzyFit toolbox.
- Identify FOPDT parameters for a range of models using *fminsearch.m* directly or via EzyFit, using ideal data (zero noise).
- Repeat with increasing noise power.

When using EzyFit, a *New User Fit* is required, with the following user-defined function. This is simply a modified version of equation (3), with the variable x replacing t in normal EzyFit notation, and initial estimates of unity:

```
Kh*(1-exp(-(x-L)/T)).*heaviside(x-L); T=1;L=1;K=1;
```
(4)

A typical result for a noise-contaminated output, for a unit step applied at t=1 s, is shown in Figure 4, where the actual model parameters were: K = 1.25; T = 2; L = 2.15; noise power Np = 0.0002 (L in Figure 4 is the time delay + the step time). The sample time was 0.01 s. The reader is invited to confirm that error-free results are obtained for ideal, noise-free data, within the bounds of MATLAB-induced numerical error.

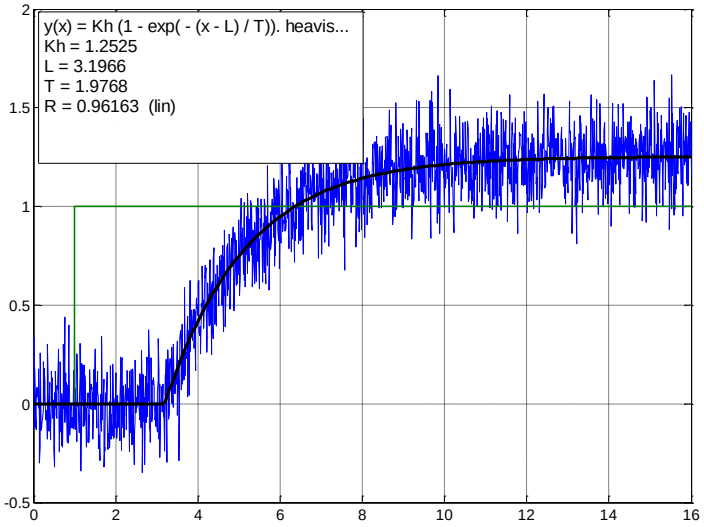


Figure 4. Example EzyFit results + noise. Green = step input applied at t = 1s; Blue = process output; Black = EzyFit model

Modified versions of equations (3) and (4) are required in the presence of an IC. However, the algorithm can fail if the initial estimate of the IC is inaccurate. This can be addressed by using an integral equation method.

Integral equation (IE) method

Equation development

In the first instance, assuming zero ICs, equation (1) can be re-written as follows:

$$TsY(s) + Y(s) = KU(s)exp(-sL) + E(s) \quad (5)$$

where *E(s)* represents any error present due to noise, nonlinearity, or other inaccuracies in the model.

An equation allowing simultaneous estimation of *K*, *T* and *L* can be obtained firstly by integrating equation (5):

$$Y(s)/s = -TY(s) + K(U(s)/s)exp(-sL) + E(s)/s \quad (6)$$

Taking inverse Laplace transforms of equation (6) yields:

$$y^{[1]}(t) = -Ty(t) + Ku^{[1]}(t - L) + e^{[1]}(t) \quad (7)$$

where the first order integrals of *y(t)* and *u(t)* are defined as follows:

$$y^{[1]}(t) = \int_0^t y(\tau)d\tau;\ u^{[1]}(t) = \int_0^t u(\tau)d\tau;\ e^{[1]}(t) = \int_0^t e(\tau)d\tau; \quad (8)$$

Equation (8) is valid for any bounded input *u(t)*. Notice that *L* remains an implicit parameter, which cannot be directly estimated. To overcome this problem, consider a step input function of height h applied at $t = 0$. The following integral holds for $t \geq L$:[14]

$$u^{[i]}(t-L) = h(t-L)^i/i! \tag{9}$$

For $t \geq L$ the estimation equation (7) can be written:

$$y^{[1]}(t) = -Ty(t) + Kh(t-L) + e^{[1]}(t) \tag{10}$$

This can be expanded as follows:

$$y^{[1]}(t) = -Ty(t) + Kht - KhL + e^{[1]}(t) \tag{11}$$

Equation (11) can be expressed in least-squares estimation form as:

$$y^{[1]}(t) = [-y(t) \quad t \quad 1]\begin{bmatrix} T \\ Kh \\ -KhL \end{bmatrix} + e^{[1]}(t) \tag{12}$$

Using more general terminology, equation (12) can be written in the form

$$\gamma = \phi(t)\theta + e(t) \tag{13}$$

where $\gamma = y^{[1]}(t)$; $\phi(t) = [-y(t) \quad t \quad 1]$; $\theta = \begin{bmatrix} T \\ Kh \\ -KhL \end{bmatrix}$; and *e(t)* is noise

Equation (13) can be written for $t = t_d+1, t_d+2, .., t_d+N$, where $d = L/T_s$ (the delay expressed in the number of sampling intervals), T_s is the sample time, and N is the total number of samples. Using all of the sampled values of $y^{[1]}(t)$, *y(t)* and *t*, equation (13) can be written:

$$\Gamma(t) = \Phi(\mathrm{t})\theta + \xi(\mathrm{t}) \tag{14}$$

$$\text{where } \Gamma = \begin{bmatrix} \gamma_1(t_d+1) \\ \gamma_2(t_d+2) \\ \gamma_3(t_d+3) \\ \vdots \\ \gamma_N(N) \end{bmatrix}; \Phi = \begin{bmatrix} \phi_1(t_d+1) \\ \phi_2(t_d+2) \\ \phi_3(t_d+3) \\ \vdots \\ \phi_N(N) \end{bmatrix} \tag{15}$$

The least squares estimation $\hat{\theta}$of θ in equation (14) is given by:

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T\Gamma \tag{16}$$

This enables *K*, *T* and *L* to be estimated from the three rows of $\hat{\theta}$:

$$\hat{T} = \hat{\theta}[1]; \hat{K} = \hat{\theta}[2]/h\ ; \hat{L} = -\hat{\theta}[3]/(Kh) \tag{17}$$

For a noise-free FOPDT process modelled by a FOPDT transfer function, exact values (i.e. 0% error) are obtained for the parameters, provided $\hat{L} \geq L$. With

data obtained from real systems, however, noise usually has a detrimental effect, resulting in biased values for the estimated parameter values.

One method of minimizing the effects of noise is to use the instrumental variable (IV) least-squares method.[12,26,27]

Although the measurement noise is assumed to be white, the integration operation performed on *y(t)* results in a coloured error term, which culminates in the parameter values being biased. To counteract this, the IV method uses a surrogate system, which uses [*K*, *T*, *L*] parameters estimated during the previous iteration, where the input is the same as the real system, but is not influenced by noise.

Algorithm implementation

Details of how to implement the method in software are presented as pseudocode in the Appendix.

Exercise 4

- Create a programme in MATLAB to implement the IE method with zero ICs.

Step response with initial condition

To analyse systems with ICs, equation (5) is modified as follows:

$$T\big(sY(s) - \mathrm{y}(0)\big) + Y(s) = KU(s)exp(-sL) + E(s) \tag{18}$$

where *y(0)* is the IC. Since there are now four unknowns rather than three, another integration is required to create an additional row in the least squares estimation matrix. Equation (12) becomes:

$$y^{[2]}(t) = \begin{bmatrix} -y^{[1]}(t) & t^2/2 & t & 1 \end{bmatrix} \begin{bmatrix} T \\ Kh \\ Ty(0) - KhL \\ KhL^2/2 \end{bmatrix} + e^{[2]}(t) \tag{19}$$

Exercise 5 (advanced)

- Starting from equation (18), derive equation (19) using the IE method.
- Develop a new MATLAB m-file to implement equation (19).

FOPDT modelling using real data

This section describes the identification of a laboratory process, the TQ Coupled Tanks apparatus shown in Figure 5. The tanks are connected by a pipe containing a valve, which allows the flow characteristic between them to be varied. Each tank also has a drain pipe at its base, with a manually operated valve that allows variable discharge into a reservoir. Liquid level is measured using a pressure sensing transducer, which results in a 0-10 V signal, corresponding to a level of 0 to 250 mm. The unit also has two pumps, each driven by a 0-10 V signal. The aim of the identification experiment was to obtain a FOPDT transfer function relating the signal applied to the tank 1 pump to the liquid level reading of tank 2.

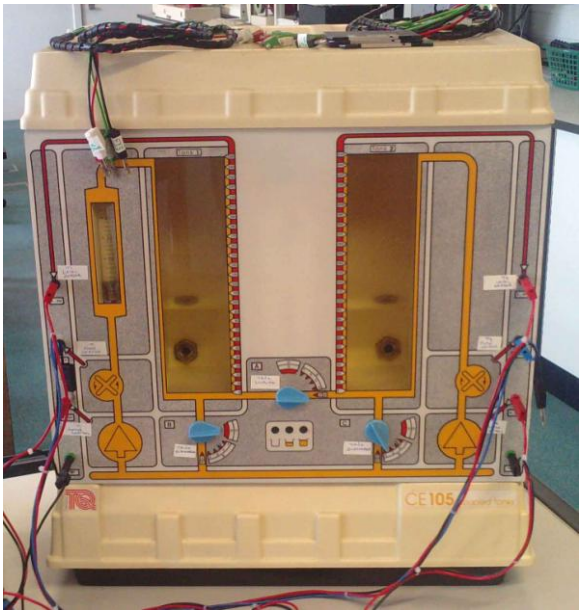


Figure 5. Coupled Tanks apparatus

The first step was to collect the raw data for a step input. Using a sample time of 1 second, 3400 data samples were collected, shown in Figure 6.

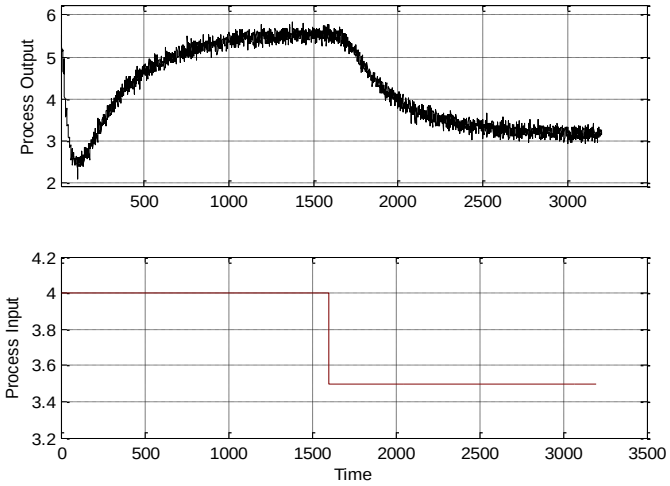


Figure 6. Raw data from two tanks apparatus

The data was cropped to remove the transient behaviour prior to the step input. The result is shown in Figure 7.

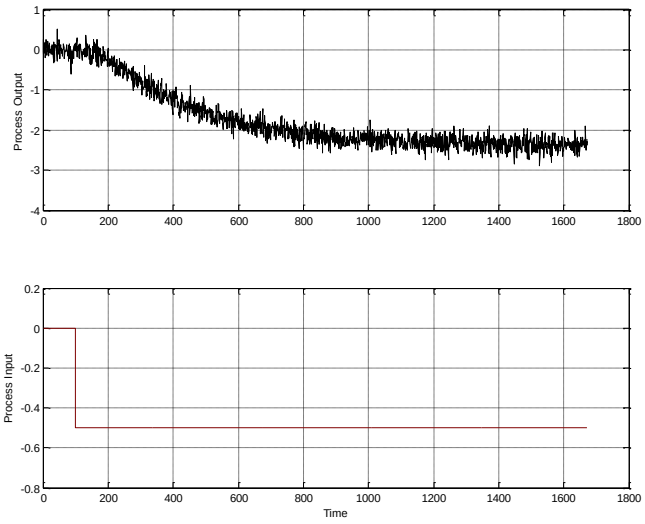


Figure 7. Cropped data, two tanks apparatus

The parameters identified by the N-M and IE methods using the data shown in Figure 7 are presented in Table 1, together with the sum of squared errors (SSE) between the experimental and model data. A visual representation of the closeness of fit of the two methods is shown in Figure 8.

Table 1
Two Tanks system, step response: FOPDT parameters and SSE

Identification method	K	T	L	SSE
N-M method	4.55	213.2	103.9	48.7
IE method	4.60	257.2	106.5	53.1

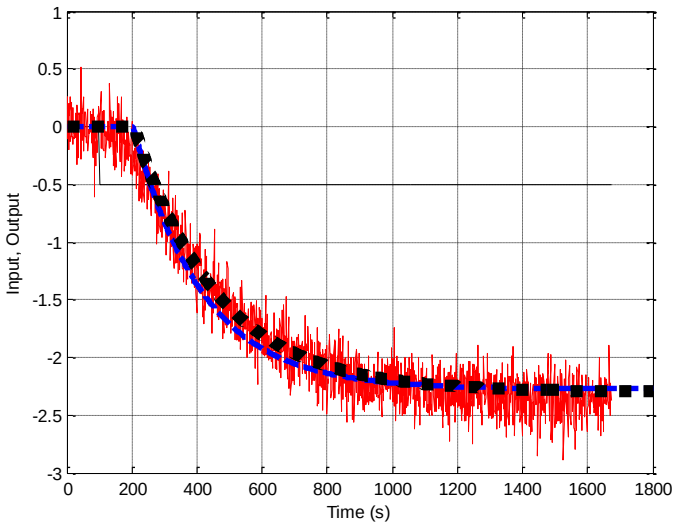


Figure 8. Two Tanks data; Dashed = N-M method; Dotted = IE method

Educational considerations

In devising these exercises, our main challenge was to embrace a selection of modern, continuous-time approaches to system identification, without necessarily demanding of our students a detailed understanding of all of the underlying mathematical concepts. However, we also wanted to maintain an appropriate level of challenge for more mathematically able students. We feel that the strategy proposed here achieves that.

The methods were originally developed using the experience gained from several recent and successful final year projects, which included in their remit the implementation of a number of system identification methods. When surveyed, students reported that learning the new methods was, in one student's words, 'a steep learning curve', but that simulation and practical studies 'significantly helped to provide a more intuitive understanding of the problem'. The importance of MATLAB and Simulink to the success of the individual projects was also highlighted. Furthermore, the methodology that scored highest amongst students in terms of their understanding was that based on step response tests. Consequently, it was decided to use step response methods, in conjunction with MATLAB, as the basis for the approach described in this paper.

During the 2017-18 academic year, elements of the paper were used in classroom and practical sessions, on an MSc Manufacturing Engineering programme. Specifically, a small cohort of twelve students was assigned exercises 1 to 4 over a three week period, as part of an introductory course in control. This involved six hours of class time: 2 hours of lectures and four hours of laboratory work. Previously, two hours was set aside for FOPDT modelling and system identification, which was largely confined to graphical methods, in a traditional classroom setting. Thus, four additional hours of computer-based laboratory time were assigned to the simulation work.

Feedback revealed that nine of the twelve students found the additional work to be both stimulating and informative, as well as improving their skills in MATLAB and Simulink. They found the most difficult part of the course to be the IE method, particularly in relation to coding the matrix algebra. It is therefore recommended that to help less confident students, a basic m-file and Simulink model are made available for the case of zero (steady state) initial conditions. This can then be modified in subsequent exercises, for example to include a non-zero initial condition.

These observations and feedback have confirmed the value of 'hands-on' work, particularly with MATLAB and Simulink, and the plan is to roll out the scheme in a second year undergraduate module in the forthcoming academic year. The authors are drawn to the conclusion that the approach reinforces the variety and excitement of control engineering, whilst improving the confidence and experience of students in their supporting studies. It is recognised that the work presented here may go beyond the requirements of some syllabi, for the type of cohort described. Nevertheless, the material provides learning opportunities for a wide range of students, as well as setting a good foundation for those wishing to follow a more academic route.

Conclusions

Three curve fitting software methods have been described, which enable the parameters of a FOPDT model to be determined from step response data. A series of exercises have also been presented to enable students to explore the methods using simulated and real data, and so improve their understanding of this important area of control engineering. Student feedback has been supportive, indicating that the hands-on approach promotes a more intuitive understanding of this type of system identification.

The work can be taken further in more advanced courses. For example, the equations can be modified to include pulse (or other) inputs and different model structures, such as second order plus dead time (SOPDT) and integral plus dead time (IPDT) models.

References

1. Y. Zhu, Multivariable System Identification For Process Control, New York: Elsevier Science Inc., 2001.
2. T. Liu, Q.-G. Wang and H.-P. Huang, “A tutorial review on process identification from step or relay feedback tests,” Journal of Process Control, vol. 23, no. 10, p. 1597–1623, 2013.
3. J. N. N. Ziegler, “Optimum Settings for Automatic Controllers,” Transactions of ASME, vol. 64, pp. 759-768, 1942.
4. R. Oldenbourg, The Dynamics of Automatic Controls, New York: ASME, 1948.
5. J. Sten, “Evaluating second-order parameters,” Instrumentation Technology, vol. 17, no. 9, pp. 39-41, 1970.
6. H. Rake, “Step response and frequency response methods,” Automatica, vol. 16, no. 5, pp. 519-526, 1980.
7. Y. Nishikawa, N. Sannomiya, T. Ohata and H. Tanka, “A method for auto-tuning of PID control parameters,” Automatica 321–332, vol. 20, p. 321–332, 1984.
8. K. Astrom and T. Hagglund, PID Controllers: Theory, Design and Tuning, Research Triangle Park: Instrument Society of America, 1995.
9. P. Balaguer, V. Alfaro and 0. Arrieta, “Second order inverse response process identification from transient step response,” ISA Transactions, vol. 50, pp. 231-238, 2011.
10. J. Arbogast, D. Cooper and R. Rice, “Graphical technique for modelling integrating processes without steady-state process data,” Chem. Eng. Commun. , vol. 194, pp. 1566-1578, 2007.
11. R. Panda, V. Vijayan, P. Deepa, D. Mananali and A. Mandal, “Parameter estimation of integrating and time delay processes using single relay feedback test,” ISA Transactions, vol. 50, pp. 529-537, 2011.
12. Q. Bi, W.-J. Cai, E.-L. Lee, Q.-G. Wang, C.-C. Hang and Y. Zhang, “Robust identification of first-order plus dead-time model from step response,” Control Engineering Practice, vol. 7, no. 1, pp. 71-77, 1999.
13. Q.-G. Wang and Y. Zhang, “Robust identification of continuous systems with dead-time from step responses,” Automatica, vol. 37, no. 3, pp. 377-390, 2001.

14. S. Ahmed, B. Huang and S. Shah, “Identification from step responses with transient initial conditions,” Journal of Process Control, vol. 18, no. 2, pp. 121-130, 2008.
15. M. Liu, Q. Wang, B. Huang and C. Hang, “Improved identification of continuous-time delay processes from piecewise step tests,” Journal of Process Control, vol. 17, no. 1, pp. 51-57, 2007.
16. S. Ahmed, “Process identification using non-ideal step inputs,” Proceedings of the 9th IFAC International Symposium on Dynamics and Control of Process Systems (DYCOPS 2010), Leuven, 5-7 July 2010.
17. S. Ahmed, C. Cox and S. Imtiaz, “Identification of integrating processes with time delay,” in 10th IFAC International Symposium on Dynamics and Control of Process Systems, Mumbai, 18-20 December 2013.
18. C. Cox, J. Tindle and K. Burn, “A comparison of software-based approaches to identifying FOPDT and SOPDT model parameters from process step response data,” Applied Mathematical Modelling, vol. 40, no. 1, pp. 100-114, 2016.
19. Y. Du, J. Tsai, H. Patil, L. Shieh and Y. Chen, “Indirect identification of continuous-time delay systems from step responses,” Applied Mathematical Modelling, vol. 35, no. 2, pp. 594-611, 2011.
20. W. Levine, The Control Handbook, Second Edition: Control System Fundamentals, Boca Raton, FL: CRC Press, 2010.
21. J. Nelder and R. Mead, “A Simplex Method for Function Minimization,” Computer Journal, vol. 7, no. 4, pp. 308-313, 1965.
22. M. H. Wright, “Nelder, Mead, and the other simplex method,” Documenta Mathematica, vol. Extra Volume: Optimization Stories, Fakultät für Mathematik, Universität Bielefeld, Germany, pp. 271-276, 2012.
23. C. Kelley, “Detection and Remediation of Stagnation in the Nelder--Mead Algorithm Using a Sufficient Decrease Condition,” SIAM J. Optim., vol. 10, no. 1, pp. 43-55, 1999.
24. The MathWorks Inc., “MathWorks Documentation: fminsearch,” 2017. [Online]. Available: https://uk.mathworks.com/help/matlab/ref/fminsearch.html. [Accessed 15 December 2017].
25. F. Moisy, “Ezyfit: A free curve fitting toolbox for Matlab,” 2014. [Online]. Available: http://www.fast.u-psud.fr/ezyfit. [Accessed 8 June 2015].
26. P. Young, “An instrumental variable method for model order identification,” Automatica , vol. 16, no. 3, pp. 281-294, 1970.
27. S. Ahmed, B. Huang and S. Shah, “Novel identification method from step response,” Control Engineering Practice, vol. 15, no. 5, pp. 545-556, 2007.

Appendix

INTEGRAL EQUATION METHOD (*t, u, y*)

1	$h = u[N] - u[1]$	// Step size
2	$T_s = t[2] - t[1]$	// Sample time
3	$Tstep = max(u[i+1] - u[i]),\ i = 1,N$	// Time at which step input occurs
4	$\boldsymbol{y1} = integral(\boldsymbol{y})$	// Numerical integration
5	$D\_guess = T_s$	// Initial guess for (L + *Tstep*) used in // iteration loop
6	$D\_guess\_z = 999$	// Previous estimate for *D_guess*
7	$D\_count = ceiling(D\_guess / T_s)$	// D_guess expressed in terms of number // of samples
8	$count = 1$	// Loop counter
9	$while \left\lvert Dguess - Dguess\_z \right\rvert > T_s$	//Convergence condition
10	*dd=ceil(Dguess/Ts);*	// Number of samples of *Dguess*
11	$\boldsymbol{t_m} = t[(dd+1)...N_t]$	// Redefine t, y, y1 data so that it starts
12	$\boldsymbol{y_m} = y[(dd+1)...N_t]$	// at sample *(dd+1)*
13	$\boldsymbol{y1_m} = y1[(dd+1)...N_t]$	
14	$\boldsymbol{phi} = [-\boldsymbol{y_m}\ \boldsymbol{t_m}\ \boldsymbol{1}]$	// Least squares estimation
15	*if count == 1*	// First loop only...
16	$\boldsymbol{psi} = \boldsymbol{phi}$	
17	*else*	
18	$\hat{y}_m = \hat{y}\left[(dd+1)\cdots N\right]$	// Estimate, calculated below
19	$\boldsymbol{psi} = [\ \hat{y}_m\ \boldsymbol{t_m}\ \boldsymbol{1}]$	
20	*end if*	
21	$\theta = (\boldsymbol{psi'}*\boldsymbol{phi})^{-1} * \boldsymbol{psi'} * \boldsymbol{y1_m}$	// Equation (11)
22	$T = \theta[1]$	// Calculate T, K and L
23	$K = \theta[2] / h$	
24	$D = -\theta[3] / (K * h)$	
25	$L = D - tstep$	
26	$if\ L < 0\ L = T_s$	// Prevents negative L causing instability
27	$G_m = Ke^{-sL} /(Ts + 1)$	// Surrogate system
28	$\hat{y} = L^{-1}[Gm(s).U(s)]$	// Inverse Laplace transform to get y // estimate
29	*Dguess_z = Dguess;*	// Update previous estimate
30	*Dguess = round(D*100)/100;*	// Accurate to within one sample period
31	*count = count + 1;*	// Update loop counter
32	*end*	// Of while loop

The MATLAB m-file implementation of the IE algorithm can be obtained by emailing the corresponding author.

Notes

- All quantities in bold are either vectors or matrices.
- The algorithm requires time, input, and output data from a step test as vector parameters (***t***, ***u*** and ***y***, respectively).
- The sampling time T_s is known and constant, and there are N samples in each of the ***t***, ***u*** and ***y*** vectors, e.g. ***u*** = *u[1], u[2], u[3], ..., u[N] etc.*

- The estimate for the time delay used in the iteration process is actually relative to time *t=0*. Hence the actual time delay is *D* – (step time). This is given the symbol *L* in the IE algorithm.
- The variable for the previous loop estimate of *D*, expressed as an integer multiple of *Ts* (*D_guess_z*), is initially set high (line 6) in order to ensure the algorithm enters the iteration loop during the first pass (line 9).