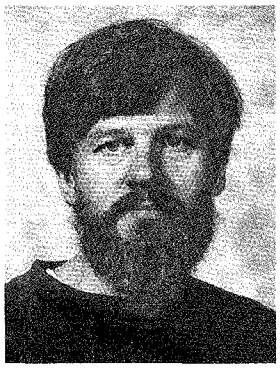
Symbolic algebra and physical-model-based control

by Peter J. Gawthrop and Donald J. Ballance

In order to achieve the best possible control of a particular system, it is clear that as much information about the system as possible should be used when designing the controller. This leads to a controller specifically tailored to the system being controlled. Unfortunately this is expensive in terms of design time and expertise and a number of approaches have been suggested to overcome this problem. The approach taken by physical-model-based control is to model the system in a generic manner and then automatically design the controller based on this model. This process requires symbolic algebra to enable the controller to be designed.

The role of symbolic algebra in the design and implementation of physical-model-based controllers is discussed and illustrated using the example of a chemical reactor. It is argued that computer algebra makes possible such a generic approach to controllers incorporating system-specific information.

Physical-model-based control has been introduced by Gawthrop, Jones, Mackenzie, Smith and Ponton[1,2,3] to provide a general methodology for building system-specific controllers which make use of system-specific information about the system to be controlled. Partially-known and nonlinear systems form two categories of systems to which such a methodology is particularly appropriate; this article concentrates on a nonlinear system for the purposes of illustration. Precisely because each control algorithm has to be specifically constructed for each system, it is essential to provide software to help in this process. The use of *computer algebra* is a vital part of such software.



Peter Gawthrop

Donald Ballance

The purpose of this article is to present the computer algebra aspects of physical-model-based control; the control theory aspects are covered in detail elsewhere.

Physical-model-based control

The structure of model-based observer control[1,2,3] is outlined in Fig. 1. This separation into controller and observer is, of course, a standard paradigm and, as such, provides a useful starting point for model-based observer control. The different aspects of the approach are outlined below:

- The overall model-based observer controller is referred to as the *compensator*. With reference to Fig. 1, the compensator is a dynamic system with two (possibly vector) inputs—the system output y_s and the corres-

ponding set-point w—and one (possibly vector) output—the system input u. It is conceptually divided into three parts: *model*, *observer feedback* and *controller feedback*.

- The block labelled 'model' is a dynamic simulation model of the system to be controlled (labelled 'system'). This system may be nonlinear.
- The *model* has the same control input u as the system. Other system inputs which can be measured could also be applied to the model in the same way to give feedforward compensation.
- The measured outputs y_s of the system are compared with the corresponding model outputs y to create a model error $e_m = y_s - y$.
- Additional inputs u_i are provided to this model—typically one for each state. The model has *observer feedback* applied to these additional inputs in such a way as to drive the model error e_m to zero.
- Additional outputs y_i are provided associated with *virtual sensors*. The controller feedback generates the control signal u using the virtual sensor signals y_i together with the set point w. Its purpose is to make the *model* behave in a desired way.
- If the model-based observer is working well, these internal signals will be the same or close to the corresponding (but possibly unmeasurable) internal signals generated within the system itself. Thus the *system* will be driven to behave in the same way as the (controlled) model.

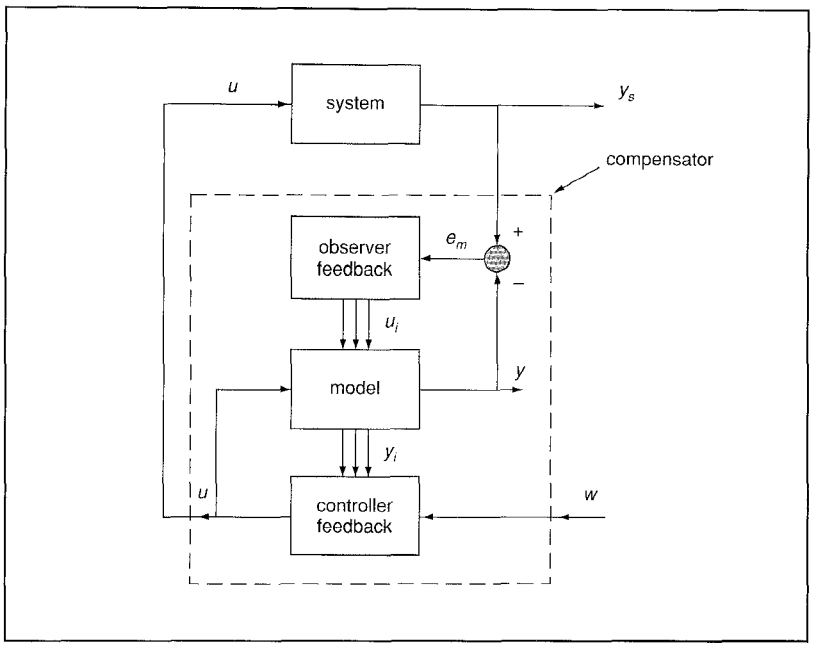


Fig. 1 Model-based observer control

Symbolic computation

The system model is at the heart of the physical-model-based controller of Fig. 1. Although this use of a physical system model has important advantages (see References 1-3 for details) it has the disadvantage that each controller is system specific; each controller is customised for a particular process. For this reason, it is essential to have a generic methodology, supported by software, for the design and implementations of these customised controllers. The use of *computer algebra* is a vital part of such software.

The following subsections indicate this generic methodology for each part of the control structure of Fig. 1.

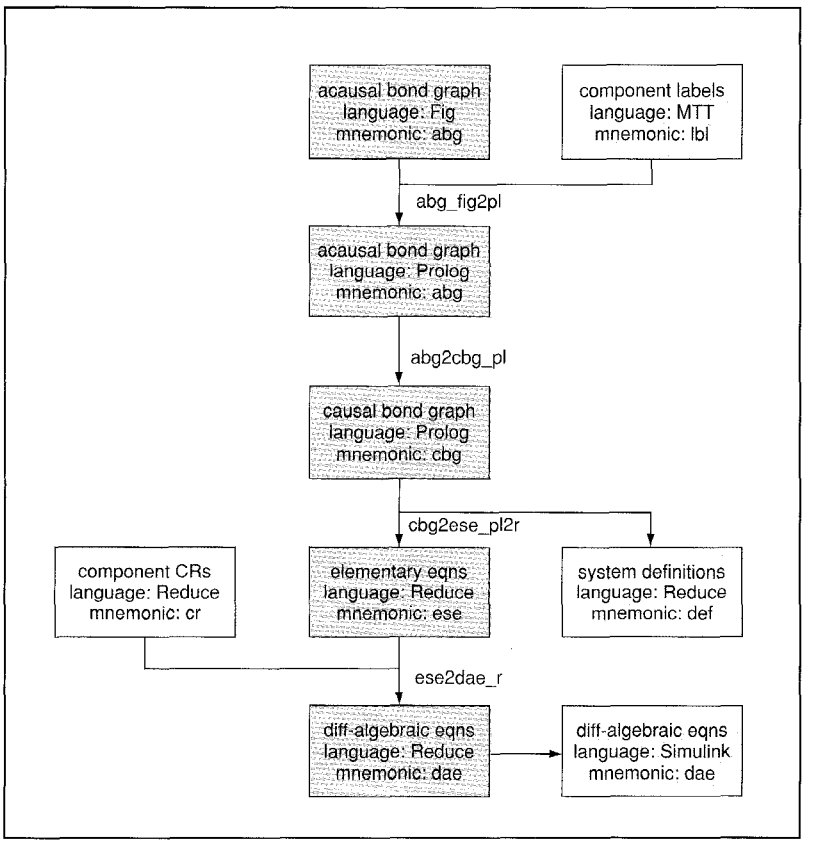


Fig. 2 Model transformation tools[5]

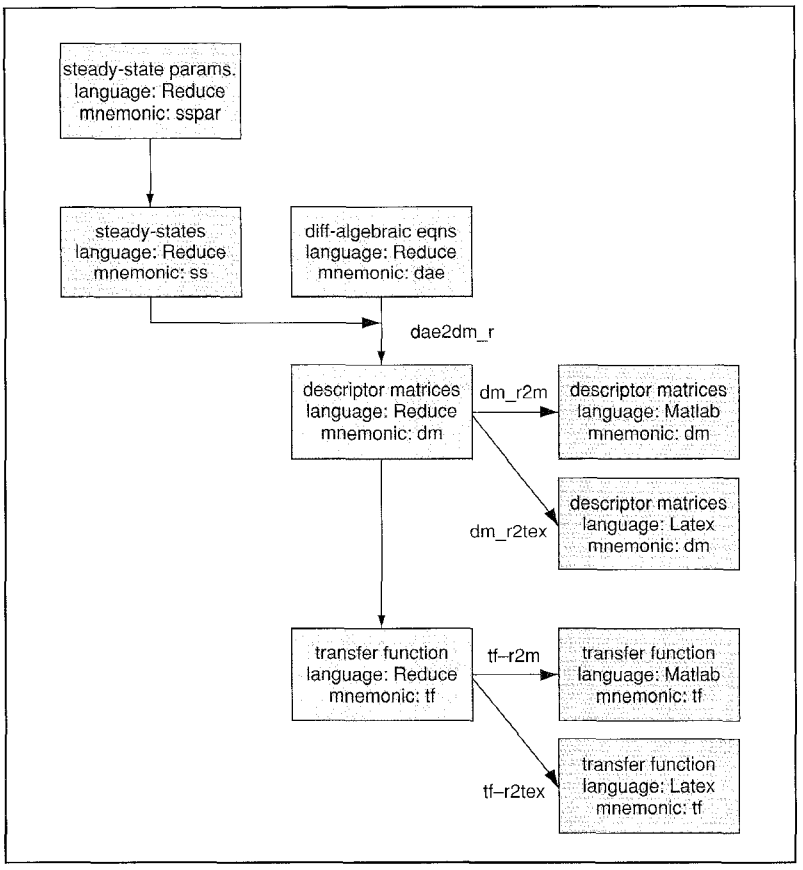


Fig. 3 More model transformation tools[1]

The model

To be able to apply computer algebra methods to design such control systems, a *representation* of the model is required which is on the one hand accessible to the control engineer and on the other is concise, precise and unambiguous so that it can be interpreted by computer. The *bond graph* representation (see, for example, References 4 and 2) has been chosen for this purpose and a corresponding set of model transformation tools (MTT) has been written. This set of tools has been implemented with the UNIX environment making use of Prolog, Reduce and standard UNIX/GNU tools.

The structure of part of the toolbox appears in Figs. 2-4. Each box represents a *representation* with three attributes: the name of the representation, the language in which the representation is expressed and a mnemonic; it is *implemented* as a text file. Each arrow represents a *transformation* between representations; it is *implemented* as a UNIX Bourne shell script using an appropriate language. The corresponding dependency tree is *implemented* as a UNIX make file, which automates the process of keeping the various different representation up to date.

It is useful to think of bond graphs as providing a high-level language for describing dynamic systems (whether mechanical, electrical, hydraulic or chemical). Within the context of Fig. 1, the resulting system description can then be '*compiled*' into a *symbolic* form suitable for:

- observer design
- controller design implementation
- system and controller simulation.

In certain cases, the model can be described by a (possibly) nonlinear state equation of the form:

$$\dot{x} = f(x,u) \quad (1)$$

$$y = g(x) \quad (2)$$

The form can also be automatically generated from the system description but can be an unwieldy object to use directly when divorced from its physical meaning. Hence we prefer to do our design directly in the physical domain rather than using eqns. 1 and 2.

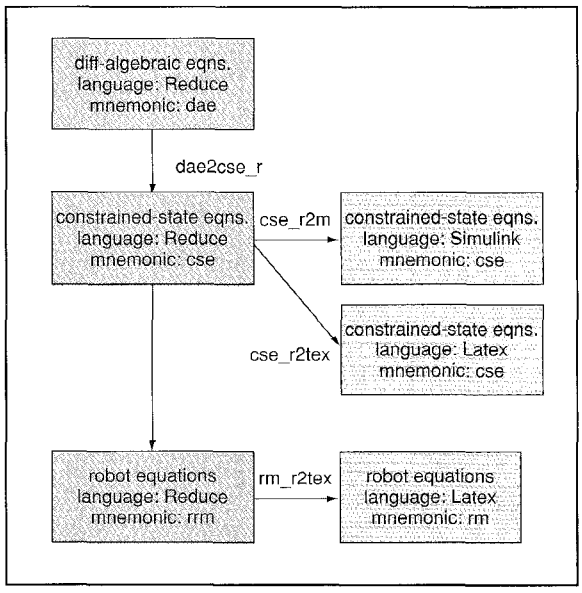


Fig. 4 More model transformation tools[3]

Chemical reactor example

An example of a chemical reactor is now used to illustrate the main points of the modelling and design procedure. The schematic diagram appears in Fig. 5; the reactor has two reaction mechanisms: A → B → C and 2A → D. The reactor mass inflow and outflow f_r are identical. q represents the heat inflow to the reactor. The two

system outputs are taken as c_b, the concentration of species b, and t_r, the reactor temperature. The controlled inputs are f_r, the reactor flow, and q the heat input. Other inputs are t_0, the inflow temperature and c_0, the inflow concentration of substance A.

The corresponding bond graph appears in Fig. 6. The accumulation of the components A and B and of enthalpy are represented by the three C components labelled *ma*, *mb* and *h_r*. The reactor flow is represented by the SS component labelled *f_r* and modulates the R components labelled *rfa* and *rfb*. The reaction kinetics A → B, B → C and 2A → D are represented by the R components labelled AB, BC and AD, respectively.

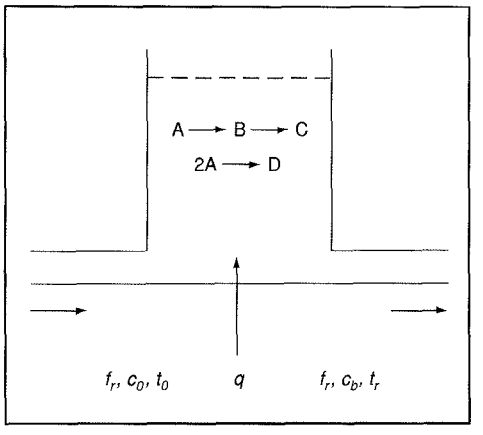


Fig. 5 Chemical reactor

The detailed interpretation of such a bond graph is discussed in detail elsewhere (see, for example, Reference 2. The purpose of this section is to indicate how such a physical model description can help in making control design decisions:

1 The art of modelling is very much the art of approximation, choosing which parts of the system to neglect in the model. Bond graphs provide a useful way of helping with such approximation. Here it has been assumed that the time constants associated with the reactor jacket are fast enough to be neglected. This simplification can be done directly from the appropriate bond graph, but the details are omitted here.

2 The control input q and output t_r are *colocated*; they are covariables of the SS element labelled t_r. The implication of this is that control of t_r using q is relatively easy. For this reason we assume perfect control of t_r and concentrate on the more difficult problem of control of c_b using f_r. This ideal control can be represented by inverting the causality of the SS element labelled t_r to give t_r as an input and q as an output—the heat input required to achieve the desired t_r.

With this approximation, and assuming constant t_r, the system equations relevant to the (isothermal) control problem become:

$$\dot{x}_1 = x_1^2\varepsilon_3 k_3 - x_1(\varepsilon_1 k_1 + u) + c_0 u \quad (3)$$

$$\dot{x}_2 = x_1\varepsilon_1 k_1 - x_2(\varepsilon_2 k_2 + u) \quad (4)$$

$$y = x_2 \quad (5)$$

where

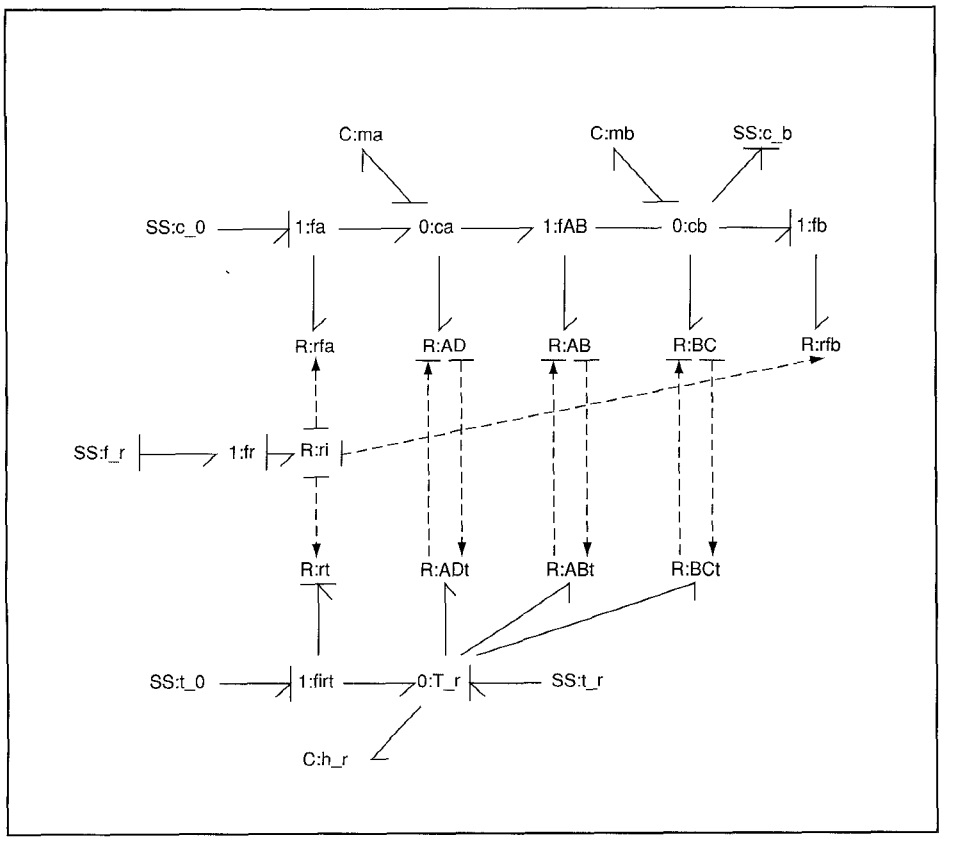


Fig. 6 Chemical reactor: bond graph

$$\varepsilon_i = \exp(q_i/T) \quad (6)$$

These are nonlinear equations.

3 The control input f_r and output c_b are not colocated, and so there is potentially a non-trivial control problem associated with this loop. Furthermore, direct manipulation of the bond graph to obtain the system inverse reveals that the inverse system has first-order zero dynamics whereas the system has second-order dynamics. This purely qualitative information is obtained directly from the bond graph.

Adding the numerical information to the symbolic expressions obtained from Fig. 6 gives the graphs of Fig. 7. The first plot shows the steady-state output y_s against f_s; it reveals that not all steady-state concentrations are achievable, and those that are may have two corresponding flows. The second and third show the poles and zero, respectively, of the *linearised* system also plotted against f_s; these plots show that, whereas the system is stable for all flows, the zero is in the right-half plane flow for low flows and migrates into the left-half plane for larger flows.

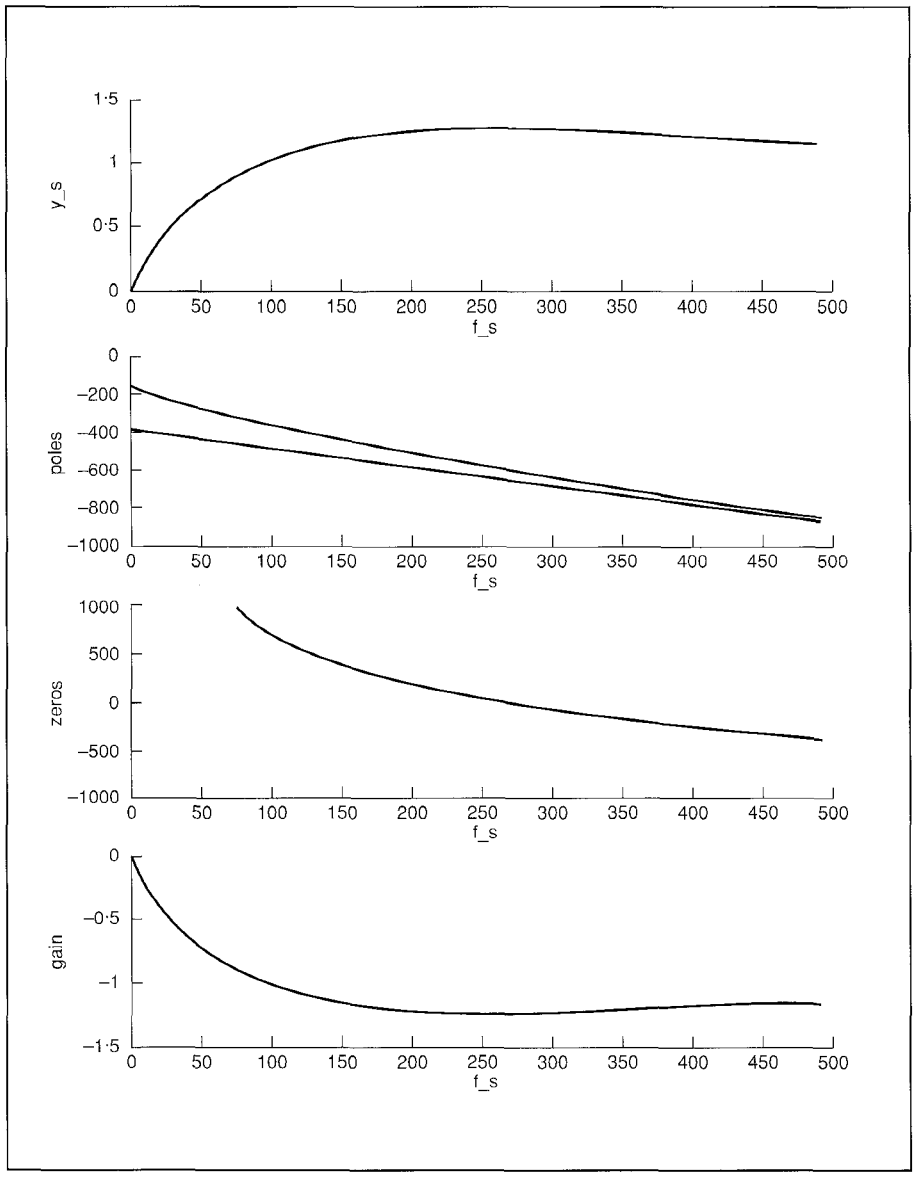
As the system is nonlinear with unstable zero dynamics, nonlinear generalised predictive control is appropriate.[6]

Observer design

Unlike the linear case, there is no general theory of state estimation for nonlinear systems. However, following the linear case, it is natural to embed the system model within the observer feedback loop of Fig. 1.

For example, observers can use linear feedback around a nonlinear model. In view of eqn. 1, such observers can

Fig. 7 Model numerical properties



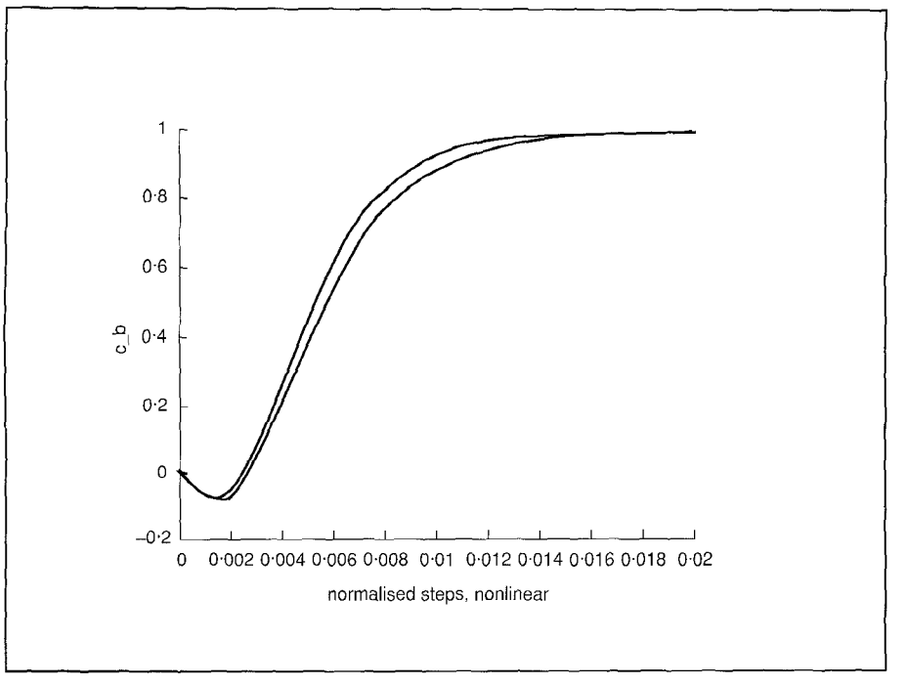


Fig. 8 Nonlinear control

be rewritten as

$$\dot{\hat{x}} = f(\hat{x},u) + L(\hat{x})e \tag{7}$$
$$\hat{y} = g(\hat{x}) \tag{8}$$
$$e = \hat{y} - y \tag{9}$$

Unlike the linear case, the stability of such an observer is not guaranteed in general and its design is non-trivial.

Once again, the approach taken here is to use *physical-model-based* observers and so build physical intuition into the design of such observers. An experimental case study is reported by Costello and Gawthrop.[5] Here, a few brief remarks illustrate some features of the approach:

1 The bond graph of Fig. 6 shows two inputs—t_0, the inflow temperature, and c_0, the inflow concentration of substance A—which are not used for control. In this form, these two additional drive the model, and hence the compensator and thus provide a form of *feedforward* control.
2 If, on the other hand, these inputs were not measured, a model representing the corresponding uncertainty can easily be incorporated. The unknown inputs then become states of the observer.

Controller design

Recent research has shown that the continuous-time generalised predictive control can be extended to cover certain non-linear systems. The algorithm requires the symbolic calculation of (possibly high-order) Lie derivatives. These are hard to generate by hand, but a tool has been added to MTT which uses symbolic computation to automatically generate the computer code necessary to implement GPC.

In particular, the algorithm requires the computation of:

$$Y_{N_y}(t) = \boldsymbol{O}(\boldsymbol{x}(t), \boldsymbol{U}_{N_u}(t)) \tag{10}$$

where $Y_{N_y}(t)$ is a column vector $(n_y(N_y + 1) \times 1)$: of output derivatives:

$$Y_{N_y}(t) = \begin{pmatrix} y \\ y^{[1]} \\ y^{[2]} \\ y^{[N_y]} \end{pmatrix} \tag{11}$$

where $^{[i]}$ indicates the ith derivative with respect to time. $U_{N_u}(t)$ is a column vector $(n_u(N_u + 1) \times 1)$: of input derivatives:

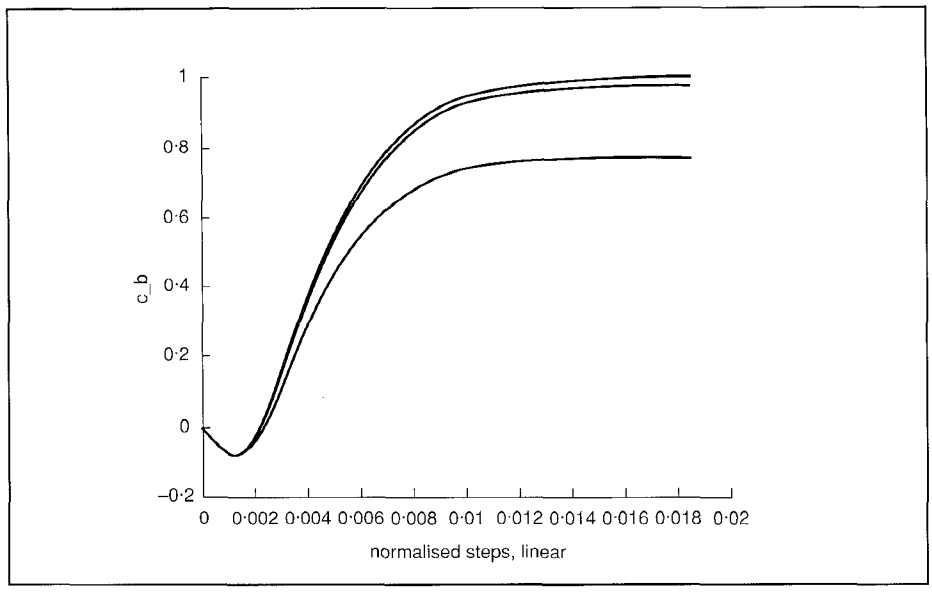
$$U_{N_u}(t) = \begin{pmatrix} u \\ u^{[1]} \\ u^{[2]} \\ u^{[N_u]} \end{pmatrix} \tag{12}$$

For example, the first four elements of Y are:

$$Y_0 = x_2$$
$$Y_1 = x_1\varepsilon_1 k_1 - x_2(\varepsilon_2 k_2 + u)$$
$$Y_2 = -x_1^2\varepsilon_1\varepsilon_3 k_1 k_3 + x_1\varepsilon_1 k_1(-\varepsilon_1 k_1 - \varepsilon_2 k_2 - 2u) + x_2(\varepsilon_2^2 k_2^2 + 2\varepsilon_2 k_2 u + u^2) + c_0\varepsilon_1 k_1 u$$
$$Y_3 = 2x_1^3\varepsilon_1\varepsilon_3^2 k_1 k_3^2 + x_1^2\varepsilon_1\varepsilon_3 k_1 k_3(3\varepsilon_1 k_1 + \varepsilon_2 k_2 + 4u) + x_1\varepsilon_1 k_1(-2c_0\varepsilon_3 k_3 u + \varepsilon_1^2 k_1^2 + \varepsilon_1\varepsilon_2 k_1 k_2 + 3\varepsilon_1 k_1 u + \varepsilon_2^2 k_2^2 + 3\varepsilon_2 k_2 u + 3u^2) + x_2(-\varepsilon_2^3 k_2^3 - 3\varepsilon_2^2 k_2^2 u - 3\varepsilon_2 k_2 u^2 - u^3) + c_0\varepsilon_1 k_1 u(-\varepsilon_1 k_1 - \varepsilon_2 k_2 - 2u)$$

In addition, the partial derivatives of $O(x,U)$ $\partial^2 O/\partial U^2$, $\partial^2 O/\partial U \partial x$ are needed. Except for the simplest of systems,

Fig. 9 Linear control



these functions are hard to generate by hand. Our approach is to generate the required functions symbolically in the form of Matlab functions.

Simulation

Simulation is often necessary to compare the performance of controllers acting on a nonlinear system. Once again, symbolic computation can help here in generating simulation code for both system and compensator. Here, two compensators are compared. Each uses identical model and observer feedback, but they differ in that one uses nonlinear GPC and the other is based on the linearisation of the nonlinear system about a form $f_r = 90$. The two controllers are compared with steps in demand concentration, about this equilibrium, of 0·002, 0·02 and 0·2 moles kg^{-1}. In each case, the outputs for the three steps are normalised and superimposed on the same graph.

Fig. 8 corresponds to the nonlinear control. The three normalised step responses are similar indicating that the nonlinear controller has approximately linearised the closed-loop system. Note that *exact* linearisation is not possible due to the unstable inverse dynamics which cause the initial negative responses.

Fig. 9 corresponds to the linear controller. The three normalised step responses are different indicating that the linear controller does not have as good performance as the nonlinear controller.

Conclusion

We believe that automating the process of modelling—via the use of symbolic computation—gives control designers the generic techniques needed to design customised controllers (incorporating a physical system model) for specific systems, thus enabling physical system understanding to be incorporated into the design.

We emphasise the following points:

- It is computational symbolic algebra that makes this approach to physical-model-based observer controllers possible.
- Reduce was used as a typical computational symbolic algebra package, but there is nothing special about its capabilities; other packages—for example Maple, Mathematic or MuPad—could equally well be used. Indeed, the interactive user interface to Reduce (or any other package) is not needed in this context: Reduce is used as a symbolic algebra programming language.
- Symbolic manipulation is not just to do with manipulating equations: it is also to do with manipulating higher-level structures, such as bond graphs.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council through grant number GR/H41942.

References

1 GAWTHROP, P. J., JONES, R. W., and MACKENZIE, S. A.: 'Bond graph based control: a process engineering example', American Control Conference, 1992
2 GAWTHROP, P. J., and SMITH, L. P. S.: 'Metamodelling: bond graphs and dynamic systems' (Prentice Hall, Hemel Hempstead, Herts., UK, 1996)
3 GAWTHROP, P. J., and PONTON, J. W.: 'Improved control using dynamic process models', *Chemical Engineering Research and Design*, **74(A1)**, pp.63-69
4 KARNOPP, D. C., MARGOLIS, D. L., and ROSENBERG, R. C.: 'System dynamics: a unified approach' (John Wiley, 1990)
5 COSTELLO, D. J., and GAWTHROP, P. J.: 'Physical model-based control: experiments with a stirred-tank heater', Technical Report CSC-95003, Glasgow University Centre for Systems and Control, 1995
6 GAWTHROP, P. J., and SILLER-ALCALA, I. I.: 'Nonlinear generalised predictive control', Technical Report CSC-95031, Glasgow University Centre for Systems and Control, 1995

The authors are with the Centre for Systems and Control, Department of Mechanical Engineering, University of Glasgow, Glasgow G12 8QE. Prof. Gawthrop is an IEE Fellow.