Automatic Tuning of Commercial PID Controllers for Single-Loop and Multiloop Applications

Peter J. Gawthrop and Panos E. Nomikos

ABSTRACT: Certain continuous-time self-tuning algorithms are shown to be capable of generating tuning parameters for commercial proportional-integral-derivative (PID) controllers. They are also shown to be capable of generating a feedforward signal that decouples the disturbance from the interaction from adjacent loops in a multivariable situation. The approach is verified using a standard Atari personal computer supervising Eurotherm 820 PID controllers. Two systems are used to test the method: a laboratory water tank system under level control and an industrial multizone plastics extruder under temperature control. The results are encouraging.

Introduction

Most industrial processes are controlled using proportional-integral-derivative (PID) controllers. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity, which allows process engineers to operate them in a simple and straightforward manner. To implement such a controller, three parameters must be determined for the given process: proportional gain, integral time constant, and derivative time constant. Usually, process engineers must tune PID controllers manually—an operation which, if done diligently, can take considerable time. The problem is exacerbated by interactions with other loops.

To overcome this problem, several methods have been developed that automatically tune PID controllers. The most well-known method is that of Ziegler and Nichols developed in 1942–1943 [1], [2]. Their method determines the parameters by observing the gain at which the plant becomes oscillatory and the frequency of this oscillation. A more useful extension of the method allows the determination of the parameters from the observation of the open-loop response of the plant to a step input change. A similar method has been developed by Hazebroek and Van Der Waerden [3]. Several other similar simple methods have been developed since, which automatically generate a special input to the process, and by observing its response, they determine the PID parameters (e.g., [4]–[6]). The simplicity of all these methods has made possible the development by the process-controllers industry of a wide range of such "autotuning" instruments that recently have appeared in the market [7]. Other approaches are given in [8]–[11].

Apart from the "initial tuning" problem, it is also true that the PID algorithm, by being so simple, cannot always effectively control systems with changing parameters, and so may need frequent manual on-line retuning. As such "difficult" systems are commonplace in real industrial plants, the need for algorithms that could cope with them encouraged the development of the adaptive control methods.

Although the PID algorithm itself is essentially simple, commercial implementations of PID controllers contain a multitude of additional features that embody the experience of many years of application. For example, integral windup prevention, integral preload, derivative limiting, and bumpless transfer are all features essential to safe and effective operation in practice. Therefore, it is prudent to retain these features when introducing self-tuning; basically sound self-tuning methods have failed because these vital details have not been applied.

To take account of these points, the approach taken here is to use a standard commercial PID controller (a Eurotherm 820) and use a continuous-time self-tuning PID controller [12]–[14] to generate its *parameters* as opposed to using the self-tuning PID controller to generate a *control signal.* A similar approach is taken by Koivo and Sorvari [15]. The self-tuning PID controller is implemented on an Atari personal computer and communicates via an RS-232 port using the Eurotherm communications protocol. This approach has several advantages.

First, the actual control signal is generated by a standard industrial PID controller which, if well tuned, is a well-known and trusted instrument giving guaranteed high performance for a given range of operating conditions. It is when that range changes that the tuner intervenes and readjusts the instrument to the new requirements.

Second, large processes usually require a large number of smaller control loops for which the performance requirements are such that a standard PID controller is sufficient. In such case, it is more reasonable to introduce few self-tuners on a higher supervisory level that will adjust only several controllers each, especially in the more critical loops. Thus, the self-tuners will become part of a more complete hierarchical structure. Moreover, in cases where PID controllers have been installed previously, it is easier to add a set of supervisory controllers instead of rebuilding the whole structure.

Third, this approach improves the safety of the plant. If, for any reason, the adaptive algorithm drifted to unstable conditions, the PID can be dropped back to a set of safe parameters, which will guarantee that the plant will remain stable. Evidently, this is more acceptable than dropping back to manual control, which is the commonly used alternative.

Fourth, the start-up of the plant can be improved greatly. Instead of injecting "artificial" perturbations into the plant to allow the tuner to tune properly, we could simply let the standard PID drive the plant until the tuner settles to an acceptable set of parameters.

The standard PID algorithm is a single-loop controller. However, most of the real industrial processes are multiloop, with several inputs that affect a number of outputs.

Peter J. Gawthrop is with the Department of Mechanical Engineering, the Glasgow University, G12 8QQ, United Kingdom. Panos E. Nomikos was with the School of Engineering and Applied Sciences, University of Sussex, Falmer, Brighton, Sussex, BN1 9QT, United Kingdom, and is now serving with the Greek armed forces.

0272-1708/90/0100-0034 $01.00 © 1990 IEEE

Changes in any of the process loops usually affect several loops; in such cases, the PID algorithms treat those interaction disturbances as any other disturbance. Therefore, the need has arisen for another algorithm that will be able to tune standard PID controllers and, at the same time, provide a means of decoupling any disturbance caused by interacting loops. Such an algorithm will be presented in this paper.

The problem in the case of multiloop processes is to decouple the interaction disturbance. Several methods have been developed for such cases; for example, Borison [16] and Koivo [17] are two of the most well known. However, in the majority of those algorithms, the design is based on a single-controller representation, where the system is represented in matrix transfer-function form, and the controller is built accordingly. Arrays of process variables are polled at each iteration, and multidimensional arrays of parameters are updated to derive the array of the control signals. Such an approach requires excessive computing power, a fact that restricts the efficiency of those controllers.

It can be argued [14], [18], [19] that a more effective alternative approach is to split the overall system representation into smaller single-loop units, with the interaction represented as additional transfer-function terms, and to design individual controllers for each loop that will be fed with information from the adjacent loops to decouple the interaction disturbance. Such a design will result in several multiple-input/single-output controllers, which individually require less computing power than before; moreover, it will be easier to adapt the design according to structural plant modifications.

A practical problem addressed and solved here is to provide for some means of decoupling the interaction, in addition to the tuning of the PID parameters. The difficulty arises from the fact that the control signal is not generated from the tuner, which could identify the interaction and adjust such a signal to decouple it, but is generated by an external instrument which, by itself, cannot take such action. The solution is to calculate a feedforward signal, which will be targeted to cancel the effects of interactions, and add it to the control signal generated by the PID using the standard "bias" facility provided in the Eurotherm 820.

Continuous-Time Self-Tuning PID Control Algorithm

This section provides a brief overview of the continuous-time self-tuning control approach; more details are given in [13], [14], and [19]. In the context of adaptive control, the controller design can be performed in either the discrete- or continuous-time domain. Although most of the contemporary algorithms are based on the former design, it has been argued in [20]–[23] that this approach causes several problems, such as migration into the instability region of the zeros of systems with relative order higher than 2, and the fact that important system dynamics apparent in a continuous-time representation may be lost in a discrete design. In particular, it is easier to relate the effects of proportional and integral action to the process dynamics if the self-tuning controller design is based on a continuous-time formulation than when it is based on a discrete-time formulation. It has been shown in [14] and [24] that it is possible to design the adaptive controller design in a continuous-time context and then discretize the algorithm for computer implementation. This is in contrast to the discrete design methods, where the discretization takes place *before* the controller design.

The continuous-time approach is summarized in the remainder of this section. More details appear in [13], [14], and [24]. The continuous-time design is based on a system model of the form shown, where $y(s)$, $u(s)$, and $z(s)$ are the Laplace transforms of the system output, input, and noise disturbance, respectively, and $A(s)$, $B(s)$, and $C(s)$ are polynomials in the s domain.

$$A(s)\,y(s) = B(s)\,u(s) + C(s)\,z(s) \qquad (1)$$

Polynomial $C(s)$ is a design polynomial, whereas $A(s)$ and $B(s)$ represent the system dynamics, which are either known (nonadaptive design) or unknown (adaptive design).

For processes with high relative order [deg (A) much greater than deg (B)], we may introduce an unrealizable quantity that will emulate the effect of taking pure derivatives as

$$\phi(s) = P(s)\,y(s) \qquad (2)$$

$$\deg(P) = \deg(A) - \deg(B) \qquad (3)$$

We call this unrealizable quantity $\phi(s)$ the *predictive* term. Since it is unrealizable, it can be shown that we may approximate it by introducing the quantity

$$\phi^*(s) = \frac{F(s)}{C(s)}\,y(s) + \frac{G(s)}{C(s)}\,u(s) \qquad (4)$$

The polynomials $F(s)$ and $G(s)$ are taken (in the nonadaptive case) from

$$\frac{P(s)\,C(s)}{A(s)} = E(s) + \frac{F(s)}{A(s)} \qquad (5)$$

$$G(s) = E(s)\,B(s) \qquad (6)$$

Polynomial $Q(s)$ is another design polynomial, called the *detuning* polynomial. For the moment, we assume that $Q(s) = 0$.

In the adaptive case, $F(s)$ and $G(s)$ can be estimated directly using a standard least-squares estimator if Eq. (4) is expressed in the form shown, where $\mathbf{X}^T$ is the data vector and $\boldsymbol{\theta}$ the vector of the corresponding parameters.

$$\phi^*(s) = \mathbf{X}^T\boldsymbol{\theta} \qquad (7)$$

$$\mathbf{X} = [\mathbf{X}^y, \mathbf{X}^u] \qquad (8)$$

$$\mathbf{X}^y = [1, s, \ldots, s^n]\,y(s)/C(s) \qquad (9)$$

$$\mathbf{X}^u = [1, s, \ldots, s^n]\,u(s)/C(s) \qquad (10)$$

Modified System Model

The preceding algorithm has been modified to allow on-line estimation of PID parameters [13]. The motive for such a modification was the need to introduce integral action into the algorithm to remove offset errors from the steady-state value of the controlled process. Some researchers have tried to introduce such integral action by forcing an integrator into the controller (e.g., [25], [26]). The approach used here is the reverse: the system is modeled to include an offset term, and the corresponding controller is found to have integral action. This approach was also adopted by Scattolini [27].

Modification of the algorithm is based mainly on the assumption that concerns the disturbance model. It is argued that real disturbances do not usually have a zero mean; such a disturbance may drift with time. We can model such a process by the following equations, where $z(s)$ is a zero-mean disturbance.

$$z'(s) = [(1 + cs)/s]\,z(s) \qquad (11)$$

$$A'(s)\,y(s) = B'(s)\,u(s) + C'(s)\,z'(s) \qquad (12)$$

For Eq. (12) to correspond with the original system model [Eq. (1)], the following relations must hold:

$$A(s) = sA'(s) \qquad (13a)$$

$$B(s) = sB'(s) \qquad (13b)$$

$$C(s) = (1 + cs)\,C'(s) \qquad (13c)$$

The integrator in the disturbance model, therefore, implies that

$$A(0) = B(0) = 0 \qquad (14a)$$

$$C(0) \neq 0 \qquad (14b)$$

Zero-Gain Predictor

The offset problem may be the result of not only a nonzero-mean disturbance, but also the incorrect choice of controller parameters. To overcome this, we simply have to constrain the design polynomials to

$$P(0) = 1 \quad \text{and} \quad Q(0) = 0 \qquad (15)$$

This means that $P(s)$ can be expressed as

$$P(s) = 1 + sP_0(s) \qquad (16)$$

$$P_0(s) = p_1 + p_2 s + \cdots + p_n s^{n-1}; \quad n = \deg(P) \qquad (17)$$

Therefore, $\phi(s)$ becomes

$$\phi(s) = y(s) + \phi_0(s) \qquad (18)$$

$$\phi_0(s) = sP_0(s)\, y(s) \qquad (19)$$

Equation (5) is now modified to

$$[(P(s) - 1)\, C(s)]/A(s) = [sP_0(s)\, C(s)]/A(s) = E'(s) + F'(s)/A(s) \qquad (20)$$

Since $A(0) = 0$, it follows that $F'(0) = 0$, and also by defining $G'(s) = B(s)\, E'(s)$ we will have the following, where $F_0(s)$ and $G_0(s)$ have a form similar to $P_0(s)$ in Eq. (17).

$$F'(s) = sF_0(s); \quad G'(s) = sG_0(s) \qquad (21)$$

In a manner similar to Eq. (4), it can be shown that the predictor $\phi_0(s)$ is given by

$$\phi_0^*(s) = sF_0(s)\, y(s)/C(s) + sG_0(s)\, u(s)/C(s) \qquad (22)$$

Since each of the two filters in the predictor has zero steady-state gain, this is termed the "zero-gain predictor."

If we wish to make the prediction $\phi_0^*(s)$ follow the set point, then a suitable control law will follow from

$$\phi_0^*(s) = w(s) - y(s) \qquad (23)$$

to become

$$u(s) = [C(s)/G_0(s)][[w(s) - y(s)]/s - [F_0(s)/C(s)]\, y(s)] \qquad (24)$$

This controller has the desirable integral action and will be used as the basis of the PI and PID controllers to be examined.

In [13], it is also proved that this zero-gain predictor can be modeled in a state-space representation very similar to that explained earlier and is, therefore, suitable for the least-squares estimator.

Although this form has integral action, it is not yet readily transformable into a PID-like control law. Another alternative would be to choose $C(s)$ to be such that $C(0) = 1$, i.e.,

$$C(s) = 1 + sC_0(s) \qquad (25)$$

Adding the term $sC_0(s)/C(s)\, y(s)$ to each side of Eq. (22), we find that

$$\phi_0^{c*}(s) = sF_0^c(s)\, y(s)/C(s) + sG_0(s)\, u(s)/C(s) \qquad (26)$$

It is shown in [13] that the quantity $\phi_0^{c*}(s)$ can be regarded as the prediction of

$$\phi^c(s) = w(s) - y(s)/C(s) \qquad (27)$$

Substituting for this alternative zero-gain predictor in a relation similar to Eq. (23), we will find a control law

$$u(s) = G_0(s)^{-1}[[w(s) - y(s)]/s + C_0(s)\, w(s) - F_0^c(s)\, y(s)] \qquad (28)$$

The polynomials $F_0^c(s)$ and $G_0^c(s)$ are given by expressions very similar to Eq. (20), and again it can be shown that this predictor as well can be expressed in a form suitable for estimation.

PI and PID Controllers

In the previous section, we have seen how we can introduce the desirable integral action in the control law [Eq. (28)]. However, this law is not yet similar to a standard PID controller. To achieve this, we need to restrict the system polynomials. In particular, the assumptions [Eq. (12)] coupled with the conditions

$$\deg(A') = 1 \quad \text{and} \quad \deg(B') = 0 \qquad (29)$$

will be shown to give a PI controller, and coupled with

$$\deg(A') = 2 \quad \text{and} \quad \deg(B') = 0 \qquad (30)$$

will be shown to give a PID controller. It is interesting to notice that this structure is also in accordance with the "second-order system with delay" structure used by most of the researchers referenced in the introduction for their adaptive PID algorithms.

It should be pointed out that Eqs. (29) and (30) do not restrict the class of the real systems (to which this algorithm can be applied) to be of second order. They restrict only the "nominal" system model. The real plant might be (and usually is) of higher order, and the difference can be modeled (or approximated) by some "neglected dynamics" terms. The importance of the terms can be investigated using the techniques presented in [14], where a method of investigating the robustness of the approach in the face of neglected high-frequency dynamics is analyzed. If the terms are found to destabilize the performance of the algorithm, then appropriate corrective action should be taken by modifying the controller design polynomials.

Assumptions (29) combined with the above-described predictor structure will give the following, where b, c, e, and f^c are real numbers.

$$C(s) = 1 + cs \qquad (31a)$$

$$E'(s) = e \qquad (31b)$$

$$F_0^c(s) = f^c \qquad (31c)$$

$$G_0(s) = eb \qquad (31d)$$

The control law [Eq. (28)] will then become

$$u(s) = (eb)^{-1}[[w(s) - y(s)]/s + cw(s) - f^c(s)\, y(s)] \qquad (32)$$

which is similar to a standard PI law, where k is the proportional gain, T_i the integral time constant, and kr_p the proportional gain on the set point.

$$u(s) = k[[w(s) - y(s)]/(T_i s) + r_p w(s) - y(s)] \qquad (33)$$

This similarity is illustrated by the relations

$$T_i = f^c \qquad (34a)$$

$$k = f^c/eb \qquad (34b)$$

$$r_p = c/f^c \qquad (34c)$$

If we now assume the system structure of Eq. (30), then the controller polynomials will be of the form

$$C(s) = c_0 s^2 + c_1 s + 1 \qquad (35a)$$

$$E'(s) = e_0 s + e_1 \qquad (35b)$$

$$F_0^c(s) = f_0^c s + f_1^c \qquad (35c)$$

$$G_0(s) = be_0 s + be_1 \qquad (35d)$$

The control law will then become

$$u(s) = [(e_1 + e_0 s)b]^{-1} \cdot [[w(s) - y(s)]/s + c_1 w(s) - f_1^c y(s) + (c_0 w(s) - f_0^c y(s))s] \qquad (36)$$

A standard PID law is typically as shown, where, in addition to Eq. (33), T_f is now the time constant of a filter to avoid derivative action at high frequencies, T_d is the derivative time constant, and r_d controls the derivative "kick."

$$u(s) = k[[w(s) - y(s)]/T_i s + (r_p w(s) - y(s)) + [r_d w(s) - y(s)]T_d s/(1 + T_f s)] \qquad (37a)$$

Comparing the two control laws, we can see that they are equivalent if the following relations apply:

$$T_f = be_1/be_0 \qquad (38a)$$

$$T_i = f_1^c - T_f \qquad (38b)$$

$$T_d = f_0^c - (T_iT_f)/T_i \qquad (38c)$$

$$k = T_i/be_1 \qquad (38d)$$

Relations (38) are used in the algorithm to derive on-line PID parameters corresponding to the estimated tuner parameters and to tune external PID controllers.

It is important to note that Eq. (37a) does not represent the only possible PID controller structure. For example, an alternative form is given by the following (where the proportional and derivative "kick" terms have been omitted for simplicity):

$$u(s) = k[1 + 1/T_is][T_ds/(1 + T_fs)] \cdot [w(s) - y(s)] \qquad (37b)$$

Equations such as (37b) would lead to a new set of conversion equations to replace Eqs. (38).

Multivariable Version of the Algorithm

As explained in the introduction, to apply the single-loop algorithm presented earlier to multiloop processes we need to introduce a way to decouple the interaction disturbance. As shown in [28], a possible representation of one loop of such a multivariable process is described in Fig. 1. The system and interaction transfer function are described as

$$S_{ij}(s) = B_{ij}(s)/A_{ij}(s); \quad i = 1, \dots, n, \quad j = 1, \dots, n \qquad (39)$$

To satisfy conditions that will be presented later, the assumption must be made that the interaction transfer functions are constants, i.e.,

$$A_{ij}(s) = 1 \qquad (40a)$$

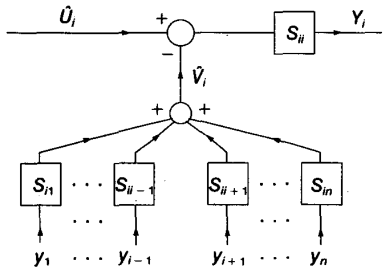


Fig. 1. ith subsystem model in a multiloop configuration.

$$S_{ij}(s) = B_{ij}(s) = \text{const}; \quad i \neq j \qquad (40b)$$

Combining Eq. (39) with Eqs. (11)-(13), we will find that the extended system representation is

$$A_{ii}'(s)\, y_i(s) = B_{ii}'(s)\Big(u_i(s) - \sum_{i\neq j}^{n} S_{ij}(s)\, y_j(s)\Big) + C_i'(s)\, z_i'(s) \qquad (41)$$

Considering Eqs. (13) and (21), we can define for each such loop

$$G_j'(s) = S_{ij}(s)\, E_i(s)\, B_{ii}(s) = sS_{ij}(s)\, G_{i0}(s) = sG_{j0}(s) \qquad (42)$$

The extended form of the zero-gain predictor, Eq. (26), will be

$$\phi_{i0}^{c*}(s) = sF_{i0}^c(s)/C_i(s)\, y_i(s) + sG_{i0}(s)/C_i(s)\, u_i(s) - \sum_{i\neq j}^{n} sG_{j0}(s)\Big/C_i(s)\, y_j(s) \qquad (43)$$

leading to the extension of the control law [Eq. (28)] as

$$u_i(s) = 1/G_{i0}(s)[[w_i(s) - y_i(s)]/s + C_{i0}(s)\, w(s) - F_{i0}^c(s)\, y_i(s)] + \sum_{i\neq j}^{n} \frac{G_{j0}(s)}{G_{i0}(s)}\, y_j(s) \qquad (44)$$

It is interesting to notice from Eq. (42) that, for each interaction term, the ratio following is the same as the interaction transfer function itself.

$$G_{j0}(s)/G_{i0}(s)\, y_j(s) = S_{ij}(s) \qquad (45)$$

This means that the sum of those ratios in Eq. (44) corresponds physically to the opposite of the disturbance from the interacting loops. Taking into account also assumptions (30), we need to have assumed Eqs. (40) so that those transfer functions are realizable.

The control signal of Eq. (44) must, therefore, compensate for two factors: dynamics of the loop under control (it must reduce the output-set-point error to zero); disturbance from the interaction.

Therefore, taking assumptions (30) for the PID controller case, the final extended control law will become

$$u_i(s) = [(e_{i1} + e_{i0}s)b_i]^{-1}[[w_i(s) - y_i(s)] \div sc_{i1}w(s) - f_{i1}^c y_i(s) + (c_{i0}w_i(s) - f_{i0}^c y_i(s))s] + \sum_{i\neq j}^{n} y_j(s)(g_{j0}s + g_{j1}) \div (g_{i0}s + g_{i1}) \qquad (46)$$

This control law has two parts. The first part is exactly the same as Eq. (28). Based on this part, the PID parameters are derived exactly as earlier. Then a feedforward bias signal is calculated based on the second part of Eq. (46), which would compensate for the interaction disturbance. It is given by

$$b_i(s) = \sum_{i\neq j}^{n} y_j(s)(g_{j0}s + g_{j1})\Big/(g_{i0}s + g_{i1}) \qquad (47)$$

If this signal is sent as a feedforward signal to the external PID controllers, the result will be the desired decoupling of the interaction. In the case of the PI controllers [assumptions (29)], the design is very similar to the PID case.

There is also scope to introduce a nonzero detuning polynomial $Q(s)$ [from Eq. (5)]. It has been shown in [14] that such an action will enhance the stability of the algorithm. A possible choice for $Q(s)$ can be

$$Q(s) = [q_0s^2 + q_1s]/C(s) \qquad (48)$$

leading to a detuned version of the control law in Eq. (46) as

$$u_i(s) = [g_{i1} + q_{i1} + (g_{i0} + q_{i0})s]^{-1} \cdot [w_i(s) - y_i(s)/sc_{i1}w(s) - f_{i1}^c y_i(s) + (c_{i0}w_i(s) - f_{i0}^c y_i(s))s] + \sum_{i\neq j}^{n} \frac{g_{j0}s + g_{j1}}{(g_{i0} + q_{i0})s + g_{i1} + q_{i1}} \cdot y_j(s) \qquad (49)$$

Now the detuned feedforward bias factor becomes

$$b_i(s) = \sum_{i\neq j}^{n} \frac{g_{j0}s + g_{j1}}{(g_{i0} + q_{i0})s + g_{i1} + q_{i1}}\, y_j(s) \qquad (50)$$

whereas the estimated PID gain, k, and derivative filter time constant, T_f, from Eqs. (38) become

$$T_f = be_1 + q_1/be_0 + q_0 \qquad (51a)$$

$$T_i = f_1^c - T_f \qquad (51b)$$

$$k = T_i/be_1 + q_1 \qquad (51c)$$

In the nonadaptive case [where the system polynomials $B_{ij}(s)$ and $A_{ij}(s)$ are assumed known], the parameters of the polynomials $F_{i0}^c(s)$, $G_{i0}(s)$, and $G_{j0}(s)$ can be calculated from relations similar to Eq. (5). Then we can solve for the PID parameters from Eqs. (38) or (51). In the adaptive case (unknown system polynomials), Eq. (43) can be expressed in the standard form as

$$\phi_{i0}^{c*}(s) = \mathbf{X}^T\boldsymbol{\theta} \qquad (52)$$

where now the data vector $\mathbf{X}^T$ is composed of the following, and the parameter vector $\boldsymbol{\theta}$ is composed of the vectors of the parameters corresponding to each element of $\mathbf{X}^T$.

$$\mathbf{X}^T = [\mathbf{X}_u; \mathbf{X}_{y1}; \ldots ; \mathbf{X}_{yn}] \qquad (53a)$$

$$\mathbf{X}_u = C_i(s)^{-1}[u_i(s), su_i(s), \ldots , s^n u_i(s)] \qquad (53b)$$

$$\mathbf{X}_{yj} = C_i(s)^{-1}[y_j(s), sy_j(s), \ldots , s^n y_j(s)], \quad j = 1, \ldots , n \qquad (53c)$$

The i factors of those vectors correspond to the main loop dynamics, and the j factors correspond to the dynamics of the interaction from the other loops.

$$\boldsymbol{\theta}^T = [\boldsymbol{\theta}_u; \boldsymbol{\theta}_{y1}; \ldots ; \boldsymbol{\theta}_{yn}] \qquad (54)$$

A standard least-squares estimation algorithm is used to estimate $\hat{\boldsymbol{\theta}}$, and the estimated parameters are used in Eqs. (38) or (51) to give the estimated PID coefficients.

First Application: A Set of Laboratory Tanks

Two sets of experiments on real processes are presented in the paper. This section describes the first: a set of mutually coupled tanks where the water level was under control. The second process (an industrial plastics extruder barrel) is discussed in the next section. Further details of both applications can be found in [19].

Experimental Equipment

Tanks A set of four adjacent tanks was constructed out of perspex and fixed above a sump of the same material. The tanks have no internal connections, but may be interconnected in a variety of configurations using rubber tubing, clamps, and valves. The tanks are numbered consecutively from 1 to 4 starting at the left. For the single-loop experiment, tanks 1 and 2 were connected and tank 2 discharged into the sump; for the two-loop experiment, tanks 1 and 2, 2 and 3, and 3 and 4 were connected. Tanks 2 and 3 also discharged into the sump. This discharge introduced a nonlinearity into the system as a result of the nonlinear pressure-flow characteristic.

Transducers Each tank was equipped with a silicon pressure sensor tightly sealed into the upper end of a plastic tube whose lower end reached the bottom of the tank. Thus, the level was measured indirectly from the pressure at the bottom of each tank. The pressure sensor output was conditioned suitably and buffered by an instrumentation amplifier to give a 0–10-V signal. For the single-loop experiment, the tank 2 level was used as output. For the two-loop experiment, the levels of tanks 2 and 3 were used as output. It was verified experimentally that the level to the measured voltage signal was linear.

The 12-Vdc submersible bilge pumps were placed in the sump and could discharge, via rubber tubes, into any of the four tanks. They were each driven, via buffers and power amplifiers, by a 0–10-V signal. For the single-loop experiments, pump 1 discharged into tank 1 and pump 2 was not used. For the two-loop experiment, pump 1 discharged into tank 1 and pump 2 into tank 4. In both cases, the discharge end of the tube was above the water surface. Experiments showed that the pumps showed a highly nonlinear and dynamic characteristic relating input voltage to flow rate. At low voltages, the pump did not deliver enough pressure to drive any water through the tube; at high voltages, the flow rate was independent of voltage. When switching from low to high voltage, there was a significant delay before the tube filled with water.

Controllers The PID controllers were industry-standard Eurotherm 820 controllers equipped with a 0–10-V interface and adjusted so that 0–10 V corresponded to a 0–100 percent range as both input and output. These controllers operated with a 50-msec sample interval and had state-of-the-art integral desaturation, bumpless transfer, and user interface. The particular feature of interest here was the ability to send P, I, and D values and receive measured variables and control signals via a standard RS232 link and an appropriate message protocol. For the two-loop experiment, the additional feature of adding an additional bias signal to the 820 control signal was utilized.

For the single-loop experiment, controller 1 measured the level in tank 2 and sent the control signal to pump 1; controller 2 was not used. For the two-loop experiment, controller 1 was connected as for the single-loop experiment, while controller 2 measured the level in tank 3 and sent the control signal to pump 2.

Tuner The self-tuning algorithm presented earlier was implemented in Pascal on an Atari 1040 ST personal computer. The PID parameters were obtained from Eqs. (51), and the proportional gain was converted to proportional band (= 100/proportional gain).

The communications protocol was implemented in C and linked to the Pascal program. The signals were requested from the updated 820 controller parameters and the PID values were returned with a 2-sec cycle time. It is important to note that the control signal itself was generated at the 820 sample interval of 50 msec. Data were displayed on the screen and also written to random-access memory disk for later copying onto the floppy disks at the end of each experiment.

Mathematical Model

An idealized mathematical model of the apparatus is presented in Fig. 2, where A is the area of the water surface, h_i the level in each tank, k_p the constant associated with the pump, k_h the constants of the output holes and the pipes, and k_s the pressure sensor constant. The signal y' is the output of the other interacting loop. There are two key features to note. First, the relationship between inflow into tank 1 (or 4) and the level in tank 2 (or 3) is predominantly second order. Second, the interaction is from the output of tank 3 to tank 2, not to tank 1. Thus, the interactive model used (output-input) involves the inverse of the transfer function relating the inflow to tank 1 to the inflow to tank 2 from tank 1.

The implication of this improper interaction model is examined in the sequel.

Single-Loop Experiments

Purpose The main purpose of these experiments was to verify our claim that the use of the self-tuning controller allows the user to specify closest loop performance in terms

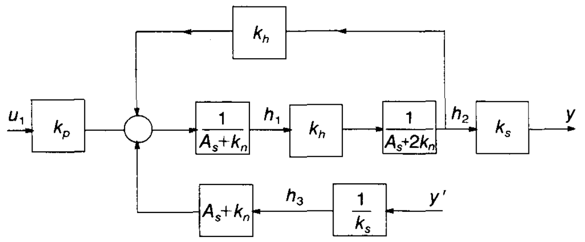


Fig. 2. Tank system and interaction block diagram.

of desired response $1/P(s)$. The PID is tuned automatically to give this desired response.

Description In each case, we chose a second-order dead-beat response specified by

$$1/P(s) = 1/(1 + sT) \tag{55}$$

A series of five experiments was conducted for $T = 10, 15, 20, 30,$ and 50 sec. Figure 3 shows the results for $T = 20$. (Figures corresponding to the other experiments are given in [29].) The figure has two sets of graphs. The upper graph shows three signals: the firm square-wave signal is the controller set point, the dashed line is that set point passed through the desired system $1/P(s)$, and the other firm line is the measured output signal. All three quantities are expressed as a percentage of full scale. The lower graph shows the control signal expressed as a percentage of full scale. The results for other values of T have a similar form, but both the desired and actual output time constants decrease, and the corresponding control signal increases, with T. Thus, T acts as a performance-oriented parameter trading off speed of response against control signal magnitude while retaining the same form of response.

Figure 4 shows the proportional band for each of the five experiments: the largest value corresponds to $T = 50$ and the smallest to $T = 10$. As expected, the proportional band decreases with desired time constant T. The corresponding integral time constants have similar behavior except that they increase with T; T_i converges to about 3.8 sec when $T = 10$ and to about 3.1 sec when $T = 50$. The derivative time constant T_d converges to about 1.1 sec for each value of T.

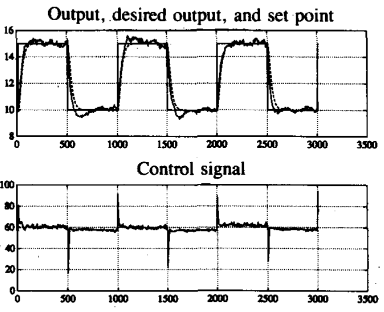


Fig. 3. Single-loop control: T = 20 sec.

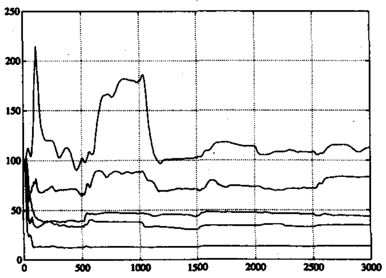


Fig. 4. Single-loop control: proportional band (%).

Discussion The overall trend is clear: the PID parameters are adjusted so that the closed-loop system is approximately as desired. The responses differ in detail from their desired values, particularly for large and small T. The former is explained by the low controller gains (large proportional band) not being sufficient to overcome nonlinear behavior; the latter is explained by control signal saturation.

Despite the nonideal system under control, the behavior is broadly as advertised.

Two-Loop Experiments

Purpose The main purpose of this sequence of experiments was to investigate the advantages of explicitly identifying process interaction as a means of reducing closed-loop interaction. In addition, we wished to verify our approach of using essentially single-loop instruments in a multiloop setting by computing an additional "bias" signal.

In hindsight, the sequence of experiments presented also illustrates the power of the continuous-time approach in analyzing and explaining our initial failure and devising methods to overcome the problem. In particular, the result noted by Gawthrop [14], [28]—that the poor robustness of self-tuning controllers to neglected dynamics can be overcome by weighting the control signal—is used to good effect.

Description As in the single-loop case, each loop had a desired response specified by

$$1/P(s) = 1/(1 + sT) \tag{56}$$

Because we wished to investigate the effect of interaction, we chose the "worst case" of the time constants used in the single-loop case: $T = 10$ sec. The following series of experiments was performed.

1. Untuned PID—no interaction term. The PID coefficients used are those generated in the corresponding single-loop experiment.
2. Tuned PID—no interaction term
3. Tuned PID—standard interaction term
4. Tuned PID—standard interaction term with weighting ($Q = 0.1$)
5. Tuned PID—set-point interaction term

In each case, loop 1 is driven by a square set point with a period of 600 sec, and loop 2 is driven by a step change. Thus, the effects of interaction are seen most clearly on loop 2.

Figure 5 is of the same form as Fig. 3, except that the lower graph contains one extra signal: a dashed line corresponding to the computed interaction, or bias term. It shows the signals corresponding to loop 2 for experiment 5. (Figures corresponding to the other experiments are given in [29].) The results of the other experiments can be described relative to this figure as follows. In each case, the description refers to loop 2, whose output would ideally settle to a constant value of 12.

1. A deviation of ± 2 units appears on output 2 when output 1 changes. The control signal is a noisy square wave moving between 60 and 100 in unison with output 1. The computed interaction is zero.
2. The results are similar in form to those of experiment 1, except that the deviation gradually reduces to about ± 0.75 and the control signal is noisier.
3. The control signal exhibits large oscillations at time 500, and the experiment is terminated.
4. The deviations in the output are of similar magnitude to those in experiment 2, and the computed interaction term moves in unison with output 1.
5. As depicted in Fig. 5, the output deviations are reduced to about ± 0.5, and the computed interaction term is not noisy. The phase-advance nature of this interaction transfer function is clear.

Discussion Comparison of experiments 1 and 2 indicates that the tuner itself is able to reduce the effect of interaction due to an adjacent loop; this is an encouraging result.

The result of experiment 3 indicates a pit-

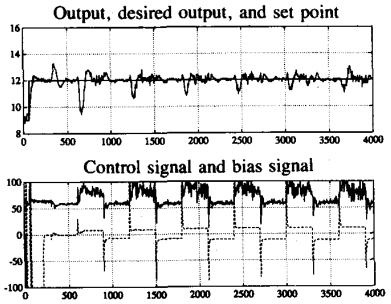


Fig. 5. Tuned PID (loop 2)—set-point interaction term.

fall of this approach: the neglected dynamics implicit in the modeling procedure lead to the tuning procedure attempting to approximate an improper interaction term with a high-gain proper term—with the unfortunate result observed. However, as discussed in [14] and [28], self-tuning controllers are sensitive to neglected dynamics unless control weighting is used; this motivates experiment 4. Note that the bias signal is now stable, but quite noisy.

As discussed in [19], the noise amplification inherent in the generation of the bias term can be avoided by replacing the output by a clean approximation; in this case, the set point itself. The result is shown in Fig. 5, where the bias signal is smoother and yet the interaction effect is reduced.

Thus, the single-loop approach is satisfactory in the two-loop situation, but well-designed decoupling gives improved performance. Because all the terms within the control and tuning algorithms are in a continuous-time format, they have a clear interpretation to the user; thus, as illustrated in this application, the approach lends itself to algorithm engineering.

Second Application: Temperature Control of a Plastics Extruder Barrel

Problems in Controlling Extrusion Processes

Plastic extrusion is a well-established method widely used in the polymer processing industry [30]. Such extruders typically consist of a large barrel divided into several temperature zones, with a hopper at one end and a die at the other. Polymer is fed into the barrel in raw and solid form from the hopper and is pushed forward by a powerful screw. Simultaneously, it is gradually heated while passing through the various temperature zones set in gradually increasing temperatures. The heat produced by the heaters in the barrel, together with the heat released from the friction between the raw polymer and the surfaces of the barrel and the screw, eventually causes the melting of the polymer, which is then pushed by the screw out from the die, to be processed further for various purposes.

Such a process is highly nonlinear and a typical example of a difficult control problem [31]. The output variables typically are the outflow from the die, or, equivalently, the pressure in the vicinity of the die, and the polymer temperature. The main controlling variable is the screw speed, since the response of the process to it is instantaneous. The input signal to the heaters is a secondary variable, mainly because the thermal inertia of the barrel seriously dampens the response of the barrel to such changes. Presently, only approximate modeling techniques have been proposed, and a complete overall modeling, designing, and controlling strategy has yet to appear [32], [33].

Control is still exercised mainly by typical PID controllers, which usually tolerate the system for short ranges of operating conditions, but which need frequent retuning as the operating and environmental conditions change with time. Thus, those controllers must be supervised constantly.

This paper does not attempt to give a total solution to the overall control problem of the extrusion process. Only the temperature control of the barrel is considered here, because the screw speed was available only for manual control. The aim of the experiment is to prove that the tuner estimates sensible parameters for a PID controller in a real industrial situation, although the final result is very much prejudiced by the ineffectiveness of temperature control.

The specific process we are dealing with is a typical extruder barrel (Fig. 6) with four main temperature zones and two more close to the die used to extract plastic to be formed into various fibers and cables. The plant was installed at an industrial site.

The temperature of the barrel had been controlled by six standard industrial PID controllers (Eurotherm 820), one for each zone, operating on the same standard PID algorithm as for the coupled tanks. Several limits and alarm settings were the only differences from the previous experiments. The self-tuner was again running from the same Atari ST exactly as described previously. Some alternative algorithms are presented in [34].

Zero Screw Speed

Purpose When there is no flow through the extruder barrel, the temperature control is essentially that of an inert thermal mass. However, because the system is physically described by a partial differential equation, an essentially *transcendental* transfer function is being controlled on the assumption that it is *rational* and second order. Therefore, it is essential that the proposed algorithm is robust in the presence of such approximations. The purpose of this experiment is to verify robustness in this practical example.

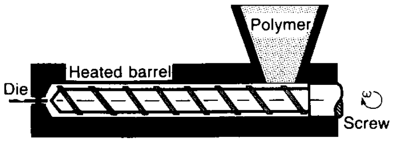


Fig. 6. Extruder barrel.

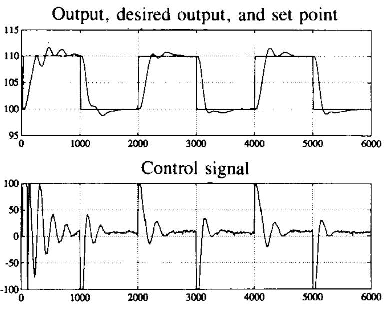


Fig. 7. Tuned PID—zero screw speed.

Description Figure 7 is of the same format as Fig. 3 and shows the results using a tuned controller. (Figures corresponding to the other experiments are given in [29].) Note that the overshoot is reduced substantially compared with a similar run of the untuned controller. The estimated proportional band settles down after about 100 points; the other PID parameters take about 10 times as long to settle.

Discussion Despite the approximations inherent in the tuner model, the tuning algorithm gives better control than the untuned controller. The imperfections in the performance appear to be a result of these approximations. We believe that a more complex structure than PID is needed to overcome these defects.

Variable Screw Speed

Purpose The rotating screw imparts a large heat flow into the system due to viscous dissipation in the polymer; therefore, a variable screw speed imparts a significant disturbance. The purpose of this experiment was to convince the industrial user that the tuner could cope with this effect.

Description Figure 8 follows the same format as Fig. 7. Note that the temperature deviations are due to changes in screw speed, and an initial set-point change is used to aid tuning.

Discussion Clearly, the self-tuning PID algorithm cannot do better than a well-tuned PID controller, as the results indicate. However, it has been demonstrated that the tuning algorithm can still operate under adverse circumstances.

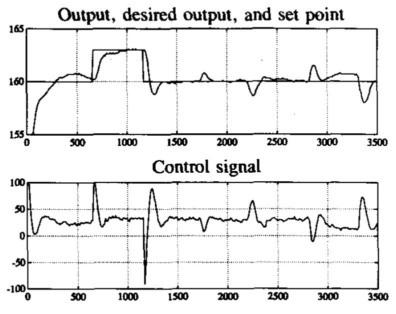


Fig. 8. Tuned PID—nonzero screw speed.

Conclusions

The continuous-time self-tuning controller, hitherto used as a complete control algorithm generating a control signal to be sent to the process, has been shown to be also effective in a new role: generating the parameters of a proportional-integral-derivative (PID) controller external to the self-tuning algorithm. These capabilities are illustrated and confirmed by application to both a laboratory-scale process and a real industrial plastic extruder.

An important advantage of using the self-tuning algorithm in this mode is that the wealth of experience built into commercial PID controllers is retained. In this case, the fact that we were using a tried and trusted industrial PID controller, well known to the company involved, was instrumental in obtaining permission to carry out the industrial experiments.

From our experience in the industrial application, we can attest to the convenience and simplicity of having a personal-computer–based automatic tuning facility interfaced to the process via a standard RS232 connection. This could well be the basis of a commercial automatic tuning instrument.

Self-tuning algorithms can be developed either within a discrete- or continuous-time context. The power of the latter approach has manifested itself in this research in two ways. First, the algorithm directly generates standard PID parameters, such as proportional band, derivative and integral time constants. Second, because all the terms within the control and tuning algorithms are in a continuous-time format, they have a clear interpretation to the user; thus, the algorithm lends itself to algorithm engineering.

As has been pointed out by Astrom [21], self-tuning algorithms may be distinguished from conventional algorithms by the fact that the user specifies performance, not how to achieve that performance. The particular algorithm given here specifies performance in terms of closed-loop performance. In particular, the performance of the second-order system considered here was essentially specified by setting the damping ratio equal to 1, and the time constant of the response was varied. It was demonstrated in the experiment that the algorithm behaved broadly as advertised; the self-tuning algorithm adjusted the PID controller so that the closed-loop response had the requested time constant.

As well as its role of automatically tuning feedback terms, the self-tuning algorithm was also used to tune feedforward terms. Because of the constraints that the controller actually used, we were not able to send feedforward parameters directly to the control algorithm, but rather generated the additional control signal directly. However, with suitable hardware, we could have passed parameters instead of control signals, which would have had the important advantage that the control system could generate these feedforward signals with the tuning device removed. This is an area for further research and development.

Acknowledgments

L. Smith of Eurotherm Ltd. provided the Eurotherm 820 controllers, suggested the application to extruder control, and gave the authors advice and encouragement. This work formed part of a project sponsored by Science and Engineering Research Council (U.K.) under Grant GR/E 61854 (ex GR/D 45642). The authors would also like to thank the anonymous referees for helpful suggestions for improving the manuscript.

References

[1] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," *ASME Trans.*, vol. 64, pp. 759–768, Nov. 1942.

[2] J. G. Ziegler and N. B. Nichols, "Process Lags in Automatic Control Circuits," *ASME Trans.*, pp. 433–444, July 1943.

[3] P. Hazebroek and B. L. Van Der Waerden, "Theoretical Considerations on the Optimum Adjustment of Regulators," *ASME Trans.*, pp. 309–315, April 1950.

[4] W. B. Field, "Adaptive Three-Mode Controller," *ISA J.*, vol. 9, no. 2, pp. 30–33, 1962.

[5] Y. Nishikawa, N. Sannomiya, T. Ohta, and H. Tanaka, "A Method for Auto-Tuning of PID Control Parameters," *Automatica*, vol. 20, no. 3, pp. 321–332, 1984.

[6] K. J. Astrom and T. Hagglund, "Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins," *Automatica*, vol. 20, no. 5, pp. 645–651, 1984.

[7] H. M. Morris, "How Adaptive Are Adaptive Process Controllers?," *Contr. Eng.*, pp. 96–100, March 1987.

[8] P. W. Gallier and R. E. Otto, "Self-Tuning Computer Adapts DDC Algorithms," *Instru. Tech.*, pp. 65–70, Feb. 1968.

[9] N. Andreiev, "A New Dimension: A Self-Tuning Controller that Continually Optimizes PID Controllers," *Contr. Eng.*, pp. 84–85, Aug. 1981.

[10] W. M. Hawk, "A Self-Tuning, Self-Contained PID Controller," *Proc. Amer. Contr. Conf.*, San Francisco, pp. 838–842, 1983.

[11] C. G. Proudfoot, P. J. Gawthrop, and O. L. R. Jacobs, "Self-Tuning PI Control of a pH Neutralization Process," *IEE Proc.*, vol. 130, pt. D, no. 5, pp. 267–272, 1983.

[12] P. J. Gawthrop, "Using the Self-Tuning Controller to Tune PID Regulators," Sussex Univ. Int. Rept. CE/T/2, 1982.

[13] P. J. Gawthrop, "Self-Tuning PID Controllers: Algorithms and Implementation," *IEEE Trans. Automat. Contr.*, vol. 31, no. 3, pp. 201–209, 1986.

[14] P. J. Gawthrop, *Continuous-Time Self-Tuning Control—Volume 1: Design*, Research Studies Press, 1987.

[15] H. N. Koivo and J. Sorvari, "On-Line Tuning of a Multivariable PID Controller for Robot Manipulators," *Proc. 24th IEEE Conf. on Decision and Control*, Fort Lauderdale, FL, pp. 1502–1504, 1985.

[16] U. Borison, "Self-Tuning Regulators for a Class of Multivariable Systems," *Automatica*, vol. 15, pp. 209–215, 1979.

[17] H. N. Koivo, "A Multivariable Self-Tuning Controller," *Automatica*, vol. 16, pp. 351–366, 1980.

[18] A. J. Morris, Y. Nazer, and R. K. Wood, "Multivariable Self-Tuning Process Control," *Optimal Contr. Applic. & Methods*, vol. 3, pp. 363–387, 1982.

[19] P. E. Nomikos, "Multivariable Self-Tuning Controllers for Industrial Applications," D.Phil. Thesis, Univ. of Sussex, England, 1988.

[20] J. M. Edmunds, "Digital Adaptive Pole-Shifting Regulators," Ph.D. Thesis, University of Manchester Institute of Science and Technology (U.K.), 1976.

[21] K. J. Astrom, P. Hagander, and J. Sternby, "Zeros of Sampled Systems," *Automatica*, vol. 20, pp. 31–38, 1980.

[22] G. C. Goodwin, R. Lozanno-Leal, D. Q. Mayne, and R. H. Middleton, "Rapprochement between Continuous and Discrete Model Reference Adaptive Control," *Automatica*, vol. 22, no. 2, pp. 199–207, 1986.

[23] M. M'Saad, R. Ortega, and J. D. Landau, "Adaptive Controllers for Discrete-Time Systems with Arbitrary Zeros: An Overview," *Automatica*, vol. 21, no. 4, pp. 413–423, 1985.

[24] P. J. Gawthrop, "A Continuous-Time Approach to Discrete-Time Self-Tuning Control," *Optimal Contr. Applic. & Methods*, vol. 3, pp. 399–414, 1982.

[25] D. W. Clarke, A. J. F. Hodgson, and P. S. Tuffs, "Offset Problem and *k*-incremental Predictors in Self-Tuning Control," *IEE Proc.*, vol. 130, pt. D, no. 5, pp. 217–225, 1983.

[26] P. R. Bellanger, "On Type 1 Systems and the Clarke-Gawthrop Regulator," *Automatica*, vol. 19, no. 1, pp. 91–94, 1983.

[27] R. Scattolini, "A Multivariable Self-Tuning Controller with Integral Action," *Automatica*, vol. 22, no. 6, pp. 619–627, 1986.

[28] P. J. Gawthrop, "Robust Self-Tuning Control of *n*-input *n*-output Systems," 7th IFAC Symp. on Identification and System Parameter Estimation, York, England, 1985.

[29] P. J. Gawthrop and P. E. Nomikos, "Automatic Tuning of Commercial PID Controllers. Single and Multi-loop Applications," Control Engineering Rept. 89.1, Dept. of Mechanical Engineering, Glasgow Univ., 1989.

[30] H. Amrehn, "Computer Control in the Polymerization Industry," *Automatica*, vol. 13, pp. 533–545, 1977.

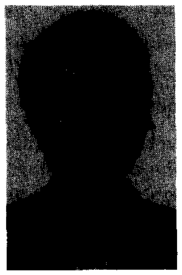
[31] A. K. Kochhar and J. Parnaby, "Dynamical Modelling and Control of Plastics Extrusion Processes," *Automatica*, vol. 13, pp. 177–183, 1977.

[32] A. K. Kochhar and J. Parnaby, "Comparison of Stochastic Identification Techniques for Dynamic Modelling of Plastics Extrusion Processes," *Proc. Inst. Mech. Engin.*, vol. 192, pp. 299–309, 1978.

[33] M. H. Costin, P. A. Taylor, and J. D. Wright, "A Critical Review of Dynamic Modelling and Control of Plasticating Extruders," *Polymer Eng. Sci.*, vol. 22, no. 7, pp. 393–401, 1982.

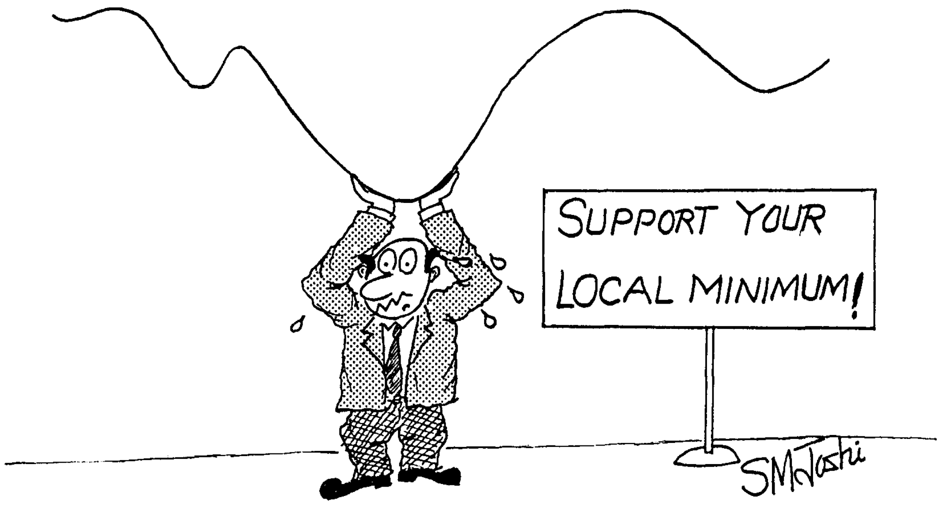
[34] P. J. Gawthrop, P. E. Nomikos, and L. Smith, "Adaptive Temperature Control of Industrial Processes—A Comparative Study," IEE Conf. Control 88, Oxford, England, April 1988.

Peter J. Gawthrop was born in Seascale, Cumberland, in 1952. He obtained his M.A. and D.Phil. degrees in engineering science from Oxford University in 1973 and 1976, respectively. Following a period as a Research Assistant with the Department of Engineering Science at Oxford University, he became W. W. Spooner Research Fellow at New College, Oxford. He then moved to the University of Sussex as a Lecturer, and later a Readers in control engineering. Since 1987, he has held the Wylie Chair of Mechanical Engineering at Glasgow University. His research interests include self-tuning control, continuous-time system identification, and system modeling. He is interested in applying control techniques to a number of areas, including process control, robotics, and macroeconomics.



Panagiotis E. Nomikos was born in Athens, Greece, in 1961. He graduated from the University of Athens with a degree in physics in 1984. He obtained his Ph.D. from the University of Sussex, England, in 1988. In his thesis, he investigated the continuous-time self-tuning control algorithm in a multiloop configuration and its applications in real-life industrial processes. He worked for 1 year with Unilever Research in Liverpool, England, as a Control Engineer, and he is now with the Greek Air Force. His interests include adaptive and self-tuning control systems and expert control systems, with particular emphasis on the application of those theories in real-time conditions.

Out of Control


SUPPORT YOUR LOCAL MINIMUM!
SMJoshi