Optimal setting for discrete PID controllers

K. Minh Vu

Indexing terms: Control theory, Process control

Abstract: An approach to obtain the optimal gains for a discrete PID controller is presented. The optimal gains are obtained by minimising the variance of the output variable under feedback. The problem is formulated similarly to the Kalman filter problem and the gains are obtained from the solution of a Riccati equation. The discussion is restricted to single input single output systems.

1 Introduction

The PID or the three-mode controller is probably the most widely used controller in the process industry. It gained its popularity for its simplicity of having only three parameters. But owing to its simplicity it has also paid the price of not having an efficient and practical way of determining optimal gains. From the time of its infancy, control engineers have worked on methods to obtain optimal gains. The work has been on both types of controllers: analogue, and discrete or digital. But, in practice, control engineers have sometimes used these indistinguishably. Early work can be credited to Ziegler and Nichols [1], who obtained the gains from the closed-loop *ultimate period*, and Cohen and Coon [2] who introduced the method of open-loop *process reaction curve* to approximate the control loop by a delayed first order system and obtained the optimal gains for minimum offset, one quarter decay ratio and minimum integral square error response to a load change. These methods have been taught in undergraduate control courses as ways to tune analogue controllers. More advanced work on analogue PID controllers used eigenvalue assignment to place the closed-loop poles for stability. Recent work on analogue PID controllers has been reported by Zervos *et al.* [3], who introduced the concept of orthonomal series to represent a system, then tuned the controller gains. A discrete version of the eigenvalue assignment method was studied by Stojić and Petrović [4], who used a grapho-analytical pole-placement procedure to obtain the gains. By using the Box–Jenkins model to represent a system, MacGregor *et al.* [5] suggested a method of obtaining the optimal gains by plotting contours of variances under feedback. The optimal gains are obtained by extrapolation from the contours of variances. Another approach was suggested by Meo *et al.* [6] who used projective control to obtain the gains. The adaptive or self-tuning PID controller was discussed by Radke and Isermann [7] who used 2-stage least squares to estimate the parameters of a system and a numerical optimisation procedure to estimate the controller gains. Åström and Hägglund [8] discussed and also compared a number of methods in their book. So far, all the methods mentioned suffer from either impracticality or time consumption or both. In this paper, a method to obtain the optimal gains for a discrete PID controller is presented. The controller gains are obtained by minimising the variance of the output variable under feedback.

Paper 8457D (C8), first received 23rd March 1990 and in revised form 13th August 1991

The author is with Measurex Devron Inc., 500 Brooksbank Avenue, North Vancouver, BC, Canada V7J 3S4

2 Optimal stochastic PID controller

Consider the following stochastic control system

$$y_{t+f+1} = \frac{\omega(z^{-1})}{\delta(z^{-1})} u_t + \frac{\theta(z^{-1})}{\phi(z^{-1})} a_{t+f+1} \qquad (1)$$

with a_t as the generating white noise. The system is required to have minimum variance under the feedback PID control law

$$u_t = k_p y_t + k_i \sum y_i + k_d(y_t - y_{t-1})$$

where k_p is the proportional gain, k_i is the integral gain and k_d is the derivative gain. By taking the difference on both sides of the above equation, one obtains the velocity form of the PID controller

$$\begin{aligned} u_t - u_{t-1} &= k_p(y_t - y_{t-1}) + k_i y_t \\ &\quad + k_d(y_t - y_{t-1} - y_{t-1} + y_{t-2}) \\ &= (k_p + k_i + k_d)y_t \\ &\quad + (-k_p - 2k_d)y_{t-1} + k_d y_{t-2} \end{aligned}$$

or

$$\begin{aligned} (1 - z^{-1})u_t &= (k_p + k_i + k_d)y_t \\ &\quad + (-k_p - 2k_d)y_{t-1} + k_d y_{t-2} \\ &= l_1 y_t + l_2 y_{t-1} + l_3 y_{t-2} \\ &= l(z^{-1})y_t \end{aligned}$$

With the above form of the controller, under feedback, eqn. 1 can be written as

$$\begin{aligned} &y_{t+f+1} \\ &\quad = \frac{\omega(z^{-1})}{\delta(z^{-1})} u_t + \frac{\theta(z^{-1})}{\phi(z^{-1})} a_{t+f+1} \\ &\quad = \frac{\omega(z^{-1})}{\delta(z^{-1})} \frac{l(z^{-1})z^{-f-1}}{(1 - z^{-1})} y_{t+f+1} + \frac{\theta(z^{-1})}{\phi(z^{-1})} a_{t+f+1} \\ &\quad = \frac{\delta(z^{-1})\theta(z^{-1})(1 - z^{-1})}{\delta(z^{-1})\phi(z^{-1})(1 - z^{-1}) - \omega(z^{-1})\phi(z^{-1})z^{-f-1}l(z^{-1})} \\ &\quad \times a_{t+f+1} \end{aligned}$$

If one rewrites the above equation as

$$y_t = \frac{\alpha(z^{-1})}{\beta(z^{-1}) - \gamma(z^{-1})z^{-f-1}l(z^{-1})} a_t$$

where

$$\alpha(z^{-1}) = 1 - \alpha_1 z^{-1} - \cdots - \alpha_n z^{-n}$$
$$\beta(z^{-1}) = 1 - \beta_1 z^{-1} - \cdots - \beta_n z^{-n}$$
$$\gamma(z^{-1}) = \gamma_0 - \gamma_1 z^{-1} - \cdots - \gamma_{n-f-3} z^{-n+f+3}$$

and defines the following matrices and vectors:

$$\boldsymbol{p} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \quad \boldsymbol{q} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ -\gamma_0 & 0 & 0 \\ \gamma_1 & -\gamma_0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & \gamma_{n-f-3} \end{bmatrix}$$

with $\boldsymbol{l}$ defined as before:

$$\boldsymbol{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} k_p \\ k_i \\ k_d \end{bmatrix} = \boldsymbol{G}\begin{bmatrix} k_p \\ k_i \\ k_d \end{bmatrix} = \boldsymbol{Gk}$$

then y_t can be put into the state space model form as below

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{A}^*\boldsymbol{x}_t + \boldsymbol{b}a_t \\ y_t &= \boldsymbol{c}^T\boldsymbol{x}_t + a_t \end{aligned} \tag{2}$$

with

$$\begin{aligned} \boldsymbol{A}^* &= \begin{bmatrix} \boldsymbol{q} & \begin{matrix} \boldsymbol{I}_{n-1} \\ \boldsymbol{0} \end{matrix} \end{bmatrix} - \boldsymbol{HGk}[1 \quad 0 \quad \cdots \quad 0] \\ &= \boldsymbol{A} - \boldsymbol{HGk}[1 \quad 0 \quad \cdots \quad 0] \\ \boldsymbol{b} &= [\boldsymbol{q} - \boldsymbol{p} - \boldsymbol{HGk}] \\ \boldsymbol{c}^T &= [1 \quad 0 \quad \cdots \quad 0] = [1 \quad \boldsymbol{0}] \end{aligned}$$

By taking the variance on both sides of eqn. 2

$$\begin{aligned} \boldsymbol{V} &= E\{\boldsymbol{x}_t \boldsymbol{x}_t^T\} \\ &= \boldsymbol{A}^*\boldsymbol{V}\boldsymbol{A}^{*T} + \boldsymbol{bb}^T\sigma_a^2 \end{aligned} \tag{3}$$

The variance of the output variable is given by

$$\begin{aligned} \sigma_y^2 &= E\{y_t y_t^T\} \\ &= \boldsymbol{c}^T\boldsymbol{Vc} + \sigma_a^2 \end{aligned}$$

and therefore

$$\sigma_y^2 = \boldsymbol{c}^T[\boldsymbol{A}^*\boldsymbol{V}\boldsymbol{A}^{*T} + \boldsymbol{bb}^T\sigma_a^2]\boldsymbol{c} + \sigma_a^2$$

or

$$\begin{aligned} \sigma_y^2 &= \boldsymbol{c}^T[(\boldsymbol{A} - \boldsymbol{HGk}[1 \quad \boldsymbol{0}])\boldsymbol{V}(\boldsymbol{A} - \boldsymbol{HGk}[1 \quad \boldsymbol{0}])^T \\ &\quad + [\boldsymbol{q} - \boldsymbol{p} - \boldsymbol{HGk}][\boldsymbol{q} - \boldsymbol{p} - \boldsymbol{HGk}]^T\sigma_a^2]\boldsymbol{c} + \sigma_a^2 \\ &= \boldsymbol{c}^T\Bigg[\boldsymbol{AVA}^T - \left[\boldsymbol{AV}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + (\boldsymbol{q} - \boldsymbol{p})\sigma_a^2\right]\boldsymbol{k}^T\boldsymbol{G}^T\boldsymbol{H}^T \\ &\quad + \boldsymbol{HGk}[1 \quad \boldsymbol{0}]\boldsymbol{V}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}\boldsymbol{k}^T\boldsymbol{G}^T\boldsymbol{H}^T \\ &\quad - \boldsymbol{HGk}[[1 \quad \boldsymbol{0}]\boldsymbol{VA}^T + (\boldsymbol{q} - \boldsymbol{p})^T\sigma_a^2] \\ &\quad + [\boldsymbol{q} - \boldsymbol{p}][\boldsymbol{q} - \boldsymbol{p}]^T\sigma_a^2 + \boldsymbol{HGk}\sigma_a^2\boldsymbol{k}^T\boldsymbol{G}^T\boldsymbol{H}^T\Bigg]\boldsymbol{c} + \sigma_a^2 \\ &= \boldsymbol{c}^T\Bigg[\boldsymbol{AVA}^T + [\boldsymbol{q} - \boldsymbol{p}][\boldsymbol{q} - \boldsymbol{p}]^T\sigma_a^2 \\ &\quad - \frac{\left[\boldsymbol{AV}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + (\boldsymbol{q} - \boldsymbol{p})\sigma_a^2\right]\left[\boldsymbol{AV}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + (\boldsymbol{q} - \boldsymbol{p})\sigma_a^2\right]^T}{\sigma_a^2 + [1 \quad \boldsymbol{0}]\boldsymbol{V}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}} \\ &\quad + \left[\boldsymbol{HGk} - \frac{\boldsymbol{AV}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + (\boldsymbol{q} - \boldsymbol{p})\sigma_a^2}{\sigma_a^2 + [1 \quad \boldsymbol{0}]\boldsymbol{V}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}}\right] \\ &\quad \times \left[\sigma_a^2 + [1 \quad \boldsymbol{0}]\boldsymbol{V}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}\right] \\ &\quad \times \left[\boldsymbol{HGk} - \frac{\boldsymbol{AV}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + (\boldsymbol{q} - \boldsymbol{p})\sigma_a^2}{\sigma_a^2 + [1 \quad \boldsymbol{0}]\boldsymbol{V}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}}\right]^T\Bigg]\boldsymbol{c} + \sigma_a^2 \end{aligned}$$

The right-hand side of the above equation consists of four terms in the square brackets. The first three of these are independent of $\boldsymbol{k}$, but the last one, which is positive, depends on $\boldsymbol{k}$. The quantity on the left hand side will obtain its minimum value when

$$\boldsymbol{HGk} - \frac{\boldsymbol{AV}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + (\boldsymbol{q} - \boldsymbol{p})\sigma_a^2}{\sigma_a^2 + [1 \quad \boldsymbol{0}]\boldsymbol{V}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}} = \boldsymbol{0}$$

Therefore the optimal gains of a PID controller are given by

$$\begin{bmatrix} k_p \\ k_i \\ k_d \end{bmatrix} = \frac{\boldsymbol{G}^{-1}[\boldsymbol{H}^T\boldsymbol{H}]^{-1}\boldsymbol{H}^T\left[\boldsymbol{AV}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + (\boldsymbol{q} - \boldsymbol{p})\sigma_a^2\right]}{\sigma_a^2 + [1 \quad \boldsymbol{0}]\boldsymbol{V}\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}} \tag{4}$$

with $\boldsymbol{V}$ as the solution of eqn. 3.

The solution of eqn. 3 for $\boldsymbol{V}$ can easily be established by an iterative approach. The approach is described below.

(i) Set up the initial condition

$$\boldsymbol{V}_0 = [\boldsymbol{q} - \boldsymbol{p}][\boldsymbol{q} - \boldsymbol{p}]^T\sigma_a^2$$

and from this equation calculate the initial gain vector

$$k_0 = \frac{G^{-1}[H^T H]^{-1} H^T \left[A V_0 \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} + (q - p)\sigma_a^2 \right]}{\sigma_a^2 + [1 \quad \mathbf{0}] V_0 \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}}$$

(ii) Use eqn. 3 to calculate the variance matrix V_{t+1} for the next iteration as below

$$V_{t+1} = A^* V_t A^{*T} + bb^T \sigma_a^2 \tag{5}$$

(iii) Use eqn. 4 to calculate the gain vector k_t as below

$$k_t = \frac{G^{-1}[H^T H]^{-1} H^T \left[A V_t \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} + (q - p)\sigma_a^2 \right]}{\sigma_a^2 + [1 \quad \mathbf{0}] V_t \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}} \tag{6}$$

(iv) Iterate between eqns. 5 and 6 until convergence, i.e.

$$V_{t+1} = V_t = V$$

(v) Use the converged value of the variance matrix V to calculate the gain vector k.

Such an approach is similar to that used to calculate the steady state Kalman filter, and eqn. 3 is a Riccati equation.

3 Optimal deterministic PID controller

In control there are two problems. In the servo problem, the load is constant but the set point undergoes change while in the regulator problem the set point is kept constant and the load undergoes change. Stochastic control models the load as a time series and therefore belongs to the regulator class. However, as there is a duality between stochastic load changes and randomly occurring deterministic set point changes [9], the theory presented in Section 2 can also be applied to servo problems. Consider the deterministic system

$$y_{t+f+1} = \frac{\omega(z^{-1})}{\delta(z^{-1})} u_t$$

with a set point change ysp_{t+f+1} at time $t + f + 1$. The above equation can be written as

$$y_{t+f+1} - ysp_{t+f+1} = y^*_{t+f+1} = \frac{\omega(z^{-1})}{\delta(z^{-1})} u_t - ysp_{t+f+1}$$

If one can express the set point changes ysp_{t+f+1} in the z domain, then the above equation is equivalent to eqn. 1 and y_t can be controlled to follow ysp_t with minimum variance. A few common set point changes are described below.

Step change to a new level

$$ysp_t = \frac{a_t}{1 - z^{-1}}$$

Ramp change to a new level

$$ysp_t = \frac{a_t}{(1 - z^{-1})^2}$$

Exponential change to a new level

$$ysp_t = \frac{a_t}{(1 - z^{-1})(1 - bz^{-1})}$$

4 Examples

In this Section two examples are given to illustrate the presented theory. First consider a stochastic control system described by the following equation

$$y_t = \frac{0.168}{1 - 0.908z^{-1}} u_{t-1} + \frac{1}{(1 - z^{-1})(1 - 0.3z^{-1} - 0.17z^{-2})} a_t$$

with a_t as the generating white noise having variance $\sigma_a^2 = 2.37$. This system gives

$$\omega(z^{-1}) = 0.168$$
$$\delta(z^{-1}) = 1 - 0.908z^{-1}$$
$$\theta(z^{-1}) = 1$$
$$\phi(z^{-1}) = 1 - 1.3z^{-1} + 0.13z^{-2} + 0.17z^{-3}$$
$$f = 0$$

and

$$\alpha(z^{-1}) = 1 - 1.908z^{-1} + 0.908z^{-2}$$
$$\beta(z^{-1}) = 1 - 3.208z^{-1} + 3.5184z^{-2} - 1.25844z^{-3} - 0.20632z^{-4} + 0.15436z^{-5}$$
$$\gamma(z^{-1}) = 0.168 - 0.2184z^{-1} + 0.02184z^{-2} + 0.02856z^{-3}$$

Therefore

$$p = \begin{bmatrix} 1.908 \\ -0.908 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad q = \begin{bmatrix} 3.208 \\ -3.5184 \\ 1.25844 \\ 0.20632 \\ -0.15436 \\ 0 \end{bmatrix}$$

$$H = \begin{bmatrix} -0.168 & 0 & 0 \\ 0.2184 & -0.168 & 0 \\ -0.02184 & 0.2184 & -0.168 \\ -0.02856 & -0.02184 & 0.2184 \\ 0 & -0.02856 & -0.02184 \\ 0 & 0 & -0.02856 \end{bmatrix}$$

The variance matrix V was found to be

$$V = \begin{bmatrix} 0.0588 & -0.0539 & -0.0022 & -0.0016 & -0.0008 & -0.0002 \\ & 0.0519 & 0.0017 & 0.0006 & -0.0000 & -0.0002 \\ & & 0.0006 & 0.0002 & -0.0001 & -0.0001 \\ & & & 0.0005 & 0.0002 & -0.0000 \\ & & & & 0.0005 & 0.0002 \\ sym & & & & & 0.0004 \end{bmatrix}$$

With the optimal gains

$$\begin{bmatrix} k_p \\ k_i \\ k_d \end{bmatrix} = \begin{bmatrix} -6.2136 \\ -2.1582 \\ 0.4433 \end{bmatrix}$$

and the feedback output variance $\sigma_y^2 = 2.4288$.

To check the validity of the result, in the following a few examples with known theoretical controller gains are considered. First consider the system

$$y_t = \frac{0.168}{1 - 0.908z^{-1}} u_{t-1} + \frac{1 - 0.2z^{-1}}{1 - z^{-1}} a_t$$

with $\sigma_a^2 = 2.0$. This system gives

$$\omega(z^{-1}) = 0.168$$
$$\delta(z^{-1}) = 1 - 0.908z^{-1}$$
$$\theta(z^{-1}) = 1 - 0.2z^{-1}$$
$$\phi(z^{-1}) = 1 - z^{-1}$$
$$f = 0$$

The minimum variance controller for this system obtained by the method of Box and Jenkins [10] is

$$(1 - z^{-1})u_t = -\frac{(1 - 0.2)(1 - 0.908z^{-1})}{0.168} y_t$$
$$= -\frac{0.8}{0.168}(y_t - 0.908y_{t-1})$$

which is a PI cpntroller with the gains $k_p = -4.3238$ and $k_i = -0.4381$. Using the method presented in this paper, it is obtained as

$$\alpha(z^{-1}) = 1 - 2.108z^{-1} + 1.2896z^{-2} - 0.1816z^{-3}$$
$$\beta(z^{-1}) = 1 - 2.908z^{-1} + 2.816z^{-2} - 0.908z^{-3}$$
$$\gamma(z^{-1}) = 0.168 - 0.168z^{-1}$$

Therefore

$$\boldsymbol{p} = \begin{bmatrix} 2.108 \\ -1.2896 \\ 0.1816 \\ 0 \end{bmatrix} \quad \boldsymbol{q} = \begin{bmatrix} 2.908 \\ -2.816 \\ 0.908 \\ 0 \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} -0.168 & 0 & 0 \\ 0.168 & -0.168 & 0 \\ 0 & 0.168 & -0.168 \\ 0 & 0 & 0.168 \end{bmatrix}$$

The variance matrix $\boldsymbol{V}$ was found to be

$$\boldsymbol{V} \times 1.0E10 = \begin{bmatrix} 0.0500 & -0.0956 & 0.0456 & 0.0000 \\ & 0.1228 & -0.0871 & 0.0000 \\ & & 0.0415 & 0.0000 \\ sym. & & & 0.0000 \end{bmatrix}$$

with the gains and feedback output variance as

$$\begin{bmatrix} k_p \\ k_i \\ k_d \end{bmatrix} = \begin{bmatrix} -4.3238 \\ -0.4381 \\ 0 \end{bmatrix} \quad \sigma_y^2 = 2.0$$

which are identical to the solutions obtained previously.

In the next example, consider the system

$$y_t = \frac{0.250}{1 - 0.8z^{-1}} u_{t-1} + \frac{1 - 0.2z^{-1}}{1 - 0.3z^{-1}} a_t$$

with $\sigma_a^2 = 2.0$. This system gives

$$\omega(z^{-1}) = 0.250$$
$$\delta(z^{-1}) = 1 - 0.8z^{-1}$$
$$\theta(z^{-1}) = 1 - 0.2z^{-1}$$
$$\phi(z^{-1}) = 1 - 0.3z^{-1}$$
$$f = 0$$

This system is known to have zero integral gain because the disturbance is stationary in nature. Using the approach presented in this paper, the following is obtained

$$\alpha(z^{-1}) = 1 - 2.0z^{-1} + 1.16z^{-2} - 0.16z^{-3}$$
$$\beta(z^{-1}) = 1 - 2.1z^{-1} + 1.34z^{-2} - 0.24z^{-3}$$
$$\gamma(z^{-1}) = 0.25 - 0.075z^{-1}$$

Therefore

$$\boldsymbol{p} = \begin{bmatrix} 2.000 \\ -1.160 \\ 0.160 \\ 0 \end{bmatrix} \quad \boldsymbol{q} = \begin{bmatrix} 2.100 \\ -1.340 \\ 0.240 \\ 0 \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} -0.250 & 0 & 0 \\ 0.075 & -0.250 & 0 \\ 0 & 0.075 & -0.250 \\ 0 & 0 & 0.075 \end{bmatrix}$$

The variance matrix $\boldsymbol{V}$ was found to be

$$\boldsymbol{V} = \begin{bmatrix} 0.1420 & -0.1375 & 0.0219 & 0 \\ & 0.1343 & -0.0213 & 0 \\ & & 0.0036 & 0 \\ sym. & & & 0.0002 \end{bmatrix}$$

with the optimal gains

$$\begin{bmatrix} k_p \\ k_i \\ k_d \end{bmatrix} = \begin{bmatrix} -0.5568 \\ 0.0000 \\ -0.1156 \end{bmatrix}$$

and the feedback output variance $\sigma_y^2 = 2.1420$.

The results of the examples presented in this Section were obtained from a computer program written in C (see Appendix 7). The program converged for many cases under testing but is known to have difficulty in cases with a delay in the system or with a nonminimum phase. These cases are under investigation and the results will be submitted at a later date.

5 Conclusion

In this paper, a method of obtaining the optimal gains for a discrete PID controller was presented. The method can be used for both stochastic or deterministic systems. A similar approach can also be derived for multivariable systems. The method can be used not only for PID controllers but also for any type of controller with known structure or orders.

6 References

1 ZIEGLER, J.G., and NICHOLS, N.B.: 'Optimum setting for PID controllers', *Trans. ASME*, 1942, **64**, pp. 759–768

2 COHEN, G.H., and COON, G.A.: 'Theoretical considerations of retarded control', *Trans. ASME*, 1953, **75**, p. 827

3 ZERVOS, C., BÉLANGER, P.R., and DUMONT, G.A.: 'On PID controller tuning using orthonomal series identification', *Automatica*, 1988, **24**, (2), pp. 165–175

4 STOJIC, M.R., and PETROVIĆ, T.B.: 'Design of digital PID stand-alone single loop controller', *Int. J. Control*, 1986, **43**, (4), pp. 1229–1242

5 MACGREGOR, J.F., WRIGHT, J.D., and HUYNH, M.H.: 'Optimal tuning of digital PID controllers using dynamic-stochastic models', *IEC Process Des. Dev.*, 1975, **14**, pp. 398–402

6 MEO, J.A.C., MEDANIC, J.V., and PERKINS, W.R.: 'Design of digital PI + dynamic controllers using projective control', *Int. J. Control*, 1986, **43**, (2), pp. 539–559

7 RADKE, F., and ISERMANN, R.: 'A parameter-adaptive PID controller with stepwise parameter optimization', *Automatica*, 1987, **23**, pp. 449–457

8 ÅSTRÖM, K.J., and HÄGGLUND, T.: 'Automatic tuning of PID controllers' (ISA, Research Triangle Park, N.C., 1988)

9 MACGREGOR, J.F., HARRIS, T.J., and WRIGHT, J.D.: 'Duality between the control of processes subject to randomly occurring deterministic disturbances and ARIMA stochastic disturbances', *Technometrics*, 1984, **26**, (4), pp. 389–397

10 BOX, G.E.P., and JENKINS, G.M.: 'Time series analysis, forecasting and control' (Holden Day, San Francisco, 1970)

7 Appendix

```
/****************************************

Module:        main( )

Description:   This module reads in the system parameters and set up the right system vectors and matrices, then it
               solves the Riccati equation and calculate the PID controller gains.

Author:        K. M. Vu      9-May-91

****************************************/

#include <c:\tc\stdio.h>
#include <c:\tc\stdlib.h>
#include <c:\tc\math.h>
#include <c:\tc\malloc.h>
#include <c:\tc\nrutil.c>

#define NDIMS 20
int    i, j, l;                      /* Do loop counters */
int    ndelay;                       /* Pure delay in the system */
int    nomega;                       /* Number of zeros of the transfer function */
int    ndelta;                       /* Number of poles of the transfer function */
int    nphi;                         /* The autoregressive order */
int    ntheta;                       /* The moving average order */
int    n;                            /* The resulting order of the time series */
int    inter;                        /* The iteration number */
int    mvflag;                       /* The minimum variance controller flag */
int    maxiter=500;                  /* The maximum iteration number allowable */

float  omega[NDIMS];                 /* The polynomial of the zeros */
float  delta[NDIMS];                 /* The polynomial of the poles */
float  phi[NDIMS];                   /* The autoregressive polynomial */
float  theta[NDIMS];                 /* The moving average polynomial */
float  var;                          /* The variance of the white noise */

float  alpha[NDIMS];                 /* The resulting time series polynomial */
float  beta[NDIMS];                  /* The resulting time series polynomial */
float  gamma[NDIMS];                 /* The resulting time series polynomial */
float  p[NDIMS];                     /*The p vector */
float  q[NDIMS];                     /* The q vector */
float  h[NDIMS][3];                  /* The h matrix */
float  k[3];                         /* The controller gain vector */

float  ht[3][NDIMS];                 /* Working matrix */
float  hth[3][3];                    /* Working matrix */
float  hthinv[3][3];                 /* Working matrix */
float  hthinvht[3][NDIMS];           /* Working matrix */
float  ginvhthinvnt[3][NDIMS];       /* Working matrix */
float  hg[NDIMS][3];                 /* Working matrix */
float  hgk[NDIMS];                   /* Working matrix */

float  a[NDIMS][NDIMS];              /* The Riccati equation matrix */
float  astar[NDIMS][NDIMS];          /* The Riccati equation matrix */
float  astart[NDIMS][NDIMS];         /* The Riccati equation matrix */
float  astarv[NDIMS][NDIMS];         /* The Riccati equation matrix */
float  vt[NDIMS][NDIMS];             /* The Riccati equation matrix */
float  vt1[NDIMS][NDIMS];            /* The Riccati equation matrix */
float  av[NDIMS][NDIMS];         /* The Riccati equation matrix */
float  av1[NDIMS];               /* The Riccati equation matrix */
float  b[NDIMS];                 /* The Riccati equation matrix */
float  bbt[NDIMS][NDIMS];        /* The Riccati equation matrix */
float  vt_sq;                    /* The current variance matrix norm */
float  vt1_sq;                   /* The next variance matrix norm */
float  b_sq;                     /* The b vector norm */
float  pc_change;                /* The % change in variance matrix norm */
float  accuracy=1.0e—06;         /* The iteration stopping criterion */
float  diff[2]={1,-1};                              /* The difference polynomial */
float  g[3][3]={{1,1,1}, {-1,0,-2}, {0,0,1}};       /* A constant matrix */
float ginv[3][3]={{0,-1,-2}, {1,1,1}, {0,0,1}};     /* A constant matrix */
float  sum;                                         /* The running sum parameter */
main( )
{
        /* Read in system parameters */
  printf("Enter the number of zeroes of the transfer function : ");
  scanf("%d", &nomega);
  for (i=0; i<=nomega; i++)
  {
  printf("Enter the omega's : ");
  scanf("%f", &omega[i]);
}
printf("Enter the number of poles of the transfer function : ");
scanf("%d", &ndelta);
for (i=0; i<=ndelta; i++)
{
  printf("Enter the delta's : ");
  scanf("%f", &delta[i]);
}
printf("Enter the delay period not counting the discretization one : ");
scanf("%d", &ndelay);
printf("Enter the autoregressive order of the disturbance : ");
scanf("%d", &nphi);
for (i=0; i<=nphi; i++)
{
  printf("Enter the phi's : ");
  scanf("%f", &phi[i]);
}
printf("Enter the moving average order of the disturbance : ");
scanf("%d", &ntheta);
for (i=0; i<=ntheta; i++)
{
  printf("Enter the theta's : ");
  scanf("%f", &theta[i]);
}
printf("Enter the variance of the white noise disturbance : ");
scanf("%f", &var);
        /* Form the necessary vectors and parameter matrix */
i =(int) max(ndelta+nphi+1,nomega+nphi+ndelay+3);
n=(int) max(i,ndelta+ntheta+1);
        /* Form vector p */
for (i=0; i<=ndelta+ntheta; i++)
{
  gamma[i]=0;
  for (j=0; j<=i; j++) gamma[i] += delta[j]*theta[i-j];
}
for (i=0; i<=ndelta+ntheta+1; i++)
{
  alpha[i]=0;
  for (j=0; j<=1; j++) alpha[i] += gamma[i-j]*diff[j];
}
for (i=0; i<n; i++)
{
  gamma[i]=0;
  if (i<=ndelta+ntheta+1) p[i]= -alpha[i+1]; else p[i]=0;
```

```
}
          /* Form vector q */
for (i=0; i<=ndelta+nphi; i++)
{
  gamma[i]=0;
  for (j=0; j<=i; j++) gamma[i] += delta[j]*phi[i-j];
}
for (i=0; i<=ndelta+nphi+1; i++)
{
  beta[i]=0;
  for (j=0; j<=1; j++) beta[i] += gamma[i-j]*diff[j];
}
for (i=0; i<n; i++)
{
  gamma[i]=0;
  if (i<=ndelta+nphi+1) q[i]--beta[i+1]; else q[i]=0;
}
          /* Form matrix h */
for (i=0; i<=nomega+nphi; i++)
{
  gamma[i]=0;
  for (j=0; j<=i; j++) gamma[i] += omega[j]*phi[i-j];
}
for (i=0; i<n; i++)
{
  h[i][0]=0;
  if (i> =ndelay&&i< =n-ndelay-2) h[i][0]= -gamma[i-ndelay];
}
for (i=0; i<n; i++)
{
  for (j=1; j<3; j++)
  {
    h[i][j]=0;
    if (i>0) h[i][j]=h[i-1][j-1];
  }
}
/*
printf("\n Print out system parameters\n");
for (i=0; i<n; i++)
{
  printf(" %8.5f %8.5f ",p[i],q[i]);
  for (j=0; j<3; j++) printf(" %8.5f ",h[i][j]);
  printf("\n");
}
*/
          /* Form matrix A */
for (i=0; i<n; i++)
{
  for (j=0; j<n; j++)
  {
    a[i][j]=0;
    if (j= =0) a[i][j]=q[i];
    if (j= =i+1) a[i][j]=1;
    if (j!=0) astar[i][j]=a[i][j];
  }
}
for (i=0; i<3; i++)
{
  for (j=0;j<n;j++) ht[i][j]=h[j][i];
}
for (i=0;i<3;i++)
{
  for(j=0;j<3;j++)
  {
    hth[i][j]=0;
    for (l=0;l<n;l++) hth[i][j]+ =ht[i][l]*h[l][j];
    hthinv[i][j]=hth[i][j];
  }
```

```
}
          /*Calculate the inverse of hth */
for (l=0;l<3;l++)
{
  /* Overwrite the lower half of the original matrix with its Cholesky triangle */
  for(j=0;j<=l-1;j++)
    hthinv[l][l]-=hthinv[l][j]*hthinv[l][j];
  hthinv[l][l]=(float)sqrt(hthinv[l][l]);
  for (i=l+1; i<3; i++)
  {
    for (j=0; j<=l-1; j++)
      hthinv[i][l]-=hthinv[i][j]*hthinv[l][j];
    hthinv[i][l]=hthinv[i][l]/hthinv[l][l];
  }
}
/* Now compute the inverse of the Cholesky triangle */
for (l=0; l<3; l++)
{
  for (j=0; j<=l; j++)
  {
    if (j==l)
      hthinv[l][j]=1,0/hthinv[l][l];
    else
    {
      sum=0;
      for (i=j; i<l; i++) sum-=hthinv[l][i]*hthinv[i][j];
      hthinv[l][j]=sum/hthinv[l][l];
    }
  }
}
/* Compute the inverse of the original matrix */
for (l=0; l<3; l++)
{
  for (j=0; j<=l; j++)
  {
    sum=0;
    for (i=l; i<3; i++) sum+=hthinv[i][l]*hthinv[i][j];
    hthinv[l][j]=sum;
    hthinv[j][l]=hthinv[l][j];
  }
}
          /* Calculate the matrices for the Riccati equation */
for (i=0;i<3;i++)
{
  for (j=0;j<n;j++)
  {
    hthinvht[i][j]=0;
    for (l=0;l<3;l++) hthinvht[i][j]+=hthinv[i][l]*ht[l][j];
  }
}
for (i=0;i<3;i++)
{
  for (j=0;j<n;j++)
  {
    qinvhthinvht[i][j]=0;
    for (l=0;l<3;l++) ginvhthinvht[i][j]+=ginv[i][l]*hthinvht[l][j];
  }
}
for (i=0;i<n;i++)
{
  for (j=0;j<3;j++)
  {
    hg[i][j]=0;
    for (l=0;l<3;l++) hg[i][j]+=h[i][l]*g[l][j];
  }
}
          /* Set up initial condition */
for (j=0;j<3;j++)
```

```
{
  k[j]=0;
  for (l=0;l<n;l++) k[j]+=ginvhthinvht[j][l]*(q[l]-p[l]);
}
for (i=0;i<n;i++)
{
  hgk[i]=0;
  for (l=0;l<3;l++) hgk[i]+=hg[i][l]*k[l];
}
for (j=0;j<n;j++) b[j]=q[j]-p[j]-hgk[j];
vt_sq=0;
for (j=0;j<n;j++)
{
  for (l=0;l<n;l++)
  {
    vt[j][l]=b[j]*b[l]*var;
    vt_sq+=vt[j][l]*vt[j][l];
  }
}
mvflag=0;
        /* And start the iteration */
iter=1;
pc_change=0.1;
while (pc_change>accuracy&&iter<maxiter&&!mvflag)
{
        /* Calculate the controller gain vector */
  for (i=0;i<n;i++)
  {
    for (j=0;j<n;j++)
    {
      av[i][j]=0;
      for (l=0;l<n;l++) av[i][j]+=a[i][l]*vt[l][j];
    }
  }
  for (j=0;j<n;j++) av1[j]=av[j][0]+(q[j]-p[j])*var;
  for (j=0;j<3;j++)
  {
  k[j]=0;
  for (l=0;l<n;l++) k[j]+=ginvhthinvht[j][l]*av1[l];
  k[j]/=var+vt[0][0];
}
/* Calculate the Riccati equation */
for (i=0;i<n;i++)
{
  hgk[i]=0;
  for (l=0;l<3;l++) hgk[i]+=hg[i][l]*k[l];
}
b_sq=0;
for (j=0;j<n;j++)
{
  astar[j][0]=q[j]-hgk[j];
  b[j]=q[j]-p[j]-hgk[j];
  b_sq +=b[j]*b[j];
}
if(b_sq<=1.0e-06&&vt_sq<=1.0e-06) mvflag=1;
for (j=0;j<n;j++)
{
  for (l=0;l<n;l++)
  {
    bbt[j][l]=b[j]*b[l]*var;
    astart[j][l]=astar[l][j];
  }
}
/*Recalculate the variance matrix */
for (i=0;i<n;i++)
{
  for (j=0;j<n;j++)
  {
    astarv[i][j]=0;
    for (l=0;l<n;l++) astarv[i][j]+=astar[i][l]*vt[l][j];
  }
}
vt1_sq=0;
for (i=0;i<n;i++)
{
  for (j=0;j<n;j++)
  {
    vt1[i][j]=bbt[i][j];
    for (l=0;l<n;l++) vt1[i][j]+=astarv[i][l]*astart[l][j];
    vt1_sq+=vt1[i][j]*vt1[i][j];
  }
}
/* And percent of accuracy */
pc_change=fabs(vt1_sq-vt_sq) / vt_sq;
/*
  printf("Iteration=%d kp=%8.4f ki=%8.4f kd=%8.4f Output variance=%8.4f\
*/
for (i=0;i<n;i++)
{
  for (j=0;j<n;j++) printf("%8.4f ",vt[i][j]);
  printf("\n");
}
printf("\n");
/* Update iteration parameters */
iter+=1;
vt_sq=vt1_sq;
for(j=0;j<n;j++)
{
  for (l=0;l<n;l++) vt[j][l]=vt1[j][l];
  }
}
          /* Print out results */
if (fabs(pc_change)<accuracy)
{
  printf("\nConvergence at iteration %d\n",iter);
  printf("\nThe variance matrix is:\n");
  for (i=0;i<n;i++)
  {
    for (j=0; j<n; j++) printf("%11.4e ", vt[i][j]);
    printf("\n");
  }
  printf("\nThe controller gains are:\n");
  printf("The proportional gain : %8.4f\n",k[0]);
  printf("The integral gain : %8.4f\n",k[1]);
  printf("The derivative gain : %8.4f\n",k[2]);
  printf("\nThe output variance is : %8.4f\n",var+vt[0][0]);
}
else if(mvflag)
{
  printf("\nThe PID is the minimum variance controller\n");
  printf("\nThe controller gains are:\n");
  printf("The proportional gain : %8.4f\n",k[0]);
  printf("The integral gain ' %8.4f\n",k[1]);
  printf("The derivative gain : %8.4f\n",k[2]);
  printf("\nThe output variance is : %8.4f\n",var);
}
else
{
  printf("\nFail to converge after %d iterations\n",iter);
  }
}
```