Center for Quality and Productivity Improvement
UNIVERSITY OF WISCONSIN
610 Walnut Street
Madison, Wisconsin 53705

(608) 263–2520
(608) 263–1425 FAX

Report No. 147

Choice of Repeated and Bounded Adjustment Schemes

Alberto Luceño, Francisco J. González, and Jaime Puig-Pey

July 1996

CQPI
UNIVERSITY OF WISCONSIN

Report No. 147

Report Series
in Quality and Productivity

Choice of Repeated and Bounded Adjustment Schemes

Alberto Luceño, Francisco J. González, and Jaime Puig-Pey

E.T.S. DE INGENIEROS DE CAMINOS, UNIVERSITY OF CANTABRIA, 39005 SANTANDER, SPAIN

ABSTRACT

An important problem in process adjustment using feedback is how often to sample the process and when and by how much to apply an adjustment. Minimum-cost feedback schemes based on simple, but practically interesting, models for disturbances and dynamics have been discussed in several particular cases. The more general situation in which there may be measurement and adjustment errors, deterministic process drift, and costs of taking an observation, of making an adjustment, and of being off target, is considered in this article. Assuming all these costs to be known, a numerical method to minimize the overall expected cost is presented. This numerical method provides the optimal sampling interval, action limits, and amount of adjustment; and the resulting average adjustment interval, mean squared deviation from target, and minimum overall expected cost. When the costs of taking an observation, of making an adjustment, and of being off target are not known, the method can be used to choose a particular scheme by judging the advantages and disadvantages of alternative options considering the mean squared deviation they produce, the frequency with which they require observations to be made, and the resulting overall length of time between adjustments. Computer codes that perform the required computations are provided in the appendices and applied to find optimal adjustment schemes in three real examples of application.

KEYWORDS: Adjustment error, average adjustment interval, mean squared deviation, minimum cost, process control, process drift, tool-wear.

Choice of Repeated and Bounded Adjustment Schemes

Alberto Luceño, Francisco J. González, and Jaime Puig-Pey

E.T.S. de Ingenieros de Caminos, University of Cantabria, 39005 Santander, Spain

1. INTRODUCTION

Two complementary approaches to process control may be called *process monitoring* and *process adjustment* (see Box, Jenkins, and Reinsel, 1994, Part IV). The former is part of what is called Statistical Process Control (SPC) and uses process monitoring charts such as Shewhart, Cusum, and Roberts' Exponentially Weighted Moving Average (EWMA) charts to verify that a process remains in a stable "state of control" and eliminate removable sources of variation from the system. The latter, sometimes called Engineering Process Control (EPC), uses, in the simplest case, manipulation of some compensating variable X to adjust a quality characteristic to be close to target employing feedback or feedforward control or sometimes a combination of them. This article concerns feedback control schemes for process adjustment or EPC.

1.1. MODELS FOR DYNAMICS AND DISTURBANCES

Let X_t be the setting of the adjustment variable at time t. For example, X_t might represent the current catalyst formulation setting, which can be manipulated to adjust the viscosity of a polymer as close as possible to some target.

Suppose that observations and opportunities for adjustment occur at discrete times $\ldots\ t-1, t, t+1\ \ldots$ equispaced at what we consider as *unit* intervals. Suppose that an adjustment $x_t = X_t - X_{t-1}$ in the "input" compensatory variable will produce gx_t units of change in the "output" quality characteristic which has its full effect in one unit time interval. Moreover, define the disturbance z_t at time t as the *deviation from target* of the "output" quality characteristic that would occur if *no* compensatory action were taken (i.e., if $x_{t-1} = x_{t-2} = \ldots = 0$). Then, at time $t+1$, the disturbance is z_{t+1} and the cumulative effect of the adjustments $x_t, x_{t-1}, \ldots$ is $g(x_t + x_{t-1} + \ldots) = gX_t$, so that the deviation from target with the adjustment variable still set at the level X_t is $\varepsilon_{t+1} = z_{t+1} + gX_t$.

A very important form of adjustment employed in EPC requires setting X_t so as just to cancel an EWMA forecast $\hat{z}_{t+1}$ of z_{t+1} computed at time t (see for example Box, Jenkins, and Reinsel, 1994), so that

$$X_t = -\hat{z}_{t+1}/g \tag{1.1}$$

with

$$\hat{z}_{t+1} = \lambda(z_t + \theta z_{t-1} + \theta^2 z_{t-2} \ldots) \tag{1.2}$$

where $\theta = 1 - \lambda$ and $0 < \lambda \leq 1$. Although this adjustment policy requires no assumptions concerning the disturbance, it was shown by Muth (1960) that the EWMA (1.2) produces a minimum mean squared error (MMSE) forecast if the disturbance is generated by the IMA(0, 1, 1) time series model

$$z_{t+1} - z_t = a_{t+1} - \theta a_t\ , \tag{1.3}$$

where the shocks a_t are independent random variables with mean 0 and variance σ_a^2.

1.2. OVERALL EXPECTED COST FUNCTION PER UNIT TIME INTERVAL

Minimum-cost schemes generally depend on the cost $C_T > 0$ of being off target for one time interval by an amount of σ_a, and the fixed costs incurred each time the process is observed C_M and/or adjusted C_A.

C_M and C_A negligible. If the operating cost of being off target is a quadratic function of the

deviation from target and is the *only* control cost, then the control scheme in which "repeated adjustments" that amount to (1.1) are made in each time interval provides minimum cost. When using this scheme, the process is sampled and adjusted every unit time interval, to take advantage that C_M and C_A are 0, and the forecasted deviation from target after an adjustment is made becomes 0.

C_M negligible. The situation in which there is a fixed adjustment cost C_A (for example, a cost of stopping a machine and changing a tool) was studied by Box and Jenkins (1963). These authors showed that, in this situation, the repeated adjustment scheme is no longer optimum and a "bounded adjustment" scheme should be used to yield minimum overall expected cost. This scheme requires that action $X_t = -\hat{z}_{t+1}/g$ such as to produce a compensation $-\hat{z}_{t+1}$ should be taken only when $|\hat{\varepsilon}_{t+1}|$ crosses either of the lines defined by $|\hat{\varepsilon}_{t+1}| = L$, where the forecasted deviation from target $\hat{\varepsilon}_{t+1} = \hat{z}_{t+1} + g X_t$ is computed assuming that no adjustment is made at time t, i.e., using $X_t = X_{t-1}$. The average interval between adjustments measured in terms of the original *unit* intervals is an increasing function of L that equals 1 if $L = 0$. The optimal value of the action limit L depends on the relative importance of the costs C_A and C_T and, in particular, is an increasing function of C_A/C_T that equals 0 when $C_A = 0$. The forecasted deviation from target after an adjustment is made becomes $|\hat{\varepsilon}_{t+1}| = 0$, as in the repeated adjustment scheme.

C_M, C_A, and C_T non-negligible. Recently, Box and Kramer (1992) generalized this minimum-cost approach by considering also the sampling cost C_M, i.e., the cost of taking each observation. Supposing the process was initially sampled at unit intervals, they considered the possibility of sampling at intervals m units apart, where m is an integer, that is, of reducing the observations and opportunities for adjustment to the discrete equispaced times ..., $tm-m$, tm, $tm+m$, ... If the disturbance is generated by the time series model (1.3), then the sampled disturbance is also an IMA(0, 1, 1) time series model like (1.3) but with parameters $\lambda_m = 1 - \theta_m$ and σ_m, where

$$\lambda_m^2 \sigma_m^2 = m\lambda^2\sigma_a^2 \ , \ \theta_m \sigma_m^2 = \theta\sigma_a^2 . \qquad (1.4)$$

The overall expected cost C per unit interval is

$$C = \frac{C_M}{m} + \frac{C_A}{\mathrm{AAI}} + C_T \frac{\mathrm{MSD}}{\sigma_a^2}, \qquad (1.5)$$

where AAI is the average interval between adjustments measured in terms of the original *unit* intervals, and MSD is the mean squared deviation defined by

$$\mathrm{MSD} = \frac{1}{\mathrm{AAI}} E\left[\sum_{j=0}^{n-1}\sum_{k=1}^{m} \varepsilon_{jm+k}^2\right], \qquad (1.6)$$

where $mn \geq m$ is the interval between adjustments and $mE(n) = \mathrm{AAI}$.

In this case, the minimum-cost scheme is also a bounded adjustment scheme requiring action $X_{tm} = -\hat{z}_{tm+m}/g$, such as to produce a compensation $-\hat{z}_{tm+m}$, only when $|\hat{\varepsilon}_{tm+m}|$ crosses either of the lines defined by $|\hat{\varepsilon}_{tm+m}| = L$, where $\hat{\varepsilon}_{tm+m} = \hat{z}_{tm+m} + g X_{tm}$ is computed with $X_{tm} = X_{tm-m}$. The AAI is an increasing function of L that equals m for $L = 0$, and the forecasted deviation from target after an adjustment is made becomes $|\hat{\varepsilon}_{tm+m}| = 0$ as when $C_M = 0$. However, the process is now sampled every m original unit intervals and the optimal values of the sampling interval m and the action limit L depend on the costs C_M, C_A, and C_T.

Several methods for obtaining approximations for the AAI and MSD in (1.5) have been given by Box and Jenkins (1963), Kramer (1989), and Srivastava and Wu (1991), and for the case $\lambda = 1$, by Crowder (1986) and Adams and Woodall (1989). More recently, Box and Luceño (1994) provided two linear integral equations that characterize exactly the AAI and MSD when $m = 1$. These integral equations are conceptually simple because they can be regarded as infinite dimensional linear systems of equations, and can be solved numerically by discretization.

Other writers including Taguchi (1981) and Srivastava and Wu (1991) have considered a similar problem; but they use charts in which action is based on the actual value z_t instead of the EWMA $\hat{z}_{t+1}$. On the assumptions mentioned above, their schemes are not optimal when the non-stationarity measure λ is different from 1 (see Adams and Woodall, 1989). Luceño (1993a) showed that the main loss of efficiency arose from making an adjustment producing a change $-z_t$ instead of the optimal change $-\hat{z}_{t+1}$.

Deterministic process drift. The drift version of this problem has been characterized as the "tool-wear" problem and has been considered by several authors; see, for example, Drezner and Wesolowsky (1989) and Schneider, Tang, and O'Cinneide, (1990). Recently, using a slightly different but equivalent formulation, Jensen and Vardeman (1993) developed optimal adjustment policies in the presence of deterministic process drift and random adjustment error for the particular case in which the sampling cost C_M was negligible. These authors showed that, disregarding a short starting period, the action limits and amount of adjustment for the optimal scheme converge quickly to limiting values. If the disturbance is now assumed to be generated by the IMA(0, 1, 1) time series model with non-random drift β per unit time interval

$$z_{t+1} - z_t = \beta + a_{t+1} - \theta a_t ,$$

then the limiting optimal scheme is such that action $X_t = (-\hat{z}_{t+1} + s)/g$ to produce a compensation $-\hat{z}_{t+1} + s$ should be taken only when $\hat{\varepsilon}_{t+1}$ crosses either of the lines defined by $L_1 < \hat{\varepsilon}_{t+1} < L_2$, where $\hat{\varepsilon}_{t+1} = \hat{z}_{t+1} + g X_t$ with $X_t = X_{t-1}$, for some non-necessarily symmetric action limits L_1 and L_2 and overcompensation s. The forecast $\hat{z}_{t+1}$ is no longer given by (1.2) but

$$\hat{z}_{t+1} = \beta / \lambda + \lambda (z_t + \theta z_{t-1} + \theta^2 z_{t-2} \dots).$$

In this case, the minimum-cost scheme is a "non-symmetric bounded adjustment" scheme, the AAI is an increasing function of $-L_1$ and L_2 that equals 1 when $L_1 = L_2$, and the forecasted deviation from target after an adjustment is made becomes $\hat{\varepsilon}_{t+1} = s$, i.e., the overcompensation. Because $C_M = 0$, the process is sampled every unit interval and the optimal values of the action limits L_1 and L_2 and overcompensation s depend on the relative importance of the costs C_A and C_T.

The main purpose of this article is to generalize the schemes described above and to provide an efficient numerical method and corresponding computer code to evaluate, not only the optimal sampling interval m, action limits L_1 and L_2, and overcompensation s, but also the resulting AAI, MSD, and minimum overall expected cost. The general situation where the costs C_M, C_A, and C_T may be non-null and there may be random measurement and adjustment errors and deterministic process drift is considered. The numerical method and computer code are also useful when the costs C_M, C_A, and C_T are not known. Thus the approach in Box and Luceño (1994) is extended to choose m, L_1, L_2, and s on the basis of how much the root mean squared error would need to increase to achieve the advantage of taking samples and making adjustments less frequently.

Detailed consideration of procedures to estimate the parameters of the models for disturbance and dynamics is beyond the scope of this article but can be found in the literature: see, for example, Ansley (1979), Box and Jenkins (1976), Box and MacGregor (1976), Ljung and Box (1979), Luceño (1993b, 1994a, b), Mélard (1984), and Shea (1987, 1989).

The article is organized as follows. The generalized integral equations that characterize exactly the AAI and MSD for the process with deterministic drift β sampled at intervals m units apart are given in Section 2. The generalized cost function when random adjustment errors are present is given in Section 3. Finally, some analytical expressions and numerical details concerning the minimum-cost schemes are given in Section 4 and three examples of application are commented in Section 5.

2. AVERAGE ADJUSTMENT INTERVAL AND MEAN SQUARED DEVIATION

Assume that the disturbance is generated by the IMA(0, 1, 1) time series model with drift

$$z_{t+1} - z_t = \beta + a_{t+1} - \theta a_t , \qquad (2.1)$$

where the shocks a_t are independent normal random variables with mean zero and variance σ_a^2, and β represents the deterministic process drift.

Then the disturbance sampled at intervals m units apart is also an IMA(0, 1, 1) time series model with drift

$$z_{tm+m} - z_{tm} = \beta_m + u_{tm+m} - \theta_m u_{tm}, \quad (2.2)$$

where the shocks u_{tm} are independent normal random variables with mean zero and variance σ_m^2, θ_m and σ_m^2 are given by the Equations in (1.4), and the drift is now $\beta_m = \beta m$. Hence the updating equation for the one step ahead forecast $\hat{z}_{tm}$ is

$$\hat{z}_{tm+m} = \theta_m \hat{z}_{tm} + \lambda_m z_{tm} + \beta_m, \quad (2.3)$$

which is equivalent to

$$\hat{z}_{tm+m} = \hat{z}_{tm} + \lambda_m u_{tm} + \beta_m. \quad (2.4)$$

Because $\hat{\varepsilon}_{tm+m} = \hat{z}_{tm+m} + g X_{tm}$, Equation (2.4) yields $\hat{\varepsilon}_{tm+m} = \hat{\varepsilon}_{tm} + \lambda_m u_{tm} + \beta_m$ if no adjustment is needed at time tm, i.e., if $x_{tm} = X_{tm} - X_{tm-m} = 0$.

Let $A\{\hat{\varepsilon}_{tm}/(\lambda_m \sigma_m)\}$ be the average adjustment interval after $tm - m$, given that the one step ahead forecasted deviation from target is $\hat{\varepsilon}_{tm}$ and the action limits are L_1 and L_2. Then, at time tm, we will conclude that the process needs an adjustment if $\hat{\varepsilon}_{tm+m} \leq L_1$ or $\hat{\varepsilon}_{tm+m} \geq L_2$, where $\hat{\varepsilon}_{tm+m}$ is computed as if $x_{tm} = 0$. In this case the adjustment interval after $tm - m$ will be equal to m. However, if $L_1 < \hat{\varepsilon}_{tm+m} < L_2$, then the process will not need adjustment for at least m unit intervals, i.e., x_{tm} will be really 0, the computed one step ahead forecast will change from $\hat{\varepsilon}_{tm}$ to $\hat{\varepsilon}_{tm+m} = \hat{\varepsilon}_{tm} + \lambda_m u_{tm} + \beta_m$, and the (conditional) average adjustment interval after $tm - m$ will be $m + A\{\hat{\varepsilon}_{tm+m}/(\lambda_m \sigma_m)\}$. Therefore

$$A\left(\frac{s}{\lambda_m \sigma_m}\right) = m$$

$$+ \int_{L_1 < s + \lambda_m u + \beta_m < L_2} A\left(\frac{s + \lambda_m u + \beta_m}{\lambda_m \sigma_m}\right) \varphi(u)\, du \quad (2.5)$$

where $\varphi(u)$ is the probability density function of the shock u, and the notation $s = \hat{\varepsilon}_{tm}$ and $u = u_{tm}$ has been used for simplicity.

Let $M\{\hat{\varepsilon}_{tm}/(\lambda_m \sigma_m)\}$ be the numerator of the MSD, i.e., the expectation in the right hand side of Equation (1.6), given that the one step ahead forecasted deviation from target is $\hat{\varepsilon}_{tm}$ and the action limits are L_1 and L_2. Then a similar argument to that used to obtain (2.5) yields

$$M\left(\frac{s}{\lambda_m \sigma_m}\right) = \sum_{k=1}^{m} E\left\{\varepsilon_{tm-m+k}^2\right\}$$

$$+ \int_{L_1 < s + \lambda_m u + \beta_m < L_2} M\left(\frac{s + \lambda_m u + \beta_m}{\lambda_m \sigma_m}\right) \varphi(u)\, du \quad (2.6)$$

where the notation $s = \hat{\varepsilon}_{tm}$ and $u = u_{tm}$ has been used again. Hence

$$M\left(\frac{s}{\lambda_m \sigma_m}\right) = m s^2 - m\beta(m-1)s$$

$$+ m\left\{\sigma_m^2 - \lambda^2 \sigma_a^2 \frac{(m-1)}{2} + \beta^2 \frac{(m-1)(2m-1)}{6}\right\}$$

$$+ \int_{L_1 < s + \lambda_m u + \beta_m < L_2} M\left(\frac{s + \lambda_m u + \beta_m}{\lambda_m \sigma_m}\right) \varphi(u)\, du, \quad (2.7)$$

as shown in Appendix A.

3. GENERALIZED COST FUNCTION

Equation (1.5) provides the overall expected cost per unit interval when there is no adjustment error, because in this case C_A is the only fixed cost incurred each time the process is adjusted.

However, if adjustment errors with variance σ_η^2 are present, C_A is no longer the only fixed cost incurred each time an adjustment is made, because the level of the process after an adjustment is made differs from the optimal level by a random quantity with variance σ_η^2. In simple cases, the adjustment error is small and dies out quickly. For example, the adjustment error might appear because a small portion of the eventual effect gx_{tm} of an adjustment x_{tm} takes place during a short period *after* the next unit time interval, possibly because of small inaccuracies in the model for the dynamics. Hence the adjustment error that ensues from an adjustment

x_{tm} introduces an additional cost that takes place at time $tm + 1$ and amounts to the cost of being off target at this time by a random quantity with variance σ_η^2. Therefore, σ_η^2 has to be added to the numerator of the MSD in (1.6). Alternatively, $C_T(\sigma_\eta^2/\sigma_a^2)$ can be added to C_A in (1.5), so that the overall expected cost per unit interval when these adjustment errors are present is given by

$$C = \frac{C_M}{m} + \frac{C_A + C_T\left(\sigma_\eta^2/\sigma_a^2\right)}{AAI} + C_T\frac{MSD}{\sigma_a^2}. \quad (3.1)$$

4. NUMERICAL SOLUTION OF THE INTEGRAL EQUATIONS AND MINIMIZATION OF THE COST FUNCTION

We now consider the analytical expressions of the exact solutions of Equations (2.5) and (2.7) and, subsequently, give a numerical method to obtain these solutions and minimize the overall expected cost per unit interval.

4.1. ANALYTICAL EXPRESSIONS OF THE SOLUTIONS OF (2.5) AND (2.7)

Equations (2.5) and (2.7) are Fredholm integral equations of the second kind for $A\{\hat{\varepsilon}_{tm}/(\lambda_m\sigma_m)\}$ and $M\{\hat{\varepsilon}_{tm}/(\lambda_m\sigma_m)\}$, respectively. They differ only in their independent terms and, therefore, their solutions can be expressed in terms of the following linear integral equation:

$$Y_k\left(\frac{s}{\lambda_m\sigma_m}\right) = C_k s^k$$

$$+ \int_{L_1 < s+\lambda_m u+\beta_m < L_2} Y_k\left(\frac{s+\lambda_m u+\beta_m}{\lambda_m\sigma_m}\right)\varphi(u)\,du, (4.1)$$

where C_k is a constant, $k = 0$, 1, or 2, and $\varphi(u) = (2\pi\sigma_m^2)^{-1/2}\exp\{-u^2/(2\sigma_m^2)\}$.

Introducing the transformation

$$w = \frac{s+\lambda_m u+\beta_m}{\lambda_m\sigma_m}$$

in the integral of Equation (4.1) and rewriting the independent term in (4.1) in an obvious way, one has

$$Y_k\left(\frac{s}{\lambda_m\sigma_m}\right) = C_k\left(\lambda_m\sigma_m\right)^k\left(\frac{s}{\lambda_m\sigma_m}\right)^k$$

$$+ \int_{L_1/(\lambda_m\sigma_m)}^{L_2/(\lambda_m\sigma_m)} \frac{Y_k(w)}{\sqrt{2\pi}}\exp\left\{-\frac{1}{2}\left(w - \frac{s}{\lambda_m\sigma_m} - \frac{\beta_m}{\lambda_m\sigma_m}\right)^2\right\}dw,$$

which shows that the solution $Y_k\{s/(\lambda_m\sigma_m)\}$ is not only a function of $s/(\lambda_m\sigma_m)$ but also of $L_1/(\lambda_m\sigma_m)$, $L_2/(\lambda_m\sigma_m)$, $\beta_m/(\lambda_m\sigma_m)$, and $C_k(\lambda_m\sigma_m)^k$, which can be written in the form

$$Y_k\left(\frac{s}{\lambda_m\sigma_m}\right) = C_k\left(\lambda_m\sigma_m\right)^k$$

$$* h_k\left(\frac{L_1}{\lambda_m\sigma_m}, \frac{L_2}{\lambda_m\sigma_m}, \frac{\beta_m}{\lambda_m\sigma_m}, \frac{s}{\lambda_m\sigma_m}\right)$$

for some function $h_k()$.

Now Equation (2.5) is a particular case of Equation (4.1) with $k = 0$ and $C_k = m$. Moreover, because Equation (2.7) is a *linear* integral equation, i.e., an infinite dimensional linear system of equations, its solution can be expressed as the sum of the solutions of Equation (4.1) with $k = 0$ and

$$C_0 = m\left\{\begin{matrix} \sigma_m^2 - \lambda^2\sigma_a^2\dfrac{(m-1)}{2} \\ +\beta^2\dfrac{(m-1)(2m-1)}{6} \end{matrix}\right\}, \quad (4.2)$$

$k = 1$ and $C_1 = -m\beta(m-1)$, and $k = 2$ and $C_2 = m$. Therefore, the AAI is given by

$$\text{AAI} \equiv A\left(\frac{s}{\lambda_m\sigma_m}\right)$$

$$= m h_0\left(\frac{L_1}{\lambda_m\sigma_m}, \frac{L_2}{\lambda_m\sigma_m}, \frac{\beta_m}{\lambda_m\sigma_m}, \frac{s}{\lambda_m\sigma_m}\right), \quad (4.3)$$

and the numerator of the MSD by

$$M\left(\frac{s}{\lambda_m\sigma_m}\right) = C_0 h_0\left(\frac{L_1}{\lambda_m\sigma_m}, \frac{L_2}{\lambda_m\sigma_m}, \frac{\beta_m}{\lambda_m\sigma_m}, \frac{s}{\lambda_m\sigma_m}\right)$$

$$-m\beta(m-1)(\lambda_m\sigma_m)$$

$$* h_1\left(\frac{L_1}{\lambda_m\sigma_m}, \frac{L_2}{\lambda_m\sigma_m}, \frac{\beta_m}{\lambda_m\sigma_m}, \frac{s}{\lambda_m\sigma_m}\right)$$

$$+m(\lambda_m\sigma_m)^2 h_2\left(\frac{L_1}{\lambda_m\sigma_m}, \frac{L_2}{\lambda_m\sigma_m}, \frac{\beta_m}{\lambda_m\sigma_m}, \frac{s}{\lambda_m\sigma_m}\right),$$

$$(4.4)$$

where C_0 is given by (4.2).

The MSD can now be obtained as the quotient of (4.4) and (4.3) so that

$$\text{MSD} = \sigma_m^2 - \lambda^2\sigma_a^2\frac{(m-1)}{2} + \beta^2\frac{(m-1)(2m-1)}{6}$$

$$-\beta(m-1)(\lambda_m\sigma_m)$$

$$* g_1\left(\frac{L_1}{\lambda_m\sigma_m}, \frac{L_2}{\lambda_m\sigma_m}, \frac{\beta_m}{\lambda_m\sigma_m}, \frac{s}{\lambda_m\sigma_m}\right)$$

$$+(\lambda_m\sigma_m)^2 g_2\left(\frac{L_1}{\lambda_m\sigma_m}, \frac{L_2}{\lambda_m\sigma_m}, \frac{\beta_m}{\lambda_m\sigma_m}, \frac{s}{\lambda_m\sigma_m}\right)$$

$$(4.5)$$

where $g_k() = h_k()/h_0()$ for $k = 1$ and 2.

In the *particular case* where $\beta = s = 0$ and $L_2 = -L_1$, the functions $h_0()$ and $g_2()$ in (4.3) and (4.5) coincide respectively with the functions $h()$ and $g()$ considered and characterized exactly by means of integral equations in Box and Luceño (1994) and for which several approximations exist. Neither the function $g_1()$ nor any particular instance of it appears to have been used previously, because $g_1()$ does not appear when $\beta = 0$ or $m = 1$.

A particularly interesting case occurs if $L_1 \geq L_2$. In this case, the integral in the right hand side of Equation (4.1) must be equal to 0 so that the solution of the integral equation satisfies $Y_k\{s/(\lambda_m\sigma_m)\} = C_k s^k$. Then (4.3) and (4.5) become AAI = m and

$$\text{MSD} = \sigma_m^2 - \lambda^2\sigma_a^2\frac{(m-1)}{2} + \beta^2\frac{(m-1)(2m-1)}{6}$$

$$-\beta(m-1)s + s^2 . \quad (4.6)$$

Finally, the MSD is often expressed in term of the percent increase in standard deviation (ISD) with respect to σ_a. This ISD is defined as ISD = $100 \times (\text{MSD}^{1/2}/\sigma_a - 1)$ and thus represents the percent inflation of the standard deviation compared with that of a repeated adjustment scheme for the process sampled at unit intervals.

4.2. NUMERICAL SOLUTIONS OF THE INTEGRAL EQUATIONS (2.5) AND (2.7)

Numerical solutions of (2.5) and (2.7) can be obtained by simply solving the integral equation (4.1) for k = 0, 1, and 2. Because (4.1) is a Fredholm integral equations of the second kind, its solution can be obtained using the routine `fred2` in Press, Teukolsky, Vetterling, and Flannery (1992, Chapter 18). This routine uses Gauss-Legendre quadrature so that the integral equation becomes a finite dimensional linear system of equations whose dimension should be chosen to obtain negligible discretization errors. The resulting linear system is then solved by *LU* decomposition. Subsequently, the routine `fredin` in Press, Teukolsky, Vetterling, and Flannery (1992, Chapter 18) can be used to perform Nystrom interpolation of the solution of (4.1) for different values of s.

The routines `fred2` and `fredin` use the routines `gauleg`, `lubksb`, and `ludcmp` to be found also in Press, Teukolsky, Vetterling, and Flannery (1992). Moreover, two functions `ak` and `g` should be provided to compute the kernel and the independent term of the integral equation. These functions together with a main program called `integral` have been written and included in Appendix B.

A detailed discussion of other methods used to obtain numerical solutions to integral equations is given by Baker (1977).

4.3. NUMERICAL MINIMIZATION OF THE OVERALL EXPECTED COST FUNCTION

The numerical minimization of the overall expected cost per unit interval, as given by (3.1), (4.3), and (4.5), with respect to L_1, L_2 and s can be performed for given values of λ, σ_a, β, m, σ_η, C_M, C_A, and C_T using the routines `brent`, `lnsrch`, and `dfpmin` in Press, Teukolsky, Vetterling, and Flannery (1992, Chapters 9 and 10). The overall expected cost can also be minimized with respect to the parameter m, which takes integer values. This may be done by minimizing the overall expected cost with respect to L_1, L_2, and s for several values of m, and choosing the value of m that provides the minimum cost.

The routine `dfpmin` uses a function called `func` and a routine called `dfunc` to compute the overall expected cost function to be minimized and the gradient of this cost. Moreover, `func` and `dfunc` use the routines `fred2`, `fredin`, `gauleg`, `lubksb`, and `ludcmp` and the functions `ak` and `g`, as in Subsection 4.2. The routine `brent` uses a function called `fs`. These functions `func` and `fs` and routine `dfunc` together with a main program called `costmin` have been written and included in Appendix C.

Two particularly interesting cases occur. First, if $C_A = 0$ and $\sigma_\eta = 0$, then there are no costs of adjustment and, consequently, the minimum cost is attained with a repeated adjustment scheme in which $L_1 = L_2$. Then the AAI equals m and the MSD is given by (4.6), so that the minimum cost is obtained by minimizing (4.6) with respect to the overcompensation s, which leads to $s = \beta(m-1)/2$. Second, if $C_M = 0$, then there is no cost of sampling and, consequently, the minimum cost is attained by sampling the process at every unit time interval, i.e., the optimum value of m is 1.

5. EXAMPLES OF THE USE OF THE COMPUTER CODES

Example 5.1. Box and Kramer (1992) give an example in which a quality characteristic subject to a disturbance has to be adjusted. After reconstruction, the disturbance was identified and estimated as an IMA (0, 1, 1) process with $\lambda = 0.3$ and $\sigma_a = 3.0$. The costs were assumed to be $C_M = \$200$, $C_A = \$600$, and $C_T = \$54$. There is neither adjustment error nor process drift, i.e., $\sigma_\eta = 0.0$ and $\beta = 0.0$.

Then the program `costmin`, given in Appendix C, yields the results in Table 1. In particular, one can see that the minimum cost is obtained using the sampling interval with $m = 10$ and action limits $L_2 = -L_1 = 2.89$. The optimal forecasted deviation from target after an adjustment is made is $s = 0.0$, because there is no process drift. The corresponding AAI and MSD are 28.25 and 16.51, respectively, which yields a percent increase in standard deviation ISD = 35.46% and a minimum overall expected cost per unit interval $C = \$140.32$.

Table 1. Minimum-cost schemes for Example 5.1 and several values of m. The overall minimum cost is attained for $m = 10$. The symmetric action limits $L_2 = -L_1$, average adjustment interval (AAI), mean squared deviation (MSD), percent increase in standard deviation (ISD) with respect to σ_a, and overall expected cost per unit interval, are shown in successive columns.

m	$L_2=-L_1$	AAI	MSD	ISD	COST
6	3.31	27.54	15.08	29.45	145.60
7	3.20	27.64	15.45	31.02	142.98
8	3.09	27.79	15.81	32.53	141.45
9	2.98	27.99	16.16	34.00	140.63
10	2.89	28.25	16.51	35.46	140.32
11	2.80	28.58	16.87	36.90	140.39
12	2.71	28.95	17.23	38.35	140.74
13	2.63	29.41	17.59	39.81	141.34
14	2.56	29.90	17.96	41.27	142.12

One can also see that values of m from 9 to 12 are close to the optimum in term of the resulting cost, and that the optimal action limit $L_2 = -L_1$ is a decreasing function of m, while the resulting AAI and MSD are increasing functions of m.

Example 5.2. Box (1991a, b) and Box and Luceño (1994) consider an example in which the thickness of a metallic film used in the manufacture of a computer chip has to be adjusted because standard SPC techniques had been only partially successful in stabilizing the process. The estimated values for $(\lambda, \sigma_a, \sigma_\eta, \beta)$ are (0.2, 11.0, 0.0, 0.0). However, the costs C_M, C_A, and C_T are not known.

Then the program `integral` in Appendix B yields the results in Table 2, which shows the AAI and ISD for 15 alternative schemes using all combinations of $L_2 = -L_1 = 0.0, 5.5$, and 11.0 with $m = 1, \ldots, 5$. This table is similar to Table 1 of Box and Luceño (1994). These authors explain how to use the results in the table to choose a control scheme. For example, one might set $L_2 = -L_1 = 5.5$ and $m = 2$ to get an AAI and ISD of 11.53 and 6.29%, respectively, or set $L_2 = -L_1 = 5.5$ and $m = 5$ with an AAI and ISD of 15.77 and 13.23%. The choice of scheme might depend, in this case, on "how capable" the controlled process was of providing product within specifications. Thus a moderate increase in the ISD might be tolerated, if the capability index were high, to attain major saving in sampling and/or adjustment costs (see Box and Luceño, 1994, Section 3).

Table 2. Some alternative schemes for the metallic thickness example of Section 5. The average adjustment interval (AAI) and percent increase in standard deviation (ISD) with respect to σ_a are shown for 15 alternative schemes using all combinations of the action limits $L_2 = -L_1 = 0.0, 5.5$, and 11.0 with the sampling interval $m = 1, \ldots, 5$.

	$L_2=-L_1$					
	0.0		5.5		11.0	
m	AAI	ISD	AAI	ISD	AAI	ISD
1	1.00	0.00	9.75	2.62	31.42	9.40
2	2.00	3.74	11.53	6.29	34.42	13.18
3	3.00	6.56	13.03	9.01	36.86	15.98
4	4.00	8.93	14.43	11.26	39.01	18.28
5	5.00	11.01	15.77	13.23	40.96	20.26
L_2/σ_a	0.0		0.5		1.0	

Example 5.3. Jensen and Vardeman (1993) consider an example in which measurements of the outer diameter of machined variator hubs were obtained for 150 consecutive hubs. The estimated values for (λ, σ_a, σ_η, β) are (0.558454, 5.65848, 0.8, 0.83), where we note that λ and σ_a are related to the standard deviations of the measurement errors σ_ε and the random shocks of the process σ_v, used in Jensen and Vardeman (1993), through the equations

$$\sigma_a^2 = \frac{1}{2}\left\{\left(\sigma_v^2 + 2\sigma_\varepsilon^2\right) + \sqrt{\left(\sigma_v^2 + 2\sigma_\varepsilon^2\right)^2 - 4\sigma_\varepsilon^4}\right\},$$

$$\lambda = 1 - \sigma_\varepsilon^2 / \sigma_a^2,$$

where the estimated values (3.76, 3.16) for (σ_ε, σ_v) have been introduced to obtain the estimates for λ and σ_a. The costs are assumed to be $C_M = C_A = 0.0$ and $C_T = 1.0$.

Then the program `costmin` in Appendix C yields the minimum-cost scheme, which is attained for the sampling interval with $m = 1$, action limits $L_1 = -0.802$ and $L_2 = 0.795$ and forecasted deviation from target after adjustment (overcompensation) $s = -0.0039$. To compare these results to those published in Jensen and Vardeman (1993), one has to subtract the drift β from L_1 and L_2, so that $L_1 - \beta = -1.632$ and $L_2 - \beta = -0.035$. The corresponding AAI and MSD are 1.238 and 32.059, respectively, which yields a percent increase in standard deviation ISD = 0.063% and minimum overall expected cost per unit interval $C = 1.01741$.

It should be noted that, in this case, the cost function is rather flat around the minimum. For example, for $L_1 = -0.84$, $L_2 = 0.76$, and $s = 0.0$, the cost is $C = 1.01742$; and so is for $L_1 = -0.76$, $L_2 = 0.84$, and $s = 0.0$; and for $L_1 = -0.84$, $L_2 = 0.84$, and $s = 0.0$. Consequently, the well-known symmetric action limits and null overcompensation are almost as good as those producing the minimum-cost scheme. The difference between them is a 0.001% in terms of the resulting overall expected cost per unit interval. Moreover, for the repeated adjustment scheme in which $L_1 = L_2 = s = 0.0$, the cost is 1.01999, which is only a 0.25% greater than the minimum cost. It is important, however, to note that the forecasted disturbance $\hat{z}_{tm}$ must be updated using Equation (2.3) with the best available estimate of the drift β_m

Table 3. Minimum-cost schemes for the outer diameter example of Section 5 for several combinations the sampling (C_M) and adjustment (C_A) costs. The optimum sampling interval (m), action limits (L_1 and L_2), overcompensation (s); and the resulting average adjustment interval (AAI), mean squared deviation (MSD), percent increase in standard deviation (ISD) with respect to σ_a, and minimum overall expected cost per unit interval, are shown in successive columns.

C_M	C_A	m	L_1	L_2	s	AAI	MSD	ISD	COST
0	0	1	−0.802	0.795	−0.004	1.24	32.06	0.06	1.017
1	0	2	−0.154	0.981	0.415	2.21	38.86	10.17	1.723
0	1	1	−5.024	4.418	−0.494	4.21	36.62	6.95	1.386
1	1	3	−2.516	3.969	0.710	4.98	46.38	20.36	1.987

If the adjustment and sampling costs are not null, then the program `costmin` can also be used to find minimum-cost schemes. Thus Table 3 gives these optimal schemes for all combinations of C_M = 0.0, 1.0 and C_A = 0.0, 1.0, and for the same values of the remaining parameters.

One can see, for example, that a unit increment in the sampling cost C_M produces much greater increments in the resulting minimum cost than a unit increment in the adjustment cost C_A, even though the effect of the unit increment in C_M has been attenuated by increasing the optimal sampling interval to $m = 2$ when $C_M = 1.0$ and $C_A = 0.0$ and to $m = 3$ when $C_M = C_A = 1.0$. Hence the assumption that $C_M = 0.0$ should be submitted to a careful scrutiny based on available data.

6. CONCLUSIONS

The problem, appearing in process adjustment using feedback, of determining the optimal sampling interval, action limits, and amount of adjustment, has been considered in this article. It is shown that two integral equations characterize exactly the AAI and MSD for a class of disturbance models containing the IMA model of Equation (2.2) for the process with deterministic drift β sampled at intervals m units apart. A computer program, called `integral`, has been written that can be used without the direct assignment of meaningful values to the costs of taking an observation, of making an adjustment, and of being off target. A second computer program, called `costmin`, has also been written to minimize the overall expected cost per unit interval when these costs are known.

The numerical method presented in the article generalizes the results in previously published papers by considering the possibility of deterministic process drift and random adjustment errors, together with the possibility of non-null sampling cost and augmented sampling intervals. Three examples of the use of the computer codes have been given, which demonstrate the usefulness of the proposed procedure.

ACKNOWLEDGMENTS

This research was partially supported by the Spanish DGICYT under Grant PB92-0502. Alberto Luceño also acknowledges the support of the National Science Foundation under Grant DMI-9414765. This article has been previously accepted for publication in the Journal of Statistical Computation and Simulation, published by Gordon and Breach Science Publishers, S. A. (see Luceño, González, and Puig-Pey, 1996).

APPENDIX A: DERIVATION OF THE INDEPENDENT TERM IN (2.7)

Using the decomposition $\varepsilon_{(t-1)m+k} = \varepsilon_{tm} - (\varepsilon_{tm} - \varepsilon_{(t-1)m+k})$, the independent term in (2.6) can be written as

$$\sum_{k=1}^{m} E\left\{\varepsilon^2_{(t-1)m+k}\right\} = m\,E\left(\varepsilon^2_{tm}\right)$$

$$+\sum_{k=1}^{m} E\left[\left\{\varepsilon_{tm} - \varepsilon_{(t-1)m+k}\right\}^2\right]$$

$$-2\sum_{k=1}^{m} E\left[\varepsilon_{tm}\left\{\varepsilon_{tm} - \varepsilon_{(t-1)m+k}\right\}\right]. \qquad \text{(A.1)}$$

Now one has $\varepsilon_{tm} = \hat{\varepsilon}_{tm} + u_{tm} = s + u$ using the notation $s = \hat{\varepsilon}_{tm}$ and $u = u_{tm}$. In addition, taking into account that the last opportunity for adjustment before tm is at $(t-1)m$, so that

$$\varepsilon_{(t-1)m+k} = z_{(t-1)m+k} + gX_{(t-1)m}$$

for $k = 1, \ldots, m$, and considering Equation (2.1), one has

$$\varepsilon_{tm} - \varepsilon_{(t-1)m+k} = z_{tm} - z_{(t-1)m+k}$$

$$= a_{tm} + \lambda \sum_{j=k+1}^{m-1} a_{(t-1)m+j} - \theta a_{(t-1)m+k} + \beta(m-k)$$

for $k = 1, \ldots, m-1$. (Clearly $\varepsilon_{tm} - \varepsilon_{(t-1)m+k} = 0$ for $k = m$.)

Hence

$$E\left(\varepsilon^2_{tm}\right) = E\left\{\left(\hat{\varepsilon}_{tm} + u_{tm}\right)^2\right\} = s^2 + \sigma^2_m, \qquad \text{(A.2)}$$

$$\sum_{k=1}^{m} E\left[\left\{\varepsilon_{tm} - \varepsilon_{(t-1)m+k}\right\}^2\right]$$

$$= \sum_{k=1}^{m-1} E\left[\left\{\varepsilon_{tm} - \varepsilon_{(t-1)m+k}\right\}^2\right]$$

$$= \sum_{k=1}^{m-1} \left[\sigma^2_a\left\{1 + \lambda^2(m-1-k) + \theta^2\right\} + \beta^2(m-k)^2\right]$$

$$= (m-1)\begin{bmatrix} \sigma^2_a\left\{1 + \lambda^2(m-2)/2 + \theta^2\right\} \\ +\beta^2\, m(2m-1)/6 \end{bmatrix} \qquad \text{(A.3)}$$

Finally, using Equations (2.1) and (2.2), one obtains

$$\varepsilon_{tm} = \hat{\varepsilon}_{tm} + u_{tm} = s + a_{tm} + \lambda \sum_{j=1}^{m-1} a_{(t-1)m+j}$$

$$-\theta a_{(t-1)m} + \theta_m u_{(t-1)m}$$

so that

$$\sum_{k=1}^{m} E\left[\varepsilon_{tm}\left\{\varepsilon_{tm} - \varepsilon_{(t-1)m+k}\right\}\right]$$

$$= \sum_{k=1}^{m-1} E\left[\varepsilon_{tm}\left\{\varepsilon_{tm} - \varepsilon_{(t-1)m+k}\right\}\right]$$

$$= \sum_{k=1}^{m-1} \left[\sigma_a^2\left\{1+\lambda^2(m-1-k)-\lambda\theta\right\}+\beta(m-k)s\right]$$

$$= (m-1)\left[\sigma_a^2\left\{1+\lambda^2(m-2)/2-\lambda\theta\right\}+\beta ms/2\right]. \tag{A.4}$$

Therefore, introducing (A.2), (A.3), and (A.4) in (A.1), one has

$$\sum_{k=1}^{m} E\left\{\varepsilon^2_{(t-1)m+k}\right\} = ms^2 - m\beta(m-1)s$$

$$+m\left\{\sigma_m^2 - \lambda^2\sigma_a^2\frac{(m-1)}{2} + \beta^2\frac{(m-1)(2m-1)}{6}\right\},$$

from which (2.7) follows immediately considering (2.6).

APPENDIX B: A COMPUTER CODE TO SOLVE THE INTEGRAL EQUATIONS (2.5) AND (2.7)

The program prompts the user to specify the name of the file containing the data or to write 0 to introduce the data interactively. Then it prompts the user to specify the name of the file to which the output will be directed. Subsequently, the following parameters should be provided: the number of points for the Gaussian quadrature (values between 40 and 60 are usually adequate); lower and upper values and increment for the lower action limit L_1, for the upper action limit L_2, for m, and for the overcompensation s; β, λ, σ_a; and the costs C_M, C_A, C_T, and σ_η. Finally, an option is provided to consider only symmetric action limits, i.e., action limits such that $L_2 = -L_1$.

The program code, which is written in FORTRAN, follows. It should be noted that the line

```
call gauleg (a,b,t,w,n)
```

has been moved from the routine `fred2` to the main program `integral` to avoid unnecessary calls to the routine `gauleg`. Hence the users should delete this line from their routine `fred2`.

```
      PROGRAM integral
      COMMON K0, BL
      INTEGER K0, M, NP, MX(3), MMIN
      PARAMETER (NPMAX=200)
      REAL t(NPMAX), w(NPMAX), h0(NPMAX), h1(NPMAX),
     & h2(NPMAX), L1X(3), L2X(3), sx(3), L1MIN, L2MIN,
     & smin, COSTMIN, BL, L1, L2, MBET, MLAMBD, MSIG,
     & BETA, LAMBDA, SIGMA, s, sdls, h0s, g1, g2, AAI,
     & MSD, ISD, CM, CA, CT, ADJSIG, COST
      CHARACTER*32 FIN, FOUT
      CHARACTER*2 SYMM
      LOGICAL NOSYMM
      EXTERNAL ak, g
```

```
C
C  K0 is k.  BL is beta/(lambdam * sigmam)
C  NP is the number of points for the Gaussian quadrature
C  MBET is betam. MLAMBD is lambdam. MSIG is sigmam
C  ADJSIG is the standard deviation of the adjustment
C         errors
C  AAI is the average adjustment interval
C  MSD is the mean squared deviation
C  ISD is the percent increase in standard deviation
C
C        Input data
C
      PRINT *,' How are you going to enter the data?'
      PRINT *,' on the screen: give 0'
      PRINT *,' on a file:     give file name'
      READ (*, '(A)') FIN
      PRINT *
      PRINT *,' Specify the name of the file to print',
     &         ' the results'
      READ (*, '(A)') FOUT
      OPEN (5, FILE = FOUT)
      PRINT *
C
      IF (FIN .EQ. '0' .OR. FIN .EQ. 'O' .OR.
     &    FIN .EQ. 'o') THEN
        PRINT *,' Specify the number of points for',
     &           ' the Gaussian quadrature'
        READ *, NP
        PRINT *,' Specify lower & upper points and',
     &  ' increment for the lower action limit:  L1'
        READ *, L1X(1), L1X(2), L1X(3)
        PRINT *,' Specify lower & upper points and',
     &  ' increment for the upper action limit:  L2'
        READ *, L2X(1), L2X(2), L2X(3)
        PRINT *,' Specify lower & upper points and',
     &  ' increment for m'
        READ *, MX(1), MX(2), MX(3)
        PRINT *,' Specify lower & upper points and',
     &  ' increment for the overcompensation s'
        READ *, sx(1), sx(2), sx(3)
        PRINT *,' Specify beta, lambda, sigma'
        READ *, BETA, LAMBDA, SIGMA
        PRINT *,' Specify costs:  CM, CA, CT, ADJSIG'
        READ *, CM, CA, CT, ADJSIG
      ELSE
        OPEN (1, FILE = FIN)
        READ (1,*) NP
        READ (1,*) L1X(1), L1X(2), L1X(3)
        READ (1,*) L2X(1), L2X(2), L2X(3)
        READ (1,*) MX(1), MX(2), MX(3)
        READ (1,*) sx(1), sx(2), sx(3)
        READ (1,*) BETA, LAMBDA, SIGMA
        READ (1,*) CM, CA, CT, ADJSIG
        CLOSE (1)
```

```
      ENDIF
      PRINT *,' Do you want symmetric action limits?'
      PRINT *,' no:  give no'
      PRINT *,' yes: give other thing'
      READ (*, '(A)') SYMM
      NOSYMM = .false.
      IF (SYMM .EQ. 'no' .OR. SYMM .EQ. 'NO' .OR.
     &    SYMM .EQ. 'nO' .OR. SYMM .EQ. 'No')
     &    NOSYMM = .true.
      COSTMIN = 1.E20
      M = MX(1)
10    L2 = L2X(1)
C
C  Computation of sigmam, lambdam, betam, a, b, and BL
C
      MSIG = 0.5 * SIGMA * (SQRT (FLOAT (M)) * LAMBDA +
     & SQRT (LAMBDA * LAMBDA * M + 4. * (1. - LAMBDA)))
      MLAMBD = (LAMBDA * SIGMA / MSIG) * SQRT (FLOAT (M))
      MBET = BETA * M
20    IF (NOSYMM) THEN
        L1 = L1X(1)
      ELSE
        L1 = - L2
      ENDIF
30    a = L1 / (MLAMBD * MSIG)
      b = L2 / (MLAMBD * MSIG)
      BL = MBET / (MLAMBD * MSIG)
C
C        Print data
C
      WRITE (*,*) 'L1 = ', L1,' L2 = ', L2,' m = ', M
      WRITE (5,*)
      WRITE (5,*)' Number of points for',
     & ' the Gaussian quadrature: ', NP
      WRITE (5,*)' Action limits:  L1 = ', L1,
     & ',  L2 = ', L2
      WRITE (5,*)' beta = ',BETA,',  lambda = ', LAMBDA,
     & ',  sigma = ',SIGMA,',  m = ', M
      WRITE (5,*)' betam = ',MBET,
     & ',  lambdam = ', MLAMBD, ',  sigmam = ',MSIG
      WRITE (5,*)' costs:  CM = ', CM, ',  CA = ', CA,
     & ',  CT = ', CT, ',  adjustment sigma =', ADJSIG
      WRITE (5,*)' standardized:  L1 = ', a,
     & ',  L2 = ', b, ',  beta = ', BL
      IF (L1 .LT. L2) THEN
C
C        Computation of Gaussian points and weights
C
        call gauleg(a,b,t,w,NP)
C
C        Computation of functions h0, h1, and h2.
C
        K0 = 0
        call fred2(NP,a,b,t,h0,w,g,ak)
```

```
          K0 = 1
          call fred2(NP,a,b,t,h1,w,g,ak)
          K0 = 2
          call fred2(NP,a,b,t,h2,w,g,ak)
C
C          Computation of the AAI, MSD, and COST.
C
        ENDIF
        s = sx(1)
40      IF (L1 .GE. L2) THEN
          AAI = FLOAT(M)
          MSD = MSIG * MSIG
     &        - (LAMBDA * SIGMA)**2 * (M - 1.) / 2.
     &        + BETA * BETA * (M - 1.) * (2. * M - 1.) / 6.
          MSD = MSD - BETA * (M - 1.) * s + s * s
          h0s = 1.
          g1 = s / (MLAMBD * MSIG)
          g2 = g1**2
        ELSE
          K0 = 0
          sdls = s / (MLAMBD * MSIG)
          h0s = fredin(sdls,NP,a,b,t,h0,w,g,ak)
          K0 = 1
          g1 = fredin(sdls,NP,a,b,t,h1,w,g,ak) / h0s
          K0 = 2
          g2 = fredin(sdls,NP,a,b,t,h2,w,g,ak) / h0s
          AAI = h0s * M
          MSD = MSIG * MSIG
     &        - (LAMBDA * SIGMA)**2 * (M - 1.) / 2.
     &        + BETA * BETA * (M - 1.) * (2. * M - 1.) / 6.
          MSD = MSD - BETA * (M - 1.) * MLAMBD * MSIG * g1
     &        + (MLAMBD * MSIG)**2 * g2
        ENDIF
        ISD = (SQRT (MSD) / SIGMA - 1.) * 100.
        COST = (CA + CT * (ADJSIG / SIGMA)**2) / AAI
     &       + CM / M + CT * MSD / SIGMA**2
C
C          print results
C
        WRITE (5,*)' s = ', s, ' h0 = ', h0s,
     &             ' g1 = ', g1,' g2 = ', g2
        WRITE (5,*)'       AAI = ', AAI, ' MSD = ', MSD,
     &             ' ISD = ', ISD, ' COST = ', COST
        IF (COST .LT. COSTMIN) THEN
          L1MIN = L1
          L2MIN = L2
          MMIN = M
          smin = s
          COSTMIN = COST
        ENDIF
        s = s + sx(3)
        IF (s .LE. sx(2)) GOTO 40
        IF (NOSYMM) THEN
          L1 = L1 + L1X(3)
```

```
        IF (L1 .LE. L1X(2)) GOTO 30
      ENDIF
      L2 = L2 + L2X(3)
      IF (L2 .LE. L2X(2)) GOTO 20
      M = M + MX(3)
      IF (M .LE. MX(2)) GOTO 10
      WRITE (5,*)
      WRITE (5,*)' the minimum cost found has been = ',
     & COSTMIN,' for:'
      WRITE (5,*)' m = ', MMIN, ' L1 = ', L1MIN,
     & ' L2 = ', L2MIN
      WRITE (*,*) ' the end'
      END

      FUNCTION g(t)
      COMMON K0, BL
      INTEGER K0
      REAL BL, g, t
      IF (K0 .EQ. 0) THEN
        g = 1.0
      ELSE IF (K0 .EQ. 1) THEN
        g = t
      ELSE IF (K0 .EQ. 2) THEN
        g = t * t
      ELSE
        STOP 'ERROR EN FUNCTION g(t)'
      ENDIF
      RETURN
      END

      FUNCTION ak(t, z)
      COMMON K0, BL
      INTEGER K0
      REAL BL, ak, t, z
      ak = EXP (-0.5 * (z - t - BL)**2) / 2.506628274631
      RETURN
      END
```

APPENDIX C: A COMPUTER CODE TO MINIMIZE THE OVERALL EXPECTED COST FUNCTION PER UNIT TIME INTERVAL

The program prompts the user to specify the name of the file containing the data or to write 0 to introduce the data interactively. Then it prompts the user to specify the name of the file to which the output will be directed and gives an option to track all the function evaluations. Subsequently, the following parameters should be provided: the number of points for the Gaussian quadrature (values around 40 are usually adequate); starting values for the action limits L_1 and L_2; β, λ, σ_a; the costs C_M, C_A, C_T, and σ_η; and, when $C_M > 0$, lower and upper values and increment for m. Finally, an option is given to modify any one of the recommended values for a convergence condition on the gradient, an increment used to compute the gradient numerically, and a factor used to scale the overall expected cost function internally.

It should be noted that the minimization routine `dfpmin` can fail if the variables are not conveniently scaled (see Press, Teukolsky, Vetterling, and Flannery, 1992, pp. 381 and 420). This has been carefully considered when writing the code so that convergence failure is very unlikely. Two advises should be given though. First, modify the input parameters L_1 and L_2 and/or the parameter to scale the function if a convergence failure happens.

Second, the variable `disc` in `lnsrch` may sporadically take a negative value, which produces an error. This problem can be easily avoided by replacing the lines

```
disc=b*b-3.*a*slope
tmplam=(-b+sqrt(disc))/(3.*a)
```

with

```
disc=b*b-3.*a*slope
if (disc .LT. 0.) then
  tmplam=-slope/(2.*(f-fold-slope))
else
  tmplam=(-b+sqrt(disc))/(3.*a)
endif
```

near the end of `lnsrch` (see Press, Teukolsky, Vetterling, and Flannery, 1992, Chapter 9). The program code, which is written in FORTRAN, follows.

```
      PROGRAM costmin
      COMMON K0, BL
      COMMON /MIN1/ NP, MSIG, BETA, LAMBDA, SIGMA, M, AAI
     & , MSD, CM, CA, CT, ADJSIG, hincr, fval, MLS, SC,
     & SYMM, s
      COMMON /FILE1/ TRACK
      LOGICAL TRACK, SYMM
      INTEGER K0, NP, M, iter, MX(3)
      REAL BL, MSIG, BETA, LAMBDA, SIGMA, AAI, MSD, CM,
     &CA, CT, ADJSIG, hincr, fval, MLS, SC, s, L1, L2,
     & ISD, gtol, gt, fret, par(2), MCOST, COST, MBET,
     & MLAMBD, scale, c1, c2, c3, c4, hess11, hess22
      CHARACTER*32 FIN, FOUT, FTRACK
      EXTERNAL dfunc, func
C
C MLS is lambdam * sigmam
C K0 is k.  BL is betam/(lambdam * sigmam)
C NP is the number of points for the Gaussian quadrature
C BET is betam.  MLAMBD is lambdam. MSIG is sigmam
C ADJSIG is the standard deviation of the adjustment
C        errors
C hincr is the increment of L1, L2, and s to compute the
C       gradient
C gtol is a convergence tolerance on zeroing the gradient
C SC is a factor to scale the cost function
C fval is SC times the current value of the cost function
C fret is the resulting minimum cost
C MCOST is the minimum cost that would result if L1=L2
C AAI is the average adjustment interval
C MSD is the mean squared deviation
C ISD is the percent increase in standard deviation
C
C        Input data
C
      PRINT *,' How are you going to enter the data?'
      PRINT *,' on the screen: give 0'
      PRINT *,' on a file:     give file name'
      READ (*, '(A)') FIN
      PRINT *
      PRINT *,' Specify the name of the file to print',
     &        ' the results'
```

```
 READ (*, '(A)') FOUT
 OPEN (5, FILE = FOUT)
 PRINT *
 PRINT *,' Would you like to keep track of all'
&        ,' function evaluations?'
 PRINT *,' no:   give 0'
 PRINT *,' yes:  give file name'
 READ (*, '(A)') FTRACK
 PRINT *
 TRACK = .false.
 IF (FTRACK .NE. '0' .AND. FTRACK .NE. 'O'
&                    .AND. FTRACK .NE. 'o') THEN
   OPEN (4, FILE = FTRACK)
   TRACK = .true.
 ENDIF
 IF (FIN .EQ. '0' .OR. FIN .EQ. 'O' .OR.
&     FIN .EQ. 'o') THEN
   PRINT *,' Specify the number of points for',
&          ' the Gaussian quadrature'
   READ *, NP
   PRINT *,' Specify the action limits:  L1, L2'
   READ *, L1, L2
   PRINT *,' Specify beta, lambda, sigma'
   READ *, BETA, LAMBDA, SIGMA
   PRINT *,' Specify the costs:  CM, CA, CT, ADJSIG'
   READ *, CM, CA, CT, ADJSIG
   IF (CM .GT. 0.) THEN
     PRINT *,' Specify lower & upper values and',
&            ' increment for m'
     READ *, MX(1), MX(2), MX(3)
     MX(1) = MAX (1, MX(1))
     MX(3) = MAX (1, MX(3))
     IF (MX(2) .LE. MX(1)) MX(2) = MX(1)
     M = MX(1)
   ELSE
     M = 1
   ENDIF
 ELSE
   OPEN (1,FILE = FIN)
   READ (1,*) NP
   READ (1,*) L1, L2
   READ (1,*) BETA, LAMBDA, SIGMA
   READ (1,*) CM, CA, CT, ADJSIG
   IF (CM .GT. 0.) THEN
     READ (1,*) MX(1), MX(2), MX(3)
     MX(1) = MAX (1, MX(1))
     MX(3) = MAX (1, MX(3))
     IF (MX(2) .LE. MX(1)) MX(2) = MX(1)
     M = MX(1)
   ELSE
     M = 1
   ENDIF
   CLOSE (1)
 ENDIF
```

```
      SYMM = .false.
      IF (BETA .EQ. 0.) THEN
        SYMM = .true.
        L2 = ABS(L2)
        L1 = -L2
      ENDIF
      IF (ABS(L2) .GT. 2*SIGMA) L2 = SIGMA
      IF (ABS(L1) .GT. 2*SIGMA) L1 = -SIGMA
C  Suggested values for gtol, hincr, and scale
      PRINT *,' Suggested values:'
      PRINT *,' gtol = 4.e-5, hincr = 1.e-3, scale = 1.'
      PRINT *,' Would you like to change any of these',
     &          ' values?'
      PRINT *,' no:   give 0., 0., 0.'
      PRINT *,' yes:  give 0.''s and/or new values'
      READ *, gtol, hincr, scale
      IF (gtol .LE. 0.) gtol = 4.e-5
      IF (hincr .LE. 0.) hincr = 1.e-3
      IF (scale .LE. 0.) scale = 1.
      PRINT *
      PRINT *,' Working'
C
C        Print data
C
      WRITE (5,*)' convergence tolerance on zeroing',
     &           ' the gradient = ',gtol
      WRITE (5,*)' increment of L1, L2, and s to',
     &           ' compute the gradient = ',hincr
      WRITE (5,*)' Number of points for the Gaussian',
     &           ' quadrature: ', NP
      WRITE (5,*)' Initial action limits:  L1 = ', L1,
     &           ',  L2 = ', L2
      WRITE (5,*)' beta = ',BETA,',  lambda = ', LAMBDA,
     &           ',  sigma = ',SIGMA
      WRITE (5,*)' costs:  CM = ', CM,',  CA = ', CA,
     &',  CT = ', CT, ',  adjustment sigma =', ADJSIG
      WRITE (5,*)
      WRITE (5,*)'             *******************',
     &           '*******************'
      WRITE (5,*)
C
C  Main loop. Computation of sigmam, lambdam, and betam
C
10    MSIG = 0.5 * SIGMA * (SQRT (FLOAT (M)) * LAMBDA +
     & SQRT (LAMBDA * LAMBDA * M + 4. * (1. - LAMBDA)))
      MLAMBD = (LAMBDA * SIGMA / MSIG) * SQRT (FLOAT (M))
      MBET = BETA * M
      MLS = MLAMBD * MSIG
      BL = MBET / MLS
C Newton-Raphson steps increase if SC increases
      SC = 8. * scale / (CT * SQRT(FLOAT(M)))
      gt = gtol * MAX(CA / 4., CT) * SC
C
C Minimization of the overall cost function
```

```
C {par(1) and par(2)} represent {L1 and L2}
C over the product of lambdam by sigmam
C
      IF ((CA .EQ. 0. .AND. ADJSIG .EQ. 0.) .OR.
     &    (ABS(L1*L2) .LT. 0.01*SIGMA)) THEN
        par(1) = 0.
        par(2) = 0.
      ELSE
        par(1) = L1 / MLS
        par(2) = L2 / MLS
      ENDIF
      n = 2
      IF (TRACK) THEN
        WRITE (*,*)
        WRITE (*,*)' going to dfpmin ****** m = ',
     &             M,' ******'
        WRITE (4,*)
        WRITE (4,*)' going to dfpmin ****** m = ',
     &             M,' ******'
      ENDIF
      CALL dfpmin(par,n,gt,iter,fret,func,dfunc)
C
C        Print data and results
C
      L1 = par(1) * MLS
      L2 = par(2) * MLS
C reevaluate variables included in COMMON
      COST = func(par) / SC
      WRITE (5,*)' m = ', M,' betam = ', MBET,
     & ',  lambdam = ', MLAMBD, ',  sigmam = ', MSIG
      ISD = MAX ((SQRT (MSD) / SIGMA - 1.) * 100., 0.)
      IF (L1 .GT. L2) THEN
        WRITE (5,*) ' L1 ≥ L2 at the optimum;'
        L1 = 0.
        L2 = 0.
      ENDIF
      WRITE (5,*)' optimal forecast after adjustment',
     &           ' and action limits:'
      WRITE (5,*)' s = ',s,',  L1 = ',L1,',  L2 = ',L2
      WRITE (5,*)' average adjustment interval,',
     &           ' mean squared deviation:'
      WRITE (5,*)' AAI = ', AAI,',  MSD = ', MSD
      WRITE (5,*)' percent increase standard',
     &           ' deviation, minimum cost:'
      WRITE (5,*)' ISD = ', ISD,',  mincost = ', COST
      WRITE (5,*)' number of iterations = ', iter
      IF (TRACK) THEN
        IF (.NOT.SYMM) THEN
          par(1) = par(1) + 0.1
          c1 = func(par)
          par(1) = par(1) - 0.2
          c2 = func(par)
          par(1) = par(1) + 0.1
        ENDIF
```

```
        par(2) = par(2) + 0.1
        c3 = func(par)
        par(2) = par(2) - 0.2
        c4 = func(par)
        IF (.NOT.SYMM)
     &  hess11 = (c1 + c2 - 2 * fret) / 0.01
        hess22 = (c3 + c4 - 2 * fret) / 0.01
        IF (SYMM) hess11 = hess22
C hessian should be kept around 1. by using scale
        WRITE (*,*)' hessian = ', hess11,'  ', hess22,
     &' SC = ', SC, ' scaled cost = ', fret
        WRITE (4,*)' hessian = ', hess11,'  ', hess22,
     &' SC = ', SC, ' scaled cost = ', fret
      ENDIF
      par(1) = 0.
      par(2) = 0.
      MCOST = func(par) / SC
      IF (MCOST .LE. COST * 1.1) THEN
        ISD = MAX ((SQRT (MSD) / SIGMA - 1.) * 100., 0.)
        s = BETA * (M - 1.) / 2.
        WRITE (5,*)
        WRITE (5,*)' the no-deadband policy is ',
     &             'optimal or close to optimal:'
        WRITE (5,*)' no-deadband optimal forecast after',
     &             ' adjustment and action limits:'
        WRITE (5,*)' s = ',s, ',  L1 = 0.0,  L2 = 0.0'
        WRITE (5,*)' average adjustment interval,',
     &             ' mean squared deviation:'
        WRITE (5,*)' AAI = ', AAI,',  MSD = ', MSD
        WRITE (5,*)' percent increase standard',
     &             ' deviation and cost:'
        WRITE (5,*)' ISD = ', ISD,',  COST = ', MCOST
        WRITE (5,*)' this cost is a ',100. *
     &  (MCOST / COST - 1.), ' % greater than mincost'
        IF (MCOST / COST .LT. 1.001)THEN
          L1 = 0.
          L2 = 0.
        ENDIF
      ENDIF
      WRITE (5,*)
      WRITE (5,*)'              *******************',
     &           '*******************'
      WRITE (5,*)
      IF (CM .GT. 0.) THEN
        M = M + MX(3)
        IF (M .LE. MX(2)) GOTO 10
      ENDIF
      PRINT *, ' the end'
      END

      FUNCTION func(par)
      COMMON K0, BL
      COMMON /MIN1/ NP, MSIG, BETA, LAMBDA, SIGMA, M, AAI
     & , MSD, CM, CA, CT, ADJSIG, hincr, fval, MLS, SC,
```

```
     & SYMM, s
      COMMON /FILE1/ TRACK
      PARAMETER (NPMAX=200)
      COMMON /FU/ a1, a2, t(NPMAX), w(NPMAX), h0(NPMAX),
     & h1(NPMAX), h2(NPMAX)
      LOGICAL TRACK, SYMM
      INTEGER K0, NP, M
      REAL BL, MSIG, BETA, LAMBDA, SIGMA, AAI, MSD,
     & CM, CA, CT, ADJSIG, hincr, fval, MLS, SC, s,
     & L1, L2, as, a1, a2, h0s, g1, g2, func, COST,
     & ax, bx, cx, fa, fb, fc, delta, tol
      PARAMETER (NMAX=2)
      REAL par(NMAX)
      EXTERNAL ak, g, fred2, fs, brent
      IF (SYMM) par(1) = -par(2)
      L1 = par(1) * MLS
      L2 = par(2) * MLS
      IF (par(1) .GE. par(2)) THEN
        par(1) = 0.
        par(2) = 0.
        s = BETA * (M - 1.) / 2.
        AAI = FLOAT(M)
        MSD = MSIG * MSIG
     &       - (LAMBDA * SIGMA)**2 * (M - 1.) / 2.
     &       + BETA * BETA * (M - 1.) * (2. * M - 1.) / 6.
        MSD = MSD - BETA * (M - 1.) * s + s * s
        COST = (CA + CT * (ADJSIG / SIGMA)**2) / AAI
     &        + CM / M + CT * MSD / SIGMA**2
        IF (TRACK) THEN
         WRITE (*,*)' L1 greater than or equal to L2 ***'
         WRITE (4,*)' L1 greater than or equal to L2 ***'
        ENDIF
        GOTO 20
      ENDIF
      a1 = par(1)
      a2 = par(2)
C Computation of Gaussian points and weights
      call gauleg(a1,a2,t,w,NP)
C Computation of functions h0, h1, and h2
      K0 = 0
      call fred2(NP,a1,a2,t,h0,w,g,ak)
      K0 = 1
      IF (.NOT.SYMM .AND. M.GT.1)
     &call fred2(NP,a1,a2,t,h1,w,g,ak)
      K0 = 2
      call fred2(NP,a1,a2,t,h2,w,g,ak)
C Computation of optimal s
      IF (SYMM) THEN
        s = 0.
        COST = fs(s)
      ELSE
C Bracketing the optimum value of as = s / MLS
        bx = BETA * (M - 1.) / (2. * MLS)
        fb = fs(bx)
```

```
        delta = 1. / (LAMBDA * SQRT(FLOAT(M)))
10      ax = bx - delta
        fa = fs(ax)
        cx = bx + delta
        fc = fs(cx)
        IF (fb .GE. fa .OR. fb .GE. fc) THEN
          delta = 2. * delta
          IF (delta .GT. 1.e+8) STOP 'error in func'
          GOTO 10
        ENDIF
C Optimizing as = s / MLS
        tol = 5.e-4
        COST = brent(ax,bx,cx,fs,tol,as)
        s = as * MLS
      ENDIF
C Print intermediate results
20    func = COST * SC
      IF (TRACK) THEN
        WRITE (*,*)' s = ', s,' L1 = ', L1,' L2 = ', L2,
     &             ' COST = ', COST
        WRITE (4,*)' s = ', s,' L1 = ', L1,' L2 = ', L2,
     &             ' COST = ', COST
      ENDIF
      fval = func
      RETURN
      END

      FUNCTION fs(as)
      COMMON K0, BL
      COMMON /MIN1/ NP, MSIG, BETA, LAMBDA, SIGMA, M, AAI
     & , MSD, CM, CA, CT, ADJSIG, hincr, fval, MLS, SC,
     & SYMM, s
      PARAMETER (NPMAX=200)
      COMMON /FU/ a1, a2, t(NPMAX), w(NPMAX), h0(NPMAX),
     & h1(NPMAX), h2(NPMAX)
      COMMON /FILE1/ TRACK
      LOGICAL TRACK, SYMM
      INTEGER K0, NP, M
      REAL BL, MSIG, BETA, LAMBDA, SIGMA, AAI, MSD,
     & CM, CA, CT, ADJSIG, hincr, fval, MLS, SC, s,
     & as, a1, a2, h0s, g1, g2, COST
      EXTERNAL ak, g, fredin
C Computation of AAI, MSD, and COST
      K0 = 0
       h0s = fredin(as,NP,a1,a2,t,h0,w,g,ak)
      K0 = 1
       g1 = 0.
      IF (.NOT.SYMM .AND. M.GT.1)
     & g1 = fredin(as,NP,a1,a2,t,h1,w,g,ak) / h0s
      K0 = 2
       g2 = fredin(as,NP,a1,a2,t,h2,w,g,ak) / h0s
      AAI = h0s * M
      MSD = MSIG * MSIG
     &    - (LAMBDA * SIGMA)**2 * (M - 1.) / 2.
```

```
     &    + BETA * BETA * (M - 1.) * (2. * M - 1.) / 6.
      MSD = MSD - BETA * (M - 1.) * MLS * g1
     &    + MLS**2 * g2
      COST = (CA + CT * (ADJSIG / SIGMA)**2) / AAI
     &     + CM / M + CT * MSD / SIGMA**2
C Testing discretization errors
      IF (h0s .LT. .9999 .OR. g2 .LT. -1.e-4) THEN
        WRITE (*,*)' Unacceptable discretization error'
        WRITE (*,*)' h0 = ',h0s,' g1 = ',g1,' g2 = ',g2
        WRITE (*,*)' Please try new  L1 and L2,'
        WRITE (5,*)' Unacceptable discretization error'
        WRITE (5,*)' h0 = ',h0s,' g1 = ',g1,' g2 = ',g2
        WRITE (5,*)' Please try new  L1 and L2,'
        STOP
      ENDIF
      fs = COST
      RETURN
      END

      SUBROUTINE dfunc(ptt,grad)
      COMMON /MIN1/ NP, MSIG, BETA, LAMBDA, SIGMA, M, AAI
     & , MSD, CM, CA, CT, ADJSIG, hincr, fval, MLS, SC,
     & SYMM, s
      COMMON /FILE1/ TRACK
      LOGICAL TRACK, SYMM
      INTEGER NP, M, I, isign
      REAL MSIG, BETA, LAMBDA, SIGMA, AAI, MSD, CM, CA,
     &  CT, ADJSIG, hincr, fval, MLS, SC, s, a1, a2,
     & fval0, temp, hh, incr, absgrad
      PARAMETER (NMAX=2)
      REAL ptt(NMAX), grad(NMAX)
      EXTERNAL func, donothing
      incr = hincr
      IF (TRACK) THEN
        WRITE (*,*) ' computing the gradient'
        WRITE (4,*) ' computing the gradient'
      ENDIF
      a1 = ptt(1)
      a2 = ptt(2)
      fval0 = fval
10    absgrad = 0.
      IF (SYMM) THEN
        temp = ptt(2) + hincr / SQRT(2.)
        call donothing(temp)
        hh = temp - ptt(2)
        ptt(2) = ptt(2) + hh
        ptt(1) = -ptt(2)
        grad(2) = (func(ptt)-fval0) / hh
        grad(1) = -grad(2)
        absgrad = ABS (2. * grad(2))
        ptt(1) = -a2
        ptt(2) = a2
      ELSE
        isign = -1
```

```
        DO 20 I = 1,2
          ptt(1) = a1
          ptt(2) = a2
          temp = ptt(I) + hincr * isign
          call donothing(temp)
          hh = temp - ptt(I)
          ptt(I) = ptt(I) + hh
          grad(I) = (func(ptt)-fval0)/hh
          absgrad = absgrad + ABS (grad(I))
          isign = 1
20      CONTINUE
        ptt(1) = a1
        ptt(2) = a2
      ENDIF
C The gradient should not be exactly 0.
      IF (absgrad .LT. 1.e-20) THEN
        hincr = 10. * hincr
        GOTO 10
      ENDIF
      IF (TRACK) THEN
        WRITE (*,*) ' gradient computed  (line search)'
        WRITE (4,*) ' gradient computed  (line search)'
      ENDIF
      hincr = incr
      RETURN
      END

      SUBROUTINE donothing(temp)
      REAL temp
      RETURN
      END
```

REFERENCES

Adams, B.M. and Woodall, W.H. (1989), "An analysis of Taguchi's on-line process control procedure under a random walk model," *Technometrics*, **31**, 401-413.

Ansley, C.F. (1979), "An Algorithm for the Exact Likelihood of a Mixed Autoregressive-moving Average Process," *Biometrika*, **66**, 59-65.

Baker, C.T.H. (1977), *The Numerical Treatment of Integral Equations*. Clarendon Press: Oxford.

Box, G.E.P. (1991a), "Feedback Control by Manual Adjustment," *Quality Engineering*, **4**, 143-151.

______ (1991b), "Bounded Adjustment Charts," *Quality Engineering*, **4**, 333-340.

Box, G.E.P. and Jenkins, G.M. (1963), "Further Contributions to Adaptive Optimization and Control: Simultaneous Estimation of Dynamics: Non-Zero Costs," *Bulletin of the International Institute*, 34th Session, Ottawa.

______ (1976), *Time Series Analysis Forecasting and Control* (2nd ed.), San Francisco: Holden-Day.

Box, G.E.P., Jenkins, G.M., and Reinsel, G.C. (1994), *Time Series Analysis Forecasting and Control* (3rd ed.), Englewood Cliffs, NJ: Prentice Hall.

Box, G.E.P. and Kramer, T. (1992), "Statistical Process Monitoring and Feedback Adjustment. A Discussion," *Technometrics*, **34**, 251-285.

Box, G.E.P. and Luceño, A. (1994), "Selection of Sampling Interval and Action Limit for Discrete Feedback Adjustment," *Technometrics*, **36**, 369-378.

Box, G.E.P. and MacGregor, J.F. (1976). "Parameter Estimation with Closed-Loop Operating Data," *Technometrics*, **18**, 371-380.

Crowder, S.V. (1986), *Kalman Filtering and Statistical Process Control*, unpublished Ph. D. dissertation, Iowa State University, Dept. of Statistics.

Drezner, Z., and Wesolowsky, G.O. (1989), "Control Limits for a Drifting Process with Quadratic Loss," *International Journal of Production Research*, **27**, 13-20.

Jensen, K. L. and Vardeman, S. B. (1993), "Optimal Adjustment in the Presence of Deterministic Process Drift and Random Adjustment Error," *Technometrics*, **35**, 376-389.

Kramer, T. (1989), "Process Control from an Economic Point of View," *Ph.D. dissertation*, University of Wisconsin-Madison.

Ljung, G.M., and Box, G.E.P. (1979), "The Likelihood Function of Stationary Autoregressive-moving Average Models," *Biometrika*, **66**, 265–270.

Luceño, A. (1993a), "Performance of EWMA Versus Last Observation for Feedback Control," *Communications in Statistics, Theory and Methods*, **22**, 241-255.

______ (1993b), "A Fast Algorithm for the Repeated Evaluation of the Likelihood of a General Linear Process for Long Series," *Journal of the American Statistical Association*, **88**, 229–236.

______ (1994a), "Fast Optimization of the Exact Likelihood of AR and ARMA Processes. *Computational Statistics & Data Analysis,* **17**, 51-63.

______ (1994b), "A Fast Algorithm for the Exact Likelihood of Stationary and Partially Nonstationary Vector Autoregressive-moving Average Processes," *Biometrika*, **81**, 555-565.

Luceño, A., González, F.J., and Puig-Pey J. (1996). Computing Optimal Adjustment Schemes for the General Tool-wear Problem. *Journal of Statistical Computation and Simulation*, **54**, 87–113.

Mélard, G. (1984), "Algorithm AS 197. A Fast Algorithm for the Exact Likelihood of Autoregressive-moving Average Models," *Applied Statistics*, **33**, 104-114.

Muth, J.F. (1960), "Optimal Properties of Exponentially Weighted Forecasts of Time Series With Permanent and Transitory Components," *Journal of the American Statistical Association*, **55**, 299-306.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992), *Numerical Recipes*, Cambridge: Cambridge University Press.

Schneider, H., Tang, K., and O'Cinneide, C. (1990), "Optimal Control of a Production Process Subject to Random Deterioration," *Operation Research*, **38**, 1116-1122.

Shea, B.L. (1987), "Estimation of Multivariate Time Series," *Journal of Time Series Analysis*, **8**, 95-109.

______ (1989), "Algorithm AS 242. The Exact Likelihood of a Vector Autoregressive-moving Average Model," *Applied Statistics*, **38**, 161-204.

Srivastava, M.S. and Wu, Y. (1991), "A second order approximation to Taguchi et al's on-line control procedure," *Communications in Statistics*, A, **20**, 2149-2168.

Taguchi, G. (1981), *On-Line Quality Control During Production*. Japanese Standard Assoc.