Motorola Digital Signal Processors

Implementation of Adaptive Controllers on the Motorola DSP56000/ DSP56001

by
Pascal Renard, Ph.D.
Strategic Marketing – Geneva, Switzerland

Table of Contents

SECTION 1
Introduction

1.1 History of Adaptive Control 1-1
1.2 Theory of Adaptive Control 1-2

SECTION 2
Numerical Domain Representation

2.1 Parametric Models 2-1
2.2 Adaptive Control Techniques 2-6

SECTION 3
Adaptive Control and Adaptive Controllers

3.1 Adaptive Control Using Reference Models 3-1
3.1.1 Introduction 3-1
3.1.2 Closed-Loop System 3-2
3.1.3 Control Law 3-4
3.1.3.1 Known System Parameters 3-4
3.1.3.2 Unknown System Parameters 3-9
3.1.4 Determination of Controller Parameters 3-10
3.1.5 Comment 3-14

3.2 Generalized Predictive Control 3-15
3.2.1 Introduction 3-15
3.2.2 Closed-Loop System 3-18
3.2.3 Control Law 3-19
3.2.3.1 Definition of Parametric Model 3-19
3.2.3.2 Definition of System Output Prediction 3-19
3.2.3.3 Determination of Polynomials 3-21
3.2.3.4 Determination of Control Law 3-21
3.2.4 Comment 3-25

Table of Contents

SECTION 4
Implementation and Simulation of Adaptive Controllers Using Reference Models

4.1 Implementation 4-1
4.1.1 Simulation 4-2
4.2 Generalized Prediction Controllers 4-2
4.2.1 Implementation 4-2

SECTION 5
Conclusion

5.1 Advantages of Adaptive Control 5-1
5.2 Advantages of DSP56000/DSP56001 Architecture 5-3

APPENDIX
The Least Mean-Square Principle

A.1 Equation Formulation A-1
A.2 Estimation of Model Parameters A-3
A.3 The Least-Squares Estimator A-6
A.3.1 Is the LSE biased? A-6
A.3.2 How accurate is the LSE? A-7
A.3.3 Conclusion (on LSE accuracy and bias) A-7
A.4 Improving the “LSE” A-8
A.4.1 What is required to minimize LSE bias? A-8
A.4.2 What is required to minimize LSE variance? A-8
A.5 Conclusion A-9

INDEX Index-1
REFERENCES Reference-1

Illustrations

Figure 1-1	Basic principles of adaptive control	1-6
Figure 2-1	Parametric model description in terms of process input and output	2-2
Figure 2-2	Adaptive controller in closed loop	2-6
Figure 2-3	Standard input and disturbance signals	2-7
Figure 2-4	Dynamic response in open loop	2-8
Figure 2-5	Desired dynamic response in closed loop	2-9
Figure 3-1	Adaptive control using reference models in closed loop	3-3
Figure 3-2	Generalized predictive control using closed loop	3-18
Figure 4-1	Simulation results for adaptive controller using parallel-serial reference models	4-3
Figure 4-2	Generalized Predictive Controller	4-4
Figure 4-3	Adaptive Controller	4-15
Figure A-1	Parametric model description in terms of process input and output	A-1

SECTION 1

Introduction

"An adaptive control system measures a certain performance rating of the system (or plant) to be controlled."

This section shows how Motorola DSP56000/DSP56001 digital signal processors can be used to solve real-time digital control problems. After reviewing the relevant basic theory of adaptive control, we look at a number of implementations.

1.1 History of Adaptive Control

Computerized industrial process control has advanced by leaps and bounds over the last ten years in hardware and methods. The development of new microcontrollers and digital signal processors (DSPs) has given rise to important changes in regulation system design. The capabilities and low cost of the latest DSPs make them ideal for a wide range of regulation applications. Further, and despite the fact that analog regulators still enjoy wide popularity, DSPs offer higher performance than their analog predecessors.

Very few of the microcontroller-based digital regulators developed to date fully exploit the key advantages of microprocessor technology. Most designers seem content to emulate the behavior of traditional PID analog regulators. Sad though it may be, this is indeed an accurate reflection of industrial reality, in many cases.Unfortunately, conventional PID controllers, whether analog or digital, are only efficient

where the system to be controlled (the plant) — or rather the model of that system represented within the controller — is characterized by constant parameters applicable at all operating points. And yet, most complex industrial systems are characterized by parameters that vary with the system operating point [REN-88], thus failing to meet the basic assumption just stated. Two examples are heat exchangers (such as those used in the production of textile fibers) and internal-combustion engines. In such cases, a control signal generated by a conventional PID controller (i.e. one for which the parameters are computed once and for all on the basis of a constant-parameter system model) will inevitably give rise to progressively more degraded operation of the overall control loop as the errors between controller and actual process parameters increase. This can only be corrected by modifying the controller coefficients . . . Which brings us to adaptive control.

1.2 Theory of Adaptive Control

Together or separately, microcontrollers and DSPs enable us to design higher performance regulation systems using more sophisticated digital control algorithms, many of which have already been developed under and tested in industrial conditions [IRV-86]. Adaptive control represents an advanced level of controller design. It is recommended for systems operating in variable environments and/or featuring variable parameters.

Although adaptive control has only been around for a few years, it has already been successfully employed in a number of industrial applications. The basic principles were published by Kalman, in 1958, for the stochastic approach, and later by Whitaker for the deterministic approach. However, the technique was not viable for two reasons:

- The solutions proposed at the time were not very “robust”.
- The hardware (computers) required for implementation were either unavailable or far too expensive.

Currently, however, the technique is rapidly gaining new supporters. This is largely a result of recent work that has improved algorithm robustness [SAM-83 and IRV-83] and of the development of microcontrollers and/or DSPs which make it possible to support and implement the new algorithms.

Adaptive control is a set of techniques for the automatic, on-line, real-time adjustment of control-loop regulators designed to attain or maintain a given level of system performance where the controlled process parameters are unknown and/or time-varying. The use of microcontrollers and/or DSPs in control loops offers the following advantages:

- Wide range of alternative strategies for controller design and mathematical modelling,Freedom to use regulation algorithms that are more complex and offer higher performance than PID
- Technique is suitable for process control applications involving time delays

- Automatic estimation of process models for different operating points
- Automatic adjustment of controller parameters
- Constant control system performance in the presence of time-varying process characteristics
- Real-time diagnostic capability

Adaptive control is based entirely on the following hypothesis: the process to be controlled can be mathematically modelled and the structure of this model (delay and order) is known in advance. The determination of the structure of a parametric system model is thus a vital step before going on to design an adaptive control algorithm. The identification technique should be selected by a specialist in automatic control. The capabilities of the adaptive control algorithm depends, to a large extent, on the faithfulness with which the model represents the system and its behavior. The chief advantage, in practical terms, of adaptive control appears to be the capability to ensure quasi-optimal system performance in the presence of a model with time-varying parameters.

Once the model and its structure have been identified, the next step is to select a control strategy. This choice depends in part on the nature of the problem (regulation or tracking) and on the system characteristics (minimum phase or not). The number of options available depends on the extent of our advance knowledge of these characteristics. The aim is to select a strategy yielding a satisfactory control law in the case where the system model and

its environment are fully determined. The strategies most commonly encountered in adaptive control are:

- For minimum-phase systems (or for systems where the non-minimum phase is fully determined): "***minimum-variance control"*** o***r "control using reference models"***
- For non-minimum-phase systems: ***"pole-placement control"*** or ***"quadratic-criterion optimal control"***

The adaptive control algorithm is then designed in accordance with the structure of the system model and the selected control strategy. As a rule, the adaptive control algorithm can be seen as a combination of two algorithms. An identification algorithm uses measurements made on the system and generates information (a succession of estimates) for input to a control law computation algorithm. This second algorithm determines, at each instant, the adaptive controller parameters and the control to be applied to the system. This type of ***adaptive control*** is termed ***indirect***. However, the breakdown into two parts is not always apparent. For example, no control law computation algorithm is required at all if the parameters characterizing the adaptive controller are directly identified. This is known as ***direct adaptive control***.

We will look first at ***adaptive control*** based on a ***direct scheme using a reference model***. There are two main reasons for this choice: first, this type of control is relatively easy to implement; second, it has already found practical applications in industrial systems [LAN-84], [DAH-82].

A discussion follows on an adaptive control system based on an indirect scheme which, to date, judging from our bibliographic research, offers the best system response. This type of control was introduced by Clarke [CLA-84]. It produces optimal control over any system, with or without time delays and irrespective of whether the inverse is stable or unstable. This scheme is known as ***generalized predictive control***.

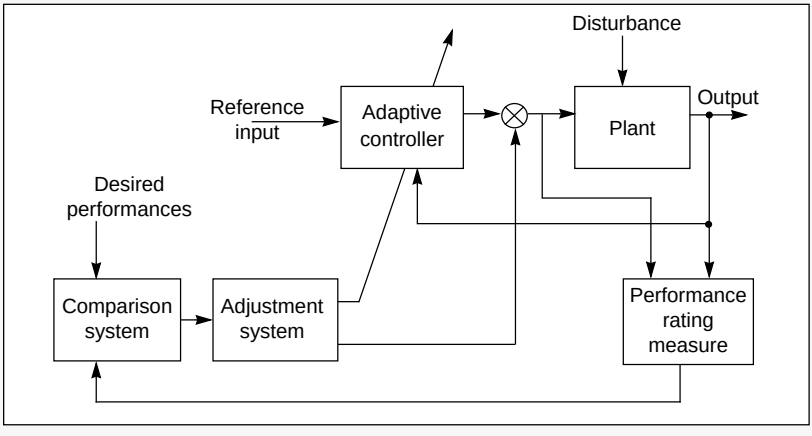

Figure 1-1 Basic principles of adaptive control

The performance rating of a system is measured and compared to the design goal. The adaptive system modifies the parameters of the adaptive controller in order to maintain the performance rating close to the desired value.

The basic principle underlying adaptive control systems is relatively simple (see Figure 1-1). An adaptive control system measures a certain performance rating of the system (or plant) to be

controlled. Starting with the difference between the desired and measured performance ratings, the adjustment system modifies the parameters of the adaptive controller (or regulator) and the control law in order to maintain the system performance rating close to the desired value(s).

Note that, in order to design and correctly adjust (or tune) a good controller, we must specify the desired performance of the regulation loop and determine the dynamic process model describing the relation between variations in control signals and output. This means we must determine the representation model which, in turn, means that we must establish the system's order and time delay.

The literature on adaptive control includes hundreds of papers on different approaches to the problem. As a result, engineers who are not specialists in adaptive control theory often find it very difficult to determine which approach they should use to solve a given problem. The aim of this application note is to introduce the reader to the two main principles of adaptive control identified to date and to guide the design engineer in the selection of control strategies applicable to a given situation.

The two principles selected for discussion were chosen on the basis of the goal of any design project, namely the determination of a real-time control law applicable to a given process. The total number of operations required to parameterize the control law is assumed to be one of the criteria most important to the design engineer. It is true that for high-speed industrial systems using microcontrollers — such as

automotive Anti-lock Braking Systems (ABS) and active suspensions, to name but two — the total number of operations assigned to the control algorithm cannot be very high. Given their internal structure (Von Neumann), conventional microcontrollers only have a limited real-time computation capability, thus directly limiting the complexity of the control algorithms.

The architecture of Motorola DSP56000/DSP56001 devices features a multi-bus processor that is highly parallel (extended Harvard architecture) and specially designed for real-time digital signal processing. In view of their computational power, these devices can be used to implement sophisticated control algorithms and thus to control high-speed industrial systems.

Apart from the fact that digital signal processing is now widely employed, the chief advantages of Motorola DSP56000/DSP56001 controllers can be summarized as follows:

- Lower system component costs because a single DSP56000/DSP56001 controller can replace not only the microcontroller but also the components required for 3-D lookup tables to digitally map model characteristics (as in the case of injection systems, active suspensions, etc.).
- Further savings can be achieved by using digital techniques (e.g. an observation model) to replace expensive sensors.
- The computations performed by DSP56000/DSP56001 devices are more accurate than

microcontroller interpolation between the inputs of a 3-D LUT (lookup table).

- A single DSP56000/DSP56001 controller can analyze and process several input parameters at a time.

Each Motorola DSP56000/DSP56001 device is both a high-speed microcontroller and a powerful digital signal processor. The DSP56001 program RAM can accommodate 512 words of 24 bits. The RAM can be loaded, following a clear, from a 2K x 8-bit EPROM or from a host processor. For mass-produced products, the DSP56000 offers a program ROM of 3.75K words of 24 bits which can be factory programmed for stand-alone applications.

Apart from the amount of memory space allocated to the program field, the DSP56000 and DSP56001 controllers are identical, with two separate memory spaces for data. A further feature is multiplication with accumulation of previous values, a capability much used by real-time control algorithms. A DSP56000/DSP56001 controller can multiply two 24-bit numbers, add the 48-bit result to the contents of the 56-bit accumulator, and simultaneously access the two data memory fields, all in a single instruction cycle.

For fast input/output, DSP56000/DSP56001 devices feature three peripheral devices in the same package, namely: a host processor interface (HI), a synchronous serial interface (SSI), and a serial communications interface (SCI). When the SCI is not required for communications, the baud rate generators can be used as timers. Depending on the way

in which the peripherals are configured, DSP56000/DSP56001 can offer up to 24 I/O lines. These features make DSP56000/DSP56001 controllers ideal for a wide range of real-time control applications where their processing power can be used to advantage. Applications include: disk drives, motor control, automotive active suspensions, active noise control, robotics, etc. ■

SECTION 2

Numerical Domain Representation

2.1 Parametric Models

For any continuous, mono- or multi-variable physical system, the search for a suitable parametric model — whether by empirical methods or on the basis of experimental data — leads to the use of linear differential equations to represent the process to be identified. These equations are of the form:

$$\frac{d^n Y(t)}{dt^n} + \alpha_1 \frac{d^{n-1} Y(t)}{dt^{n-1}} + \ldots + \alpha_n Y(t) = \beta_0 \frac{d^m U(t)}{dt^m} + \ldots + \beta_m U(t) \qquad \text{Eqn. 2-1}$$

In nature, no system is rigorously linear in the mathematical sense. However, most processes approach linear behavior over a limited operating range.

Contrary to non-parametric models (finite impulse response), parametric models depend on a specific structure. The parametric model characterizes the dynamic behavior of a physical system in terms of its transmittance or transfer function. This may be deduced using a z-transform. Applying such a transform to expression Eqn. 2-1, we obtain:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{z^{-d}(b_0 + \ldots + b_m z^{-m})}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} = \frac{z^{-d} B(z^{-1})}{A(z^{-1})} \qquad \text{Eqn. 2-2}$$

where:

- *(a_1,. . .,a_n) and (b_0,. . .,b_m) represent the parameters of the sampled model*
- *d represents the time delay (for $i \leq d$ then, $b_i = 0$)*
- *n determines the order of the model ($n \geq m$)*
- *U (z) is the model input*
- *Y (z) is the model output*

The most widely used parametric model is illustrated in Figure 2-1:

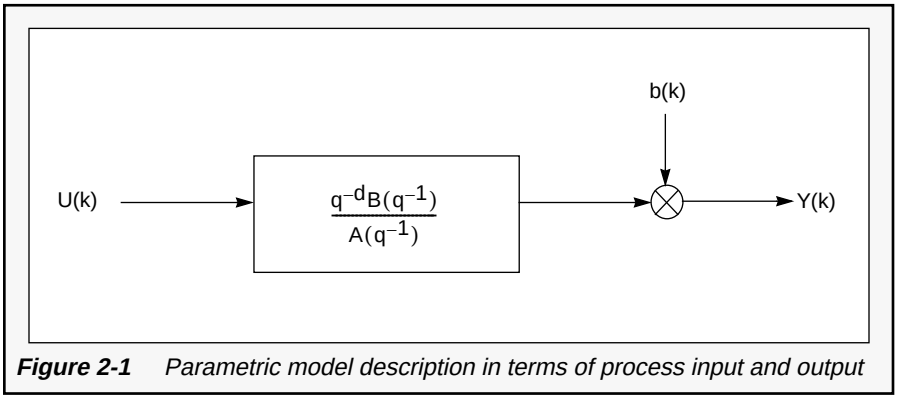

Figure 2-1 *Parametric model description in terms of process input and output*

with:

- *q^{-1} is the time delay operator.*
- *$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_n q^{-1}$*
- *$B(q^{-1}) = b_0 + \ldots + b_m q^{-1}$*
- *b(k) represents all noise sources expressed in terms of their equivalent effect on output.*

The model described by equation Eqn. 2-2 is known in the literature as the polynomial parametric model. Expression Eqn. 2-2 is solely in terms of the process input and output. The model can also be represented as a first-order differential equation by converting expression Eqn. 2-1. This representation is known as the parametric state model and is defined in accordance with equation Eqn. 2-3. Throughout the remainder of this application note we will assume that polynomial B(q-1) is of the same degree as polynomial A(q-1).

$$X_{k+1} = P \cdot X_k + Q \cdot U_k$$
$$Y_k = C \cdot X_k \qquad \text{Eqn. 2-3}$$

where:

- *X_k is the state vector of dimension ((n+d) x 1)*
- *P is the state matrix of dimension ((n+d) x (n+d))*
- *Q is the input vector of dimension ((n+d) x 1)*
- *C is the output vector of dimension (1 x (n+d))*
- *n is the order of the system*

The relation between these two representations of the parametric model is given by:

$$P = \begin{bmatrix} -a_1 & 1 & 0 & \cdots & 0 \\ 0 & & & & \\ -a_n & & & & 0 \\ 0 & & & & 1 \\ 0 & 0 & & \cdots & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ b_0 \\ \vdots \\ b_n \end{bmatrix} \} d \text{ (vector from Eqn. 2-2)} \quad C = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \qquad \text{Eqn. 2-4}$$

While it is true that the parametric model approximates the behavior of the physical system, one must be cautious when it comes to the physical interpretation of the parameters contributing to the model's structure.

The purpose of the parametric model is to approximate as closely as possible the behavior of the system by ensuring the closest possible match between predicted and observed output. This is done, moreover, within the limits of an accuracy vs. simplicity trade-off that the automatic control specialist defines when choosing the parametric model to generate the control law.

The advantages of the parametric model approach lie in its structure:

- *It enables us to describe, sufficiently accurately, the dynamics of an arbitrary physical process using fewer parameters that are required by the non-parametric model (finite impulse response).*

- *It is relatively simple to implement on the controller. Using a well-known property of the z-transform (time delay theorem), we can proceed from the polynomial parametric model to the difference equation of the following form:*

$$Y(k) = b_0 \cdot U(k-d) + \ldots + b_n \cdot U(k-n-d) - a_1 \cdot Y(k-1) - \ldots - a_n \cdot Y(k-n) + e(k)$$

Eqn. 2-5

with:
- *e (k) representing the generalized or residual noise*
- *e (k) = b (k).(1 + $a_1 q^{-1}$ +. . .+$a_n q^{-1}$)*

Given that we now have the time-history of the input and output signals, we can readily predict the model output values. This important point is widely used in modern regulation theory.The state parametric model is useful for describing multivariable systems.

The chief drawback of the parametric model is the difficulty of determining the order of the system. If the designer underestimates the process order, model predictions will not match actual system behavior. On the other hand, if the designer overestimates the order, the increased complexity of the model will mean longer computation times. This same comment also applies to the estimation of pure time delays. The automatic control specialist must therefore pay careful attention to this phase of the modelling procedure. With most industrial systems, we do not have access to the states values, which is a major handicap for the state parametric model. There are state observer techniques allowing the

state estimation, but having a heavy penalty in terms of computation time.

2.2 Adaptive Control Techniques

Consider the two adaptive control techniques applied to a closed-loop physical system as shown in Figure 2-2:

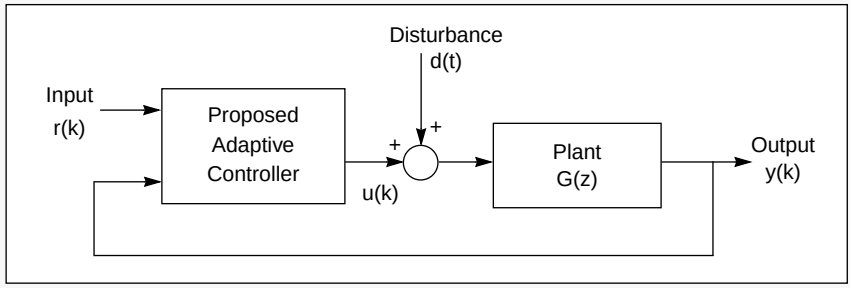

Figure 2-2 Adaptive controller in closed loop

This system will be used to measure the performance of the adaptive controller as it works to maintain (or equal) the desired response in the presence of a disturbance.

In these examples, G(z), the plant transfer function, is defined as follows:

$$G(z) = \frac{z^{-1} \cdot (b_0 + b_1 z^{-1})}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{Y(z)}{U(z)} \qquad \text{Eqn. 2-6}$$

The nominal values for the process parameters are:

$b_0 = 0.039$

$h = 0.031$

$a_1 = -1.457$

$a_2 = 0.527$

The performance of the adaptive controls presented here will be evaluated on the basis of the system's capacity to equal the closed-loop response (Figure 2-2) with the desired performance. Before going on to make the different comparisons, we must first define the standard (input and output) signals to be used with the simulated system and the desired closed-loop performance. The input and disturbance signals are shown in Figure 2-3. Note, these signals will be assumed to be fixed throughout the remainder of this section.

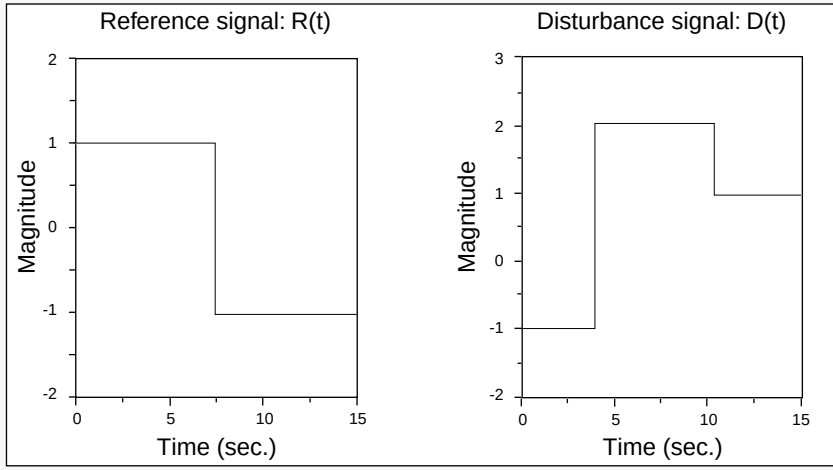

Figure 2-3 *Standard input and disturbance signals*

Noise free representation of the reference input and disturbance signals

In order to characterize the dynamic response of the uncorrected (i.e. without regulation) simulated system, we apply the input signal defined in Figure 2-3. The system dynamic response is illustrated in Figure 2-4.

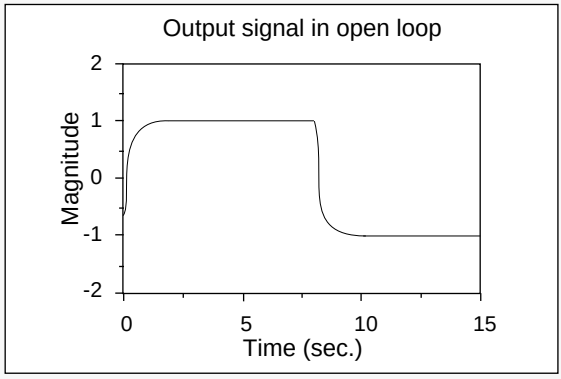

Figure 2-4 *Dynamic response in open loop*

Dynamic response of the uncorrected system when it has been excited by the reference signal of Figure 2-3.

The function of the different regulators presented in the following pages is to improve the dynamic behavior of the simulated system. Two constraints are imposed on these regulators. These constraints will be used, at first, to determine the desired performance of the closed-loop system. The constraints are defined as follows:

- *In order to respond more rapidly to variations in the reference value and/or the level of disturbance, we require that the simulated system have a settling time of no more than 2.5 seconds.*

- *During the dynamic response, the variation in the output signal of the simulated system shall be set for an overshoot of 70% relative to the final value.*

The desired system dynamic response is illustrated in Figure 2-5 (without disturbance):

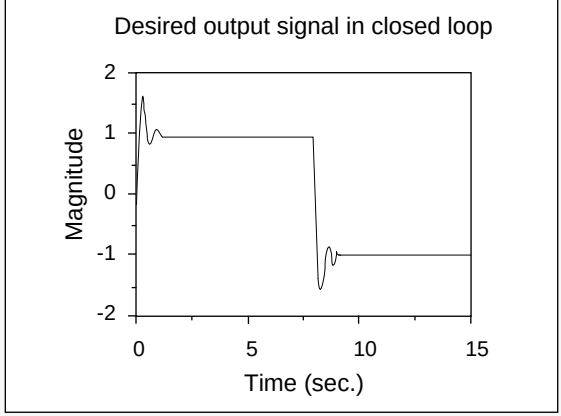

Figure 2-5 *Desired dynamic response in closed loop*

Dynamic response without disturbance of the closed loop system.

To illustrate and compare the performance of the different adaptive controllers, we introduce, for each method of adaptive regulation, a variation in the reference value (R(t), Figure 2-3), followed, as soon as the closed-loop system response has stabilized, by a disturbance (D(t), Figure 2-3). The impact of the disturbance is then monitored. ■

SECTION 3

Adaptive Control and Adaptive Controllers

3.1 Adaptive Control Using Reference Models

3.1.1 Introduction

"The main advantage of generalized predictive control is that the control is always stable irrespective of the nature of the system to be regulated (the plant). "

An adaptive controller may be of conventional design or it may be more complex in structure, including adjustable coefficients such that their tuning, using a suitable algorithm, either optimizes or extends the operating range of the process to be regulated. The different methods of adaptive control differ as to the method chosen to adjust (or tune) the control coefficients.

This section discusses adaptive control using parallel-serial reference models which, along with self-tuning control, are the only control schemes to have found practical applications to date. The adaptive control scheme using parallel reference models (i.e. located in parallel on the closed-loop system) was originally proposed by Whitaker in 1958. The version proposed at the time offered a solution to the tracking problem, but not the regulation problem. Note that a tracking problem is defined when the reference value

(r(k)) varies and when no disturbances (d(k)) are present in the output (y(k)). A regulation problem is defined when the reference value is zero or steady and when there is a disturbance in the output such that its effect must be reduced by the control (u(k)).

The parallel model structure is suitable for solving the tracking problem and is demonstrated by the fact that the model requires reasonable control signals; the structure is *not* suitable for solving regulation problems and is demonstrated by the fact that, in this case, the model requires unreasonable control signals. We obtain unreasonable control signals because the estimated error (differences between the output of the parallel reference model and that of the system) converges to zero during a single sampling interval. To attenuate the control signal, a serial reference model (i.e. in series with the estimated error) can be added to the general structure. This imposes a converge-to-zero requirement, with a chosen dynamic response, that is less severe than in the previous case [IRV-85]. Let us now look at this adaptive control method using parallel-serial reference models more closely.

3.1.2 Closed-Loop System

An adaptive control system comprises not only a feedback-type control loop (or inner loop) including an adaptive controller, but also an additional, or outer, loop acting on the controller parameters in order to maintain system performance in the presence of

variations in the process parameters. This second loop also has a feedback-loop-type structure, the controlled variable being the performance of the control system itself. The arrangement is schematically shown in Figure 3-1.

where: • *e_P represents the parallel estimated error*

• *e_S represents the serial estimated error*

This type of adaptive scheme offers the advantage of being able to accommodate separately both tracking and regulation problems. This is because the desired performance of the controlled system are defined by a parallel model for a tracking problem and by a serial model for a regulation problem.

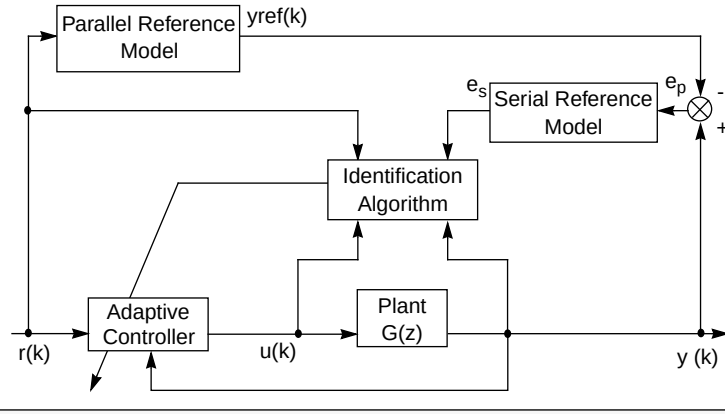

Figure 3-1 *Adaptive control using reference models in closed loop*

Note that this system not only has a feedback type control loop that includes an adaptive controller but also an outer loop that acts on the controller to maintain performance in the presence of disturbances.

3.1.3 Control Law

The dynamic behavior of the simulated system is defined by a parametric model. We recall that its general structure is given by the relation:

$$A(q^{-1}) \cdot Y(k) = q^{-d} \cdot B(q^{-1}) \cdot U(k) \qquad \text{Eqn. 3-1}$$

where:

- *n = 2 (order)*
- *d = 1 (time delay)*
- *$A(q^{-1}) = 1 + a_1.q^{-1} + a_2.q^{-2}$*
- *$B(q^{-1}) = b_0 + b_1.q^{-1}$*

The order of polynomials $A(q^{-1})$ and $B(q^{-1})$ and also the time delay of the parametric model enable us to correctly dimension the control law. To bring us nearer to the formulation of the adaptive control law, we first consider the case where the system parameters are known.

3.1.3.1 Known System Parameters

With the objectives of tracking and regulation being independent, we can formalize their respective equations as: A: Regulation (r(k) = 0).

The problem here is to determine a control (u(k)) that will eliminate an initial disturbance (d(k)) with a dynamic response defined by the relation:

$$A_r(q^{-1}) \cdot Y(k + d) = 0 \qquad \text{Eqn. 3-2}$$

with:

- *d = 1*
- *n = 2 (order)*

$$A_r(q^{-1}) = 1 + a_{r1}.q^{-1} + a_{r2}.q^{-2} \qquad \text{Eqn. 3-3}$$

The polynomial $A_r(q^{-1})$ is determined by the design engineer to be asymptotically stable for order n. The polynomial represents the serial (or regulation) model. B: Tracking (d(k) = 0).

The problem here is to determine a control (u(k)) such that the system output (y(k)) satisfies a relation of the form:

$$A_p(q^{-1}) \cdot Y(k + d) = B_p(q^{-1}) \cdot R(k) \qquad \text{Eqn. 3-4}$$

where:

- *n = 2 (order)*
- *d = 1 (time delay)*
- $A_p(q^{-1}) = 1 + a_{p1}.q^{-1} + a_{p2}.q^{-2}$
- $B_p(q^{-1}) = b_{p0} + b_{p1}.q^{-1}$

This corresponds to tracking a trajectory defined by the following reference model:

$$G_p(q^{-1}) = \frac{q^{-d} \cdot B_p(q^{-1})}{A_p(q^{-1})} \qquad \text{Eqn. 3-5}$$

In general, one may assume that there is some link between the tracking dynamic response $A_p(q^{-1})$ and the regulation dynamic response $A_r(q^{-1})$. However, in this application note, and for the sake of simplicity, we shall assume identical dynamic response to a variation in either load or reference value, i.e. we shall assume $A_p(q^{-1}) = A_r(q^{-1})$. We

shall further assume a reference model such that the output is described by the relation:

$$A_p(q^{-1}) \cdot Yref(k+d) = B_p(q^{-1}) \cdot R(k) \qquad \text{Eqn. 3-6}$$

Under these conditions, the aims to be achieved by the control signal can be expressed in the form:

$$e_s(k+d) = A_p(q^{-1}) \cdot [Y(k+d) - Yref(k+d)] = 0 \qquad \text{Eqn. 3-7}$$

The control law, with a parallel-serial reference model, can be deduced by minimizing the following quadratic criterion:

$$J(k+d) = e_s^2(k+d) = [A_p(q^{-1}) \cdot [Y(k+d) - Yref(k+d)]]^2 \qquad \text{Eqn. 3-8}$$

In the case of unit time delay (d = 1), we can determine the control law directly by minimizing criterion Eqn. 3-8 relative to u(k). The problem may be different, however, if the pure time delay of the controlled system is equal to or greater than twice the sampling period. In order to obtain a causal regulator, i.e. one such that u(k) is of the form:

$$U(k) = F_u(Y(k), Y(k-1), \ldots\ldots, U(k-1), \ldots) \qquad \text{Eqn. 3-9}$$

We must first rewrite the process output prediction in terms of the quantities measurable at time k and prior to time k. The prediction can be expressed in the form:

$$A_p(q^{-1}) \cdot Y(k+d) = F_y(Y(k), Y(k-1), \ldots\ldots, U(k), U(k-1), \ldots)$$

Eqn. 3-10

In the literature, an expression of this form is known as a “d-step-ahead predictive model”. An expression such as Eqn. 3-10 can be obtained directly using the general polynomial identity:

$$A_p(q^{-1}) = A(q^{-1}) \cdot S(q^{-1}) + q^{-d} \cdot R(q^{-1}) \qquad \text{Eqn. 3-11}$$

where:

- $S(q^{-1}) = 1 + s_1.q^{-1} + \ldots + s_{d-1}.q^{-d+1}$
- $R(q^{-1}) = r_0 + r_1.q^{-1} + \ldots + r_{n-1}.q^{-n+1}$

This relation yields a unique solution for polynomials $S(q^{-1})$ and $R(q^{-1})$ when the degree of $S(q^{-1})$ is d–1. Polynomials $S(q^{-1})$ and $R(q^{-1})$ can be obtained either recursively or by dividing polynomial $A_p(q^{-1})$ by polynomial $A(q^{-1})$. Polynomial $S(q^{-1})$ then corresponds to the quotient while $q^{-d}.R(q^{-1})$ corresponds to the remainder. Multiplying both sides of Eqn. 3-11 by y(k+d) and taking into account expression Eqn. 3-1, we obtain:

$$A_p(q^{-1}) \cdot Y(k+d) = R(q^{-1}) \cdot Y(k) + B(q^{-1}) \cdot S(q^{-1}) \cdot U(k)$$

Eqn. 3-12

This can be rewritten in the form:

$$A_p(q^{-1}) \cdot Y(k+d) = R(q^{-1}) \cdot Y(k) + b_0 \cdot U(k) + B_s(q^{-1}) \cdot U(k-1)$$

Eqn. 3-13

where: $B(q^{-1}).S(q^{-1}) = b_0 + q^{-1}.B_S(q^{-1})$

Substituting Eqn. 3-13 into criterion expression Eqn. 3-8, we obtain:

$$J(k+d) = [R(q^{-1}) \cdot Y(k) + b_0 \cdot U(k) + B_s(q^{-1}) \cdot U(k-1) - A_p(q^{-1}) \cdot Yref(k+d)]^2$$

Eqn. 3-14

The criterion can now be minimized by determining the control u(k) for which:

$$\frac{\delta J(k+d)}{\delta U(k)} = 0$$

Eqn. 3-15

Combining this with expression Eqn. 3-14, we obtain:

$$\frac{\delta J(k+d)}{\delta U(k)} =$$

$$b_0 \cdot [R(q^{-1}) \cdot Y(k) + b_0 \cdot U(k) + B_s(q^{-1}) \cdot U(k-1) - A_p(q^{-1}) \cdot Yref(k+d)] = 0$$

Eqn. 3-16

Now, using expression Eqn. 3-6, we obtain the required control in the form:

$$U(k) = \frac{1}{b_0} \cdot [B_p(q^{-1}) \cdot R(k) - R(q^{-1}) \cdot Y(k) - B_s(q^{-1}) \cdot U(k-1)]$$

Eqn. 3-17

where polynomials Bp(q-1), R(q-1) and Bs(q-1) are defined by:

$$B_p(q^{-1}) = b_{p0} + b_{p1}.q^{-1}$$

$$R(q^{-1}) = (a_{p1} - a_1) + (a_{p2} - a_2).q^{-1} = r_0 + r_1.q^{-1}$$

$$B_s(q^{-1}) = b_1 = b_{s0}$$

The control expressed in relation Eqn. 3-17 thus has the property of reducing criterion Eqn. 3-8 to zero while independently meeting the requirements of both tracking and regulation.

In other words, in the case of regulation (r(t) = 0), criterion expression Eqn. 3-8 represents a minimum-variance condition on the process output. Physically, this criterion implies minimizing the mean energy of the "filtered error" expression in relation Eqn. 3-7.

The equations presented in this section were made possible by the fact that we knew the parameters of the controlled process. Let us now look at the case where these process parameters are unknown.

3.1.3.2 Unknown System Parameters

In the adaptive case, the structure of the controller is the same as for known system parameters, except that we replace the fixed parameters by variable ones. With the role of the adaptive, or outer, loop being to determine the correct values of these parameters, the self-tuning controller equation can be derived from Eqn. 3-17 and written as:

$$U(k) = \frac{1}{\hat{b}_0(k)} \cdot [B_p(q^{-1}) \cdot R(k) - \hat{R}(k, q^{-1}) \cdot Y(k) - \hat{B}_S(k, q^{-1}) \cdot U(k-1)]$$

Eqn. 3-18

where: *$b_o(k), r_o(k), \ldots, b_{S1}(k), \ldots$, are the controller parameter estimates at time k*

By defining the tuning vector $\theta(k)$ and the measurement vector $\Psi(k)$ by the following expressions:

$$\hat{\theta}^T(k) = \left[\hat{b}_0(k) \quad \hat{b}_{s0}(k) \quad \hat{r}_0(k) \quad \hat{r}_1(k) \right]$$

Eqn. 3-19

$$\Psi(k) = \left[U(k) \quad U(k-1) \quad Y(k) \quad Y(k-1) \right]$$

The controller equation can be rewritten in the form:

$$B_p(q^{-1}) \cdot R(k) = \hat{\theta}^T(k) \cdot \Psi(k)$$

Eqn. 3-20

The next step is to determine the recursive parameter-vector self-tuning algorithm.

3.1.4 Determination of Controller Parameters

The self-tuning controller parameters are determined by recursive minimization of a least-squares type criterion starting from asymptotic stability conditions dictated by the model-process error. The aim then is to estimate the parameter vector at time k in such a way that it minimizes the sum of the squares of the filtered errors between the process and the model over a time-horizon of k measurements. This is expressed by the relation:

$$J_1(k) = \sum_{i=1}^{k} e_s^2(i) = \sum_{i=1}^{k} [A_p(q^{-1}) \cdot (Y(i) - Yref(i))]^2$$

Eqn. 3-21

This same condition can also be expressed in the form:

$$J_1(k) = \sum_{i=1}^{k} [B_p(q^{-1}) \cdot R(i) - \hat{\theta}^T(i) \cdot \Psi(i)]^2 \qquad \text{Eqn. 3-22}$$

The values of $\theta(k)$ which minimize criterion Eqn. 3-22 are obtained by determining the value of $\theta(k)$ which cancels in the expression:

$$\frac{\delta J_1(k)}{\delta \hat{\theta}(k)} = 0 \qquad \text{Eqn. 3-23}$$

Applying relation Eqn. 3-23 to relation Eqn. 3-22, we obtain:

$$\frac{\delta J_1(k)}{\delta \hat{\theta}(k)} = -\sum_{i=1}^{k} [\Psi(i) \cdot [B_p(q^{-1}) \cdot R(i) - \hat{\theta}^T(i) \cdot \Psi(i)]] = 0$$

Eqn. 3-24

From equation Eqn. 3-24 we have:

$$\hat{\theta}(k) = \left[\sum_{i=1}^{k} \Psi(i) \cdot \Psi^T(i)\right]^{-1} \cdot \sum_{i=1}^{k} B_p(q^{-1}) \cdot R(i) \cdot \Psi(i)$$

Eqn. 3-25

In the previous expression, we now let:

$$\hat{\theta}(k) = F(k) \cdot \sum_{i=1}^{k} B_p(q^{-1}) \cdot R(i) \cdot \Psi(i) \qquad \text{Eqn. 3-26}$$

where: $F^{-1}(k) = \left[\sum_{i=1}^{k} \Psi(i) \cdot \Psi^T(i) \right]^{-1}$

Expression Eqn. 3-25 corresponds to the non-recursive least-squares algorithm. To obtain a recursive algorithm, we recompute the optimal value of θ(k+1) for the minimization condition J(k+1) and express θ(k+1) as a function of θ(k). This yields:

$$\hat{\theta}(k+1) = \hat{\theta}(k) + F(k+1) \cdot \Psi(k) \cdot e_S(k+1) \qquad \text{Eqn. 3-27}$$

where: $F^{-1}(k+1) = F^{-1}(k) + \Psi(k+1) \cdot \Psi^T(k+1)$

Here, F(k+1) represents the estimator tuning gain. This is an important variable since it gives us an indication of the quality of estimation (covariance of parameter estimates).

It has been shown elsewhere [LJU-83] that if k (experiment time) increases, the θ(k) estimates tend towards constants. In this case, the variance of the estimates tends towards zero (F(k+1) = 0). The

least-squares algorithm briefly presented here has progressively less effect on new measurement values. This is acceptable if the process is unvarying in time. However, this is not the case in this application note since the system parameters are explicitly assumed variable. This problem can be resolved by modifying the $J_1(k)$ criterion. We need to arrange for the criterion to "forget" earlier measurement values by adding a suitable weighting factor. When this is done, the criterion to be minimized becomes:

$$J_2(k) = \sum_{i=1}^{k} \lambda^{k-i} \cdot e_S^2(i) \qquad \text{Eqn. 3-28}$$

where: λ *represents the weighting, or "forgetting factor" (0 < l ≤ 1)*

The thus modified least-squares algorithm is detailed in **APPENDIX A**. The main difference between algorithms is which variables are contained in vector $\Psi(k)$. In the literature, this quantity is referred to as the **"measurement vector"** while $e_S(k)$ is termed the **"post-prediction tuning error".**

In order to ensure the stability of the overall system, the recursive least-squares identification algorithm must meet the following three conditions:

- *The rapid decrease in the prediction error ($e_S(k)$) must occur during the periods when $\theta(k)$, the unknown parameter of the system to be identified, is constant.*

- *Irrespective of any variations in the domain bounded by $\theta(k)$, the adjusted parameter $\theta(k)$ of the identifier must remain within the appropriate bounded domain.*
- *The variation $\theta(k) - \theta(k-1)$ in the estimated parameter must decrease at the same time as the prediction error $e_S(k)$. If $e_S(k)$ is below a certain threshold, then $\theta(k) - \theta(k-1)$ must be zero.*

These conditions can only be met by making further changes to the recursive least-squares algorithm. Several authors [IRV-85 and BOD-87] have already tackled this problem. We have used their results to improve the robustness of the controller parameter estimation algorithm.

3.1.5 Comment

The use of a control strategy based on an output-signal minimum-variance criterion theoretically requires that the system to be regulated (the plant) have a stable inverse (i.e. $b_0 > b_1$). It is therefore important to have some prior knowledge of the nature of the plant, and its behavior over its entire operating range. Parametric identification is used to determine not only the structure of the representation model (order and time delays), but also the nature of the system to be regulated (i.e. whether it is a minimum-phase system or not). Note also that this type of controller can be used to define tracking and regulation performance totally independently.

The main disadvantage of a control strategy using a minimum-variance criterion applied to the variable

to be regulated is that it always leads to a direct adaptive scheme. This presents a problem for the other control strategies where the control law parametering is broken down into two distinct steps, namely:

- *Estimation of parameters of the system representative model, and*
- *Adjustment of controller parameters using system parameters.*

This method of breaking down control law parametering leads to an indirect adaptive scheme. Note, however, that it can be an advantage to have a means of monitoring system dynamic response in real time. Thus, the estimation of process parameters can be used for diagnostics, monitoring, etc. Let us now look at this indirect adaptive scheme more closely.

3.2 Generalized Predictive Control

3.2.1 Introduction

The adaptive control scheme presented in the previous section is useful when the system to be controlled has a stable inverse. This leads to investigations to see if other control schemes, associated with the least-squares identification method, can generate stable control signals irrespective of the nature of the system to be controlled. Given that the

minimization of the mean tracking error energy (Eqn. 3-21) is not sufficient to ensure control stability in the case of a so-called “non-minimum phase system”, it seems fairly natural to investigate what happens if one introduces a control weighting term into the expression for the criterion to be minimized [SAM-83]. This is expressed by the relation:

$$J_3(k) = \sum_{i=1}^{k} [A_p(q^{-1}) \cdot (Y(i) - Yref(i))]^2 + \alpha \cdot U(k)^2$$

Eqn. 3-29

where: α is the strictly positive weighting term

An improvement in this criterion has been suggested on the basis of the following observation. A car driver does not need to have a complex mathematical model in mind in order to be able to drive. All he needs is the ability to recall a set of images of possible trajectories produced by a corresponding set of control actions on the car steering wheel. Given the driver's view of the road to be followed, the human control algorithm chooses the control action (or signal) that will produce the vehicle trajectory closest to the desired trajectory [IRV-85].

From this we conclude, in other words, that to obtain a robust control scheme, we can use the predictions obtained from the identification of the system to be controlled and minimize a least-squares criterion involving the difference between the predicted desired trajectory and the predicted trajectories in response to the control signals. This

criterion has been formulated by Clarke [CLA-84] and is expressed in the form:

$$J_4(k) = \sum_{i=0}^{N_y-1} [Yref(yk+i+d) - \hat{Y}(k+i+d)]^2 + \sum_{i=0}^{N_u-1} \alpha \cdot \Delta \cdot U(k+1)^2$$

Eqn. 3-30

where:

- $\hat{Y}(k+i+d)$ *is the output prediction with Ny*
- *Yref is the predicted output of the reference model over horizon* N_y
- *U(k+i) represents the predicted control over* N_u
- *Ny determines the horizon on the outputs*
- *Nu determines the horizon on the control*
- α *is the control weighting factor*
- Δ *represents the differentiation operator* ($\Delta = 1-q^{-1}$)

Thus, the control weighting term (α) ensures control stability in all cases where the system has an unstable inverse, provided the time delay is greater than unity. The differentiation operator (Δ) enables us to obtain a control that is free of static error in the variable to be controlled (Y) relative to the reference trajectory (Yref).

On the basis of our bibliographic research, generalized predictive control is considered to be the best control technique currently available. This is why we chose to discuss it in detail in this application

note and why we made it the subject of our simulation studies. An industrial application of this technique is described in [LIM-89].

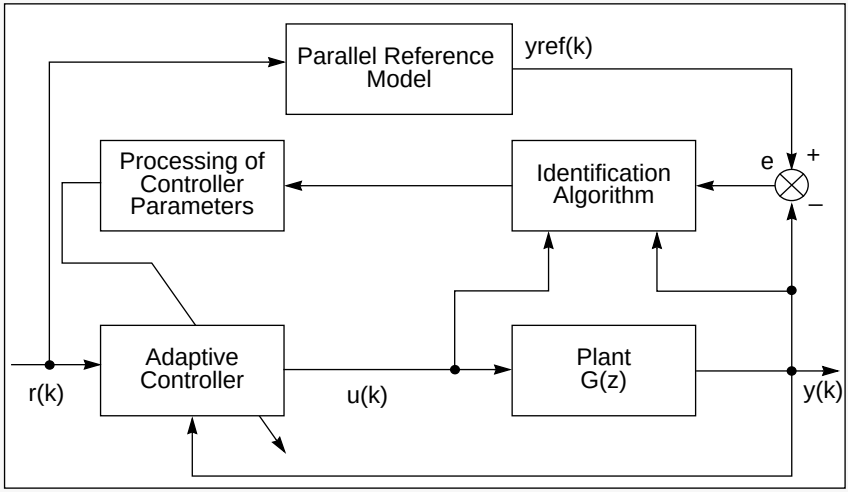

Figure 3-2 *Generalized predictive control using closed loop*

Note the presence of an additional loop to perform system identification on the process. The advantage is that the process parameters would be accessible with the main disadvantage that there is an increase time in the computation of the control law.

3.2.2 Closed-Loop System

As with all adaptive control systems, the system discussed in this section features not only a conventional servo-type feedback loop, but also an additional loop designed to identify the on-line process and determine the parameters to be adjusted on the basis of the process parameters. The arrangement is schematically shown in Figure 3-2.

The main advantage of this indirect adaptive scheme is that it gives access to the process parameters, which is important for monitoring, diagnostics, and the like. The main disadvantage is the increased computation time required to parameter the control law.

3.2.3 Control Law

3.2.3.1 Definition of Parametric Model

The parametric model required to formulate the control law was defined earlier on. Recall that the mathematical structure is of the form:

$$\hat{A}(k, q^{-1}) \cdot Y(k) = q^{-d} \cdot \hat{B}(k, q^{-1}) \cdot U(k) \qquad \text{Eqn. 3-31}$$

where: A(k,q-1) and B(k,q-1) are the polynomials estimated by the identifier at each sampling interval.

In the remainder of this application note, we will simplify the mathematical notation by omitting the ^ symbols (indicating estimated variables) and the (k) portion of the different terms indicating that the variable is estimated at each sampling interval k.Te.

3.2.3.2 Definition of System Output Prediction

The prediction of the parametric model output — which is to say the probable behavior of the process output between time k_1 and some future time kj — is deduced using a j-step-ahead prediction model.

The general expression for such a model is:

$$1 = E_j(q^{-1}) \cdot A(q^{-1}) \cdot \Delta + q^{-d} \cdot F_j(q^{-1}) \qquad \text{Eqn. 3-32}$$

where:

- *j = 1..kj*
- *kj is the given future time-horizon*
- $F_j(q^{-1}) = f_{0j} + ... + f_{(kj-1)j}.q^{-kj+1}$
- $E_j = 1 + ... + e_{j-1}.q^{-j+1}$

This expression is known as a Diophantine equation. The expression for the predicted model output can be deduced by multiplying the two sides of equation Eqn. 3-30 by Ej.Δ, then substituting the expression for Ej.A(q-1).Δ from equation Eqn. 3-32. This gives:

$$(1 - q^{-j} \cdot F_j(q^{-1})) \cdot Y(k) = q^{-d} \cdot E_j(q^{-1}) \cdot B(q^{-1}) \cdot \Delta \cdot U(k)$$

Eqn. 3-33

The expression for the predicted model output can now be rewritten in the form:

$$\hat{Y}(k + j) = F_j(q^{-1}) \cdot Y(k) + G_j(q^{-1}) \cdot \Delta \cdot U(k - d + j)$$

Eqn. 3-34

where: $G_j(q^{-1}) = E_j(q^{-1}).B(q^{-1})$

The sequence of predicted parametric model outputs can now be represented by vector $\underline{Y}$. Note that for all future sampling intervals smaller than or equal to the system time delay (i.e. for $j \leq d$), the Y(k+j) values can be computed using the input and output data available up to time k. For sampling intervals greater than the system time delay (i.e. for $j > d$),

we need to know the future control U(k+j). These assumptions form the basis of generalized predictive control.

3.2.3.3 Determination of Polynomials $F_j(q^{-1})$ and $G_j(q^{-1})$

In **Section 3.1.3.1** we showed that we could obtain the polynomials S(q-1) and R(q-1) by simple division. The disadvantage, however, of this technique is that it is very time consuming. Clarke proposes a recursive method for the determination of polynomials $F_j(q\text{-}1)$ and $E_j(q\text{-}1)$. This is the solution we have adopted. Readers interested in this reformulation of the polynomials in recursive form should refer to the bibliography, and particularly to [BOD-87], [CLA-85], and [AST-84]. Note, the control algorithm encoded in Motorola DSP56000/DSP56001 digital signal processors is based on the same solution.

3.2.3.4 Determination of Control Law

Above, we derived an expression (Eqn. 3-34) for predicting the behavior of the process output signal. The behavior of the reference model output signal, on the other hand, is predicted by expression (Eqn. 3-35). We must now solve this equation from sampling time (k+d) to the chosen time-horizon (k+d+N–1).

$$Yref(k+d) = -A_p^*(q^{-1}) \cdot Yref(k+d-1) + B_p(q^{-1}) \cdot R(k)$$

Eqn. 3-35

where: $A_p^*(q^{-1}) = a_1 \cdot q^{-1} + \ldots + a_n \cdot q^{-n}$

Determining the output of the parallel reference model at a future time is not difficult since the polynomials A_p and B_p are known and are constant at each sampling time. The sequence of reference model outputs from sampling time (k+d), can be expressed in vector form as follows:

$$\underline{Yref} = [Yref(k+d) \ldots\ldots Yref(k+d+N-1)] \qquad \text{Eqn. 3-36}$$

Recall that the parallel reference model plays the same role as that defined for the adaptive controller of a parallel-serial model (**Section 3.1**). In order to compare the performance of the two adaptive regulators, we choose the same structure (n = 2 and d = 1) for the parallel model and the same dynamic response.

The prediction error at future time k+j is given by:

$$e(k+j) = Yref(k+j) - \hat{Y}(k+j) \qquad \text{Eqn. 3-37}$$

Proceeding as previously described, let us now define the prediction error vector:

$$\underline{e} = [e(k+d) \ldots\ldots e(k+d+N-1)] \qquad \text{Eqn. 3-38}$$

Thus:

$$\underline{e} = \underline{Yref} - \underline{Y} \qquad \text{Eqn. 3-39}$$

We saw earlier that at time k, certain elements of vector $\underline{Y}$ are functions of known and unknown data. Among the unknowns, we can define the predicted

control vector (U) as follows:

$$\underline{U} = [\Delta.U(k) \ldots\ldots \Delta.U(k+N-1)] \qquad \text{Eqn. 3-40}$$

A clever decomposition of expression Eqn. 3-39 enables us to separate the terms that depend on known data at time k and those that are unknown at time k, such as vector U. We thus obtain:

$$\underline{e} = \underline{Yref} - \underline{G} \cdot \underline{U} - \underline{f} \qquad \text{Eqn. 3-41}$$

where: G is a triangular matrix of dimension N.N

The elements of G are generated by reformulating the Diophantine equation in recursive form.

$$\underline{G} = \begin{bmatrix} g_0 & 0 & \bullet & \bullet & \bullet & \bullet & 0 \\ g_1 & g_0 & \bullet & & & & \bullet \\ \bullet & \bullet & \bullet & \bullet & & & \bullet \\ \bullet & \bullet & & \bullet & \bullet & & \bullet \\ \bullet & \bullet & & & \bullet & \bullet & \bullet \\ \bullet & \bullet & & & & \bullet & 0 \\ g_N & g_{N-1} & \bullet & \bullet & \bullet & \bullet & g_0 \end{bmatrix} \qquad \text{Eqn. 3-42}$$

The elements of vector $\underline{f}$ are the components of the part of the prediction depending on known data at time k. This vector can be written in the form:

$$\underline{f} = \begin{bmatrix} F_d(q^{-1}) \cdot Y(k) + q\left[G_d(q^{-1}) - g_0\right] \cdot \Delta \cdot U(k-1) \\ F_{d+1}(q^{-1}) \cdot Y(k) + q^2 \cdot \left[G_{d+1}(q^{-1}) - g_0 - g_1 \cdot q^{-1}\right] \cdot \Delta \cdot U(k-1) \\ \vdots \\ F_{d+N-1}(q^{-1}) \cdot Y(k) + q^N \cdot \\ \quad [G_{d+N-1}(q^{-1}) - g_0 - g_1 \cdot q^{-1} \ldots\ldots g_{N-1} \cdot q^{-N+1}] \cdot \Delta \cdot U(k-1) \end{bmatrix}$$

Eqn. 3-43

The control vector, $\underline{U}$, can be determined by minimizing criterion $J_4(k)$ as expressed in equation Eqn. 3-40. In vector form, this criterion is given by:

$$J_4 = \underline{e}^T \cdot \underline{e} + \alpha \cdot \underline{U}^T \cdot \underline{U} \qquad \text{Eqn. 3-44}$$

It can be shown that this criterion has a simple optimal solution for:

$$\underline{U} = [\underline{G}^T \cdot \underline{G} + \alpha \cdot I]^{-1} \cdot \underline{G}^T \cdot [\underline{Yref} - \underline{f}] \qquad \text{Eqn. 3-45}$$

Control u(k) is computed from $\Delta u(k)$ using the following expression:

$$U(k) = U(k-1) + \Delta \cdot U(k-1) \qquad \text{Eqn. 3-46}$$

The power of generalized predictive control can be gauged from the fact that it allows us to reduce the

control prediction time-horizon. The only difficulties are the mathematical problems posed by the inversion of matrix ($G^T.G + a.I$) of dimension N.N and the associated computation times. But this can be overcome by defining a control prediction time-horizon such that $N_u < N$ and an output prediction time-horizon such that $N_y = N$. This reduces the number of columns of matrix ($G^T.G$) so that we can now write:

$$\Delta \cdot U(k + N_u) = \Delta \cdot U(k + N_u + 1) = \ldots\ldots = 0$$

Eqn. 3-47

3.2.4 Comment

The main advantage of generalized predictive control is that the control is always stable irrespective of the nature of the system to be regulated (the plant). Thus, without making any changes to the control law obtained by minimizing criterion Eqn. 3-40, generalized predictive control can readily control systems with an unstable inverse matrix.

The approach suffers, however, from one serious drawback. The weighting factor ‘a’ plays a determining role in system dynamic response, enabling us to obtain reasonable control signals for trajectory tracking or for attenuating the effects of disturbance. This is not, however, very satisfactory since ‘a’ defines the dynamic response of the loop system in a fashion that is difficult to determine in advance. In our study, the only way we found of approaching the desired performance through the adjustment of a was by iterative trial and error. This is a step backwards compared to the asymptotic performance of adaptive

control using parallel-serial reference models where the desired dynamic response was explicitly contained in polynomials A_p and A_r. Worse still, when the parameters of an industrial system vary from one operating point to another, weighting factor a must be modified to match the variation in the process dynamic response.

This is not to say that generalized predictive control is not promising, but rather that further improvements are still required. Irving [IRV-85], for instance, proposes generalized predictive control with dual parallel-serial reference models — one to attenuate tracking and/or regulation dynamic response, the other to moderate control dynamic response. ■

SECTION 4

Implementation and Simulation of Adaptive Controllers Using Reference Models

4.1 Implementation

"The identification algorithm in this program and the one which follows is a slightly more complex algorithm but it enables us to avoid the problem of divergent real-time estimated parameters while ensuring improved parameter tracking during transitions."

The scheme for adaptive control using parallel-serial reference models presented in Section 3.1 has been used to implement an adaptive controller on Motorola DSP56000/DSP56001 digital signal processors. The controller logic is programmed in assembler using a Motorola DSP56000CLASA macro cross-assembler.

The following pages present listings of the linked adaptive control program and the object code generated by the macro cross-assembler. Note that the identification algorithm in this program and the one which follows is a slightly more complex algorithm but it enables us to avoid the problem of divergent real-time estimated parameters while ensuring improved parameter tracking during transitions.

The adaptive controller using parallel-serial reference models presented requires about 5,000 instruction

cycles from input signal acquisition (r(k), y(k)) to control (u(k)) output. With the DSP56000/ DSP56001 devices running at 27 MHz, this represents a minimum sampling period of 410 μs.

4.1.1 Simulation

Using the same conventions for input signals (r(k)) and disturbance (d(k)) as in **SECTION 2**, Figure 4-1 shows the simulated system output signal (y(k)) and the control (u(k)) generated by the adaptive controller using parallel-serial reference models. In this simulation, the subroutine “saturation” was not activated.

4.2 Generalized Prediction Controllers

4.2.1 Implementation

The scheme for generalized prediction controller presented in **Section 3.2** has been used to implement an adaptive controller on a Motorola DSP56000 Digital Signal Processor. It is coded with C language using the Motorola GNU 56K C compiler. This compiler offers two important advantages: first it enables us to insert assembly language code into the source code, secondly, it allows us to generate an object code when the compiling and linking have been completed.

The following pages present a C language implementation of the Generalized Predictive Controller and the Adaptive Controller with Parallel and Serial Models.The identification algorithm takes about 3,100 instruction cycles. The control algorithm, which is based on the theory of the prediction of the output of the identified system (y(k)) at a given time horizon (ny) and the prediction of the control (u(k)) also at a given time-horizon (nu) takes about 8,200 instruction cycles.

The generalized predictive controller presented in Figure 4-2 requires about 12,000 instruction cycles from input signal acquisition to control output. With the DSP56001 running at 27 MHz, this represents a minimum sampling period of 890 microseconds.

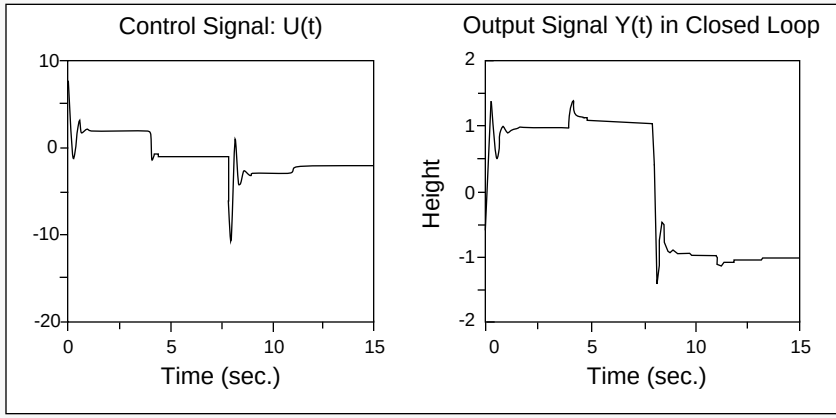

Figure 4-1 *Simulation results for adaptive controller using parallel-serial reference models*

Note the effects of the disturbance on the output signal in this closed loop.

```
/*************************************************************************
MOTOROLA GENEVA
*************************************************************************

        Name:      Pascal RENARD
        Type:           C_Program
        Version:   1.1
        Last Change:1 April 1992
        Function:  Generalized Predictive Controller
        Device:    DSP56000/1
        File name: main.c
*************************************************************************/

Conversion to GNU based C compiler
Based on Version 1.0 14 June 1990 Program
Ed Martinez
Motorola Inc.
Digital Signal Processing Operations
Austin, TX 78735

/*************************************************************************
Global variables
These variables are declared at the global level. They will be accessed from
the different assembly language sections within the program
*************************************************************************/
int Ynew_u__asm("Ynew_u");
int Ynew_y__asm("Ynew_y");
int Ynew_r__asm("Ynew_r");

/*************************************************************************
The following are variables that are defined later on using
in-line assembly language. They are not available to the
c program directly.
Yin_y     $ffe0
Yin_r     $ffc0
Yout_u    $ffe1

/*************************************************************************
parallel parameters with the following structure:
*************************************************************************/

bm[0].z-1 + bm[1].z-2
-----------------------------
1 + am[0].z-1 + am[2].z-2
*************************************************************************/
typedef struct {
        float am[2],bm[2];
        } e_param;

/*************************************************************************
parallel model parameters
*************************************************************************/
typedef struct {
        float am[2],bm[2];
        }para_param;
```

Figure 4-2 *Generalized Predictive Controller described in Section 3.2 has a sampling period of 890 μs using a 27MHz DSP56001.(sheet 1 of 11)*

```
/*********************************************************************
identification parameters
structure of the estimated model

theta[2].z-1 + theta[3].z-2
-------------------------------
1 + theta[0].z-1 + theta[1].z-2

*********************************************************************************/
typedef struct {
                  int n,na;
                  float theta[4],fi[4];
                  float p0,v_conv,tr_min,tr_max;
                  float eps,eps1,lambda,trace;
                  float diag[4],offdiag[7];
                  } estpartyp;

/*********************************************************************************
controller parameters
*********************************************************************************/
typedef struct {
                  float lambda2,du,u_low,u_high;
                  int nu,ny;
                  } r_param;

/*********************************************************************************
input and output signals
*********************************************************************************/
typedef struct {
                  float tab_du[7],ui[7], yi[4], ri[4], ymi[4];
                  } io_param;

/*********************************************************************************
pointer initializations
*********************************************************************************/
e_param ep,*pep = &ep;
estpartyp est,*pest = &est;
r_param rp,*prp = &rp;
io_param iop,*piop = &iop;

/*********************************************************************************
main program
*********************************************************************************/

main ()
{
        __asm("\nYin_y    equ     $ffe0");
        __asm("\nYin_r    equ     $ffc0");
        __asm("\nYout_u   equ     $ffe1");



initialize(pest,prp,piop,pep);/* parameter initialization */
for (;;)          /* infinite loop */
        {

        __asm volatile ("movep y:Yin_y,y0" ::: "y0"); /* Obtain system input */
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 2 of 11)*

```
__asm volatile ("move y0,Y:Ynew_y" ::: "y0");
__asm volatile ("movep y:Yin_r,y0" ::: "y0"); /* Obtain system reference */
__asm volatile ("move y0,y:Ynew_r" ::: "y0");


datain_shift(piop,&Ynew_r,&Ynew_y);  /*(yi[], ri[]) data shift */
ls(piop,pest);                       /* identification */
gene_pre_com(piop,pest,pep,prp);     /* control signal computing */

datau_shift(piop,prp);               /* ui[] data shift */
saturation(piop,prp,&Ynew_u);        /* saturation of ui[0] */
                                     /* output of control signal */

__asm("move y:Ynew_u,y0" ::: "y0");   /* output of control signal */
__asm("movep y0,y:Yout_u" ::: "y0");

datadu_shift(piop);                  /* tab_du[] data shift */
       }

}

/*********************************************************************
initialization
*********************************************************************/
initialize(pest,prp,piop,pep)

estpartyp *pest;
r_param *prp;
io_param *piop;
e_param *pep;
{
int i;                    /* identification parameters initialization */
pest->n = 4;              /* number of parameter estimated */
pest->na = 2;             /* system order */
     /* this parameter is proportional to variance of estimates */
pest->p0 = 1000.0;
pest->v_conv = 0.1;       /* convergence speed of estimates */
pest->tr_min = 5.0;       /* minimum of trace */
pest->tr_max = 15.0;    /*maximum of trace used to avoid estimate divergence */
pest->eps = 0.0;          /* output of serial model at time tk */
pest->eps1 = 0.0;         /* output of serial model at time tk-1 */
pest->lambda = 1.0;       /* forgotten factor of identification algorithm */
pest->theta[0] = -1.936;  /* preset value for estmates */
pest->theta[1] = 0.937;   /* " */
pest->theta[2] = 0.0001;  /* " */
pest->theta[3] = 0.0001;  /* " */
pest->fi[0] = -1.0;       /* preset value for input matrix */
pest->fi[1] = -1.0;       /* " */
pest->fi[2] = 0.5;        /* " */
pest->fi[3] = 0.5;        /* " */
for (i=0 ; i<=(pest->n)-1 ; i++)
       pest->diag[i] = pest->p0; /* initialization of diagonal matrix */
for (i=0 ; i<=((pest->na)*4)-2 ; i++)
       pest->offdiag[i] = 0.0;
                         /* initialization of upper-triangular matrix */
                          /* controller parameters initialization */
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 3 of 11)*

```
prp->nu = 3;               /* horizon on the control signal */
prp->ny = 8;               /* horizon on the plant output */
prp->u_low = -5.0;         /* negative saturation for control signal */
prp->u_high = 5.0;         /* positive saturation for control signal */
prp->lambda2 = 0.5;        /* weighting factor on control signal criterion */

                                   /* I/O parameters initialization */
for (i=0 ; i<=((pest->na)*4)-2 ; i++)
        piop->tab_du[i] = 0.0;     /* vector of control signal variation */

for (i=0 ; i<=((pest->na)*4)-2 ; i++)
        piop->ui[i] = -0.5;        /* preset value on control signal*/

for (i=0 ; i<=(pest->n)-1 ; i++)
        piop->yi[i] = -1.0;        /* preset value on model output */

for (i=0 ; i<=(pest->n)-1 ; i++)
        piop->ri[i] = -1.0;/* preset value on reference signal */

for (i=0 ; i<=(pest->n)-1 ; i++)
        piop->ymi[i] = -1.0;       /* preset value on parallel model output */

                /* parallel model parameters initialization */

pep->am[0] = -1.294;   /* preset value allowing to define desired performance */
pep->am[1] = 0.630;        /* " */
pep->bm[0] = 0.181;        /* " */
pep->bm[1] = 0.155;        /* " */
}

/*****************************************************************************
ri[], yi[] data shift
reference signal vector : ri[]
plant output vector : yi[]
*****************************************************************************/
datain_shift(piop,pnew_r,pnew_y)
io_param *piop;
float   *pnew_r,*pnew_y;
{
        piop->ri[1] = piop->ri[0];/* r(k-1) = r(k) */
        piop->ri[0] = *pnew_r;/* r(k) = new_r(k) */
        piop->yi[1] = piop->yi[0];
        piop->yi[0] = *pnew_y;
}


/
*****************************************************************************
ui[] data shift
control signal vector : ui[]
*****************************************************************************/
datau_shift(piop,prp)
io_param *piop;
r_param *prp;
{
        piop->ui[1] = piop->ui[0];   /* u(k-1) = u(k) */
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 4 of 11)*

```
        piop->ui[0] += prp->du;      /* u(k) = u(k) + du(k) */
}

/
*****************************************************************************
tab_du[] data shift
control signal variation vector : tab_du[]
*****************************************************************************/
datadu_shift(piop)
io_param *piop;
{
piop->tab_du[1] = piop->tab_du[0];   /* tab_du(k-1) = tab_du(k) */
piop->tab_du[0] = piop->ui[0] - piop->ui[1];/* tab_du(k) = u(k) - u(k-1)*/

}

/
*****************************************************************************
saturation of control signal
*****************************************************************************/
saturation(piop,prp,pnew_u)
io_param *piop;
r_param *prp;
float *pnew_u;
{
        if (piop->ui[0] < prp->u_low)
            piop->ui[0] = prp->u_low;         /* u(k) = u_low */
        else
            if (piop->ui[0] > prp->u_high)
                piop->ui[0] = prp->u_high;  /* u(k) = u_high */
        *pnew_u = piop->ui[0];
}

/
*****************************************************************************
identification by using a recursive method (see appendix : equation A-11)
This algorithm is based on the Bierman's and Thornton's theory. This sub-
routine computes the least squares estimate using the U-D method.
The recursion involves an inversion of matrix F(k) which is
not well-suitable from a numerical point of view. Another way to do the cal-
culations is to use the U-D algorithm. This method is based on a factorization
of F as : F = U.D.UT , where D is diagonal (called diag[] in algorithm)
 and U is upper-triangular matrix (called offdiag[] in algorithm).
This program gives estimates of the parameters of the process :
        y(k) = - theta[0]*y(k-1) - theta[1]*y(k-2) + theta[2]*u(k-1) + the-
ta[3]*u(k-3) + e(k)
which has the same behaviour than the plant.
*****************************************************************************/
ls(piop,pest)
io_param *piop;
estpartyp *pest;
{

float   fj,vj,alphaj,ajlast,pj,w,perr,k[7];
int     kf,ku,i,j;
        perr = piop->yi[0];
for (i=0 ; i<=(pest->n)-1 ; i++)
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 5 of 11)*

```
        perr -= pest->theta[i] * pest->fi[i];  /* calculate prediction error */
pest->eps1 = pest->eps;
pest->eps = perr;
fj = pest->fi[0];
vj = pest->diag[0] * fj;   /* calculate gain and covariance using U-D method */
k[0] = vj;
alphaj = 1.0 + vj * fj;
pest->diag[0] = pest->diag[0]/alphaj/pest->lambda;
if (pest->n > 1)
        {
        kf = 0;
        ku = 0;
        for (j=1 ; j<=(pest->n )-1 ; j++)
                {
                fj = pest->fi[j];
                for (i=0 ; i<=j-1 ; i++)
                        {                       /* f = fi*U */
                        kf = kf + 1;
                        fj = fj + pest->fi[i] * pest->offdiag[kf];
                        }
        vj = fj * pest->diag[j];/* v = D*f */
        k[j] = vj;
        ajlast = alphaj;
        alphaj = ajlast + vj * fj;
        pest->diag[j] = pest->diag[j] * ajlast/alphaj/(pest->lambda);
        pj = -fj/ajlast;
        for (i=0 ; i<=j-1 ; i++)
                {
                ku = ku + 1;
                w = pest->offdiag[ku] + k[i] * pj;
                k[i] = k[i] + pest->offdiag[ku] * vj;
                pest->offdiag[ku] = w;
                }
        }
}
for (i=0 ; i<=(pest->n)-1 ; i++)
        pest->theta[i] += perr * k[i] / alphaj;/* update parameter estimates */
for (i=0 ; i<=(pest->n)-2 ; i++)
        pest->fi[(pest->n)-1-i] = pest->fi[(pest->n)-2-i];/* updating of fi */
pest->fi[0] = -(piop->yi[0]);
pest->fi[(pest->na)] = piop->tab_du[0];
pest->trace = 0.0;
for (i=0 ; i<=(pest->n)-1 ; i++)
        pest->trace += pest->diag[i];  /* computing of the D matrix trace */
if ((abs(pest->eps) - abs(pest->eps1)) < pest->v_conv)
                                        /* test variation of prediction error */
        if (pest->trace < pest->tr_max)   /* test if trace > trace_max */
                pest->lambda = 0.9;
                else pest->lambda = 1.0;
        else if (pest->trace < pest->tr_min)  /* test if trace < trace_min */
                {
                for (i=0 ; i<=(pest->n)-1 ; i++)
                        pest->diag[i] = 1.005 * (pest->diag[i]);
                        pest->lambda = 0.9;
                        }
                        else pest->lambda = 1.0;
}
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 6 of 11)*

```
/************************************************************************
computing of control signal
main program
************************************************************************/
gene_pre_com(piop,pest,pep,prp)
io_param *piop;
estpartyp *pest;
e_param *pep;
r_param *prp;
{
float   f[10],y[5],du_futur[5],g[10][10],gtg[5][5];
int     i,j;

        for (i=0 ; i<=(prp->ny)-1 ; i++)
                  f[i] = 0.0;                   /* initialization */
        for (i=0 ; i<=(prp->nu)-1 ; i++)
                  {
                  y[i] = 0.0;                   /* " */
                  du_futur[i] = 0.0;            /* " */
                  }

        for (i=0 ; i<=(prp->ny)-1 ; i++)
                  for (j=0 ; j<=(prp->ny)-1 ; j++)
                           g[i][j] = 0.0;       /* " */

        for (i=0 ; i<=(prp->nu)-1 ; i++)
                  for (j=0 ; j<=(prp->nu)-1 ; j++)
                           gtg[i][j] = 0.0;     /* " */
        prepare(prp,pest,piop,g,f);  /* initialise f[i] and g[i][j] */
        square(g,gtg,prp);           /* calculate gtg */
        add_lambda(prp,gtg);         /* calculate gtg + lambda2 */
        do_y(g,f,y,prp,pep,piop);/* calculate plant output prediction */
        gauss(gtg,y,du_futur,prp);/* calculate y[i]*[gtg]-1 = du_futur[i] */
        prp->du = du_futur[0];
}

/************************************************************************
calculate f[i] and g[i][j] of the equation :
y[i] = g[i][j] * (y_par[j+2] - f[j])
************************************************************************/
prepare(prp,pest,piop,g,f)
r_param *prp;
estpartyp *pest;
io_param *piop;
float   g[10][10],f[10];
{
int     i,j,nn,nk,n1,j_delay;
float   rj,e[10],fk[4],w[12],par_a[4],par_b[3],ad[4],delta[2];
        n1 = pest->na + 1;
        nk = n1 - 2;
        for (i=0 ; i<=(prp->ny)-1 ; i++)
                  e[i] = 0.0;
        for (i=0 ; i<=(prp->ny)+(pest->na)-1 ; i++)
                  w[i] = 0.0;
        for (i=0 ; i<=(pest->na)+1 ; i++)
                  fk[i] = 0.0;
        for (i=0 ; i<=(prp->ny)-1 ; i++)
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 7 of 11)*

```
        for (j=0 ; j<=(prp->ny)-1 ; j++)
                g[i][j] = 0.0;
for (i=1 ; i<=(pest->na) ; i++)
        par_a[i] = pest->theta[i-1];
par_a[0] = 1.0;
for (i=0 ; i<=(pest->na)-1 ; i++)
        par_b[i] = pest->theta[i+(pest->na)];
delta[0] = 1.0
delta[1] = -1.0;
mul_pol(delta,0,1,par_a,0,pest->na,ad,0,n1);
for (j=0 ; j<=(prp->ny)-1 ; j++)
        {
        if (j=0)
                {
                e[0] = 1.0;
                for (i=0 ; i<=n1-1 ; i++)
                        fk[i] = -ad[i];
                }
        else
                {
                rj = fk[0];
                for (i=0 ; i<=n1-1 ; i++)
                        fk[i] = fk[i+1] - ad[i] * rj;
                e[j-1] = rj;
                }
        mul_pol(e,0,j,par_b,0,(pest->na)-1,w,0,pest->na+j);
        j_delay = j;
        if (j_delay >= 0)
                {
                f[j_delay] = 0.0;
                for (i=0 ; i<=n1-1 ; i++)
                        f[j_delay] += fk[i] * piop->yi[i];
                for (i=1 ; i<=nk ; i++)
                        f[j_delay] += w[j_delay+i] * (piop->tab_du[i]);
                for (i=0 ; i<=j_delay ; i++)
                        g[j_delay][i] = w[j_delay-i];
                }
        }
}
/*************************************************************************
first call ===> ad(q-1) = a(q-1) * (1 - q-1)
second call ===> w(q-1) = e(q-1) * b(q-1)
*************************************************************************/
mul_pol(mat1,l1,h1,mat2,l2,h2,resu,l3,h3)
float   mat1[],mat2[],resu[];
int     l1,h1,l2,h2,l3,h3;
{
int     i,j;
        for (i=l3 ; i<=h3-l3 ; i++)
                resu[i] = 0.0;
        for (i=l1 ; i<=h1 ; i++)
                for (j=l2 ; j<=h2 ; j++)
                        resu[i+j] += mat1[i] * mat2[j];
}
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 8 of 11)*

```
/*************************************************************************
computing of ===> gtg
*************************************************************************/
square(g,gtg,prp)
r_param *prp;
float   g[10][10],gtg[5][5];
{
int     i,k,l;
        for (k=0 ; k<=(prp->nu)-1 ; k++)
                  {
                  for (l=0 ; l<=k ; l++)
                           {
                           gtg[k][l] = 0.0;
                           for (i=0 ; i<=(prp->ny)-1 ; i++)
                                    {
                                    gtg[k][l] += g[i][l] * g[i][k];
                                    if (k!=l)
                                         gtg[l][k] = gtg[k][l];
                                    }
                           }
                  }
}

/*************************************************************************
add lambda2 to gtg
lambda2 is weighting factor of the control signal
*************************************************************************/

add_lambda(prp,gtg)
r_param *prp;
float   gtg[5][5];
{
int     i;
        for (i=0 ; i<=(prp->nu)-1 ; i++)
                  gtg[i][i] += prp->lambda2;
}

/*************************************************************************
computing of the system output prediction
*************************************************************************/
do_y(g,f,y,prp,pep,piop)
r_param *prp;
e_param *pep;
io_param *piop;
float   f[10],y[5],g[10][10];
{
int     i,j;
float   y_par[12];
        for (i=0 ; i<=(prp->ny)+1 ; i++)
                  y_par[i] = 0.0;
        y_par[1] = piop->ymi[0];
        y_par[0] = piop->ymi[1];
        for (i=0 ; i<=(prp->nu)-1 ; i++)
                  {
                  y[i] = 0.0;
                  for (j=0 ; j<=(prp->ny)-1 ; j++)
                           {
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 9 of 11)*

```
                    y_par[j+2] = - pep->am[0] * y_par[j+1] - pep->am[1] * y_par[j]
                      + pep->bm[0] * piop->ri[0] + pep->bm[1] * piop->ri[1];
                    y[i] += g[j][i] * (y_par[j+2] - f[j]);
                    }
          if (i=0)
                    {
                    piop->ymi[1] = piop->ymi[0];
                    piop->ymi[0] = y_par[2];
                    }
          y_par[0] = y_par[1];
          y_par[1] = y_par[2];
          }
}
/*************************************************************************
computing of ===> y[] = du_futur[] * gtg[][]
*************************************************************************/
gauss(gtg,y,du_futur,prp)
float   gtg[5][5],y[5],du_futur[5];
r_param *prp;
{
float   b[5][5],w[5],save1,sum,t,ab,big;
int     i,j,i1,k,l,n;
char    err;
          err = 'f';
          n = prp->nu;
          for (i=0 ; i<=n-1 ; i++)
                    {
                    for (j=0 ; j<=n-1 ; j++)
                              b[i][j] = gtg[i][j];
                    w[i] = y[i];
                    }
          for (i=0 ; i<=n-2 ; i++)
                    {
                    big = abs(b[i][i]);
                    l = i;
                    i1 = i + 1;
                    for (j=i1 ; j<=n-1 ; j++)
                              {
                              ab = abs(b[j][i]);
                              if (ab>big)
                                        {
                                        big = ab;
                                        l = j;
                                        }
                              }
                    if (big=0.0)
                              err = 't';
                    else
                              {
                              if (l!=i)
                                        {
                                        for (j=0 ; j<=n-1 ; j++)
                                             {
                                             save1 = b[l][j];
                                             b[l][j] = b[i][j];
                                             b[i][j] = save1;
                                             }
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 10 of 11)*

```
                                save1 = w[l];
                                w[l] = w[i];
                                w[i] = save1;
                                }
                        for (j=i1 ; j<=n-1 ; j++)
                                {
                                t = b[j][i] / b[i][i];
                                for (k=i1 ; k<=n-1 ; k++)
                                    b[j][k] -= t * b[i][k];
                                w[j] -= t * w[i];
                                }
                        }
                }
        if (b[n-1][n-1]=0.0)
                err = 't';
        else
                {
                du_futur[n-1] = w[n-1] / b[n-1][n-1];
                for (i=n-2 ; i>=0 ; i--)
                        {
                        sum = 0.0;
                        for (j=i+1 ; j<=n-1 ; j++)
                                sum += b[i][j] * du_futur[j];
                        du_futur[i] = (w[i] - sum) / b[i][i];
                        }
                }
}
```

Figure 4-2 *Generalized Predictive Controller* *(sheet 11 of 11)*

```
/*************************************************************************
MOTOROLA GENEVA
*************************************************************************
        Name :              Pascal RENARD
        Type :                        C_Program
        Version :           1.0
        Last Change :       07-June-90
        Function :          Adaptive Controller with Parallel and Serial Models
        Device :            DSP56000/1
        File name :         maincad.c

/*************************************************************************
Global variables
These variables are declared at the global level. They will be accessed from
the different assembly language sections within the program
*************************************************************************/
int Ynew_u__asm("Ynew_u");
int Ynew_y__asm("Ynew_y");
int Ynew_r__asm("Ynew_r");

/*************************************************************************
parallel model parameters
*************************************************************************/
typedef struct {
                  float am[2],bm[2];
                  }para_param;

/*************************************************************************
serial model parameters
*************************************************************************/
typedef struct {
                  float ep1,ep2,ep3,es,esp;
                  }serial_param;

/*************************************************************************
identification parameters
*************************************************************************/
typedef struct {
                  int n,na;
                  float theta[4],fi[4];
                  float p0,lambda;
                  float diag[4],offdiag[7];
                  } ls_param;

/*************************************************************************
controller parameters
*************************************************************************/
typedef struct {
                  float u_low,u_high;
                  } reg_param;
```

Figure 4-3 *Adaptive Controller with Parallel and Serial Models described in Section 3.1 has a sampling period of 410 µs using a 27 MHz DSP56001.* *(sheet 1 of 6)*

```
/*****************************************************************************
input and output signals
*****************************************************************************/
typedef struct {
                  float ui[4], yi[4], ri[4], ymi[4];
                  } io_param;

/*****************************************************************************
pointer initializations
*****************************************************************************/
para_param pap,*ppap = &pap;
serial_param sep,*psep = &sep;
ls_param lsp,*plsp = &lsp;
reg_param rep,*prep = &rep;
io_param iop,*piop = &iop;
/*****************************************************************************
main program
*****************************************************************************/
main ()
{
     __asm("\nYin_y    equ     $ffe0");
     __asm("\nYin_r    equ     $ffc0");
     __asm("\nYout_u   equ     $ffe1");

    initialize(plsp,prep,piop,ppap,psep);/* parameter initialization */
    for (;;)                             /* infinite loop */
      {
    __asm volatile ("movep y:Yin_y,y0" ::: "y0"); /* Obtain system input */
    __asm volatile ("move y0,Y:Ynew_y" ::: "y0");
    __asm volatile ("movep y:Yin_r,y0" ::: "y0"); /* Obtain system reference */
    __asm volatile ("move y0,y:Ynew_r" ::: "y0");

    datain_shift(piop,&Ynew_r,&Ynew_y;   /* (yi[], ri[]) data shift */
    model_par(ppap,piop);                /* parallel model output */
    model_ser(psep,ppap,piop);           /* serial model output */
    identify(piop,plsp,ppap,psep);       /* identification */
    regulate(prep,piop,plsp,ppap,&Ynew_u);/* control signal computing */
    saturate(piop,prep,&Ynew_u);         /* saturation of control signal */
    datals_shift(plsp,piop);             /* fi[] data shift */

                                         /* output of control signal */

    __asm("move y:Ynew_u,y0" ::: "y0");  /* output of control signal */
    __asm("movep y0,y:Yout_u" ::: "y0");

    dataout_shift(piop,&Ynew_u);         /* ui[] data shift */
      }
}

/*****************************************************************************
initialization
*****************************************************************************/
initialize(plsp,prep,piop,ppap,psep)
ls_param *plsp;
reg_param *prep;
io_param *piop;
```

Figure 4-3 *Adaptive Controller* *(sheet 2 of 6)*

```
para_param *ppap;
serial_param *psep;

{
int i;

        plsp->n = 4;       /* identification parameters initialization */
        plsp->na = 2;
        plsp->p0 = 1000.0;
        plsp->lambda = 0.98;

        plsp->theta[0] = 0.02;
        plsp->theta[1] = -0.001;
        plsp->theta[2] = 0.33;
        plsp->theta[3] = 0.305;

        plsp->fi[0] = 1.0;
        plsp->fi[1] = 1.0;
        plsp->fi[2] = 0.5;
        plsp->fi[3] = 0.5;

        for (i=0 ; i<=(plsp->n)-1 ; i++)
                  plsp->diag[i] = plsp->p0;

        prep->u_low = -5.0;      /* controller parameters initialization */
        prep->u_high = 5.0;

        for (i=0 ; i<=(plsp->n)-1 ; i++)/* I/O parameters initialization */
                  piop->ui[i] = -0.5;
        for (i=0 ; i<=(plsp->n)-1 ; i++)
                  piop->yi[i] = -1.0;
        for (i=0 ; i<=(plsp->n)-1 ; i++)
                  piop->ri[i] = -1.0;
        for (i=0 ; i<=(plsp->n)-1 ; i++)
                  piop->ymi[i] = -1.0;

        ppap->am[0] = -1.294;   /* parallel model parameters initialization */
        ppap->am[1] = 0.630;
        ppap->bm[0] = 0.181;
        ppap->bm[1] = 0.155;

}

/*****************************************************************************
ri[], yi[] data shift
*****************************************************************************/
datain_shift(piop,pnew_r,pnew_y)
io_param *piop;
float   *pnew_r,*pnew_y;
{
        piop->ri[1] = piop->ri[0];/* ri_1 = ri */
        piop->ri[0] = *pnew_r;/* ri = new_r */
        piop->yi[1] = piop->yi[0];
        piop->yi[0] = *pnew_y;
}
```

Figure 4-3 *Adaptive Controller* *(sheet 3 of 6)*

```
/*****************************************************************************
parallel model
*****************************************************************************/
model_par(ppap,piop)
para_param *ppap;
io_param *piop;
{
float yk_m;

        yk_m = - ppap->am[0] * piop->ymi[0] - ppap->am[1] * piop->ymi[1]
               + ppap->bm[0] * piop->ri[0] + ppap->bm[1] * piop->ri[1];

        piop->ymi[1] = piop->ymi[0];
        piop->ymi[0] = yk_m;
}


/*****************************************************************************
serial model
*****************************************************************************/
model_ser(psep,ppap,piop)
serial_param *psep;
para_param *ppap;
io_param *piop;
{
        psep->ep3 = psep->ep2;
        psep->ep2 = psep->ep1;
        psep->ep1 = piop->ymi[0] - piop->yi[0];
        psep->esp = psep->es;
        psep->es = psep->ep1 + ppap->am[0] * psep->ep2 + ppap->am[1] * psep->ep3;
}

/*****************************************************************************
identification
*****************************************************************************/
identify(piop,plsp,ppap,psep)
io_param *piop;
ls_param *plsp;
para_param *ppap;
serial_param *psep;
{
float   fj,vj,alphaj,ajlast,pj,w,perr,k[7];
int     kf,ku,i,j;

        perr = psep->es - ppap->bm[0] * piop->ri[0] - ppap->bm[1] * piop->ri[1];
        for (i=0 ; i<=(plsp->n)-1 ; i++)
                perr -= plsp->theta[i] * plsp->fi[i];
        fj = plsp->fi[0];
        vj = plsp->diag[0] * fj;
        k[0] = vj;
        alphaj = 1.0 + vj * fj;
        plsp->diag[0] = plsp->diag[0] / alphaj / plsp->lambda;
        if (plsp->n > 1)
                {
                kf = 0;
                ku = 0;
                for (j=1 ; j<=(plsp->n )-1 ; j++)
```

Figure 4-3 *Adaptive Controller* *(sheet 4 of 6)*

```
                        {
        fj = plsp->fi[j];
        for (i=0 ; i<=j-1 ; i++)
                {
                kf = kf + 1;
                fj = fj + plsp->fi[i] * plsp->offdiag[kf];
                }
        vj = fj * plsp->diag[j];
        k[j] = vj;
        ajlast = alphaj;
        alphaj = ajlast + vj * fj;
        plsp->diag[j] = plsp->diag[j] * ajlast / alphaj / (plsp->lambda);
                pj = -fj/ajlast;
                for (i=0 ; i<=j-1 ; i++)
                        {
                        ku = ku + 1;
                        w = plsp->offdiag[ku] + k[i] * pj;
                        k[i] += plsp->offdiag[ku] * vj;
                        plsp->offdiag[ku] = w;
                        }
                }
        }
                for (i=0 ; i<=(plsp->n)-1 ; i++)
                        plsp->theta[i] += perr * k[i] / alphaj;
}

/
*****************************************************************************
computing of control signal
*****************************************************************************
/
regulate(prep,piop,plsp,ppap,pnew_u)
io_param *piop;
ls_param*plsp;
para_param *ppap;
reg_param *prep;
float *pnew_u;
{
        *pnew_u = (ppap->bm[0]*piop->ri[0] + ppap->bm[1]*piop->ri[1]
                - plsp->theta[0]*piop->yi[0] - plsp->theta[1]*piop->yi[1]
                - plsp->theta[3]*piop->ui[0]) / plsp->theta[2];
}

/
*****************************************************************************
saturation of control signal
*****************************************************************************
/
saturate(piop,prep,pnew_u)
io_param *piop;
reg_param *prep;
float *pnew_u;
{
        if (piop->ui[0] < prep->u_low)
                piop->ui[0] = prep->u_low;
        else
                if (piop->ui[0] > prep->u_high)
```

Figure 4-3 *Adaptive Controller* *(sheet 5 of 6)*

```
                        piop->ui[0] = prep->u_high;
        Ynew_u = piop->ui[0];
}

/*****************************************************************************
fi[] data shift
*****************************************************************************/
datals_shift(plsp,piop)
ls_param *plsp;
io_param *piop;
{
        plsp->fi[1] = plsp->fi[0];
        plsp->fi[0] = - piop->yi[0];
        plsp->fi[3] = plsp->fi[2];
        plsp->fi[2] = - piop->ui[0];
}

/*****************************************************************************
ui[] data shift
*****************************************************************************/
dataout_shift(piop,pnew_u)
io_param *piop;
float *pnew_u;
{
        piop->ui[1] = piop->ui[0];/* ui[] data shift */
        piop->ui[0] = *pnew_u;
}
```

Figure 4-3 *Adaptive Controller* *(sheet 6 of 6)*

SECTION 5

Conclusion

5.1 Advantages of Adaptive Control

"The unique architecture of Motorola DSP56000/ DSP56001 devices enables them to function as both powerful microcontrollers and as fast digital signal processors."

Adaptive control using parallel-serial reference models is an effective method of regulation for industrial systems with rapidly varying parameters (as a function of operating point) or slowly varying parameters as a result of wear, etc. The main advantage of this technique is that the target performance (specifications) for tracking and regulation can be explicitly defined and incorporated into the parallel and serial models respectively. We note also that this type of regulation enables us to obtain quasi-optimal performance and to comply faithfully with imposed specifications, independent of any variations in the process parameters. An industrial system featuring this type of regulation may be expected of offer high efficiency over its entire operating range. The disadvantage of the adaptive control using parallel-serial reference models lies in the fact that the control signal can become unreasonable according to the desired performance in closed-loop.

Despite the fact that generalized predictive control can be used to regulate any physical system, we saw in **Section 4.2** that this system suffers from a major drawback. Thus, the control weighting factor (α) appli-

cable to an industrial application where the system dynamic response varies over the long term will need to be modified from time to time by a person with expert knowledge of automatic control theory. This in itself significantly reduces the performance of this method of adaptive regulation. A further drawback is the fact that the regulation dynamic response cannot be defined beforehand as in the case of adaptive control using parallel-serial reference models. However, one may also expect to achieve energy savings during transitions since the basic principle of generalized predictive control is to generate control signals that minimize the variance of the output signal (y(k)) and the variance of the control signal (u(k)).

Until a few years ago, adaptive control was not very popular with control engineers, first because the processors (computers) were relatively slow, and secondly because implementation was expensive. In addition, the schemes available at the time were not very robust. However, significant progress has been made recently in the theory of physical system identification and the synthesis of adaptive control which, in turn, has increased the stability of the overall method. In addition, microelectronics has made rapid strides so that higher performance tools are now available at lower unit cost than ever before.

5.2 Advantages of DSP56000/DSP56001 Architecture

The unique architecture of Motorola DSP56000/DSP56001 devices enables them to function as both powerful microcontrollers and as fast digital signal processors. The data memory spaces (X, Y) can accommodate parallel implementations of control algorithms including adaptive controllers. Control engineers used to open-loop type regulation systems or to systems using a 3-D lookup table can now use Motorola DSP56000/DSP56001 processors to develop their own control algorithms using either analytical equations or exact models.

Automotive engineering, telecommunications and the television industry use a variety of electronic control systems which will benefit greatly from digital signal processing. The internal design of Motorola DSP56000/DSP56001 processors provides the processing power to solve a wide range of control problems in these industries and many other besides. ■

APPENDIX

The Least Mean-Square Principle

A.1 Equation Formulation

Consider the following parametric model:

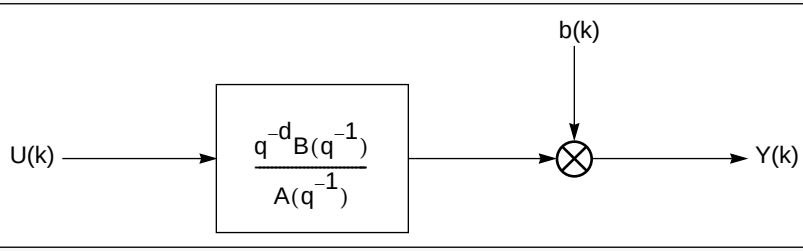

Figure A-1 *Parametric Model Description in Terms of Process Input and Output*

where: $A(q^{-1}) = 1 + a_1.q^{-1} + \ldots + a_n.q^{-n}$

$B(q^{-1}) = b_0 + b_1q^{-1} + \ldots + b_n.q^{-n}$

q^{-1} *is the pure time-delay operator* $(y(k).q^{-1} = y(k-1))$

d is the time delay

b(k) is white noise, including system measurement noise

The order and time delay of the model are assumed to be known. The difference equations can be deduced from the following expression:

$$A(q^{-1}) \cdot y(k) = B(q^{-1}) \cdot U(k) + A(q^{-1}) \cdot b(k)$$

Eqn. A-1

whence:

$$y(k) = -a_1 \cdot y(k-1) - \ldots - a_n \cdot y(k-n) + b_0 \cdot u(k-d) + b_1 \cdot u(k-d-1) + \ldots + b_n \cdot u(k-d-n) + e(k)$$

Eqn. A-2

where: *e(k) is generalized or residual noise*

e(k) = A(q-1).b(k)

Expression Eqn. A-2 can be restated using matrix notation in the form:

$$y(k) = \Psi(k) \cdot \theta(k) + e(k)$$

Eqn. A-3

where: $\Psi(k) = [-y(k-1) \ldots -y(k-n)\ u(k-d) \ldots u(k-d-n)]$

$\theta^T(k) = [a_1 \ldots a_n\ b_0 \ldots b_n]$

$\theta^T(k)$ *is the transpose of vector* $\theta(k)$

Using N input and output measurements, the above expression can be generalized to yield:

$$Y = \Phi \cdot \theta + E$$

Eqn. A-4

where: $Y^T = [y(n+1) \ldots y(N)]$

$q^T = [a_1 \ldots a_n \ b_0 \ldots b_n]$

$E^T = [e(n+1) \ldots e(N)]$

$$\Phi = \begin{bmatrix} -y(n) & \cdots & -y(1) & u(n-d+1) & \cdots & u(1-d) \\ \vdots & & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots \\ -y(N-1) & \cdots & -y(N-n) & u(N-d) & \cdots & u(N-d-n) \end{bmatrix}$$

A.2 Estimation Of Model Parameters

If the previous values of the real system inputs and outputs are known and if the model parameter vector (θ) is assumed to be identical to the actual system parameter vector, then the best possible prediction of the model output is given by:

$$\hat{y}(k) = \Psi(k) \cdot \hat{\theta}(k) \qquad \text{Eqn. A-5}$$

Clearly, this represents the ideal case. The situation is, however, quite different in practice: first, because measurements are contaminated by noise, and second, because the model parameter vector (θ) differs from the actual system parameter vector. This can be expressed as a prediction error or by the error equation:

$$e(k) = y(k) - \hat{y}(k) \qquad \text{Eqn. A-6}$$

The quality of the approximation of an actual system by a model system can be expressed as the variation in the prediction error around its mean value. Let us attempt to minimize this variation, by choosing a criterion defined by:

$$C = \sum_{k=n+1}^{N} e^2(k) = E^T \cdot E \qquad \text{Eqn. A-7}$$

This can be re-expressed in the form:

$$C = (Y - \Phi \cdot \theta)^T \cdot (Y - \Phi \cdot \theta) \qquad \text{Eqn. A-8}$$

This criterion can be minimized as a function of the parameter vector we seek in such a way that:

$$\frac{\delta C}{\delta \hat{\theta}} = [\hat{\theta} - [\Phi^T \cdot \Phi]^{-1} \cdot \Phi^T \cdot Y]\Phi^T \cdot \Phi[\hat{\theta} - [\Phi^T \cdot \Phi]^{-1} \cdot \Phi^T \cdot Y] = 0$$

Eqn. A-9

The solution to the above expression is given by the least-squares estimator. Thus:

$$\hat{\theta} = [\Phi^T \cdot \Phi]^{-1} \cdot \Phi^T \cdot Y \qquad \text{Eqn. A-10}$$

This solution is used in situations where off-line processing is a viable option. Vectors Φ and Y contain all system measurement pairs recorded to that time. One might add that the more measurements there are available, the more accurate the parameter vector (θ) estimates.

An alternative solution is to seek equations to compute sequences of parameter vector ($\theta(\kappa)$) estimates each time new measurement data become available. This method leads to recursive equations that are readily usable in on-line applications. A typical set of such equations is:

$$\hat{\theta} = \hat{\theta}(k-1) + F(k-1) \cdot \Psi(k-1) \cdot \Gamma(k)$$

$$F(k) = \frac{1}{\lambda_1(k)}\left[F(k-1) - \frac{F(k-1) \cdot \Psi(k-1) \cdot \Psi^T(k-1) \cdot F(k-1)}{\frac{\lambda_1(k)}{\lambda_2(k)} + \Psi^T(k-1) \cdot F(k-1) \cdot \Psi(k-1)}\right]$$

$$\Gamma(k) = \frac{y(k) - \hat{\theta}^T(k-1)\Psi(k-1)}{1 + \Psi^T(k-1) \cdot F(k-1) \cdot \Psi(k-1)}$$

Eqn. A-11

where:

- $0 < \lambda_1(k) \leq 1$
- $0 \leq \lambda_2(k) \leq 2$
- $F(0) > 0$
- *$\Psi(k-1)$ is the system measurement vector (i.e. the vector containing measurement data acquired from the system) which is updated at each processing cycle*
- *$\Gamma(k)$ represents the a priori tuning error.*
- *F(k-1), from Eqn. A-11, represents the tuning gain. The structure of this term is quite general.*

Depending on the choice of sequences l1(k) and l2(k), we can obtain a self-tuning algorithm with:

- decreasing gain, for:
  $l_1(k) = l_2(k) = 1$
  (This option is used to identify stationary systems)

- forgetting factor, for:
  $l_1(k) = l(k)$
  $l_2(k) = 1$

- constant trace, for:
  $l_1(k) = C.l_2(k)$ where $C > 0$

(*The last two options for $\lambda_1(k)$ and $\lambda_2(k)$ are used to identify systems that vary slowly in time.*)

- constant gain, for:
  $\lambda_1(k) = 1$
  $\lambda_2(k) = 0$

A.3 The Least-Squares Estimator

In order to simplify the notation in the following, we use the least-squares estimator (LSE) given by expression Eqn. A-10 and suitable for off-line applications.

A.3.1 Is the LSE biased?

Recall that an estimator is said to be statistically unbiased if:

$$E_M[\hat{\theta}] = 0 \qquad \text{Eqn. A-12}$$

where: $E_M[x]$ *is the expected value of* x

This means that the mean value of the parameter vector estimate converges to the true value of the system parameter vector. For the LSE, we have:

$$E_M[\hat{\theta}] = E_M[[\Phi^T \cdot \Phi]^{-1} \cdot \Phi^T \cdot Y] = \theta + E_M[[\Phi^T \cdot \Phi]^{-1} \cdot \Phi^T \cdot E]$$

Eqn. A-13

From this we conclude that the LSE is biased, i.e. it introduces an error into the estimate of vector θ.

A.3.2 How accurate is the LSE?

The quality of the estimates produced by an estimator is determined by computing the variance of vector θ around its mean value, i.e. by computing the variance-covariance matrix represented by $C_{\theta\theta}$. The accuracy of the LSE is given by matrix elements $C_{\theta i \theta i}$. The matrix is given by:

$$C_{\theta\theta} = E_M[[\hat{\theta} - \theta][\hat{\theta} - \theta]^T]$$

Eqn. A-14

$$= E_M[[\Phi^T \cdot \Phi]^{-1} \cdot \Phi^T \cdot E \cdot E^T \cdot \Phi \cdot [\Phi^T \cdot \Phi]^{-1}]$$

A.3.3 Conclusion

(concerning the bias and accuracy of the LSE)

An ideal estimator is characterized by zero bias and minimum variance. Examination of expressions Eqn. A-14 and Eqn. A-13 reveals that the LSE suffers from an unavoidable bias while the variance-covariance matrix $C_{\theta\theta}$ is certainly not minimum.

A.4 Improving The “LSE”

A.4.1 What is required to minimize LSE bias?

The answer to this question is obtained by considering the case where the generalized noise e(k) has the following properties:

$$E_M[\Phi, E] = 0$$
$$E_M[E] = 0 \qquad \text{Eqn. A-15}$$

This amounts to saying that there must be no correlation between e(k) and Ψ(k) and that e(k) must have a symmetrical distribution.

Applying these properties to expression Eqn. A-13, we indeed come back to expression Eqn. A-12. The mean value of the estimated vector is thus identical, under these conditions, to that of the parameter vector we seek.

A.4.2 What is required to minimize LSE variance?

LSE variance can be minimized if the following property is met:

$$E_M[E^T \cdot E] = \alpha^2 \quad \text{(variance)}$$
$$E_M[E^T \cdot E] = 0 \quad \text{(covariance)} \qquad \text{Eqn. A-16}$$

This amounts to saying that e(k) represents white noise.

Applying this property to expression Eqn. A-14, the LSE accuracy is given by:

$$C_{\theta\theta} = \alpha^2[\Phi^T \cdot \Phi]^{-1} \qquad \text{Eqn. A-17}$$

A.5 Conclusion

Unfortunately, the properties expressed by relations Eqn. A-15 and Eqn. A-16— which should be properties of the generalized noise e(k)—are not met by the least-squares estimator. The very structure of the parametric model, illustrated in Figure A-1, makes it impossible to obtain white noise for e(k) since we set out with the equality:

$$e(k) = A(q^{-1}) \cdot b(k) \qquad \text{Eqn. A-18}$$

The non-correlation property expressed by relation Eqn. A-15 cannot be met because e(k) is correlated with y(k) through the dynamic response of the model, A(q-1).

In summary, the simple LSE described above introduces systematic errors into the estimated parameters. In order to improve the quality of model parameter estimates, other estimators must be sought. The generalized least-squares estimator, for instance, is based on the following principle:

If e(k) is white noise and uncorrelated with Ψ(k), then the above-mentioned properties can be met by adding a filter to the generalized noise. This implies that consideration must be given to changes in the structure of the parametric model.

Principles on which other estimators can be based will not be discussed in this application note. Readers interested in the methods of system identification will find the answers in the relevant journals. Three references of special interest are: [FOU-86], [SAM-83], and [LJU-83]. ■

INDEX

—A—

ABS brakes 7
Active Suspensions 7
Adaptive 3

—G—

Generalized predictive control 5

—N—

Non-parametric models 10

—P—

Parametric model 12
Performance of adaptive controls 14
Polynomial parametric model 13

REFERENCES

[DSP-89]: DSP56000UM/AD
DSP56000/DSP56001 Digital Signal Processor User's Manuel, Motorola, 1989.

[KUO-87]: B. Kuo
Automatic Control Systems
Englewood Cliffs, NJ : Prentice-Hall, 1987.

[REN-88]: P. Renard
Etude et mise au point de modèles d'aide à la conception de régulateurs et application aux moteurs diesel.
Thèse de l'Université de Haute Alsace, Mul house (FR), 1988.

[IRV-86]: E. Irving
Commande adaptative
Cours "Ecole Supérieure d'Electronique" (FR), 1986.

[SAM-83]: C. Samson
Problèmes en identification et commande de systèmes dynamiques.
Thèse d'état, Rennes (FR), 1983.

[IRV-83]: E. Irving
Commande adaptative
Cours "Ecole Supérieure d'Electronique" (FR), 1983.

[LAN-84]: I.D. Landau
Colloque sur la commande adaptative.
Aspects pratiques et théoriques.
E.N.S.I.E.G. (FR), 1984.

[DAH-82]: B. Dahhou, K. Najim, M. M'saad
Commande adaptative d'un four industriel de séchage de phosphate basée sur les systèmes adaptatifs avec modèle de référence.
RAIRO, vol. 16, n° 3, 1982.

[CLA-84]: D.W. Clarke, P.S. Tuffs, C. Mohtadi
Self-tuning control of a difficult process.
Colloque sur la commande adaptative.
Aspects pratiques et théoriques.
E.N.S.I.E.G. (FR), 1984.

[CLA-85]: D.W. Clarke
Implementation of self-tuning controllers.
IEE Control engineering series, vol. 15,
pp. 144-165, 1985.

[LJU-83]: L.Ljung, T.Soderstrom
Theory and practice of recursive identifcation.
Cambridge, M.I.T., 1983.

[BOD-87]: J.C. Bodart
Modélisation et commande d'une machine à thermofixer en continu des fibres synthétiques.
Thèse 3ème cycle, Mulhouse (FR), 1987.

[LIM-89]: K.W. Lim and K.V. Ling
Generalized predictive control of a heat exchanger.
IEEE, Control Systems Magazine, Vol. 9, Num. 6, Oct. 1989.

[AST-84]: K.J. Astrom, B. Wittenmark
Computer controlled systems.
Prentice hall, Englwood Cliffs, 1984.

[FOU-86]: C. Foulard, S. Gentil, J.P. Sandraz
Commande et régulation pour calculateur numérique.
Eyrolles, 1986. ■