A Comparative Study of Delta- and Z-Domain Based Output Feedback Pole-Placement Adaptive Controllers

M. Flynn[*] and J.V. Ringwood[**]

[*]Amdahl Ireland Ltd, Balheary, Swords, Co. Dublin, Ireland
[**]School of Electronic Engineering, Dublin City University, Glasnevin, Dublin 9, Ireland

ABSTRACT

This paper selects a popular z-domain adaptive controller based on output feedback pole placement, utilising recursive least squares for system identification. The adaptive algorithm is recast in the delta-domain with a corresponding reformulation of the identification algorithm. Simulation results, incorporating finite word-length effects, are convincingly used to demonstrate the improvement in the delta formulation, when smaller word-lengths are used. Small word lengths are a feature of most popular industrial controllers. In addition, a pilot-scale rig is used as an application example to demonstrate the effectiveness of the delta controller in real-life implementation. The paper concludes by comparing and contrasting the z- and delta-controllers not only in performance terms but also in terms of design complexity, intuitive appeal and cost benefits.

1. INTRODUCTION

The primary purpose of this paper is to demonstrate the effectiveness of designing and implementing digital adaptive controllers in the delta-domain as opposed to the z-domain. This alternative transform domain was popularised in 1985 by Middleton and Goodwin [1]. Instead of representing signal values in absolute terms, the concentration is instead focussed on *differences* between the signal values. Such an approach is common in related areas, such as time series modelling [2], where a differencing transformation is applied to the data prior to modelling. However, as yet, the delta transformation has not gained wide acceptance in the control community.

The δ-operator provides a number of advantages for the design, analysis and implementation of control systems. The principal advantage is that the delta operator formulation gives better coefficient representation and less round-off noise when compared with the usual z-domain formulation. Such an effect becomes more exaggerated as the word length of the controller implementation is decreased. An added advantage of the delta operator formulation is that the discrete δ-operator model coefficients converge on their continuous-time (s-domain) counterparts, drawing a strong connection between models generated on different sampling periods and creating a unifying framework for the examination of both continuous-time and discrete-time systems. Such an equivalence is not available for z-domain models generated with different sampling periods.

In addition to [1], the numerical properties of the δ-operator have been studied by Salgado *et al* [3] in a digital filtering context and by Terrett and Downing [4] in a system identification application. Both report significant improvements in a δ-operator formulation. Other encouraging results have been provided in areas such as z to δ transformations [5] and sensitivity properties of δ-based pole assignment control system designs [6].

This paper attempts to examine the potential benefit of utilising a δ-operator formulation in the implementation of a self-tuning regulator (STR). The explicit STR proposed incorporates recursive least squares (RLS) identification of the process parameters with a pole/zero placement output feedback control system design. Self-tuning controllers have been shown to have considerable performance benefits over conventional (fixed) controllers in a variety of applications, where process parameters are either unknown or vary with time.

The paper is organised as follows. Section 2 describes the δ-operator and some of its properties. The self-tuning controller is described next for both z and δ, the controller design given Section 3 and the identification algorithm in Section 4. Section 5 looks at how finite wordlength effects arise and how they might be simulated with simulation and implementation results given in Sections 6 and 7 respectively. Finally, conclusions are drawn in Section 8.

2. THE DELTA OPERATOR

The z operator is widely used to describe discrete trnasfer functions. However, a disadvantage of the z operator is that it is not at all like its continuous counterpart, d/dt (or s in Laplace form). It is therefore logical to assume that a better correspondence between continuous and discrete systems would result if a difference (corresponding to a numerical derivative) operator was used to represent discrete time signals instead of the usual z operator. To fulfill this role, the δ-operator is defined as:

$$\delta = \frac{z-1}{T_s} \tag{1}$$

where T_s is the sampling period. Note that the relationship between z and δ is a linear one, allowing for easy system transformations from z to δ and vice versa. Although δ models are more like models in s, z models are generally simpler functions and describe the sequential nature of discrete-time signals. However, because of the linear transformation in (1), equivalent transfer functions in z and δ are always of the same order.

2.1 Properties of the δ Operator

For the forthcoming analysis, some properties of the δ operator need to be documented. One important aspect is the region of stability in the δ domain. Recalling that the stability regions in the s and z planes are given by the left half plane (LHP) and unit disk respectively, it may be noted that these stability regions are invariant with sampling period. In the δ domain, however, the region of stability is given by a circle of centre $(-1/T_s)$ and radius $1/T_s$. Thus the stability region expands as the sampling period descreses and converges on the stability region in the s domain as $T_s \to 0$.

One other property worth documenting is the final value theorem. This is required when evaluating the steady-state gain of systems or for selecting reference models with a unity d.c. tracking property. The final value theorem for the δ operator is given as:

$$\lim_{t\to\infty} f(t) = \lim_{\delta\to\infty} \{\delta F(\delta)\} \tag{2}$$

2.2 δ Transfer Function Realisation

In order to implement a δ transfer function, it must first realised in state space form and then iterated in a simulation or control loop. For a generalised proper transfer function of the form:

$$H(\delta) = \frac{b_m\delta^m + b_{m-1}\delta^{m-1} + \ldots + b_0}{\delta^n + a_{n-1}\delta^{n-1} + \ldots + a_0} = \frac{y_t}{u_t} \tag{3}$$

where m≤n, the following controller canonical form state space realisation may be formed:

$$\begin{aligned} x_t &= \delta^{-1}\{Ax_t + Bu_t\} \\ y_t &= Cx_t + Du_t \end{aligned} \tag{4}$$

where

$$A = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \cdots & -a_1 & -a_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & \cdots & 0 & b_m & b_{m-1} & \cdots & b_1 & b_0 \end{bmatrix} \tag{5}$$

In th above realisation, δ^{-1} is used as the basic building block, acting as an accumulator, with a behaviour similar to an integrator. To simulate the model, the following equations are iterated:

$$\begin{aligned} y_{t+1} &= Cx_t + Du_t \\ x_{t+1} &= x_t + T_s(Ax_t + Bu_t) \end{aligned} \tag{6}$$

3. POLE PLACEMENT CONTROLLER DESIGN

In this section, the calculations required to evaluate a controller which places the poles and zeros of the overall closed-loop system at desired locations are given. It is assumed that a plant model in rational polynomial form is available; the determination of such

a model using system identification techniques will be considered in the following section. The controller has two degrees of freedom and has a structure as shown in Fig.1.

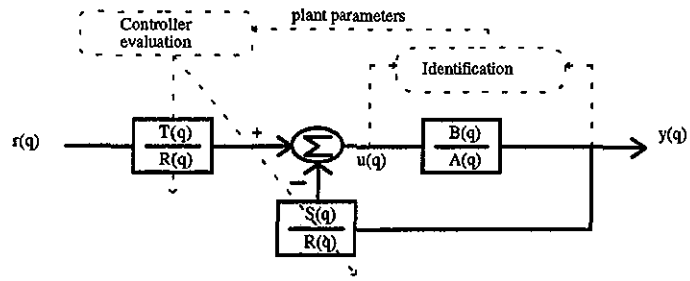


Fig.1: Pole Placement Self-Tuning Regulator

The controller claculation problem may be stated as: Given a plant description B(q)/A(q), determine controller polynomials T(q), R(q) and S(q) such that an overall closed loop transfer function of $B_m(q)/A_m(q)$ is achieved. In the adaptive or self-tuning controller, these calculations are performed on-line. The desired closed loop transfer function, $B_m(q)/A_m(q)$, may be arbitraraily specified, subject to certain constraints.

The derivation of the output feedback pole placement controller is given in a number of places in the literature. A basic development, for both z and δ is given in [1]. However, this development does not cover the case of plants with non-minimum phase zeros. A superior development is given in [7], but only for the z domain. This development is adopted for the current investigation, with modifications made to allow its use with the δ domain. The following development is therefore presented in terms of a generalised operator, q, to cover both z and δ cases, with appropriate notes given in areas of conflict between the two.

3.1 Choice of Closed Loop Transfer Function

The closed loop transfer function:

$$H_m(q) = \frac{B_m(q)}{A_m(q)} \tag{7}$$

must be chosen so that:

$$\begin{gathered} \deg A_m - \deg B_m \geq \deg A - \deg B \\ A_m \text{ and } B_m \text{ relatively prime} \end{gathered} \tag{8}$$

Apart from the conditions in (8), B_m and A_m may be chosen to give a desired transient response, with unity d.c. gain if required (as in (2)), with correspondingly larger control signals for faster responses.

3.2 Pole Zero Cancellations

From Fig.1, the CLTF may be evaluated as:

$$H_m(q) = \frac{BT}{AR + BS} = \frac{B_m}{A_m} \tag{9}$$

Care must be taken that non-minimum phase zeros (zeros of B(q)) are not cancelled, since this will result in the introduction of unstable controller poles. To ensure this condition, B(q) is factorised as:

$$B(q) = B^+(q)\,B^-(q) \tag{10}$$

where $B^+(q)$ is monic and contains only zeros within the region of stability, and
$B^-(q)$ contains only zeros outside the region of stability

Since the regions of stability for δ and z are different, the claculations for δ and z differ in this respect. The $B^+(q)$ zeros are cancelled, which implies that R(q) must contain $B^+(q)$ as a factor. Thus,

$$R(q) = B^+(q)\,R^*(q) \tag{10}$$

and

$$\frac{T}{AR^* + B^-S} = \frac{B_m^*}{A_m} \tag{11}$$

with

$$B_m(q) = B^-(q)\,B_m^*(q) \tag{12}$$

3.3 Observer Polynomial

For realisability, an observer polynomial is included in equation (11), which is cancelled in the closed loop system:

$$\frac{T}{AR^* + B^-S} = \frac{B_m^* A_0}{A_m A_0} \tag{13}$$

A_0 is chosen as a stable polynomial, and a popular choice is for deadbeat response. This will result in different polynomials in δ and z. A solution for the controller polynomials, T, R and S may now be obtained using the numerator and denominator components of equation (13) as:

$$T(q) = B_m^*(q)\, A_0(q) \tag{14}$$

$$A(q)\, R^*(q) + B^-(q)\, S(q) = A_0(q)\, A_m(q) \tag{15}$$

Equation (15) is a Diophantine equation, the solution of which is considered in Section 3.5.

3.4 Degree of Controller Solution

The degrees of the controller and observer polynomials [7] are chosen subject to the following equations:

$$\deg S = \deg A - 1 \tag{16}$$

$$\deg A_0 \geq 2\deg A - \deg A_m - \deg B^+ - 1 \tag{17}$$

$$\deg R^* = \deg A_0 + \deg A_m - \deg A \tag{18}$$

$$\deg T = \deg B_m - \deg B^- + \deg A_0 \tag{19}$$

3.5 Diophantine Equation Solution

The polynomial Diophantine equation in (15) may be solved using the following matrix formulation [8]:

$$\begin{bmatrix} a_0 & 0 & \cdots & 0 & b_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \cdots & 0 & b_1 & b_0 & \cdots & 0 \\ \vdots & a_1 & \ddots & 0 & \vdots & b_1 & \ddots & 0 \\ a_n & \vdots & \ddots & a_0 & b_m & \vdots & \ddots & b_0\ 0 \\ 0 & a_n & \vdots & a_1 & 0 & b_m & \vdots & b_1 \\ 0 & 0 & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & a_n & 0 & 0 & 0 & b_m \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_t \\ s_0 \\ \vdots \\ s_p \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ . \\ . \\ . \\ c_j \end{bmatrix} \tag{20}$$

4. RECURSIVE LEAST SQUARES IDENTIFICATION

The A(q) and B(q) plant model denominator and numerator polynomials in Section 3 are provided using a recursive identification technique. Recursive least squares is a straightforward and effective identification technique which may be formulated for both z and δ models. The cost function used is:

$$V(\theta, N) = \tfrac{1}{2}\sum_{i=1}^{N} \lambda^{N-i}(y_i - \Phi_i^T\, \theta) \tag{21}$$

where y is the actual plant output and Φ and θ the regressor and parameter vectors respectively. The effect of the forgetting factor, λ, is to weight more recent data with a progressively heavier weight, allowing improved operation where system parameters change with time. The system description is of the from:

$$y_t = \Phi_t^T\, \theta \tag{22}$$

with the parameter vector, θ, for a δ or z transfer function of the form in (3) given by:

$$\theta^T = [-a_{n-1}, \ldots\ldots, -a_0, b_m, \ldots\ldots, b_0] \tag{23}$$

4.1 Z-Domain Algorithm

The z domain algorithm is well understood and can be implemented using the following three equations [9]:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t)-\Phi^T(t)\hat{\theta}(t-1)) \quad (24)$$

$$K(t) = P(t)\Phi(t) = P(t-1)\Phi(t)(\lambda I+\Phi^T(t)P(t-1)\Phi(t))^{-1} \quad (25)$$

$$P(t) = (I-K(t)\Phi^T(t))P(t-1)/\lambda \quad (26)$$

where P is the covariance matrix and K the effective adaptation gain.

The regressor vector, Φ, for the z domain case is:

$$\Phi_{t+1}^T = [y_t, \ldots\ldots, y_{t-n+1}, u_{t-n+m+1}, \ldots\ldots, u_{t-n+1}] \quad (27)$$

4.2 δ-Domain Algorithm

The δ-domain formulation for RLS [1], [4] is not dissimilar from the z-domain formulation, but differs in a number of respects. First of all, the regressor vector is composed as:

$$\Phi_t^T = [\delta^{n-1}y_t, \ldots\ldots, y_t, \delta^m u_t, \ldots\ldots, u_t] \quad (28)$$

If the differences $\delta^{n-1}y_t$, $\delta^{n-2}y_t$, ... are available, then the δ domain RLS can be implemented directly. However this is not usually the case, since plant outputs and inputs are normally measured in absolute terms. If only absolute measurements are available, then these differences need to be generated. Unfortunately, these differences may not be evaluated directly, for a number of reasons:

- the difference (or differential) operator is very susceptible to noise, and
- there is an inherent accumulator effect when the model is formulated in negative powers of δ, leading to poor numerical performance.

To counteract these effects, a stable operator, F(δ) is introduced, which effectively filters the differences and places an upper bound on any accumulations. In accordance with this strategy, equation (3) is recast as:

$$\frac{A(\delta)}{F(\delta)}y_t = \frac{B(\delta)}{F(\delta)}u_t \quad (29)$$

where:

$$F(\delta) = \delta^n + f_{t-1}\delta^{n-1} + \ldots\ldots + f_0$$

The regressor model in (22) is now replaced by:

$$s_t = [-a_{n-1} \quad -a_{n-2} \quad \cdots \quad -a_0 \quad b_m \quad b_{m-1} \quad \cdots \quad b_0]\Phi_t \quad , \quad \Phi_t = \begin{bmatrix}\phi_t \\ \varphi_t\end{bmatrix} \quad (30)$$

where:

$$s_t = \frac{\delta^n}{F(\delta)}y_t \quad (31)$$

$$\phi_t = \left[\frac{\delta^{n-1}}{F(\delta)}y_t \quad \frac{\delta^{n-2}}{F(\delta)}y_t \quad \cdots \quad \frac{1}{F(\delta)}y_t\right] \quad (32)$$

$$\varphi_t = \left[\frac{\delta^{n-1}}{F(\delta)}u_t \quad \frac{\delta^{n-2}}{F(\delta)}u_t \quad \cdots \quad \frac{1}{F(\delta)}u_t\right] \quad (33)$$

For computer implementation, (31) may be reconfigured to give:

$$s_t = y_t - [f_{t-1} \quad f_{t-2} \quad \cdots \quad f_0]\phi_t \quad (34)$$

and using (1) and (34), the regressor, ϕ, may be updated using:

$$\phi_{t+1} = \phi_t + \begin{bmatrix} -f_{t-1} & -f_{t-2} & \cdots & f_1 & f_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}\phi_t + T_s\begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix}y_t \quad (35)$$

Similarly

$$\varphi_{t+1} = \varphi_t + \begin{bmatrix} -f_{l-1} & -f_{l-2} & \cdots & f_1 & f_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \varphi_t + T_s \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u_t \qquad (36)$$

The δ-domain RLS algorithm is as in (24) to (26), with y_t replaced by s_t.

5. FINITE WORD LENGTH EFFECTS

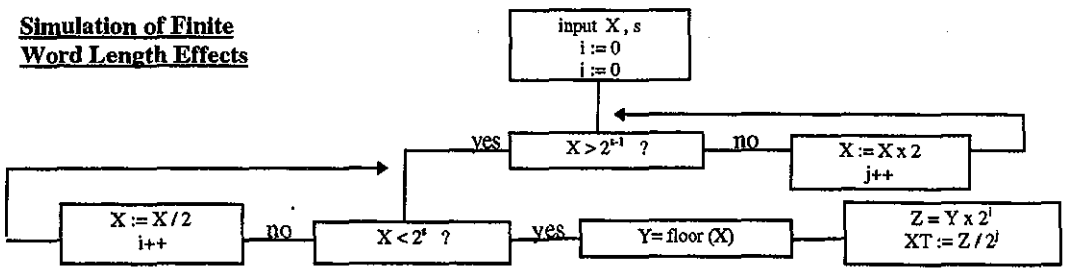
The primary purpose for utilising a δ operator formulation is to produce an algorithm which is less sensitive to finite word length effects. Such effects arise due to the limited precision with which computations are performed. They may be classified in three ways:

- A/D quantisation errors,
- Parameter storage errors, and
- Multiplication errors

A/D errors are unavoidable, but may be minimised with higher precision devices, subject to noise limitations. No improvement in performance, with respect to this error type, is likely from the δ algorithm, since plant input and output signals are measured in absolute terms. Given a particular word length, controller parameter storage errors, which can cause offsets in the controller poles and zeros, are likely to be similar for z and δ, but because of the difference in the transfer function representations between z and δ, some benefits are likely to accrue for the δ representation. The main benefit of the δ representation is in the reduction in the effective magnitude of multiplication errors. Because signals are represented in *difference* form, more significant bits are retained due to the reduction in number magnitudes.

z and δ forms for a floating point representation with a finite number of mantissa bits will be examined. This representation is simulated in C using the following algorithm:

Fig.2: **Simulation of Finite Word Length Effects**



X is the number to be truncated, s is the required wordlength in number of binary bits and XT the truncated value. Each parameter, variable and intermeduate variable is subjected to the above truncation in the simulation.

6. SIMULATION RESULTS

Both initial tuning and adaptation performance of both z and δ algorithms were examined by specifying a process model which changed over the course of the simulation. The process models in the different forms are:

Model 1: $$G_1(z) = \frac{0.75z - 0.1409}{z^2 - 1.2769z + 0.4066}, \qquad G_1(\delta) = \frac{0.75\delta + 0.9}{\delta^2 + 0.73\delta + 0.1297}$$

Model 2: $$G_2(z) = \frac{0.6z - 0.1}{z^2 - 1.3z + 0.4}, \qquad G_2(\delta) = \frac{0.8\delta + 1.0}{\delta^2 + 0.8\delta + 0.05}$$

The reference model was chosen to have unity dc gain as:

$$G_{ref}(z) = \frac{0.1z}{z^2 - 1.3z + 0.4}, \qquad G_{ref}(\delta) = \frac{0.1\delta + 0.1}{\delta^2 + 0.7\delta + 0.1}$$

The sampling period for the simulation is 1 sec., the forgetting factor set to λ=0.995 and the covariance matrix initialised as P = 1000 I. Initially, the number of significant bits was set to 32 (the machine limit) and, as expected, both algorithms performed

satisfactorily. As the number of bits approached 12, the performance differences became apparant. Figs.3 and 4 show the regulation properties and parameter convergence for the z-based algorithm, while the superior performance of the δ-based algorithm is shown in Figs.5 and 6. The vertical axis on Figs. 3 and 4 has been bounded, whereas no bounding has been applied to the δ results.

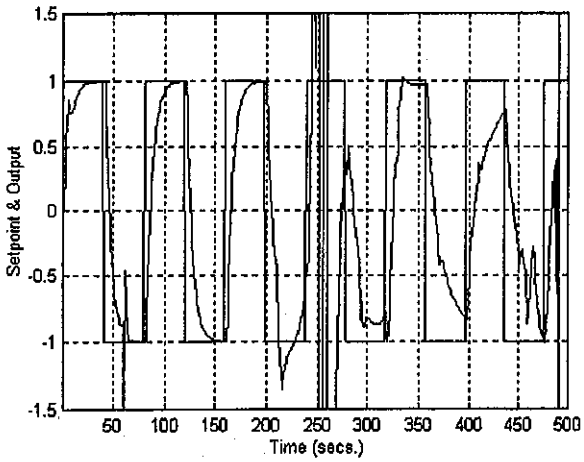


Fig.3 Regulation properties of z controller

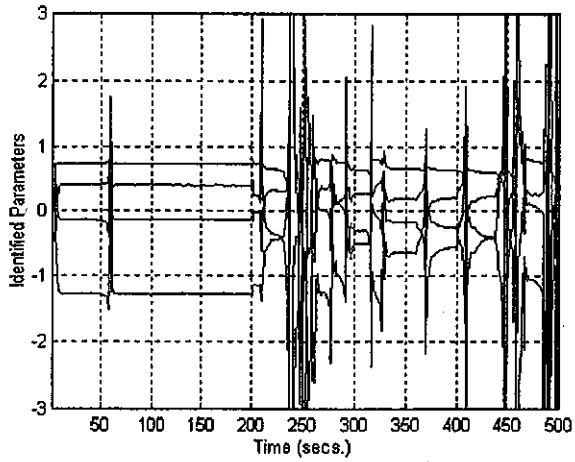


Fig.4 Parameter estimates in z-domain

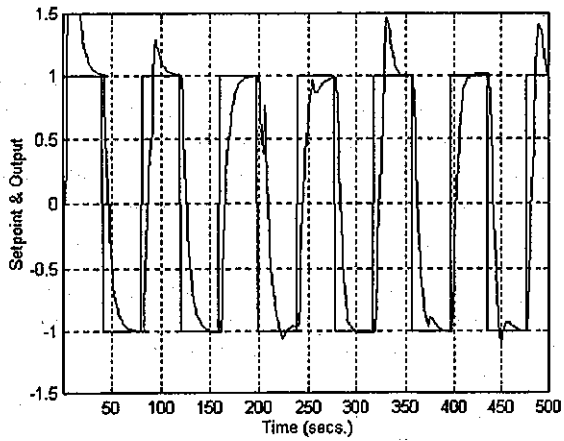


Fig.5 Regulation properties of δ controller

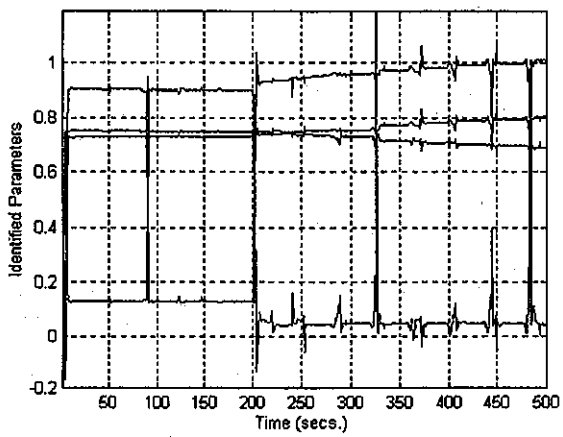


Fig.6 Parameter estimates in δ-domain

Below 11-bit resolution, the z-based algorithm goes unstable, due to poor parameter estimates, particularly during readaptation after the process model has changed. The δ-domain algorithm provides reasonable control down to about 9 bits resolution, after which it fails to achieve reasonable reconvergence of the process parameters. It should be borne in mind that no extra excitation signal eas added to assist identification in closed loop.

7. CONTROLLER IMPLEMENTATION

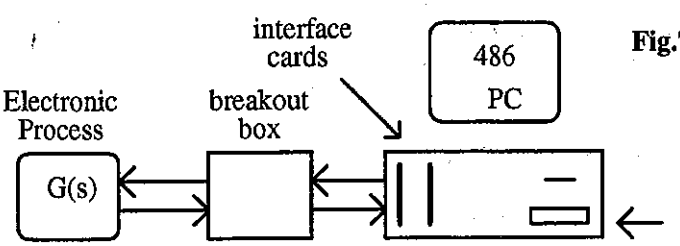


Fig.7: Hardware Set-up

The hardware set-up is as shown in Fig.7. A pilot-scale electronic process was connected to a PC running the adaptive pole-placement software via 8-bit analogue interface cards. Unfortunately, due to the considerable amount of looping in the truncation algorithm (see Fig.2), which is applied to each operation result, it was not possible to implement the reduced precision algorithm in real time with a suitable sampling period. A full 32-bit precision experiment was performed, however, with the process dynamics changed after 250 seconds, in order to examine the

adaptation qualities of z and δ algorithms. Figs. 8 and 9 show the variations in the prediction error returned from the z and δ identification algorithms respectively for λ=0.99. Although, no great difference is apparent between graphs, the mean absolute prediction error is about 15% better for δ over z. This improvement is consistent over a broad range of forgetting factors. The percieved differences between z and δ controllers are likely to be relatively small because of the inherent precision limitations in using 8-bit A/D and D/A.

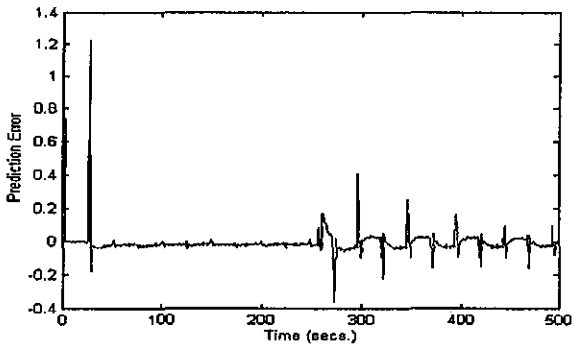


Fig.8: Prediction error for z-domain

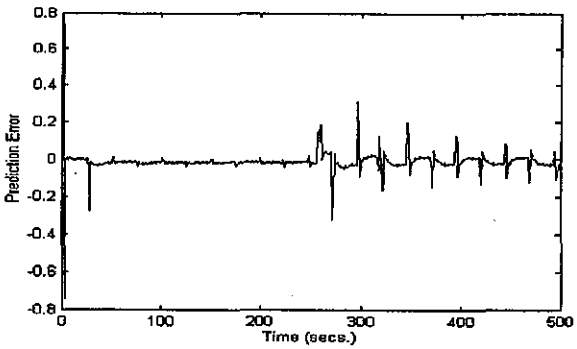


Fig.8: Prediction error for δ-domain

8. CONCLUSIONS

This paper demonstrates that the δ formulation has superior properties in a self-tuning pole-placement application. The performance of the adaptive controller is dominated by the ability of the identification to produce quality parameter estimates, especially following a change in the plant parameters. This masks any differences which may exist in reduced precision implementations of the z and δ pole-placement algorithms, but separate tests conducted by the authors confirm the superiority of the δ formulation. In those tests, the δ controller retained good fidelity down to 6 bits, whereas the performance of the z controller degraded dramatically below10 bits.

Differences between z and δ algorithms for reduced precision *floating-point* implementations were considered in this paper. It is likely that any differences between the algorithms seen here will be exaggerated in a reduced precision *fixed-point* implementation. However, since the numerical precision of most controllers implemented on fixed point devices is enhanced through appropriate variable scaling, the current analysis is probably more realistic.

Finally some comment on the choice of the polynomial F(δ) in equation (29) is pertinent. Apart from being a mathematical requirement for properness, the addition of F(δ) can be regarded as providing a filtering action on input and output prior to differencing. Since the effective filter transfer function is 1/F(δ), which is required to be strictly proper, only low pass forms are possible. This means that any high frequency output disturbances will be attenuated, which may help to avoid biased estimates. Low frequency disturbances, however, must be removed by additional external filtering.

REFERENCES

1. Middleton, R.H. and Goodwin, G.C. *Digital Control and Estimation: A Unified Approach*, Prentice-Hall, 1990.
2. Box, G.E. and Jenkins, G.M. *Time Series Analysis: Forecasting and Control*, Holden-Day, 1976.
3. Salgado, M., Middleton, R.H. and Goodwin, G.C. *Connections Between Continuous Discrete Ricatti Equations With Applications to Kalman Filtering*, IEE Proc. D, Vol.135, No.1, Jan 1988.
4. Terrett, P. and Downing, C.J. *System Identification and Modelling on the TMS320C25 Fixed-Point Processor Using the δ Operator*, Proc Irish Colloquium on DSP and Control, Dublin, June 1993.
5. Neuman, C.P. *Transformations Between Delta and Forward Shift Operator Transfer Function Models*, IEEE Trans. on Syst. , Man and Cyber., Vol.23, No.1, Jan-Feb. 1993.
6. Middleton, R.H. and Goodwin, G.C. *Improved Finite Word Length Characteristics in Digital Control Using Delta Operators*, IEEE Trans. on Auto. Control, Vol.AC-31, Nov. 1986.
7. Astrom, K.J. and Wittenmark, B. *Computer Controlled Systems - Theory and Design*, Prentice-Hall, 1984.
8. Feinstein, J. and Barr-Ness, Y. *The solution of the matrix polynomial equation A(s)X(s)+B(s)Y(s)=C(s)*, IEEE Trans. on Auto. Control, Vol.AC-29, No.1, Jan. 1984.
9. Astrom, K.J. and Wittenmark, B. *Adaptive Control*, Addison-Wesley, 1989.