IMPLICIT STATE-SPACE REPRESENTATION : A UNIFYING FRAMEWORK FOR FWL IMPLEMENTATION OF LTI SYSTEMS

T. Hilaire [*,***] **P. Chevrel** [*,**] **Y. Trinquet** [*]

[*] *IRCCyN, UMR CNRS 6597, 1 rue de la Noë 44321 NANTES, FRANCE*
[**] *EMN, 4 rue Alfred Kaster, La Chantrerie 44307 NANTES, FRANCE*
[***] *PSA Peugeot Citroën, 18 rue des Fauvelles 92256 La Garenne, FRANCE*

Abstract: Practically, some intermediary realizations are used in order to simulate, numerically, dynamic systems. One of the most popular is the state-space realization. It reveals to be very useful to study the impact of Finite Word Length implementation, especially in the case of embedded controller. Numerous works concerned the design of the "best" realization concerning parameterisation, numerical noise minimisation or saving computation. This paper points out however that a standard state-space realization is too basic to take into account some interesting realizations. On the contrary, it highlights that implicit state-space realizations allows a more direct link with the macroscopic computations to be performed. It is necessary to describe some popular algorithms simulating LTI systems. Moreover, such a representation has the important property to unify different ways of research considering differently the possibilities offered by using the shift, δ or γ operators.

Keywords: parametrization, implicit systems, state-space realization, implementation, control algorithms, digital control

1. INTRODUCTION

Finite Word Length (FWL) implementation leads to deterioration of dynamic systems used for filtering or control. The degradation introduced (quantification of the coefficients, roundoff errors in numerical computations) may be formalised as parametric errors and numerical noises. It depends on the realization used. That is the reason why the filtering and control community have made extensive works in order to find the "best" realization.
What a "best" realization means in this context may vary. Hence, some works try to save computations while others attempt to reduce the parametric sensitivity or the roundoff noise gain. Some constraints may also be taken into account such as legibility of the resulting algorithm or antiwindup properties.

In modern automotive industry, hundreds of controllers are digitally implemented on embedded processors in a single car. Numerous applications, from engine control to vehicule dynamics, require precise and performant controllers. Most of them, for cost reason, are realized in fixed-point processors (in opposition to floating-point processors), and require a specific attention to avoid numerical

difficulties.
Even for more advanced processors, there is a real need for methodology in order to manage better the compromise between the on-line computational efforts (number of operations, memory size) and the quality of the results. Such a methodology concerns not only the controller designer but also the engineers working on the real-time software developpement.

State-space realizations are indeed a powerful tool in order to support such a methodology. The state vector gathers the variable to be stored at each computation step (between two sampling time). This paper shows however that the standard state-space realization can not handle important points on implementation and proposes a generalised form. It focuses on the problem of parameterization and macroscopic computational representation.

This paper first presents a brief state of the art in optimal FWL controller implementation. Then section 3 highlights the interest of the implicit state-space representation. Using this formalism, realizations with δ and γ operator, state-estimate feedback controller, cascade and direct form I realizations are reformulated in section 4. Some concluding remarks are given in section 5.

2. NOTATION AND PROBLEM FORMULATION

Although often considered (Middleton and Goodwin, 1990), transfer function based representations are not considered in the following as they can be associated in a one to one manner to a state space representation using the companion form. Let us consider the discrete time LTI **system** $\mathcal{S}$ defined by one of its **realization** $R = (A_q, B_q, C_q, D_q)$, and the input/output equations are :

$$\mathcal{S} \left\{ \begin{array}{rcl} qX_k &=& A_q X_k + B_q U_k \\ Y_k &=& C_q X_k + D_q U_k \end{array} \right. \quad (1)$$

where the q-operator is the shift-operator defined by

$$qX_k \triangleq X_{k+1} \quad (2)$$

The associed **transfer fonction** is

$$H_{\mathcal{S}}(z) = \frac{Y}{U} = C_q(zI - A_q)^{-1}B_q + D_q \quad (3)$$

Contrary to Gevers and Li (1993), realization, representation and parametrization will have different meaning here. The term **parametrization** will designate all the coefficients involved in the calculations (ie all the reprensentative coefficients, different from 0 or ± 1, in the matrix of the realization). For example, two realizations derived from the transfer fonction, like Direct Form I and Direct Form II can share the same parametrization (the same parametric coefficients) but have different realizations (Direct Form I is not minimal, see 4.4).

Indeed, all the realizations of the form

$$R_T = (T^{-1}A_qT, T^{-1}B_q, C_qT, D_q) \quad (4)$$

where T is a non-singular matrix, are equivalent in infinite precision (they represent the same system, they have the same input/output relationship) *but* not any more in finite-precision. The coefficients of the state-space matrices are then numerically quantified as well as the computations.

The degradation resulting from the quantization depends on the realization used. This is also the case for the number of computations to be performed during one sampling time and the memory required.

The two FWL effects (roundoff noise and coefficients approximation) have been widely studied in digital signal processing and in control design. Some works concern the problem of finding, starting from a given state-space realization (see eq. 1), the change of coordinate leading to the *optimal* realization according to different criterion : parametric sensibility (in order to find the closest transfer function once the coefficients are quantized), roundoff noise gain, sparsity, stability, ... (Istepanian and Whidborne, 2001; Gevers and Li, 1993; Rotea and Williamson, 1995; Wu *et al.*, 2000*a*; Wu *et al.*, 2001; Tavşanoğlu and Thiele, 1984; Istepanian *et al.*, 1996). All realizations considered then are the same dimension, most often considered as minimal.

Other works analyse the interest of using alternative discrete-time operators (eg δ or γ-operator) for FWL implementation (Wu *et al.*, 2000*b*), such as δ or γ-operators defined by

$$\delta \triangleq \frac{q-1}{\Delta}, \qquad \gamma \triangleq \frac{2}{\Delta}\frac{q-1}{q+1} \quad (5)$$

where Δ is a positive real constant [1] .
With these operators, state-space systems are

$$\left\{ \begin{array}{rcl} \delta X_k &=& A_\delta X_k + B_\delta U_k \\ Y_k &=& C_\delta X_k + D_\delta U_k \end{array} \right. \quad (6)$$

$$\left\{ \begin{array}{rcl} \gamma X_k &=& A_\gamma X_k + B_\gamma U_k \\ Y_k &=& C_\gamma X_k + D_\gamma U_k \end{array} \right. \quad (7)$$

It can be easily shown that the realization (1), (6) and (7) are equivalent in infinite precision if

$$A_\delta = \frac{A-I}{\Delta},\ B_\delta = \frac{B}{\Delta},\ C_\delta = C,\ D_\delta = D \quad (8)$$

[1] in Middleton and Goodwin's (1990) definition, Δ corresponds to the sampling period, but this contraint is removed by Gevers and Li (1993)

and

$$\begin{aligned} A_\gamma &= \frac{2}{\Delta}\left(I + A_q\right)^{-1}\left(A_q - I\right) \\ B_\gamma &= \frac{2}{\Delta}\left(I + A_q\right)^{-1} B_q, \quad C_\gamma = C_q \\ D_\gamma &= D_q - C_q\left(I + A_q\right)^{-1} B_q \end{aligned} \tag{9}$$

The δ-operator has shown numerical advantages for FWL implementation (Gevers and Li, 1993; Wu *et al.*, 2000*a*), and section 4.1 shows that implementation using δ-operator make use of more variables than implementation using shift-operator. This points that a non minimal realization should not be *a priori* rejected when looking for an optimal FWL realization. A larger class of equivalence including non-minimal realization could be defined using the Inclusion Principle (Stanković and Šiljak, 2001).

Some related works are concerned with the numerical implementation aspects (residue feedback (Williamson, 1986), quantization after or before multiplication (Rotea and Williamson, 1995)), with conversion from floating-point to fixed-point algorithms for DSP processors (Keding *et al.*, 1998; Kum *et al.*, 2000) and with evaluation of the accuracy of fixed-point algorithms (Menard and Sentieys, 2002).
As long as it is possible to emulate in software every calculation at arbitrary fixed precision, it also exists, for a given parametrization, an infinite choice of software technics. For example, the same transfer function could be implemented with the same realization but with differents software technics, like the same fixed-point representation for each coefficient with truncation, fixed-point representation with roundoff, or fixed-point representation with residue feedback. They don't have neither the same accuracy nor the same execution time, but the same realization.
This gives an extra degree of freedom for the implementation, but it is not used in the search for an optimal FWL controller relatively to a certain measure. Most of the time, a unique implementation is implicitly associated with a given parametrization.
Rotea and Williamson (1995) propose a unified representation of FWL implementation : the structure of their model is independent of the specific implementation, and they can formalize only some particular FWL implementation, like quantization before and after multiplication, and error feedback. But this is not enough to represent other existent implementations.
This points a paradox : some realizations (for example, those using γ-operator) can not be realistically implemented, whereas some implementations cannot be associated with classical state-space realizations. The important point is generally to make closer system realization and control algorithms.

3. IMPLICIT STATE-SPACE REALIZATIONS

This section proposes to use implicit state-space realizations as a unifying framework to allow a more detailled description of the control or filtering algorithms. Such an implicit realization for linear system can be written as :

$$E\begin{pmatrix} Z_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} Z_k \\ U_k \end{pmatrix} \tag{10}$$

For a special case where $E = \begin{pmatrix} E_{11} & 0 \\ E_{21} & I \end{pmatrix}$, the system is said to be singular (and cannot be easily implemented) iff E_{11} is singular (Dai, 1989).

The specialized form (11) of the realization (10) will be used to make explicit the parametrization and the distinction between intermediate and stored variables.

The model of the proposed unifying framework for FWL implementations of LTI systems is :

$$\begin{pmatrix} J & 0 & 0 \\ -K & E & 0 \\ -L & 0 & I \end{pmatrix}\begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix}\begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix} \tag{11}$$

where

- the J matrix is lower triangular with 1 on the diagonal

$$\begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ \star & \ddots & 0 & & \vdots \\ \vdots & \star & 1 & 0 & \vdots \\ \vdots & & \star & \ddots & 0 \\ \star & \dots & \dots & \star & 1 \end{pmatrix} \tag{12}$$

- E is non-singular, and, most often to be taken equal to identity
- T_{k+1} is the intermediate variable in the calculations of step k (the column of 0 in the second matrix shows that T_k is not used for the calculation at step k : that defines the concept of intermediate variables)
- X_{k+1} is the stored state-vector (it is effectively stored from one step to the next)

T_{k+1} and X_{k+1} form the state-vector : X_{k+1} is stored from one step to the next, and T_{k+1} is used in the calculations.

It is implicitly considered throught the paper that the computations associated to realization (11) are ordered from top to bottom. So the following algorithm is associated in a one to one manner to (11) :

[1] $J.T_{k+1} = M.X_k + N.U_k$:
calculation of the intermediate variables. J is lower triangular, so $T^{(0)}_{k+1}$ is first calculated, and then $T^{(1)}_{k+1}$ using $T^{(0)}_{k+1}$ and so on ...
[2] $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
[3] $Y_k = L.T_{k+1} + R.X_k + S.U_k$

J and E beeing non-singular, with an infinite precision, equation (11) is equivalent to the classical state-space form :

$$\left(\frac{X_{k+1}}{Y_k}\right) = \left(\frac{KJ^{-1}M + P \,\middle|\, KJ^{-1}N + Q}{LJ^{-1}M + R \,\middle|\, LJ^{-1}N + S}\right)\left(\frac{X_k}{U_k}\right)$$

Indeed, the parametrization is changed in this case (as well as the number of state variables).

4. ILLUSTRATING EXAMPLES

4.1 δ-operator

As seen in section 2, the δ-operator is an alternative discrete-time operator promoted by Middleton and Goodwin (1990). This operator proposes a theorical interesting unified formulation of continuous and discrete time filter (when $Te \to 0$) and has also shown numerical advantages for FWL implementation(Gevers and Li, 1993; Wu *et al.*, 2000*a*). According to equations (5) and (6), the algorithm to implement a δ-realization is the following :

[1] $T_{k+1} = A_\delta X_k + B_\delta U_k$
[2] $X_{k+1} = X_k + \Delta T_{k+1}$
[3] $Y_k = C_\delta X_k + D_\delta U_k$

where T is an intermediate vector used for the calculation. A δ-realization can be written using the shift-operator with the implicit proposed state-space form :

$$\begin{pmatrix} I & 0 & 0 \\ -\Delta I & I & 0 \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix}\begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

It is important to notice that (1) and (6) are equivalent, *but* the parametrization is completely different. There is no singular matrix T to change the base of the realization in the explicit form to change from (1) to (6), but the optimal δ-realization has better closed-loop stability margin than the optimal q-realization in FWL implementation (Wu *et al.*, 2000*b*). It is due to a non minimal realization and a different organization of the computations.

4.2 the implicit γ-operator

This discrete-time operator was introduced by Gevers and Li (1993) (this operator also exists in a generalized form, called the π-operator (Back *et al.*, 1996)).
While the inverse δ-operator geometrically represents an optimal first order explicit numerical integrator, the inverse γ-operator represents an optimal first order implicit method : the trapezoidal integrator (Rostgaard *et al.*, 1993). This explains the important role played by γ-operator in the unification of continuous and discrete time systems (Rabah and Bergeon, 2001).
However, a state-space system like (7) could not be directly implemented due to the implicit nature of γ.

Rostgaard *et al.* (1993) propose an iterative algorithm to implement γ-based state-space, but this implementation is rather expensive computationally. Świder (1998) also proposes another method to implement it, without iterations. Its algorithm requires a state-space in canonical form and exploits some numerical particularities of $\left(I - \frac{\Delta}{2}A_\gamma\right)^{-1}$.

4.3 Cascade realization

A method that is widely used to implement high-order controller consists to consider the whole controller like an interconnection of multiple subsytems (such that the output of one subsystem is the input of another). Each subsystem is a low order controller (first or second degree controllers in most cases, and each one is implemented in an usual form (direct form II, for example).
Let's consider two systems $\mathcal{S}_1$ and $\mathcal{S}_2$ with realizations (A_1, B_1, C_1, D_1) and (A_2, B_2, C_2, D_2) , and with correct dimensions :

$$\begin{array}{l} \mathcal{S}_1 \left\{ \begin{array}{rcl} X^{(1)}_{k+1} &=& A_1 X^{(1)}_k + B_1 U^{(1)}_k \\ Y^{(1)}_k &=& C_1 X^{(1)}_k + D_1 U^{(1)}_k \end{array} \right. \\ \mathcal{S}_2 \left\{ \begin{array}{rcl} X^{(2)}_{k+1} &=& A_2 X^{(2)}_k + B_2 U^{(2)}_k \\ Y^{(2)}_k &=& C_2 X^{(2)}_k + D_2 U^{(2)}_k \end{array} \right. \end{array} \qquad (13)$$

A realization of $\mathcal{S}$, the serie connection of $\mathcal{S}_1$ and $\mathcal{S}_2$ is $R = (A, B, C, D)$:

$$\begin{array}{ll} A = \begin{pmatrix} A_1 & 0 \\ B_2 C_1 & A_2 \end{pmatrix} & B = \begin{pmatrix} B_1 \\ B_2 D_1 \end{pmatrix} \\ C = \begin{pmatrix} D_2 C_1 & C_2 \end{pmatrix} & D = D_2 D_1 \end{array} \qquad (14)$$

Pratically, the computation performed do not correspond to (14), in that the ouput of $\mathcal{S}_1$ is computed first to be used then as input for $\mathcal{S}_2$. The following implicit state-space form exhibits the right parametrization and computations :

$$\begin{pmatrix} I & 0\ 0 \\ \begin{pmatrix} 0 \\ -B_2 \end{pmatrix} & I\ 0 \\ -D_2 & 0\ I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ \begin{pmatrix} X_{k+1}^{(1)} \\ X_{k+1}^{(2)} \end{pmatrix} \\ Y_k^{(2)} \end{pmatrix} = \begin{pmatrix} 0 & \begin{pmatrix} C_1 & 0 \end{pmatrix} & D_1 \\ 0 & \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} & \begin{pmatrix} B_1 \\ 0 \end{pmatrix} \\ 0 & \begin{pmatrix} 0 & C_2 \end{pmatrix} & 0 \end{pmatrix} \begin{pmatrix} T_k \\ \begin{pmatrix} X_k^{(1)} \\ X_k^{(2)} \end{pmatrix} \\ U_k^{(1)} \end{pmatrix} \quad (15)$$

With $T_{k+1} = Y_k^{(1)} = U_k^{(2)}$.
The second realization required the use of intermediary variables but has the advantage to permit modular implementation (Williamson, 1992).

4.4 Direct Form I

Considering a transfer fonction

$$H(z) = \frac{b_0 z^n + \cdots + b_{n-1} z + b_n}{a_0 z^n + a_1 z^{n-1} + \cdots + a_n} \quad (16)$$

with $a_0 \neq 0$. The easiest and most natural way to implement it is to write

$$\left(\sum_{i=0}^{n} a_i q^{-i} \right) Y_k = \left(\sum_{i=0}^{n} b_i q^{-i} \right) U_k \quad (17)$$

This leads to the recurent equation

$$Y_k = \frac{1}{a_0} \left(\sum_{i=0}^{n} b_i U_{k-i} - \sum_{i=1}^{n} a_i Y_{k-i} \right) \quad (18)$$

This form is not minimal and the associated realization is order $2n$ with :

$$\left(\begin{array}{cccccccc|c} 0 & & & & & & & & 1 \\ 1 & \ddots & & & & & & & 0 \\ & \ddots & \ddots & & & & & & \vdots \\ & & 1 & 0 & & & & & 0 \\ \frac{b_1}{a_0} & \cdots & \cdots & \frac{b_n}{a_0} & -\frac{a_1}{a_0} & \cdots & \cdots & -\frac{a_n}{a_0} & \frac{b_0}{a_0} \\ & & & & 1 & 0 & & & 0 \\ & & & & & \ddots & \ddots & & \vdots \\ & & & & & & 1 & 0 & 0 \\ \hline \frac{b_1}{a_0} & \cdots & \cdots & \frac{b_n}{a_0} & -\frac{a_1}{a_0} & \cdots & \cdots & -\frac{a_n}{a_0} & \frac{b_0}{a_0} \end{array} \right)$$

The calculation of Y_k appears two times, because it is also stored in the state-vector. In the real implemented algorithm however, it appears only once, and the coefficients $(b_i)_{0 \leqslant i \leqslant n}$ are used, instead of $\left(\frac{b_i}{a_0}\right)_{0 \leqslant i \leqslant n}$.
Taking this into account is not a problem with the implicit state-space form (11) with :

$$\begin{array}{c} J = (a_0)\,,\ L = (1)\,,\ N = (b_0)\,,\ S = (0) \\ K = \left(0 \ \cdots \ \cdots \ 0 \middle| 1\ 0 \ \cdots \ 0 \right)^\top \\ M = \left(b_1 \ \cdots \ \cdots \ b_n \middle| -a_1 \ \cdots \ \cdots \ -a_n \right) \\ P = \left(\begin{array}{cccc|cccc} 0 & & & & & & & \\ 1 & \ddots & & & & & & \\ & \ddots & \ddots & & & & & \\ & & 1 & 0 & & & & \\ \hline & & & & 0 & & & \\ & & & & 1 & \ddots & & \\ & & & & & \ddots & \ddots & \\ & & & & & & 1 & 0 \end{array} \right)^\top \\ Q = \left(1\ 0 \ \cdots \ 0 \middle| 0 \ \cdots \ \cdots \ 0 \right)^\top \\ R = \left(0 \ \cdots \ \cdots \ 0 \middle| 0 \ \cdots \ \cdots \ 0 \right) \end{array} \quad (19)$$

In a DSP [2], the computation of the scalar product T_{k+1} could be process in the Multiply Accumulate Unit with an increase precision (for example, 40 bits instead of 16) (Roger and Aubenas, 2001). Exhibit the calculations permits to have different wordlengths in the algorithm, and to take account software and hardware improvements.

4.5 Observer-State-feedback Form

Let us now consider the following state-feedback control :

$$\begin{cases} \hat{X}_{k+1} = A\hat{X}_k + BU_k + L(Y_k - C\hat{X}_k) \\ u_k = -K\hat{X}_k \end{cases} \quad (20)$$

This system is implemented by :

[1] $T_{k+1} = -K\hat{X}_k$
[2] $\hat{X}_{k+1} = (A - LC)\hat{X}_k + BT_{k+1} + LY_k$
[3] $U_k = T_{k+1}$

It can be easily represented without changing the parametrization thanks to the implicit state-space representation :

$$\begin{pmatrix} I & 0\ 0 \\ -B & I\ 0 \\ -I & 0\ I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ \hat{X}_{k+1} \\ U_k \end{pmatrix} = \begin{pmatrix} 0 & -K & 0 \\ 0 & A - LC & L \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} T_k \\ \hat{X}_k \\ Y_k \end{pmatrix}$$

Indeed, the realization (20) is equivalent, in infinite precision, to $(A - BK - LC, L, -K, 0)$. It has however many appealing properties for implementation (eg antiwindup, meaningful state-variables).

5. CONCLUSION

This article highlights the interest of the implicit state-space representation in the context of FWL implementation problems. With this type of representation, no more δ or γ-operators are needed to

[2] Digital Signal Processor

introduce a relevant parametrization for FWL implementation. Moreover, the link with the macroscopic computations to be performed are made more explicit and the gap between the software FWL implementation and its formal description is reduced. As a perspective, promising researches will consist to re-examine previous works on optimal realizations (relating to parametric sensitivity or FWL stability measure, etc...) using the implicit state-space form. The roundoff noises and their propagation will also be reconsidered in this framework.

6. ACKNOWLEDGEMENT

The authors wish to thank PSA Peugeot Citroën for their interest and financial support.

REFERENCES

Back, A.D., A.C. Tsoi, B.G. Horne and C.L. Giles (1996). Alternative discrete time operators and their application to nonlinear models. *IEEE Transactions on Signal Processing, special issue on Applications of Neural Networks to Signal Processing*.

Dai, L. (1989). *Singular Control Systems*. lecture note in control and information sciences ed.. Springer-verlag.

Gevers, M. and G. Li (1993). *Parametrizations in Control, Estimation and Filtering Probems*. Springer-Verlag.

Istepanian, R. and Whidborne, J., Eds.) (2001). *Digital Controller implementation and fragility*. Springer.

Istepanian, R., I. Pratt, R. Goodall and S. Jones (1996). Effect of fixed-point parametrization on the performance of active suspension control systems. In: *13th IFAC World Congress*.

Keding, H., M. Willems, M. Coors and H. Meyr (1998). FRIDGE : A fixed-point design and simulation environment. *EDAA*.

Kum, KI., J. Kang and W. Sung (2000). AUTOSCALER for C : An optimizing floating-point to integer C program converter for fixed-point digital signal processors. *IEEE Transactions on Circuits and Systems* **47**(9), 840–848.

Menard, D. and O. Sentieys (2002). Automatic evaluation of the accuracy of fixed-point algorithms. In: *Proceedings of DATE02 (Design Automation and Test in Europe)*.

Middleton, R. and G. Goodwin (1990). *Digital Control and Estimation, a unified approach*. Prentice-Hall International Editions.

Rabah, R. and B. Bergeon (2001). On state-space representation for linear discrete-time systems in hilbert spaces. In: *Kharkov University Vestnik*. Vol. 514.

Roger, A. and C. Aubenas (2001). Application note : Signal processing with ST10-DSP. Technical report. ST Microelectronics.

Rostgaard, M., NK. Poulsen and O. Ravn (1993). A rapprochement between discrete-time operators. In: *ECC93*. Vol. 2. pp. 426–431.

Rotea, M. and D. Williamson (1995). Optimal realizations of finite wordlength digital filters and controllers. *IEEE Transactions on Circuits and Systems* **42**(2), 61–72.

Stanković, S. and D. Šiljak (2001). Contractibility of overlapping decentralized control. *System & Control Letters*.

Świder, Z. (1998). Realization using the γ-operator. *Automatica* **43**(11), 1455–1457.

Tavşanoğlu, V. and L. Thiele (1984). Optimal design of state-space digital filters by simultaneous minimization of sensibility and roundoff noise. In: *IEEE Trans. on Acoustics, Speech and Signal Processing*. Vol. CAS-31.

Williamson, D. (1986). Roundoff noise minimization and pole-zero sensibivity in fixed-point digital filters using residue feedback. In: *IEEE Trans. on Acoustics, Speech and Signal Processing*. Vol. ASSP-43.

Williamson, D. (1992). *Digital Control and Implementation, Finite Wordlength Considerations*. Prentice-Hall International Editions.

Wu, J., S. Chen, G. Li and J. Chu (2000*a*). Optimal finite-precision state-estimate feedback controller realizations of discrete-time systems. *IEEE Transactions on Automatic Control* **45**(8), 1550–1554.

Wu, J., S. Chen, J. Whidborne and J. Chu (2001). Optimal realizations of floating-point implemented digital controllers with finite wordlength considerations. *International Journal of Control* **77**(5), 427–440.

Wu, J., S. Chen, R. Istepanina and J. Chu (2000*b*). Shift and delta operator realizations for digital controllers with finite-word-length considerations. *IEE Proc. Control Theory and Applications*.