April, 1981
LIDS-P-1084
To be presented at 1981 Jount Automatic Control Conference, Charlottesville, VA
June 1981

REALIZATION OF A/D AND D/A CODERS

D.G. Wimpey, Graduate Student, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Mass.

T. L. Johnson, Visiting Scientist, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Mass.

M. E. Kaliski, Associate Professor, Department of Electrical Engineering, Northeastern University, Boston, Mass.

ABSTRACT

The notion of a discrete-time coder as a device which converts real vector-valued sequences into sequences over a finite alphabet is formalized. A hierarchical classification of all coders, in terms of their input-output mappings, is sought. This classification is based on a canonical structure theory being developed for coders. An algebraic approach is used to define three classes of coders which have simple canonical realizations, i.e., ones for which known synthesis procedures may be used. It is proposed that coders be viewed as acceptors of real languages, and the hierarchy of the real languages be used in conjunction with the hierarchy suggested by these three coders to achieve a complete classification.

NOMENCLATURE

C	coder mapping $R^{p*} \to W$
$\ell(u)$	the length of a sequence u, $\ell(\Lambda) = 0$
q	a memoryless quantization mapping, $q: R^p \to W$
R^p	p-dimensional real Euclidean space
W	finite set of coder output symbols
ε	set membership symbol
δ	next-state mapping of a finite state system
β	readout mapping of a finite state system
τ	right shift transformation
σ	left shift transformation
Λ	the empty string
ϕ	the empty set
$\approx$	an equivalence relation
$X/\approx$	the set whose elements are the equivalence classes of X modulo $\approx$
$\circ$	denotes function composition
$\forall$	for all

SUPERSCRIPTS

X^*	the free monoid generated by the set X
X^+	the free semigroup generated by the set X
f^*	the causal extension of the function f

INTRODUCTION

Coders and decoders here are devices (such as A/D and D/A converters) which transform real-valued sequences[1] into sequences of symbols over some finite alphabet and vice-versa. They form the interconnection between systems whose variables evolve on the continuum and systems, such as digital computers, which have a discrete state and input set. Coders and decoders are therefore inherent subsystems in hybrid control systems (1), where the plant state variables and outputs take values in the reals, and the controller is modelled as an automaton.

In the development of any general compensation scheme involving an automaton as controller, the choice of the coder and decoder should be included in the overall design process; the design of the coder and decoder is in fact central in the compensator synthesis. While various hierarchies of automata structures exist (finite-state, linear-bounded, pushdown, etc.) providing the necessary design constraints, no such classifications exist for coders and decoders. A constraint on the coder may be that its "continuous-state part" must be in the same class as the plant (for example finite-dimensional) and its "discrete-state part" in the same class as the automaton. Thus it becomes necessary to develop a canonical structure theory for these systems.

Some examples of coders commonly found in practice are memoryless quantizers, quantizers with hystersis (2), differential quantizers (3) and resettable integrators. A quantizer with hystersis is shown in Figure 1 below and may be realized as a quantizer (different from q) followed by a finite automaton. We will call coders which can be decomposed this way finitary coders. A coder which is not finitary is given in Figure 2.

We will view a coder as a map

$$C: R^{p*} \to W$$

where R^{p*} is the set of all finite length sequences of vectors in R^p, and W is the finite set of output

[1] Any sampling in time is assumed to have taken place prior to conversion.

symbols of the coder. A decoder performs the inverse operation. It has already been shown (4) that any coder may be realized, for $n \geq 1$, as a composition of an n-dimensional discrete-time system followed by a memoryless quantizer (Figure 3), i.e., as a composition of maps $C_1: R^{p*} \to R^n$ and $q: R^n \to W$. While this decomposition is completely general it is not the most useful one in terms of coder synthesis. This is since any part of the coder that would normally be synthesized using digital logic circuitry is treated as part of the discrete-time system with input output map C_1, with states taking values in R^n. Our aim will be to develop conditions on the mapping C for the coder to be synthesized using standard circuit synthesis techniques. It is thus desirable that these conditions result in realizations of C in which the inherently analog and inherently discrete parts are identifiable. The results that are presented here are preliminary and pertain to certain "simple but practically useful" coders; in general the problem concerns the realization of nonlinear discontinuous mappings and is difficult. The results for decoders are similar and are not given.

NOTATION AND DEFINITIONS

Definition
A coder is any function

$$C: R^{p*} \to W$$

where W is a finite set consisting of the coder output symbols. Sometimes the domain of C will be the semigroup $R^{p+} = R^{p*} - \{\Lambda\}$ where Λ is the empty string. In the sequel the domain of C is always assumed to be R^{p*} unless otherwise indicated.

To view a coder as a mapping from strings to strings we define the causal extension of C to be the mapping

$$C^*: R^{p*} \to W^+$$

obtained by extending C as follows:

$$C^*(\Lambda) = C(\Lambda)$$

$$C^*(y_1 \ldots y_k) = C(\Lambda)C(y_1)\ldots C(y_1 \ldots y_k)$$

Definition
Let X be any set,

(a) The left shift transformation $\sigma: X^* \to X^*$ is defined as follows:

$$\sigma x = \begin{cases} x(2)\ldots.x(k) & \text{if } x = x(1)x(2)\ldots x(k) \text{ for } x(i) \in X,\ k \geq 1 \\ \Lambda & \text{if } x \in X \cup \{\Lambda\} \end{cases}$$

We extend this to multiple shifts by defining σ^0 to be the identity map, $\sigma^1 = \sigma$ and $\sigma^{n+1} = \sigma^n\sigma$.

(b) The right shift transformation $\tau: X^* \to X^*$ is defined as

$$\tau^u x = ux \qquad \forall u, x \in X^*$$

Definition
Let C be a coder and consider the associated set of conjugate transformations $\overline{C} = \{C\tau^u : u \in R^{p*}\}$. If $\overline{C}$ is finite we say C is finitary, and if $\overline{C} = \{C\}$ we say C is unitary.

This definition is just Raney's (5) definition modified to handle sequences over R^p. Note that this notion of a unitary coder is only useful if the domain of C is R^{p+}. A minor modification in the definition is necessary if one wishes to define unitary coders on R^{p*}. We will clarify this later on.

Example
A quantizer is a memoryless coder with domain R^{p+} given by

$$C(y_1 \ldots y_k) = q(y_k)$$

where $q: R^p \to W$. C is clearly unitary.

Example
$C: R^* \to W$ is defined as

$$C(\Lambda) = 1$$

$$C(y_1 \ldots y_k) = \begin{cases} 0 & \text{if } y_k \geq 0 \text{ and the number of nonnegative terms in } y_1 \ldots y_{k-1} \text{ is either even or zero} \\ 1 & \text{otherwise} \end{cases}$$

Then $\overline{C} = \{C_1, C_2, C_3\}$, $C_i: R^* \to \{0,1\}$ where

$C_i = C\tau^u$, $u \in U_i \subset R^*$ for $i = 1,2,3$, and

$U_1 = \{u \in R^*$: the number of nonnegative terms in u is odd, and the last term is negative$\}$

$U_2 = \{u \in R^*$: the number of nonnegative terms in u is odd, and the last term is nonnegative$\}$

$U_3 = \{u \in R^*$: the number of nonnegative terms in u is even or zero$\} \cup \{\Lambda\}$

and thus C is finitary.

Example
The quantizer with hystersis of Figure 1 is defined as

$$C(\Lambda) = w_o$$

$$C(y) = q(y + ad(w_o)) \qquad y \in R$$

$$C(y_1 \ldots y_k) = q(y_k + ad\{C(y_1 \ldots y_{k-1})\}) \qquad k = 2,3\ldots.$$

where $a \in R$, $w_o \in W$ are given and d is an injection of W into R. Suppose $W = \{\alpha, \beta\}$, $d(\alpha) = -2$, $d(\beta) = 1$, $a = 1$, $w_o = \alpha$ and $q: R \to \{\alpha, \beta\}$ is the mapping

$$q(y) = \begin{cases} \alpha & \text{if } y \geq 0 \\ \beta & \text{otherwise.} \end{cases}$$

Then it is easy to see that the finite state system of Figure 4 is a realization of C. We will see that this implies C is finitary. Note that a decomposition of this coder in the form of Figure 3 appears unnatural.

Alternate descriptions of unitary and finitary coders may be obtained via the mechanism of Nerode

equivalence.

Definition
Let u,v be sequences in R^{p*} and define the (Nerode) equivalence relation (6) $\approx$ on R^{p*} as

$$u \approx v \iff C(ux) = C(vx) \quad \forall\, x \in R^{p*}$$

It is immediate that $\approx$ is a right-congruence on R^{p*}. The following proposition is also evident, and the proof is left to the reader.

Proposition
C is finitary iff $\approx$ has finite index.

A coder which is not finitary is the shift-unitary coder defined below.

Definition
For each $u \in R^{p*}$ and some fixed $\theta \in R^{p}$ define the shift-conjugate functions $C_u: R^{p*} \to W$ of C as follows:

$$C_u(\Lambda) = C\tau^{\theta}(\Lambda)$$

$$C_u(y_1 \ldots y_k) = C\tau^{\theta}(y_1 \ldots y_k) \quad k = 1,2,\ldots,\ell(\theta)-1$$

$$C_u(y_1 \ldots y_k) = C\tau^{u}(y_1 \ldots y_k) \quad k = \ell(\theta), \ell(\theta)+1,\ldots$$

Then we say C_+ is shift-unitary if $\{C_u: u \in R^{p*}\} = \{C\}$ for some $\theta \in R^{p}$.

Example
Consider the coder C given by

$$C(\Lambda) = 0$$

$$C(y) = \operatorname{sgn}(y^2-1) \qquad y \in R$$

$$C(y_1 \ldots y_k) = \operatorname{sgn}(y_k^2 - y_{k-1}^2) \quad y_i \in R;\ k = 2,3,\ldots$$

where $\operatorname{sgn}: R \to \{0,1\}$ is the mapping

$$\operatorname{sgn}(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Then C is shift-unitary with $\theta = (\theta_1, 1)$ for any $\theta_1 > 1$.

The definition of a shift-unitary coder for the case where $\ell(\theta) = 1$ is precisely the definition of a unitary coder with domain R^{p*}. The fact that a shift-unitary coder is not finitary (except for when it is unitary) will become evident in the next section.

The following definition will be useful in characterizing finitary coders.

Definition
A threshold finite automaton (TFA) is the 5-tuple

$$M_T = (Q, Y, W, \delta, \beta)$$

where

Q = the finite set of states, $Q = \{q^1, \ldots, q^r\}$

Y = the input set, $Y \subset R^p$

W = the finite output set, $W = \{w^1, \ldots w^m\}$

δ: $Q \times Y \to Q$ is the next-state function given by
$\delta(q^i, y) = q^j$ if $y \in A_{ij} \subset Y$

β: $Q \to W$ is the readout function

A finite automaton (6) is defined similarly except that Y is a finite set and the notation M is used instead of M_T.

Note that the specification of δ defines the sets A_{ij}, and that for fixed i, the sets A_{ij}, $j = 1,\ldots,r$ form a partition of Y.

For a particular initial state $q_1 \in Q$, the response function of M_T is the map

$$M_{T,q_1}: R^{p*} \to W$$

given by

$$M_{T,q_1}(y_1 \ldots y_k) = \beta(\delta^*(q_1, y_1, \ldots, y_k)) \qquad k = 1,2,\ldots$$

$$M_{T,q_1}(\Lambda) = \beta(q_1)$$

Definition
C has memory span N if N is the smallest non-negative integer such that

$$C(y_1 \ldots y_k) = C(y_{k-N} \ldots y_k) \qquad k = N+1, N+2, \ldots.$$

If no such N exists, then we say that C has infinite memory span.

CODER REALIZATION, SYNTHESIS

A unitary coder is the simplest of all coders; it is memoryless. This is the statement of the following Theorem.

Theorem
$C: F^{p^+} \to W$ is unitary iff there exists a map $q: R^p \to W$ such that for all $y_1 \ldots y_k \in R^{p^+}$,

$$C(y_1 \ldots y_k) = q(\sigma^{k-1} y_1 \ldots y_k) \qquad k = 1,2,\ldots$$

Proof. Necessity.
First define the sets A_i for each $w_i \in W$ as follows:

$$A_i = \{y \in R^p: C(y) = w_i\}$$

Then

$$C(y_1 \ldots y_k) = (C\tau^{y_1 \ldots y_{k-1}})(y_k)$$

$$= C(y_k) \qquad \text{since C is unitary}$$

$$= w_i \quad \text{if } y_k \in A_i$$

Now define the function $q: R^p \to W$ as

$$q(y) = w_i \quad \text{if } y \in A_i$$

Then

$$C(y_1 \ldots y_k) = q(\sigma^{k-1} y_1 \ldots y_k)$$

Sufficiency.
Suppose

$$C(y_1 \cdots y_k) = q(\sigma^{k-1} y_1 \cdots y_k)$$

for some map $q: R^p \to W$. Then for $u \in R^{p*}$,

$$(C\tau^u)(y_1 \cdots y_k) = q(\sigma^{\ell(u)+k-1} y_1 \cdots y_k) = q(y_k) \qquad k = 1,2,\ldots$$

and hence C is unitary. Q.E.D.

The synthesis of a unitary coder therefore involves the synthesis of the map $q: R^p \to W$. Note that this may not always be practical; consider for example the map

$$q(y) = \begin{cases} 1 & \text{if } y \text{ is rational} \\ 0 & \text{otherwise} \end{cases}$$

We will not attempt to define a "well-behaved" quantizer here.

Finitary coders are more interesting; they are dynamic and in general have infinite memory span. The finitary coders are precisely those coders which are finite-state realizable. This is stated in the following theorem; the equivalent result in automata theory is standard (6), (7).

Theorem
C is finitary iff C is the response function of some (minimal) TFA.

Proof. Suppose C is finitary. Define the sets

$$A_{ij} = \{y \in R^p : uy \in U_j \ \forall u \in U_i\}$$

where the $U_i \subset R^{p*}$ are the congruence classes of the right congruence $\approx$. For fixed i, the sets A_{ij}, $j = 1,\ldots,r$ clearly form a partition of R^p. We now construct the (minimal) TFA $M_T = (Q, R^p, W, \delta, \beta)$ as follows:

$Q = \overline{C}$, i.e., $q^i = \overline{C}_i = C\tau^u$ for $u \in U_i$

$\delta: Q \times R^p \to Q$ via

$$\delta(\overline{C}_i, y) = \overline{C}_i \tau^y$$

$\beta: Q \to W$ via

$$\beta(\overline{C}_i) = \overline{C}_i(\Lambda)$$

Then

$$\overline{C}_i \tau^y = C\tau^{uy} \quad \text{for } u \in U_i$$
$$= \overline{C}_j \quad \text{if } y \in A_{ij}$$

and

$$\beta(\overline{C}_i) = \overline{C}_i(\Lambda)$$
$$= C\tau^u(\Lambda) \quad \text{for } u \in U_i$$
$$= C(u).$$

We take $q_1 = C\tau^\Lambda = \overline{C}_k$ where $\Lambda \in U_k$, and

$$M_{T,q_1}(y_1 \cdots y_k) = \beta(\delta^*(q_1, y_1 \cdots y_k))$$
$$= \beta(C\tau^\Lambda \tau^{y_1 \cdots y_k})$$
$$= C(y_1 \cdots y_k)$$

and $M_{T,q_1}(\Lambda) = \beta(\delta^*(q_1, \Lambda)) = C(\Lambda)$.

The proof of the converse is left to the reader. Q.E.D.

The coder of Figures 1 and 4 is therefore finitary. The following result separates out the threshold-type operations that occur in the TFA realization of a finitary coder from the dynamic part, and hence tells us how to go about synthesizing the coder. This decomposition should be compared with that of Figure 3.

Theorem
C is finitary iff C may be realized as the composition of maps

$$C = C_1 \circ C_2^*$$

where $C_2: R^{p*} \to V$ is unitary, V is a finite set, and $C_1: V^* \to W$ is finite-state realizable.

Proof. Suppose that C is finitary. Then by the previous theorem, C is the response function of a reduced TFA $M_T = (Q, R^p, W, \delta, \beta)$. Define the functions $\overline{q}_i$ as follows:

$$\overline{q}_i: R^p \to \{1,\ldots,r\} \qquad i = 1,\ldots,r$$

via

$$\overline{q}_i(y) = j \quad \text{if } y \in A_{ij}$$

where the $A_{ij} \subset R^p$ were defined in the proof of the previous theorem, r is the cardinality of Q. Now define $\overline{q}: R^p \to \{1,\ldots,r\}^r \triangleq V$ as

$$\overline{q}(y) = (\overline{q}_1(y), \ldots, \overline{q}_r(y))$$

and take $C_2: R^{p*} \to V$ to be the map

$$C_2(\Lambda) = \overline{v}$$

$$C_2(y_1 \cdots y_k) = \overline{q}(y_k) \qquad k = 1,2,\ldots$$

where $\overline{v}$ is any vector in V with $\overline{v}_1 = 1$.
Clearly C_2 is unitary. Define the finite automaton $M = (\overline{C}, V, W, \overline{\delta}, \overline{\beta})$ as follows:

(a) $\overline{\delta}: \overline{C} \times V \to \overline{C}$ via

$$\overline{\delta}(\overline{C}_i, v) = \overline{C}_{p_i(v)} \qquad v \in V$$

where $p_i: V \to \{1,\ldots,r\}$ is the projection mapping

$$p_i(v) = \text{ith component of } v$$

(b) $\bar{\beta}: \bar{C} \to W$ via

$$\bar{\beta}(\bar{C}_i) = \beta(\bar{C}_i)$$

Let C_1 be the response function $M_{q_1} : V^* \to W$ where $q_1 = \bar{C}_1$ (suppose $\Lambda \in U_1$). Then we have that, for $y \in R^p$,

$$\begin{aligned}\bar{\delta}(\bar{C}_i, \bar{q}(y)) &= \bar{C}_{p_i(\bar{q}(y))} \\ &= \bar{C}_{\bar{q}_i(y)} \\ &= \bar{C}_j \quad \text{if } y \in A_{ij} \\ &= \delta(\bar{C}_i, y)\end{aligned}$$

Now,

$$\begin{aligned}(C_1 o C_2^*)(y_1 \cdots y_k) &= C_1(C_2(\Lambda)C_2(y_1)C_2(y_1y_2)\cdots \\ &\quad \cdots C_2(y_1 \cdots y_k)) \\ &= C_1(\bar{v}\ \bar{q}(y_1)\cdots\bar{q}(y_k)) \\ &= \beta(\bar{\delta}^*(q_1, \bar{v}\ \bar{q}(y_1)\cdots\bar{q}(y_k)) \\ &= \beta(\bar{\delta}^*(q_1, \bar{q}(y_1)\cdots\bar{q}(y_k))) \quad \text{since} \\ &\qquad \bar{\delta}(q_1, \bar{v}) = q_1 \\ &= \beta(\delta^*(q_1, y_1 \cdots y_k)) \\ &= C(y_1 \cdots y_k).\end{aligned}$$

The proof of the converse is left to the reader.

Q.E.D.

Application of these results to the coder of Figure 4 results in the realization shown in Figure 5.

Not all coders with inherently discrete dynamics are finitary; the coder of Figure 2 has a countable state set. Extensions of the above decomposition result, where the isolated automaton is a deterministic pushdown automaton (6) are currently being investigated. Note that in this case a feedback-free decomposition cannot be obtained in general.

The shift-unitary coders are perhaps the simplest class of coders with realizations that have inherently analog dynamics and are finite-dimensional. These coders have finite memory-span (i.e. nilpotent) and may be implemented with a single (real number) storage register and a quantizer. This is the result of the following theorem.

Theorem

If C is shift-unitary then there exists a map $q: R^p \times R^N \to W$ such that

$$C(y_1 \cdots y_k) = \begin{cases} q(y_{k-N+1} \cdots y_k) & k \geq N \\ q(\theta_{k+1} \cdots \theta_N y_1 \cdots y_k) & 1 \leq k < N \end{cases}$$

$$C(\Lambda) = q(\theta)$$

where R^{p^+} is the sequence (of length $N \geq 1$) appearing in the definition of a shift-unitary coder.

Proof. For any sequence $y_1 \cdots y_k \in R^{p^+}$ and $1 \leq i, j \leq k$ define

$$B(y; i,j) \triangleq y_i \cdots y_j$$

Also define the sets Y_i, A_i for each $w_i \in W$ as follows:

$$Y_i \triangleq \{y \in R^{p^*} : \ell(y) \geq N \text{ and } C(B(y; 1,N)) = w_i\}$$

$$A_i \triangleq \{B(y; 1,N) : y \in Y_i\}.$$

Now define the map $q: R^p \times R^N \to W$ as

$$q(y) = w_i \text{ if } y \in A_i.$$

Then, for $y \in R^{p^+}$ with $\ell(y) = N$,

$$\sigma^N C^*(y)(1) = C(y_1 \cdots y_N) = w_i \Leftrightarrow q(B(y; 1,N)) = w_i$$

and for $y \in R^{p^+}$, with $\ell(y) = k = N+m$ for some integer $m \geq 0$,

$$\begin{aligned}\sigma^k C^*(y)(1) = C(y_1 \cdots y_k) &= C\tau^{y_1 \cdots y_m} \sigma^m(y_1 \cdots y_k) \\ &= C\sigma^m(y_1 \cdots y_k) \quad \text{since } \ell(\sigma^m y_1 \cdots y_k) \geq \ell(\theta) \\ &= \sigma^N C^*(\sigma^m y)(1) \quad \text{by defn. of } C^* \\ &= q(B(\sigma^m y; 1,N)) \text{ since } \ell(\sigma^m y) = N \\ &= q(B(y; k-N+1, k))\end{aligned}$$

For $\ell(y) = k$, $1 \leq k < N$,

$$\begin{aligned}C(y) &= C_u(y) \\ &= C\tau^{\theta}(y) \\ &= C\tau^{\theta_1 \cdots \theta_k}(\sigma^k \tau^{\theta} y) \\ &= C(\sigma^k \tau^{\theta} y) \quad \text{since } \ell(\sigma^k \tau^{\theta} y) \geq \ell(\theta) \\ &= q(B(\sigma^k \tau^{\theta} y; 1,N)) \\ &= q(\sigma^k \theta y)\end{aligned}$$

Finally, for $y = \Lambda$

$$\begin{aligned}C(\Lambda) = C_u(\Lambda) &= C\tau^{\theta}(\Lambda) \\ &= C(\theta) \\ &= q(B(\theta; 1,N)) \\ &= q(\theta).\end{aligned}$$

Q.E.D.

It should be clear that the memory span of a shift-unitary coder is finite, and is equal to $\ell(\theta)-1$. (Note that although θ may not be unique, $\ell(\theta)$ is.) For these coders, $u \approx v$ is equivalent to the statement

$$\sigma^{\ell(u)}\theta\, u = \sigma^{\ell(v)}\theta\, v$$

and hence $R^{p^*}/\approx$ is isomorphic to R^{pxN}. The extension to shift-finitary coders is currently under investigation.

The example of the shift-unitary coder given above may be realized as shown in Figure 6.

CONCLUSIONS AND DISCUSSION

We have formalized the notion of a discrete-time coder and have exhibited canonical structures for three classes of coders, the unitary, finitary and shift-unitary coders. We have indicated that, from the point of view of synthesis, coders and decoders should be viewed as hybrid-state systems; the major task is to define classes of coders and decoders which have identifiable discrete-state and continuous-state parts. A finitary coder realized in the general form shown in Figure 3 may still be easily synthesizable (the coder of Figure 7 is an example). However, this is not always the case and is the reason for the algebraic approach we have adopted.

We have also shown that a definite hierarchical classification of coders exists. To aid in this classification, a coder may be viewed as an acceptor (6) of real languages. A hierarchy of these languages exists similar to the hierarchy of languages studied in the computer science literature (regular, context-free, context-sensitive, etc.). Real context-free languages and their generating grammars have been studied by Lemone (8). The language accepted by the coder of Figure 2 can be shown to be context-free, while no language accepted by a shift-unitary coder is context-free unless the coder is unitary. The real regular languages form a proper subclass of the real context-free languages, and are precisely the languages accepted by the finitary coders (9).

Coders form one subclass of the nonlinear discontinuous mappings for which a realization theory can be developed. It is the fact that the domain of a coder mapping is a finite set that has enabled us to draw on many of the ideas and results in the field of computer science.

ACKNOWLEDGEMENTS

This research has been performed at the M.I.T. Laboratory for Information and Decision Systems and the Northeastern University Department of Electrical Engineering with support provided by the U.S. Air Force Office of Scientific Research, under contract number F49620-80-C-0002.

The views expressed in this paper do not necessarily represent those of the U.S. Government.

REFERENCES

1. Johnson, T.L., "Finite-State Compensation of Continuous Processes," Proceedings of the IFAC Triennial World Congress, June 1978, Helsinki, Finland, pp. 1823-1828.

2. Widrow, B., "Statistical Analysis of amplitude Quantized Sampled-Data Systems," Transactions of the AIEE (Appl. & Ind.), Vol. 79, Jan. 1961, pp. 555-568.

3. Limb, J.O., and Mounts, F.W., "Digital Differential Quantizer for Television," BSTJ, Vol. 48, 1969, pp. 2583-2599.

4. Kaliski, M.E., and Lemone, K., "Discrete Codings of Continuous-Valued Signals," Proc. 14th Annual Conference on Information Sciences and Systems, Johns Hopkins University, Dept. of Electrical Engineering, March 1980.

5. Raney, G., "Sequential Functions," Assoc. for Comp. Mach. Journal, No. 5, 1958, pp. 177-180.

6. Arbib, M.A., Theories of Abstract Automata, Prentice-Hall, Englewood Cliffs, N.J., 1969.

7. Hellerman, L., Duda, W.L., and Winograd, S., "Continuity and Realizability of Sequence Transformations," IEEE Trans. Elec. Comp., Vol. EC-15, No. 4, 1966, pp. 560-569.

8. Lemone, K., "Languages Over the Real Numbers", Ph.D. Thesis, Dept. of Math., Northeastern Univ., Boston, Mass., 1979.

9. Wimpey, D.G., "Towards a Structure Theory for Coders of Real-Valued Signals," LIDS-TM-1052, Oct. 1980, Mass. Inst. of Tech., Laboratory for Inf. and Decision Systems, Cambridge, Mass.

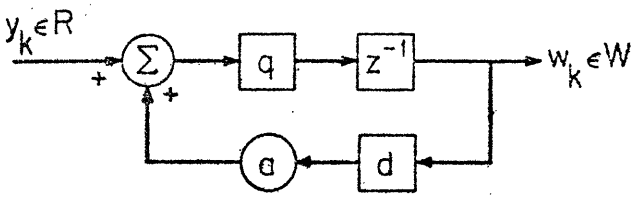


Figure 1 Quantizer q with hysteresis. q: R → W and d: W → R

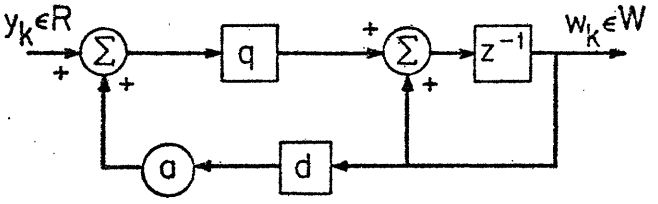


Figure 2 Differential quantizer
q: R → W and d: W → R

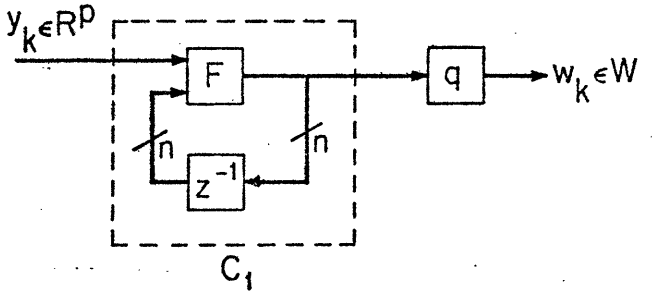


Figure 3 General coder decomposition
C_1: $R^{p^*} \to R^n$, q: $R^n \to W$

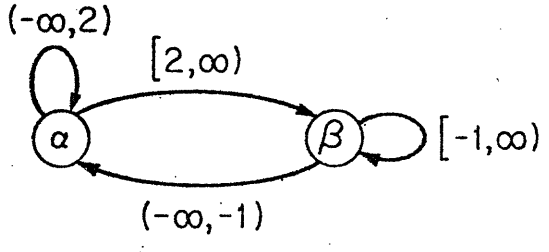


Figure 4 Finite-state realization of the quantizer with hysteresis

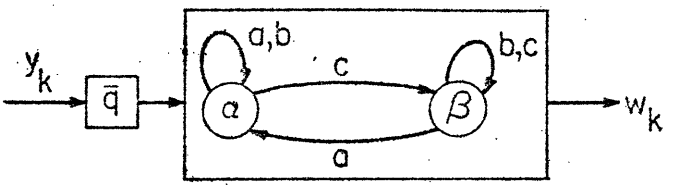


Figure 5 Decomposition of coder of Figure 4
$\bar{q}$: R → {a,b,c} via

$$\bar{q}(y) = \begin{cases} a & \text{if } y\in(-\infty,-1) \\ b & \text{if } y\in[-1,2) \\ c & \text{if } y\in[2,\infty) \end{cases}$$

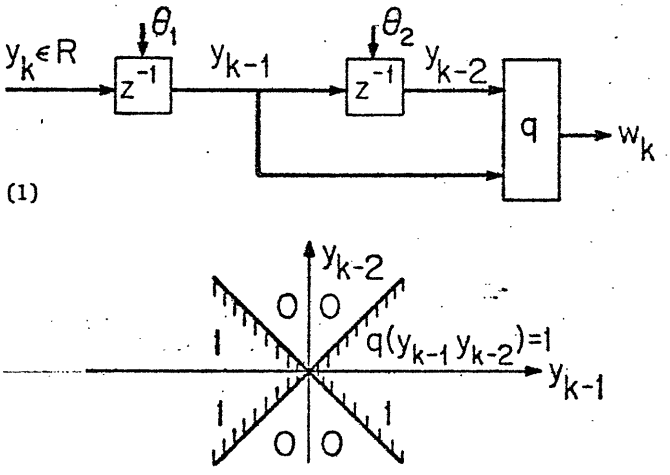
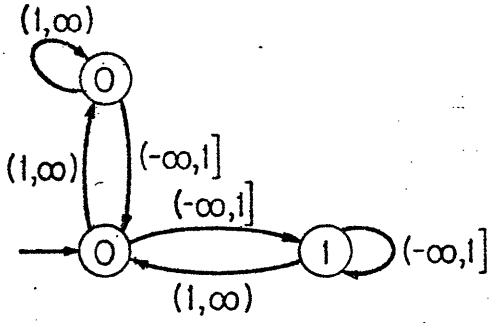


Figure 6 Example of shift-unitary coder
(1) Coder decomposition
(2) q: $R^2 \to \{0,1\}$



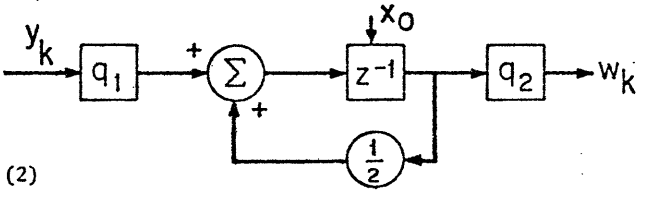


Figure 7 A finitary coder
(1) Specification as a TFA
(2) Realization in the form of Figure 3:
q_1: R → {-1,1} via

$$q_1(y) = \begin{cases} 1 & \text{if } y\in(1,\infty) \\ -1 & \text{otherwise} \end{cases}$$

q_2: R → {0,1} via

$$q_2(y) = \operatorname{sgn}(y - \tfrac{1}{6})$$

$x_0 \in (-\frac{1}{6}, \frac{1}{6})$