T. L. Johnson page 1

Finite-State Compensators for Continuous Processes

NGL-22-009-124

ESL-P-812

FOLDOUT FRAME

FINITE-STATE COMPENSATORS FOR CONTINUOUS PROCESSES

T. L. Johnson

Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139 U.S.A.

Abstract. Mathematical representation of feedback systems composed of both continuous and finite-state processes is discussed. Such a representation provides a new viewpoint for the design of computer control systems for industrial processes, and clarifies areas in which further research is required. Some motivating examples are provided.

Keywords. Automata theory; direct digital control; system theory; switching theory; computer control; discrete systems.

INTRODUCTION

The feedback interconnection of a continuous finite-dimensional dynamic system and a finite state sequential machine is shown in Figure 1. Because the natural input and output spaces of such systems are defined over different fields, two coupling elements, termed a coder and decoder; are necessary to define the transformations of plant output to compensator input, and conversely. There is an extensive literature dealing with each of the foregoing elements, for instance: Willems (1972) defines a dynamical systems theory of continuous processes (the plant); Bo- (the sequential machine); and Gallager (1968) treats communications theory (the coder and decoder). Considerable progress has been made in the last decade, with the emergence of mathematical systems theory, toward a unifying view of these disciplines--the works of Kalman, Falb and Arbib (1969) and Padulo and Arbib (1970) being representative of a broader literature.

The purpose of the present note is to pinpoint the shortcomings of existing theories when applied to the situation of Fig. 1 and

continuous and finite-state processes is discussed. Such a representation provides a new viewpoint for the design of computer control systems for industrial processes, and clarifies areas in which further research is required. Some motivating examples are provided.

Keywords. Automata theory; direct digital control; system theory; switching theory; computer control; discrete systems.

INTRODUCTION

The feedback interconnection of a continuous finite-dimensional dynamic system and a finite state sequential machine is shown in Figure 1. Because the natural input and output spaces of such systems are defined over different fields, two coupling elements, termed a coder and decoder; are necessary to define the transformations of plant output to compensator input, and conversely. There is an extensive literature dealing with each of the foregoing elements, for instance: Willems (1972) defines a dynamical systems theory of continuous processes (the plant); Bobrow and Arbib (1974) review automata theory (the sequential machine); and Gallager (1968) treats communications theory (the coder and decoder). Considerable progress has been made in the last decade, with the emergence of mathematical systems theory, toward a unifying view of these disciplines--the works of Kalman, Falb and Arbib (1969) and Padulo and Arbib (1970) being representative of a broader literature.

The purpose of the present note is to pinpoint the shortcomings of existing theories when applied to the situation of Fig. 1 and to propose a representation for the state of

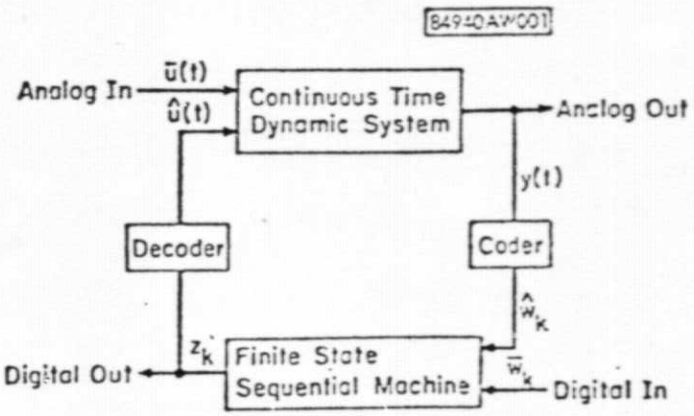


Figure 1: Block Diagram of Finite-State Compensation of a Continuous Process

FOLDOUT FRAME

T. L. Johnson

such systems. Such a representation provides the cornerstone of a theory for the direct design of finite-state compensators for continuous processes, which subsumes purely continuous control systems and purely finite-state feedback systems as special cases.

THE STATUS QUO: SOME NECESSARY INNOVATIONS

In motivating the new representation, it is most appropriate to recount the status quo:

(a) Continuous Dynamic Systems:

These are characterized (Willems and Mitter, 1971) by sets of the form $\Sigma = (U, \mathcal{U}, Y, \mathcal{Y}, X, \phi, r)$ where

- U: input set
- $\mathcal{U}$: input space
- Y: output set
- $\mathcal{Y}$: output space
- X: state set
- ϕ: state transition map
- r: read-out map

where U, Y are usually taken as Euclidean vector spaces on the field of real numbers, R, $\mathcal{U}$: $T \to U$ and $\mathcal{Y}$: $T \to Y$ are normed linear spaces on the time interval $T \subset R$. The system Σ will be assumed finite-dimensional in this discussion and X is correspondingly taken as a Euclidean vector space. The state transition mapping is continuous, with

ϕ: $T \times T \times \mathcal{U} \times X \to X$, and the readout map

r: $T \times T \times U \times X \to Y$ is continuous.

In addition, ϕ is required to have the identity, causality, and semigroup properties:

(b) Discrete Dynamic Systems:

These are characterized by sets of the form $M = (W, Z, Q, \delta, \lambda)$ where the finite sets W, Z, and Q are the input, output, and state sets, respectively; and δ: $I \times W \times Q \to Q$ λ: $I \times W \times Q \to Z$ are the next-state and current output mappings, respectively. The time-set I is taken as a sub-interval of the integers. (The notation has been slightly modified from that of the literature to conform with part (a)[1].) The machine is thought of as the recursion $q_{k+1} = \delta(k, u_k, q_k)$ for any $k \geq k_0 \in I$, and the output map $\lambda(k, u_k, q_k) = z_k$. It should be noted that ϕ of part (a) is analogous to $\delta(k, u_k, \delta(k-1, u_{k-1}, \delta(k-2, \ldots, u_{k_0}, q_{k_0})\ldots))$, the composition of δ with itself $k-k_0$ times.

(c) Coding and Decoding:

Let A, B denote (finite) alphabets and let A^+ (B^+) denote the set of non-empty finite-length sequences of elements from A(B). Let h: $A^+ \to B^+$ be a homomorphism; then the set $h(A) = \{h(a) \mid a \in A\}$ is termed a code. h(A) is uniquely decodable iff h is an injection.

These definitions summarize standard results from the theory of noiseless channel coding; see also Gallager (1968).

The objective of the next section is to characterize the interconnection shown in Fig. 1 as a dynamic system of some general class. Evidently, (a)-(c) are not directly compatible in their present condition. This is partly a problem of notation and technicalities, but there remain some fundamental difficulties. In (c), for instance, one would like the alphabet A to be real rather than finite-valued; but so long as B is finite, h(•) could never be an homomorphism. In (b) and (c), the real-time sequence of state transitions has been abstract-

ϕ, λ) where

U: input set
$\mathcal{U}$: input space
Y: output set
$\mathcal{Y}$: output space
X: state set
ϕ: state transition map
r: read-out map

where U, Y are usually taken as Euclidean vector spaces on the field of real numbers, R, $\mathcal{U}$: $T \to U$ and $\mathcal{Y}$: $T \to Y$ are normed linear spaces on the time interval $T \subset R$. The system Σ will be assumed finite-dimensional in this discussion and X is correspondingly taken as a Euclidean vector space. The state transition mapping is continuous, with

ϕ: $T \times T \times \mathcal{U} \times X \to X$, and the readout map

r: $T \times T \times U \times X \to Y$ is continuous.

In addition, ϕ is required to have the identity, causality, and semigroup properties:

(i) $\phi(t,t, u(\cdot), x) = x$ for all $t \in T$, $u(\cdot) \in \mathcal{U}$, $x \in X$.

(ii) For any $t \geq t_0$, both elements of T, and any $x_0 \in X$ if $u_1(\tau) \equiv u_2(\tau)$, $\tau < t$, then $\phi(t,t_0, u_1(\cdot), x_0) = \phi(t,t_0, u_2(\cdot), x_0)$.

(iii) For any $t_2 \geq t_1 \geq t_0$, all elements of T, any $u(\cdot) \in \mathcal{U}$, and any $x_0 \in X$, $\phi(t_2,t_0, u(\cdot), x_0) = \phi(t_2,t_1, u(\cdot), \phi(t_1,t_0, u(\cdot), x_0))$.

The system is thought of as the pair of equations $x(t) = \phi(t,t_0, u(\cdot), x(t_0))$ and $y(t) = r(t, t_0, u(t), x(t))$, $t \geq t_0$ both in T. Referring to Fig. 1, the input u(t) may be taken as the pair $(u(t), \hat{u}(t))$.

(b) Finite-State Sequential Machines (Finite Automata):

These are usually characterized by sets of

Let A, B denote (finite) alphabets and let A^+ (B^+) denote the set of non-empty finite-length sequences of elements from A(B). Let h: $A^+ \to B^+$ be a homomorphism; then the set $h(A) = \{h(a) \mid a \in A\}$ is termed a code. h(A) is uniquely decodable iff h is an injection.

These definitions summarize standard results from the theory of noiseless channel coding; see also Gallager (1968).

The objective of the next section is to characterize the interconnection shown in Fig. 1 as a dynamic system of some general class. Evidently, (a)-(c) are not directly compatible in their present condition. This is partly a problem of notation and technicalities, but there remain some fundamental difficulties. In (c), for instance, one would like the alphabet A to be real rather than finite-valued; but so long as B is finite, h(·) could never be an homomorphism. In (b) and (c), the real-time sequence of state transitions has been abstracted, and only the ordering of transitions remains; but for continuous dynamical systems as in (a), actual transition times in $\hat{u}(t)$ must be preserved. The basic operations on the fields in (a) and (b) are different, and thus the class of possible functions generated by ϕ and λ are inherently different. Finally, it might be necessary to generalize the input space of (a) to include distributions (i.e., impulse trains).

FOLDOUT FRAME 2

PROPOSED REPRESENTATION

By introducing some formal assertions, we demonstrate how definitions (b) and (c) may be modified so that the system of Fig. 1 may be interpreted as a generalized dynamical system. The demonstration that these assertions are

---

[1] By allowing δ, λ to depend on $k \in I$, the structure of the next-state and current-output mappings may vary with k. While W could denote any input string of finite length, it is proper here to associate it with an input set as used in (a). The initial state q_{k_o} and index k_o are not specified in the definition of M.

T.L Booth ... FOLDOUT FRAME

consistent is beyond the scope of this brief note. First, a definition is introduced that generalizes the machine M to have a real-time structure.

Definition 1: A finite-state dynamic system, on an interval $T \subset R$, is a set $\Gamma = (\tilde{W}, \tilde{\mathcal{W}}, \tilde{Z}, \tilde{\mathcal{Z}}, \tilde{Q}, \tilde{\delta}, \tilde{\lambda})$. The sets $\tilde{W} = T \times W$ and $\tilde{Z} = T \times Z$ for finite input and output sets W, Z respectively. $\tilde{\mathcal{W}}$ and $\tilde{\mathcal{Z}}$ are spaces of sequences of elements in $\tilde{W}$ and $\tilde{Z}$ for which the first index is monotonically increasing. For a positive integer n and finite set $\tilde{Q}$, the state set has the structure $\tilde{Q} = (T \times Q)^n$. The state-transition map $\tilde{\delta}$: $T \times T \times \tilde{\mathcal{W}} \times \tilde{Q} \to \tilde{Q}$ satisfies the following properties: Given a sequence $\{\tilde{w}\} = \{t_k, w_k\} \in \tilde{\mathcal{W}}$, let T_k denote the projection of $\tilde{w}$ onto T, i.e., the set of input transition-times $\{t_k\}$, $k = k_0, k_0+1, k_0+2, \ldots$. For any $\{\tilde{w}\} \in \tilde{\mathcal{W}}$, t_i, t_j, $t_\ell \in T_k$, $\tilde{q} \in \tilde{Q}$, then

(i) $\tilde{\delta}(t_i, t_k, \{\tilde{w}\}, \tilde{q}) = \tilde{q}$

(ii) for any $t_i < t_j$ and $\{\tilde{w}\} \in \tilde{\mathcal{W}}$ such that $\{\tilde{w}\}^1 \equiv \{\tilde{w}\}$ on $[t_i, t_j]$, then $\tilde{\delta}(t_j, t_i, \{\tilde{w}\}^1, q) = \tilde{\delta}(t_j, t_i, \{\tilde{w}\}, q)$.

(iii) whenever $t_i < t_j < t_\ell$, $\tilde{\delta}(t_\ell, t_i, \{\tilde{w}\}, \tilde{q}) = \tilde{\delta}(t_\ell, t_j, \tilde{\delta}(t_j, t_i, \{\tilde{w}\}, \tilde{q}))$.

Finally, the readout map $\tilde{\lambda}$: $T \times T \times \tilde{\mathcal{W}} \times \tilde{Q} \to Z$. #

A more concise definition of a finite-state dynamic system could be given if it were not for our attempt to clarify how the conventional notions of dynamic system and finite-state sequential machine have been reconciled. Actually, the ideas are quite simple. The key concept is that all "spontaneous" state transitions should ultimately be attributed to the passage of time, i.e., the elapsed time since the most recent (or n most recent) state transition(s), at the current input transition time, t_k. Between input transitions, the state may, of course, undergo a large (but finite) number of transitions at intervals determined by the internal system delays. However, these are completely predictable given the past state and the current input, and hence the next state can be computed. This is considered to be the key defining property of the "state" of a system. A major technical obstacle in defining real-time automata has thus been successfully removed--it would otherwise be impossible to project the "next state" of such a system until the "next" input transition were known! Consider now the natural structure of the state-transition map, $\tilde{\delta}$ between input-transitions t_k and t_{k+1}, i.e., $\tilde{\delta}(t_{k+1}, t_k, \{(t_j, w_j)\}, (\tau_k, q_k)^n)$, which is the analog of the next-state map, δ, in a finite-state sequential machine, M. By causality, the map $\tilde{\delta}$ can only depend on the single pair $(t_{k+1}\ w_{k+1})$ of the sequence $\{(t_j, w_j)\}$ for a given k. $\tilde{\delta}$ may always be represented as two functions in this single-transition case:

$$\tilde{\delta} = (\tilde{\delta}_\tau, \tilde{\delta}_q)$$

where $\tau^\ell_{k+1} = \tilde{\delta}^\ell_\tau(t_{k+1}, t_k, w_{k+1}, (\tau_k, q_k)^n)$

$$q^\ell_{k+1} = \tilde{\delta}^\ell_q(t_{k+1}, t_k, w_{k+1}, (\tau_k, q_k)^n)$$

$\ell = 1, 2, \ldots, n$.[2] The function $\tilde{\delta}^\ell_\tau$ projects the ℓ-th state transition time, τ^ℓ_{k+1} prior to the current input transition, while the function $\tilde{\delta}^\ell_q$ projects the current state q^ℓ_{k+1}.

For any $\{\tilde{w}\}\varepsilon\,\tilde{W}$, t_i, t_j, $t_\ell \varepsilon T_k$, $\tilde{q}\varepsilon\tilde{Q}$, then

(i) $\tilde{\delta}(t_i,t_k,\{\tilde{w}\},\tilde{q}) = \tilde{q}$

(ii) for any $t_i < t_j$ and $\{\tilde{w}\}\varepsilon\tilde{W}$ such that $\{\tilde{w}\}^1 \equiv \{\tilde{w}\}$ on $[t_i,t_j]$, then $\tilde{\delta}(t_j,t_i,\{\tilde{w}\}^1, q) = \tilde{\delta}(t_j, t_i, \{\tilde{w}\}, q)$.

(iii) whenever $t_i<t_j<t_\ell$, $\tilde{\delta}(t_\ell,t_i,\{\tilde{w}\},\tilde{q}) = \tilde{\delta}(t_\ell,t_j,\tilde{\delta}(t_j,t_i,\{\tilde{w}\},\tilde{q}))$.

Finally, the readout map $\tilde{\lambda}$: $T\times T\times\tilde{W}\times\tilde{Q} \to Z$. #

A more concise definition of a finite-state dynamic system could be given if it were not for our attempt to clarify how the conventional notions of dynamic system and finite-state sequential machine have been reconciled. Actually, the ideas are quite simple. The key concept is that all "spontaneous" state transitions should ultimately be attributable to either the initial state or some input transition; the rate at which such transitions occur is due to any inherent finite delays for any implementation of the machine. Consequently, each "input" can be characterized by

(a) the time of the current input transition, t_k

(b) the new input value (for $t \geq t_k$)

This is denoted as $\tilde{w} = \{t_k,w_k\}$ above. Notice that when there are several input lines, a transition on <u>any one of them</u> is considered a transition time. The state set is also a product space; each "state variable" takes values in the finite set Q. The time-parameter carried with it is the time of the most recent state-transition strictly prior to the current input-transition, and $q\varepsilon Q$ is associated with the value of the state following that state transition; thus elements of $\tilde{Q}$ take the form $(\tau_k,q)^n$, as it may in general be necessary to keep count of a finite number of prior transitions. Again it is important to note that τ_k is always by convention the

map, $\tilde{\delta}$ between input-transitions t_k and t_{k+1}, i.e., $\tilde{\delta}(t_{k+1},t_k,\{(t_j,w_j)\},(\tau_k,q_k)^n)$, which is the analog of the next-state map, δ, in a finite-state <u>sequential</u> machine, M. By causality, the map $\tilde{\delta}$ can only depend on the <u>single pair</u> $(t_{k+1}\ w_{k+1})$ of the sequence $\{(t_j,w_j)\}$ for a given k. $\tilde{\delta}$ may always be represented as two functions in this single-transition case:

$$\tilde{\delta} = (\tilde{\delta}_\tau, \tilde{\delta}_q)$$

where $\tau^\ell_{k+1} = \tilde{\delta}^\ell_\tau(t_{k+1},t_k,w_{k+1},(\tau_k,q_k)^n)$

$$q^\ell_{k+1} = \tilde{\delta}^\ell_q(t_{k+1},t_k,w_{k+1},(\tau_k,q_k)^n)$$

$\ell=1,2,\ldots,n$.[2] The function $\tilde{\delta}^\ell_\tau$ projects the ℓ-th state transition time, τ^ℓ_{k+1} prior to the current input transition, while the function $\tilde{\delta}^\ell_q$ projects the current state due to the current input-transition (t_{k+1},w_{k+1}). It is readily verified that this definition applies to both synchronous <u>and</u> asynchronous machines. Finally, one can appeal to the viewpoint taken by Allen and Gallager (1977) in viewing the (initial) state of a machine to include both "internal registers" and "mass storage"-thus the compensation algorithm <u>itself</u> becomes part of the machine's initial state. In other words, selecting the control algorithm corresponds to choosing an initial data set for the machine. The formulation is sufficiently general to allow that the algorithm execution changes the machine structure itself. It is approximately correct to view the map $\tilde{\delta}$ as the implementation of a machine's "control stru

---

[2] Note that $(\tau_k,q_k)^n$ is shorthand for $(\tau^1_k,q^1_k,\tau^2_k,q^2_k,\ldots,\tau^n_k,q^n_k)$. In general, n may also depend on k, but is bounded above.

ture", and $\tilde{\delta}_q$ as implementing a "data flow structure", in the terminology of computer science.

Returning to the context of the compensation problem, it remains to define the coder and decoder. The following definitions would appear natural:

Definition 2: A dynamic code is a causal mapping $C: \tilde{Y} \to \tilde{W}$. A dynamic decoder is a causal mapping $\mathcal{D}: \tilde{Z} \to \tilde{U}$, where $\tilde{Y},\tilde{U},\tilde{W},\tilde{Z}$ are as in the definitions of a continuous dynamical system and a finite-state dynamical system, respectively. Some of the issues of defining codes of this type have been recently discussed by Ziv and co-workers (1977).

Examples

Example 1: Stabilization of a first-order system. Figure 2 shows an unstable first-order lag, which is to be stabilized (if possible) by a D-type flip-flop. The forward-loop is a dynamical system with time-set $T = [0,\infty)$, characterized by[3]

$$\Sigma = \{R, C(T), R, C(T), R, \phi, r\}$$

where $\| f \|_{C(T)} = \sup_{t\in T} |f(t)|$ takes values in the extended real numbers, and

$$\phi(t,t_0,u(\cdot),x_0) = e^{a(t-t_0)} \cdot x_0 + \int_{t_0}^{t} e^{a(t-\tau-t_0)} u(\tau)d\tau \doteq x(t) \qquad (4.1)$$

and

$$r(t,t_0,u(\cdot),x_0) = x(t) \doteq y(t) \qquad (4.2)$$

The encoder is a threshold function with level θ and binary output, which is sampled at regular intervals $t_k = k\Delta$. The coder input takes values in the space

$$\tilde{W} = \{\{t_k,\hat{w}_k\} | k=0,1,2\ldots;\ \hat{w}_k \in W,\ t_k \in R\}$$

where $W = \{0,1\}$. The coder is then defined by the map $C: C(T) \to \tilde{W}$ where for $y \in C(T)$, $Cy \doteq \{t_k,\hat{w}_k\}$ and

[3] $R \doteq$ real numbers; $C(T) \doteq$ cont. functions on T.

Figure 2(a): Stabilization of an Unstable Lag by a D-Type Flip Flop

system, respectively. Some of the issues of defining codes of this type have been recently discussed by Ziv and co-workers (1977).

Examples

Example 1: Stabilization of a first-order system. Figure 2 shows an unstable first-order lag, which is to be stabilized (if possible) by a D-type flip-flop. The forward-loop is a dynamical system with time-set $T = [0,\infty)$, characterized by[3]

$$\Sigma = \{R, C(T), R, C(T), R, \phi, r\}$$

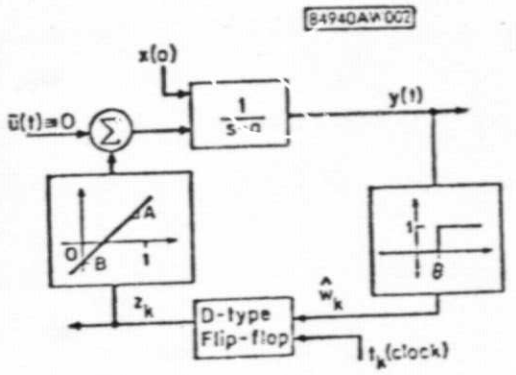
The encoder is a threshold function with level θ and binary output, which is sampled at regular intervals $t_k = k\Delta$. The coder input takes values in the space

$$\tilde{W} = \{\{t_k, \hat{w}_k\} \mid k=0,1,2\ldots;\ \hat{w}_k \in W,\ t_k \in R\}$$

where $W = \{0,1\}$. The coder is then defined by the map $C: C(T) \to \tilde{W}$ where for $y \in C(T)$, $Cy \doteq \{t_k, \hat{w}_k\}$ and

[3] $R \doteq$ real numbers; $C(T) \doteq$ cont. functions on T.

Figure 2(a): Stabilization of an Unstable Lag by a D-Type Flip Flop



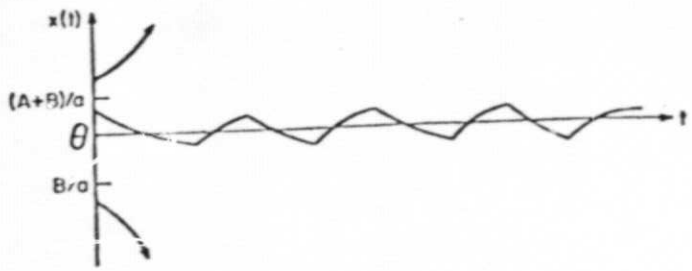


Figure 2(b): Sketch of Typical Responses for Case 2(ii)

T. L. Johnson

FOLDOUT FRAME

$$\hat{w}_k = \begin{cases} 1 & y(t_k) \geq \theta \\ 0 & y(t_k) < \theta \end{cases}$$

The flip-flop is almost the simplest example of a finite-state dynamical system; it is characterized by

$$\Gamma \doteq \{\tilde{W}, \tilde{\tilde{W}}, \tilde{z}, \tilde{Z}, \tilde{Q}, \tilde{\delta}, \tilde{\lambda}\}$$

where $\tilde{W} = Rx\{0,1\}$, $\tilde{\tilde{z}} = Rx\{0,1\}$, and for $Z = \{0,1\}$,

$$\tilde{Z} = \{\{t_k, \hat{z}_k\} | k=0,1,2,\ldots; \hat{z}_k \in Z;\ t_k \in R\}$$

The state-set is $\tilde{\tilde{Q}} = (\tau_k, q_k)$ where $\tau_k \in R$, $q_k \in Q \doteq \{0,1\}$. In this device, the state merely stores the value of the past input, which was always sampled Δ seconds previously, i.e. $(\tau_k, q_k) = (\Delta, \hat{w}_{k-1})$. Thus the state-transition map is implemented by

$$\tilde{\delta}(t_i, t_j, \{t_k, \hat{w}_k\}, (\Delta, q_j)) = \begin{cases} (\Delta, w_{i-1}) & i-j \geq 1 \\ (\Delta, q_j) & i-j=0 \end{cases}$$

and the output map is

$$\mathcal{D}: \tilde{Z} \to C(T)$$

where $\{t_k, z_k\} \to \hat{u}(t)$ is defined by $\hat{u}(t) = Az_k + B$ whenever $t \in [t_k, t_k+\Delta)$ and A, B are real numbers. The design parameters in this feedback system are thus θ, A and B.

The problem of determining which values of

(ii) If $\theta \leq 0$, the system is only stable if $x_0 = A/a$, and then $x(t) \equiv A/a$.

(iii) If $\theta > A/a$, the system is not stabilizable for any x_0.

(iv) If $0 < \theta < A/a$ is chosen properly, a limit cycle is obtained if $0 < x_0 < A/a$; $x_0 = 0$ and A/a are critically stable initial conditions.

Case 2: $B \neq 0$

(i) If $B > 0$, the results are similar to Case 1.

(ii) If $B < 0$, B must be chosen so that $(A+B) > 0$, and $\theta \in [B/a, (A+B)/a]$ leads to a limit cycle about θ. If $(A+B) \leq 0$, results similar to Case 1 are obtained. Initial states $x_0 \in [B/a, (A+B)/a]$ may be stabilized.

Most interesting is Case 2, (ii), and a representative trajectory is sketched in Fig. 2(b). The quantity $-a\Delta$ affects the amplitude of limit cycle oscillations.

The purpose of this example is to illustrate how the general notation applies in a specific synchronous finite-state compensation problem, and to illustrate that indeed stabilization of continuous systems may be achieved with a very economical finite-state compensator which can be synthesized with a threshold device, flip-flop, 2-level supply and gate. The proposed representation was used to conceptualize the problem in this case, but is not in itself a design method. Note that the product space for representing the closed-loop system is immediate, given

merely stores the value of the past input, which was always sampled Δ seconds previously, i.e. $(\tau_k, q_k) = (\Delta, \hat{w}_{k-1})$. Thus the state-transition map is implemented by

$$\tilde{\delta}(t_i, t_j, \{t_k, \hat{w}_k\}, (\Delta, q_j)) = \begin{cases} (\Delta, w_{i-1}) & i-j \geq 1 \\ (\Delta, q_j) & i-j=0 \end{cases}$$

and the output map is

$$\mathcal{D}: \tilde{Z} \to C(T)$$

where $\{t_k, z_k\} \to \hat{u}(t)$ is defined by $\hat{u}(t) = Az_k + B$ whenever $t \in [t_k, t_k+\Delta)$ and A, B are real numbers. The design parameters in this feedback system are thus θ, A and B.

The problem of determining which values of x_0 and $a>0$ yield a Lagrange stabilized system by appropriate choice of θ, A and B has been solved by D. G. Wimpey (1977). Since the system is synchronous, the state-transition maps ϕ, δ may be replaced by one-step recursions, yielding the augmented system[4]

$$\begin{bmatrix} x(t_{i+1}) \\ q_{i+1} \end{bmatrix} = \begin{bmatrix} e^{a\Delta} & (A/a) & (1-e^{a\Delta}) \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} x(t_i) \\ q_i \end{bmatrix}$$

$$+ \begin{bmatrix} (B/a)\ (1-e^{a\Delta}) \\ \operatorname{sgn}\{e^{a\Delta} x(t_i) + a^{-1}(Aq_i+B)(1-e^{a\Delta}) - \theta\} \end{bmatrix}$$

Careful consideration of these equations leads to the following cases:

Case 1: B=0

(i) If $x_0<0$, the system is not stabilizable.

(ii) If $x_0>0$ and $(A+B) \leq 0$, results similar to Case 1 are obtained. Initial states $x_0 \in [B/a, (A+B)/a]$ may be stabilized.

Most interesting is Case 2, (ii), and a representative trajectory is sketched in Fig. 2(b). The quantity $-a\Delta$ affects the amplitude of limit cycle oscillations.

The purpose of this example is to illustrate how the general notation applies in a specific synchronous finite-state compensation problem, and to illustrate that indeed stabilization of continuous systems may be achieved with a very economical finite-state compensator which can be synthesized with a threshold device, flip-flop, 2-level supply and gate. The proposed representation was used to conceptualize the problem in this case, but is not in itself a design method. Note that the product space for representing the closed-loop system is immediate, given Σ and Γ.

Example 2: A Finite-State Dynamical System with Feedback. A simple asynchronous finite-state system, Johnson and Kovacs (1977) with delays in an internal feedback loop is shown in Fig. 3. No general but explicit theory for this class of system is known to the author, yet it is readily represented as a finite-state dynamical system using the proposed framework. Due to space limitations, we shall not spell out the formalisms but only indicate the key ideas. Let $\{t_k, w_k\}$ denote the sequence of input transition times t_k, where $w_k=1$ if $d(t_k^+) \geq \theta$ and 0 if $d(t_k^+) < \theta$; the set $\{t | d(t)=\theta\}$ is assumed to have measure zero, and $d(\cdot)$ is also assumed to have a degree of smoothness such that $|t_{k+1}-t_k| > \Delta_1$. One (minimal) representation is obtained by taking as states

---

[4] $\operatorname{sgn}[f] = \begin{cases} 1; & f \geq 0 \\ 0; & f < 0 \end{cases}$

Finite-State Components for Continuous Processes

FOLDOUT FRAME

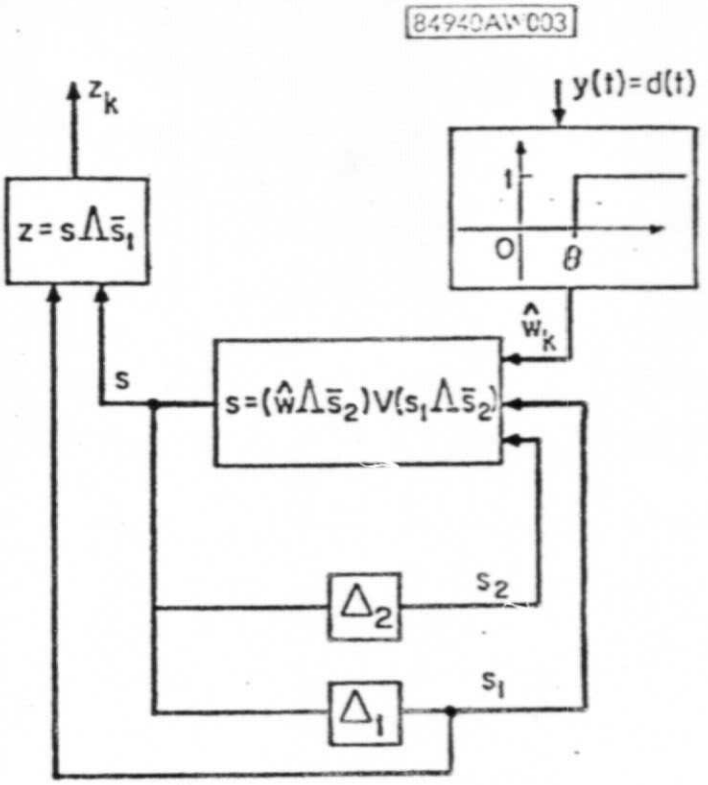


Figure 3: A Finite-State System with Feedback through delays Δ_1, Δ_2

$$q_k^{\ell} = \begin{bmatrix} s(t_k-\tau_k^1)^+ \\ \hat{w}_k(t_k-\tau_k^2)^+ \end{bmatrix} \quad \tau_k = \begin{bmatrix} \tau_k^1 \\ \tau_k^2 \end{bmatrix}; \quad \ell = 1,2$$

where τ_k^1 is the elapsed time since the most

transition will occur at $t_{k-1}-\tau_k+\Delta_2$ and $\tau_k=0$; or (ii) if $s(\tau_{k+1}^+) = 0$, no further changes will occur until t_k and also $\tau_k=0$. If $\hat{w}_{k-1} = 1$, and (τ_k^2) is large a cycle of period $2\Delta_2$ and duty cycle 1 will be propa-

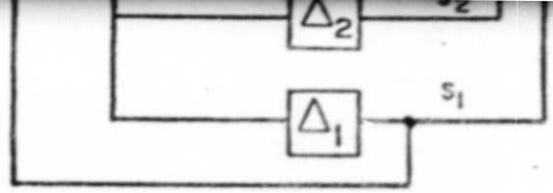


Figure 3: A Finite-State System with Feedback through delays Δ_1, Δ_2

$$q_k^{\ell} = \begin{bmatrix} s(t_k-\tau_k^1)^+ \\ \hat{w}_k(t_k-\tau_k^2)^+ \end{bmatrix} \quad \tau_k = \begin{bmatrix} \tau_k^1 \\ \tau_k^2 \end{bmatrix}; \quad \ell = 1,2$$

where τ_k^1 is the elapsed time since the most recent switching of s for $t \leq t_k$ ($\tau_k = 0$ if s switches at t_k), and $\tau_k^2 = t_k - t_{k-1}$ is the elapsed time since the last input transition. The key issue is to verify that δ is well-defined for one time step i.e., given $(\tau_{k-1}^{\ell}, q_{k-1}^{\ell})$ and $(t_k, \hat{w}_k)$, find $(\tau_k^{\ell}, q_k^{\ell})$. Clearly the new input defines the last state component of q_k. If $\tau_{k-1}>0$, s_1 will switch at $t_{k-1}-\tau_{k-1}^1 + \Delta_1$ and if also $\tau_{k-1}^1>\Delta_2$, s_2 will switch at $t_{k-1}-\tau_{k-1}^1 + \Delta_2$, but by definition, these do not lead to further transitions of s in $(t_{k-1}-\tau_{k-1}^1 + \Delta_2, t_{k-1})$, and thus s_1, s_2 remain constant, until t_{k-1}. Thus for all $\tau_{k-1}^1 \geq 0$, q is known at t_{k-1}^+ (if $\tau_{k-1}=0$, $q_{k-1}=q(t_{k-1}^+)$. From t_{k-1} to t_k, the input has constant value $\hat{w}_{k-1}$, so it is simply a matter of simulating the transitions up to t_k, an interval of length t_k-t_{k-1} and then determining if $\hat{w}_k$ produces an s-transition at t_k. Clearly this can be carried out. In fact, if $\hat{w}_{k-1}=0$ and (τ_k^2) is large, then (i) if $s(\tau_{k-1}^+) = 1$, a single transition will occur at $t_{k-1}-\tau_k+\Delta_2$ and $\tau_k=0$; or (ii) if $s(\tau_{k+1}^+) = 0$, no further changes will occur until t_k and also $\tau_k=0$. If $\hat{w}_{k-1} = 1$, and (τ_k^2) is large a cycle of period $2\Delta_2$ and duty cycle 1 will be propagated, and τ_k may be determined by $(\tau_k^2 + \tau_k^1) \bmod 2\Delta_2$. Note that the system of Fig. 3 is a particular instance of the combined coder-finite state dynamic system portion of the general feedback system shown in Fig. 1.

DISCUSSION AND CONCLUSIONS

The results are still preliminary. The following issues can now be addressed: (1) proving when or if such a representation is canonical; (2) examining the properties of closed-loop dynamics obtained by state augmentation; (3) developing design and simulation methods for such systems; (4) examination of the natural operator algebra on the product space; (5) generalizations to stochastic-finite-state systems.[5]

[5] While technically involved, the generalization of these ideas to the stochastic case should be feasible. It requires synthesis of results on continuous-time stochastic systems, stopping times, and Markov chain theory, which are already well-developed in the literature.

The primary contribution of this note is the a systematic and general method for analyzing the real-time behavior of both synchronous and asynchronous finite-state sequential machines has been discovered, and that this is representation is convenient for analyzing feedback systems. A design method based on this representation has the potential for yielding digital control algorithms which are directly implementable, thus avoiding a tedious process of approximating controllers based on ordinary calculus or, on the other hand, on approximate representations of plant dynamics.

REFERENCES

Allen, J. and R. Gallager. (to be published). Computation Structures.

Bobrow, L. and M. Arbib. (1974). Discrete Mathematics. W. B. Saunders Co., Philadelphia, Pa.

Gallager, R. G. (1968). Information Theory and Reliable Communication. J. Wiley & Sons, New York, N.Y.

Johnson, T.L. and Z. Kovacs. (1977). Asynchronous control of lower leg reflexes via the spinal motor neuron pool. Proc. 1977 IEEE Con. Dec. and Control. (to appear).

Kalman, R. E., P. L. Falb, and M. Arbib. (1969). Topics in Mathematical Systems Theory. McGraw-Hill, New York.

Padulo, L. and M. Arbib. (1970). System Theory: A Unified Approach to Continuous and Discrete Time Systems. W. B. Saunders

REFERENCES

Allen, J. and R. Gallager. (to be published). Computation Structures.

Bobrow, L. and M. Arbib. (1974). Discrete Mathematics. W. B. Saunders Co., Philadelphia, Pa.

Gallager, R. G. (1968). Information Theory and Reliable Communication. J. Wiley & Sons, New York, N.Y.

Johnson, T.L. and Z. Kovacs. (1977). Asynchronous control of lower leg reflexes via the spinal motor neuron pool. Proc. 1977 IEEE Con. Dec. and Control. (to appear).

Kalman, R. E., P. L. Falb, and M. Arbib. (1969). Topics in Mathematical Systems Theory. McGraw-Hill, New York.

Padulo, L. and M. Arbib. (1970). System Theory: A Unified Approach to Continuous and Discrete Time Systems. W. B. Saunders Co., Philadelphia, Pa.

Willems, J. C. (1972). Dissipative dynamical systems. Parts I, II. Archive for Rational Mechanics and Analysis., 45, 321-393.

Willems, J. C. and S. K. Mitter. (1971). Controllability observability, pole allocation, and state reconstruction. IEEE Trans. Autom. Control., AC-16, 6, 582-596.

Wimpey, D. G. (1977). Finite-state compensation of a first-order system. (M.I.T. project report).

Ziv, J. and A. Lempel. (1977). A universal algorithm for sequential data compression. IEEE Trans. Infor. Theory., IT-23, 3, 337-343.

This research was performed at the M.I.T. Electronic Systems Laboratory with support provided by the Air Force Office of Scientific Research, Grant 77-3281 and NASA Ames Research Center NGL-22-009-124.