Annual Technical Report

# DISCRETE CONTROL OF CONTINUOUS PROCESSES

Contract F49620-80-C-0002 (as amended)

10/1/80-9/30/81

To: Directorate of Mathematical and Information Sciences
Air Force Office of Scientific Research
Building 410
Bolling Air Force Base, D.C. 20332

From: Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Room 35-205B
Cambridge, Massachusetts 02139

Acting Contract Monitor: Dr. Bernard Epstein

Principal Investigators:

Timothy L. Johnson, Visiting Research Scientist and Lecturer
Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

Martin E. Kaliski, Professor
Northeastern University
Department of Electrical Engineering

CONTENTS

F. T. L. Johnson, "On Feedback Laws for Robotic Systems", Proc. 8th IFAC World Congress, Kyoto, Japan, August 1981.

G. M. E. Kaliski, "Essentially Continuous Functional Real Coders", unpublished report.

H. D. G. Wimpey, "Finite-State Control of Discrete-Time Continuous Processes", M.I.T., Dept. of Electrical Engineering and Computer Science, Nov. 1981 (preliminary copy).

## II. Statement of Research Objectives

The research objectives, as stated in the amended contract, are:

(a) Investigate real language extensions

(b) Develop control design methodology

(c) Develop asynchronous coding

(d) Study the representation of states (in asynchronous hybrid systems)

(e) Formulate improved controller design procedures for synchronous hybrid systems

## III. Status of the Research Effort

The objectives of this phase of the proposed research on discrete control of continuous processes have been substantially achieved. This annual technical status report is divided into two unequal segments, dealing with research at M.I.T. and at Northeastern University. The second and shorter segments constitute the final report of the Northeastern University Subcontract, under the direction of Professor Kaliski; however, only 4 weeks' effort was budgeted due to his sabbatical leave of absence.

### A. Research at M.I.T.

#### 1. Coder Synthesis for Synchronous Systems

We have already motivated our study of coders in the context of digital control systems; in fact, in many cases the entire system in the feedback loop may be viewed as a coder. To recapitulate, we viewed a coder simply as a map

$$G: \quad R^{p*} \to W$$

where $R^{p*}$ is the free monoid generated by $R^p$ (p-dimensional real Euclidean Space) and W is an alphabet (a finite set). Some examples of coders are (memoryless) A/D convertors (or more generally quantizers) and delta modulators. In a more theoretical view we may wish to model our digital controller as a quantizer followed by a transducer (such as a finite-state transducer or a pushdown transducer, etc.) with the following implications: That all coders may be decomposed into the cascade connection of a memoryless quantizer followd by a transducer, and that the most general coder may be represented as a quantizer in cascade with a Turing transducer. Simple counter-examples show, however, that not all coders may be realized this way. Our research has been directed toward establishing general

canonical realizations for coders.

We have identified three basic approaches for the development of the realization theory which we loosely term the analytic, the algebraic and the linguistic (or syntatic) approaches. The former approach involves obtaining physical realizability constraints for Kaliski's general coder decomposition theorem [1]; this viewpoint is currently being pursued although no results have yet been obtained. We summarize here the results of the last two approaches. Here we have begun to develop a hierarchy of coders of increasing complexity, motivated by the already existing hierarchy of automata.

The success of the algebraic approach hinges on the definition of appropriate right-congruence relations on $R^{p*}$. We have defined a generalized Nerode equivalence relation $\equiv_\theta$ for C which may have finite index even though the minimal realization of C is not finite-state. The relation $\equiv_\theta$ is defined as follows: $u \equiv_\theta v$ iff $C_{u,\theta}(y) = C_{v,\theta}(y)$ for all $y \in R^{p*}$, where the shift-conjugate function $C_{u,\theta}: R^{p*} \to W$ is defined below.

$$C_{u,\theta}(y) = C\tau^{\theta}(y) \qquad y \in R^{p*} \text{ with } 0 \leq \ell(y) < \ell(\theta)$$

$$C_{u,\theta}(y) = C\tau^{u}(y) \qquad y \in R^{p*} \text{ with } \ell(y) \geq \ell(\theta)$$

$\theta \in R^{p*}$ is a parameter. Then a coder is defined to be shift-finitary if the minimum length $\theta \in R^{p*}$ for which $\equiv_\theta$ has finite index is one (with the implication such that a $\theta$ exists). Special cases are as follows:

(i) The index of $\equiv_\theta$ is 1; C is said to be shift unitary

(ii) $\equiv_\theta$ has finite index and the minimum length of $\theta$ is zero; C is said to be finitary

(iii) $\equiv_\theta$ has unit index and the minimum length of $\theta$ is zero; C is said to be unitary.

The most general coder in this hierarchy, the shift-finitary coder, has a minimal hybrid state-space $R^n x Q$, where $n = \ell(\theta)$ and Q is a finite set. The structure of the coder is shown in Figure 1. In the case of a shift-unitary coder, the finite-automaton in Figure 1 has only one state. Clearly, the shift-register of Figure 1 is not present in the realization of unitary and finitary coders; these coders are finite-state realizable, and may be decomposed into the cascade connection of a quantizer followed by a finite-state transducer.

In the linguistic or syntatic approach we view the coder as an acceptor of a real language L, which is a subset of $R^{p*}$. Here we take W to be the two element set {0,1}, and C accepts L as follows:

$$C(y) = \begin{cases} 1 & \text{if } y \varepsilon L \\ 0 & \text{otherwise} \end{cases}$$

The most general grammars that we have studied for generating real languages are the real context-free grammars; these are a generalization of the real context-free grammars of Lemone [2]. We have generalized the notion of a pushdown automaton to accept real-valued signals; this new "machine" has a hybrid control module (state-space $Q_1 x R^n$ for $Q_1$ an alphabet) and a stack with a hybrid stack set $Q_2 x R^p$, with $Q_2$ an alphabet. Since context-free languages are nondeterministic in general, this acceptor is also nondeterministic.

Since the continuous-state part of this pushdown coder is infinite

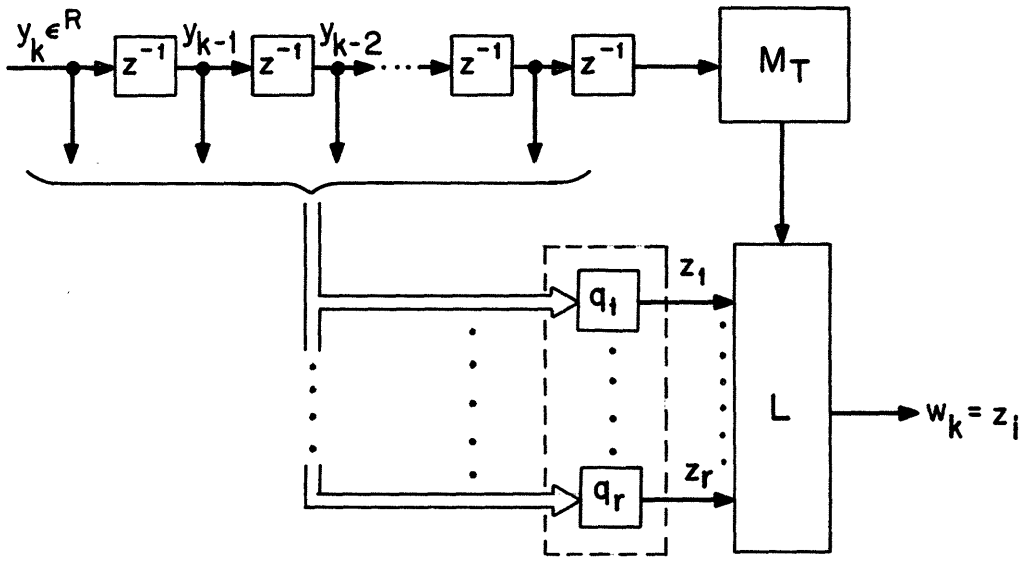


Figure 1.

Realization of a shift-unitary coder. L is a multiplexor, $M_T$ is a finite automaton and $q_i : R^n \to Z$ is a quantizer.

dimensional, the coder may be considered unrealizable, unless of course an equivalent finite-dimensional realization can be found. In principle this can always be done (according to Kaliski's general coder realization theorem mentioned above), but in practice this may only be feasible for special examples. A subset of the real context-free languages are the "bounded real embedding " (BRE) context-free languages, where the acceptors for these languages have a finite stack set. These languages may thus be viewed as the largest general class of real context-free languages which represent realizable coders (although for the purposes of coding we should restrict ourselves to the deterministic subset of these languages).

The right-linear real languages* are a proper subset of the deterministic BRE context-free real languages. Here there is no stack at all in the acceptor realization; the coder is finite dimensional (with a state-space $QxR^n$, Q an alphabet; see Figure 2). An acceptor of a right-linear real language is only finite-state realizable if the grammar is of order one (the order of a grammar was defined by Lemone [2]). In fact the coders which accept languages generated by first order grammars are precisely those coders referred to above which may always be decomposed into the cascade connection of a memoryless quantizer and an automaton.

The syntactic approach to coder synthesis has clearly been illuminat-

---

*Productions of a right-linear real grammar take the form

$$A \to wB \qquad w \in \phi$$

or $$A \to B$$

where A, B are nonterminals, w is a string of terminal symbols of length n, and $\phi \subset R^n$ is a replacement set.

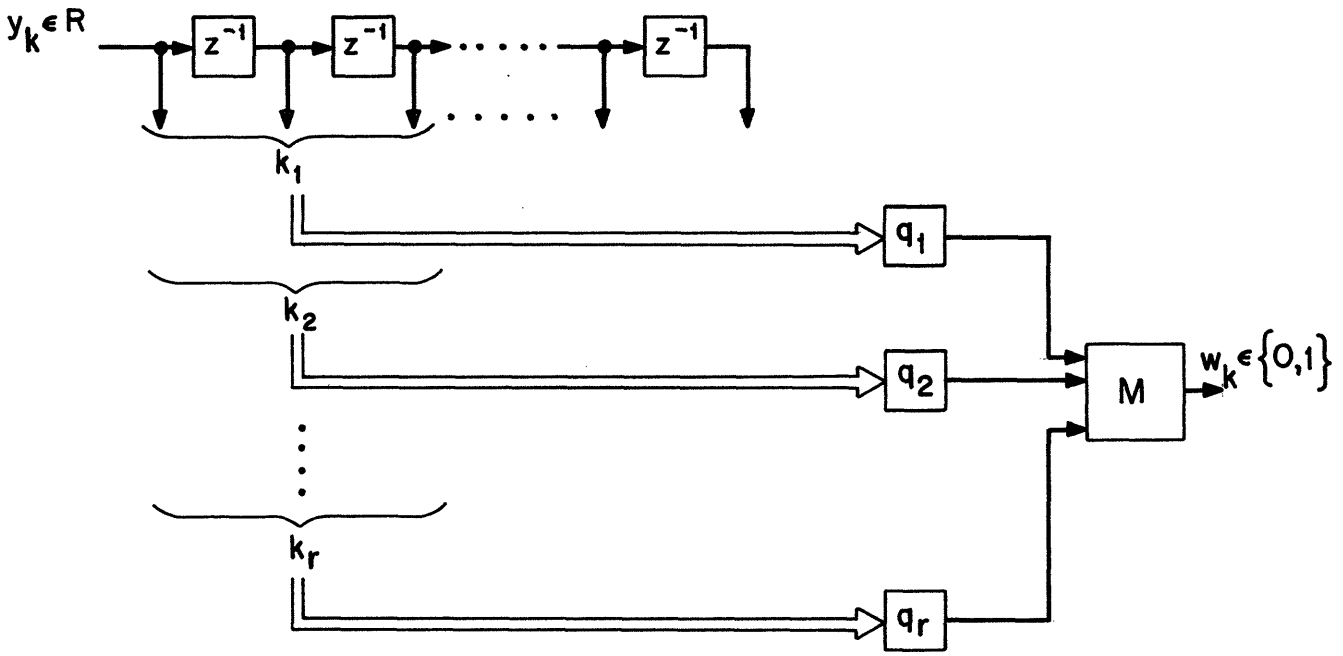


Figure 2.

Structure of a Coder Specified by a Right Linear Real Language

ing. We note that our extension from finitary to shift-finitary by the addition of a shift-register appears natural in the context of right-linear real languages (any language describing a shift-finitary acceptor is right-linear). It would be premature though to say that the natural structure of coders are those corresponding to acceptors. Further developments are needed along the lines of the analytic approach mentioned above and an extension in the syntatic approach to more general phrase-structure grammars.

2. Discrete Control of Synchronous Systems

The design of a digital controller for a continuous-state system usually involves the design of three subsystems: a coder, a control algorithm (e.g., to reside in a digital computer) and a decoder. If the coder and decoder are allowed to be dynamic systems and the control algorithm is modelled as an automaton, then it is clear that the digital controller may be viewed as a coder connected directly to a decoder. The structure theory we have developed for coders, some of which may be extended to decoders, then provides general models for the controller. An interesting viewpoint yet to be fully explored is to view the controller as a translator T of real-valued sequences into real-valued sequences ($T; R^{p*} \to R^{m*}$) using syntax-directed schemas [3] to describe T (these consist of pairs of real grammars, one called an input-grammar and the other called the output grammar).

In the case where the coder and decoder are constrained to be finitary, the entire controller may be viewed as a finitary coder $C: R^{p*} \to W$ where W is a finite subset of $R^m$, the plant input set. A finitary controller may thus always be decomposed into the cascade connection of a memoryless

quantizing coder, a finite-state transducer and a memoryless decoder, the latter being simply a realization of a mapping from the finite output alphabet of the transducer to W. All of the procedures discussed in the literature for designing finite-state controllers concentrate on designing the transducer; the design of the coder and decoder are omitted. Also, since these schemes involve either approximation or learning via ad hoc adaptive algorithms, the theories provide no performance guarantees.

We have developed an algorithm for designing finitary coders for a very general class of nonlinear deterministic discrete-time systems: those that are piecewise continuous. The algorithm provides an effective procedure for computing the coder, transducer and decoder for the following problem formulation: Given the plant F, find a finitary feedback controller that will drive the plant state, starting initially in $X_o$, to the target set Z in a finite number of steps. This is the finite-horizon problem; in the infinite horizon problem, the plant state is required to remain in Z for an infinite period of time once it has reached Z. In both of these cases the regulator initial state is fixed and depends on $X_o$. This has been termed the synchronous control problem by Gatto and Guardabassi [4] and the weak regulator problem by Sontag [5].

The controller is designed in two steps: First an aggregate finitary model of the plant is obtained, and second, a finitary controller is designed for this aggregate model. The aggregate model has the following properties:

(i) it is a nondeterministic finite-state system

(ii) the states of the model are in one-to-one correspondence with the blocks of a finite partition P of the plant state space X

and (iii) the set of possible states that the model may be in at any time k, viewed as a subset of X, always contains the actual plant state at time k.

It is the task of the designer to choose the partition P. Once this partition is chosen, the aggregate model may be completely specified. At this stage in the design, the only requirement on the choice of P is that the aggregated sets $X_o$ and Z--specified new in terms of blocks of P--still provide an acceptable aggregate control problem formulation (e.g P is well-suited to the approximation of $X_o$ and Z).

The solution of the aggregate control problem requires the design of a finite-state controller for a nondeterministic finite-state plant. The problem formulation now appears to have similarities with the usual "unknown but bounded" stochastic control problems (see Sira Ramirez [6]) and the control problems for deterministic finite automata (Gatto and Guardabassi [4]), although neither of the solutions to these problems are directly applicable.

Since the states of the controller may be viewed as state-estimates of the aggregate model, the specification of P also provides us directly with the coder (since there are only finitely many different state estimates). The remainder of the controller is designed in three stages (for the infinite horizon problem). The first part involves the computation of all those state-estimates contained within Z for which there exist control inputs, guaranteeing that future state estimates have the same property. Specification of these admissable control inputs defines a finite partition on the plant input space, once again because the set of

all possible state-estimates is finite, and this partition partially specifies the decoder. The second part involves sequentially identifying all those state estimates that are controllable in finite time to the estimates computed already, until a state-estimate is obtained that contains $X_o$. Again, the admissable control inputs define a finite partition on the plant input space. The design is completed by selecting from each of the blocks of the input space partition (obtained in the first two parts of the design procedure) a representative input. This completes the design of the decoder.

The straightforward application of this algorithm has the following potential disadvantages:

(i) A controller may not exist for a particular partition P

(ii) If P has a large number of blocks, it may be impractical to compute the controller because it will have too many states.

To keep the design feasible, the number of blocks in P must be kept as small as possible. However with coarser partitions it becomes unlikely that a controller may be found. To overcome this dilemma, we have developed a hierarchical design procedure, wherein the controller is computed in stages. A two stage design proceeds as follows (the extension to an n-stage design is obvious). A coarse partition $P_1$ is chosen, together with a "large" target set $Z_1 \supset Z$, and a controller is designed for F, $X_o$ and $Z_1$, based on $P_1$. Then, a refinement of $P_1$; called $P_2$, is chosen, and a controller is designed for F, $Z_1$, and Z. Note that if at either stage no controller can be found, it becomes necessary to try to finer partitions and/or larger target sets. Design experience with actual systems

will clarify what choice may be appropriate. At each stage in the hierarchical design procedure, the controller with the fewest* states is sought by choosing the coarsest partition possible. At each stage then we are dealing with a nondeterministic automaton as an aggregate model of the plant, which has a state set which is not too large to make the controller design impractical. Details of these results appear in Appendix H.

3. Analysis of Asynchronous Feedback Systems

The most important phenomenon encountered in asynchronous hybrid systems which does not occur in synchronous systems is sliding mode behavior--"infinitely fast" switching which may occur along a discontinuity in the state transition function and which results in a trajectory shape which could not occur in the absence of the discontinuity. Another phenomenon is that the class of systems which admit finite-dimensional realizations is less pervasive in practice than for synchronous systems.

The problem of representing sliding mode behavior in asynchronous hybrid systems arose even in the restricted context of diced systems (Appendix C) in the form of a condition that the sequence of switching times for such a system be "asymptotic"--i.e. have an infinite limit. We have constructed a counter-example of a diced system which is not asymptotic (included in the continuation proposal for this research). This shows that there are cases where sliding mode behavior may achieve exact equilibrium in finite time; however there are other cases of sliding where this does not occur. In such non-asymptotic cases, the use of switching-times as hybrid state variables is not sufficient to fully characterize the behavior of a diced system.

This paradox implies that care must be taken in developing a realization theory for asynchronous (continuous-time) hybrid systems. One possibility is to consider a class of realization which enforces a separation of control and that flow at inputs and outputs and to place conditions on the maximum switching rate of control signals. Many problems are known to arise when both control and dtat are combined on a single channel (e.g. Witsenhausen's counterexample in stochastic control). Some examples of hybrid systems which arise naturally in robotics are given in Appendix F.

An important class of applications which can be represented as asynchronous hybrid systems arise in multitasking systems where the tasks involve interaction with continuous processes (timing processes). It has been shown (Appendices D,E) that under certain conditions such systems admit a synchronous hybrid representation. These conditions place a limit on the number of tasks (to ensure finite-dimensionality), on the minimum delay in switching tasks, and on the degree of continuity of the continuous processes. An example of representing a temperature-control system in this manner has been provided in Appendix E.

B. Research at Northeastern University

Much of our work is based upon the key decomposition result reported in [ 1 ]. Specifically we cite the following Theorem: Let C: $R^* \to [0,1]$ be an arbitrary coder. Then for any $n \geq 1$, C may be decomposed as $C = C_2 \cdot C_1$ where $\cdot$ is functional composition. $C_1$: $R^* \to R^n$ and $C_2$: $R^n \to [0,1]$. Further, $C_1$ is realizable by a finite-dimensional discrete-time system of dimension $n$. Although such a result is primarily only of theoretical interest it serves as a starting point for relating properties of $C_1$ and $C_2$ to those of C and conversely.

1. Well-Behaved Coders from the Input/Output Viewpoint

The research undertaken here sought to identify and, if possible, make rigorous, the notion of a physically well-behaved coder. This notion is an intuitive one, e.g. the coder defined by

$$C_\alpha(u_1 \ldots u_k) = \begin{cases} 0 & \text{if } u_1 + \ldots + u_k \geq 0 \text{ or if u is null} \\ 1 & \text{otherwise} \end{cases}$$

is intuitively well-behaved, but the coder $C_\beta$ defined by

$$C_\beta(u_1 \ldots u_k) = \begin{cases} 0 & u = \lambda, \text{ the null string} \\ 0 & u_1 \ldots u_k \text{ is a palindrome} \\ 1 & \text{otherwise} \end{cases}$$

is not. Clearly some implicit notion of continuity is involved here, although we must be careful in using the term "continuity", as the range of the coder is discrete. We developed several potential definitions of "well-behaved" during this research period that embody these concepts. Further research should try to relate these various definitions to one another and to the more formal algebraic/topological approaches already explored. We give some of these definitions below

Definition 1: C is well-behaved at $u_1 \ldots u_k$ if $\exists\ \varepsilon > 0$, dependent upon $u_1 \ldots u_k$, such that if $v_1 \ldots v_k \in R^*$ obeys $\sum_{i=1}^{k} |u_i - v_i| \leq \varepsilon$ then $C(u_1 \ldots u_k) = C(v_1 \ldots v_k)$.

Definition 2: C is strongly well-behaved of order $k \geq 1$ at $u_1 \ldots u_k$ if $\exists\ \varepsilon_L > 0$, dependent upon both $u_1 \ldots u_k$ and L, such that if $v_1 \ldots v_k \in R^*$ obeys $\sum_{i=1}^{k} |u_i - v_i| \leq \varepsilon_L$ and $w_1 \ldots w_p \in R^*$ is arbitrary, $p \leq L$ then $C(u_1 \ldots u_k\ w_1 \ldots w_p) = C(v_1 \ldots v_k w_1 \ldots w_p)$. We say C is strongly well-behaved at $u_1 \ldots u_k$ if it is strongly well-behaved of order L at $u_1 \ldots u_k$ for all $L \geq 1$ and $\varepsilon^* = \min_L \{\varepsilon_L\} > 0$.

Note that this latter concept embodies the notion that "close together strings" (within $\varepsilon^*$) of the same length map to the same values regardless of what is appended onto their ends. The values, of course, depend upon what is appended. If C is strongly well-behaved at $u_1 \ldots u_k$ then the

$\varepsilon^*$-neighborhood of $u_1 \ldots u_k$ consists of length k strings that are Nerode equivalent to $u_1 \ldots u_k$ (as well as other strings, possibly, of length different from k). Interpreting, correctly, "well-behaved" (Def. 1) as strongly well-behaved of order 0, we see that each of these properties are successively more stringent.

We posit two other difenitions that follow a somewhat different tack.

Definition 3: C is causally well-behaved of order $\varepsilon > 0$ if whenever $C(u_1 \ldots u_{k_1}) = C(v_1 \ldots v_{k_2})$ then if $w_1 \ldots w_p$ in $R^*$ obeys $|w_1| + \ldots + |w_p| \leq \varepsilon$ then $C(u_1 \ldots u_{k_1}, w_1 \ldots w_p) = C(v_1 \ldots v_{k_2}\ w_1 \ldots w_p)$. If the above is true for any $\varepsilon > 0$ then we say that C is causally well-behaved.

Note that all of the above definitions are "independent" of the decomposition maps $C_1$ and $C_2$ discussed earlier. They are suggested to indicate the flavor of the thinking undertaken here, and must be viewed as "first efforts".

2. Well-Behaved Coders from the Decomposition Viewpoint

A second area of research involved the problem of "behavior" attacked via the decomposition maps $C_1$ and $C_2$. The one-dimensional case (n=1), being general, was studied. Appendix B to this report details a theory of "good behavior" for the map C, oriented about the constraints of boundedness and Lipschitz continuity. We thus do not discuss the details of this here. Work during our first contract year sought to characterize intuitively well-posed "threshold" maps $C_2$, and such work was reported last year. Some effort was made during the current contract year to topologize $\{0,1\}^*$ so as to generalize the bounded uniform Lipschitz constraint to C directly--its preliminary nature does not justify its inclusion here.

3. "Similarity" of Coders and Related Notions

The notion of similarity of coders $C_\alpha$ and $C_\beta$ is an important one. As seen elsewhere in this report many synthesis results for finitary and other types of coders have been developed.
Knowing that a given coder is "similar" to one of these "canonical" forms is useful, for, within the allowable tolerance of "practical realizations" it may be sufficient to realize the similar, canonical coder in lien of the given one. Furthermore placing a metric on coders may allow us to appropriate arbitrary coders C: $R^* \rightarrow \{0,1\}$ by (sequences of) coders with finite domain ([ 2 ] has pursued a similar approach using formal grammars)

For these reasons a brief study of coder sililarity was undertaken, motivated by work in pattern recognition theoretic similarity ([ 7 ]). By positing the definition that two coders $C_\alpha$ and $C_\beta$ are similar if their associated languages $L(C_\alpha)$ and $L(C_\beta)$ are similar as sets of strings of real numbers (recall that for C: $R^* \rightarrow \{0,1\}$ a coder $L(C) = C^{-1}\{1\}$), the problem can be reduced to exploring similarity measures amongst subsets of $R^*$. These can be generalized from notions already developed in finite set similarity theory or from similarity measures of singleton sets (i.e. between strings).

4. Asynchronous Coders...The Underlying Framework

Asyncrhonous systems must explicitly incorporate the notion of time--for it is at irregular, often event-driven times that transitions can take place, etc. Several approaches to this incorporation are possible. One method involves explicitly adding a time variable to the input and induced output sequences of the coder map. This approach will be explored in

future research and as such will not be presented here. The second involves eliminating, at least formally, the notion of discrete time and working with coders whose domains are functions, instead of sequences. Appendix G to this report examines such a construct called a functional real coder. We chose the range to be real to simplify the overall flow and to better isolate the theoretical constructs needed. This appendix attempts to generalize the fundamental notion of unitary coder to the functional coder case and shows that an "extra" condition, that of essential continuity, must be introduced.

5. Concluding remarks: Software Realizations/Implementation of Coder Concepts

Many useful algorithms constructs have been developed over the past two years. Although the work reported above was theoretical, the potential for syntax-directed, linguistic models for coders can be made a reality. Further research can, and should be undertaken, to help completely automate coder design for a wide class of coder structures.

C. References

[1] 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.

[2] Lemone, K., "Languages Over the Real Numbers", Ph.D. Thesis, Dept. of Math., Northeastern University, Boston, Mass., 1979.

[3] Aho, A.V. and Ullman, J.D., The Theory of Parsing Translation, and Compiling, Vol. I: Parsing, Prentice Hall, 1972.

[4] Gatto, M. and Guardabassi, G., "The Regulator Theory for Finite Automata", Information and Control, V. 31, N. 1, 1976.

[5] Sontag, E. D., "Nonlinear Regulation, the Piecewise Linear Approach", IEEE Trans. on Auto. Control, Vol. AC-26, No. 2, April, 1981.

[6] Sira Ramirez, H.J., "Set Theoretic Control of Large Scale Uncertain Systems", M.I.T., Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, May 1977.

[7] Lemone, K., "Similarity Measures Defined Over Sets of Strings of Real Numbers", submitted for publication to IEEE Trans. of Pattern Recognition and Machine Intelligence, 1981.

IV. Publications and Reports

See citations of Appendices A-H in the Table of Contents of this report.

V. Personnel

The personnel participating in this project have been Dr. T. L. Johnson, Professor M. E. Kaliski, and Professor D. G. Wimpey, whose doctoral research has been completed during this contract.

VI. Interactions

Regular meetings (and correspondence, with Professor Kaliski, during his sabbatical leave in France) have been held throughout the course of this research, and a strong interaction has been maintained amont the participants.

In June, 1981, we initiated an Invited Session on Discontinuous Processes at the Joint Automatic Control Conference held in Charlottesville, Va., which also provided a unique opportunity for discussion among workers in this field of research. The results of Appendix A were presented here. The results of Appendices B-F have or will be also presented at conferences according to the citations given.

Professor Kaliski had the opportunity to discuss the results of this research with French scholars and students during his sabbatical leave in Paris, and to explore the relationship to Petri-net methods used there with asynchronous control representations explored in this research.

## VII. New Discoveries, Inventions and Specific Applications

The coder realization theories presented in Appendix A lead to practical synthesis algorithms for an important class of devices. Comparable algorithms already find very wide use in analog filter and linear circuit design and in digital circuit design, and we recommend that the development of algorithms for automatic synthesis of coders and decoders be pursued as an important technical advance.

Algorithms for feedback control design have also been suggested in Appendix H. While these results are more preliminary, they should definitely be pursued because they represent a synthesis method for global nonlinear feedback laws, and no simple or effective general solution to this problem are currently available.

Potential applications to robotics and multitasking computer operating systems have been described in Appendices E and F.

## VIII. Appendices

# Appendix A

## 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 |
| $\varepsilon$ | set membership symbol |
| $\delta$ | next-state mapping of a finite state system |
| $\beta$ | readout mapping of a finite state system |
| $\tau$ | right shift transformation |
| $\sigma$ | left shift transformation |
| $\Lambda$ | the empty string |
| $\phi$ | 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 $\Lambda$ 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:

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

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

**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 $\sigma^0$ 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

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

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

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

$$C(\Lambda) = 1$$

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

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

$$C(\Lambda) = w_o$$

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

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

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, $W = \{w^1,\ldots w^m\}$
$\delta: Q\times Y\to Q$ is the next-state function given by
$\delta(q^i,y) = q^j$ if $y\in A_{ij}\subset Y$
$\beta: Q\to W$ 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 $\delta$ 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

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

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

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

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

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

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

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 $M = (\overline{C}, V, \overline{W}, \overline{\delta}, \overline{\beta})$ 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) $\overline{\beta}$: $\overline{C} \to W$ via

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

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

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

Now,

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

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 $m \geq 0$,

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

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 $\theta$ 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.

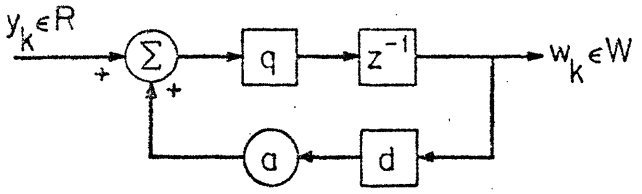


Figure 1 Quantizer q with hysteresis. q: R → W and d: W → R

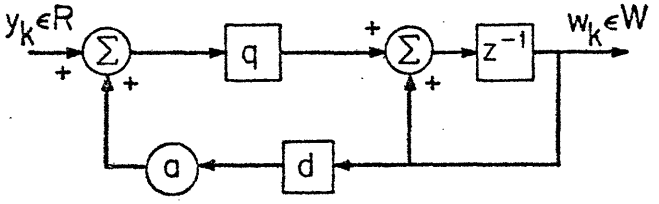


Figure 2 Differential quantizer
q: R → W and d: W → R

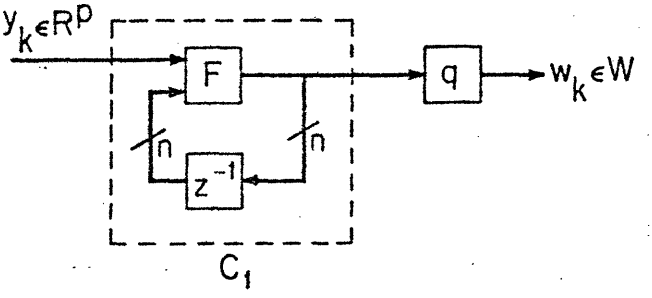


Figure 3 General coder decomposition
$C_1$: $R^{p^*} \to R^n$, q: $R^n \to W$

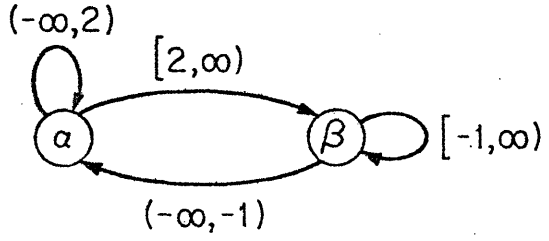


Figure 4 Finite-state realization of the quantizer with hysteresis

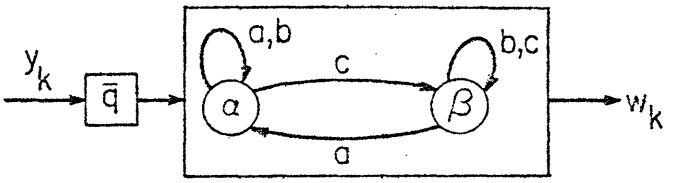


Figure 5 Decomposition of coder of Figure 4
$\bar{q}$: R → {a,b,c} via

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

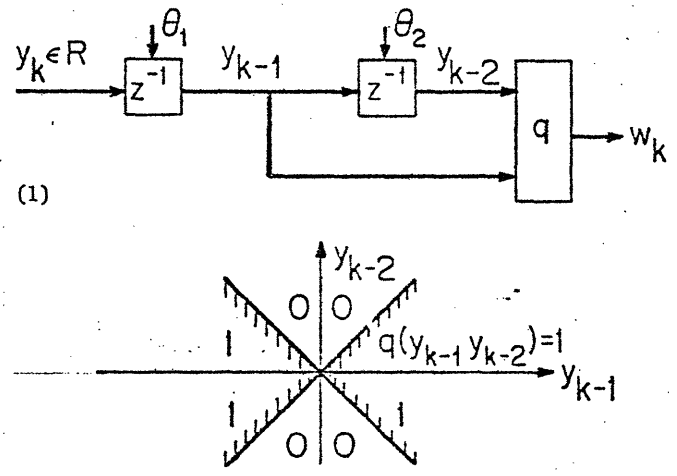


Figure 6 Example of shift-unitary coder
(1) Coder decomposition
(2) q: $R^2 \to \{0,1\}$

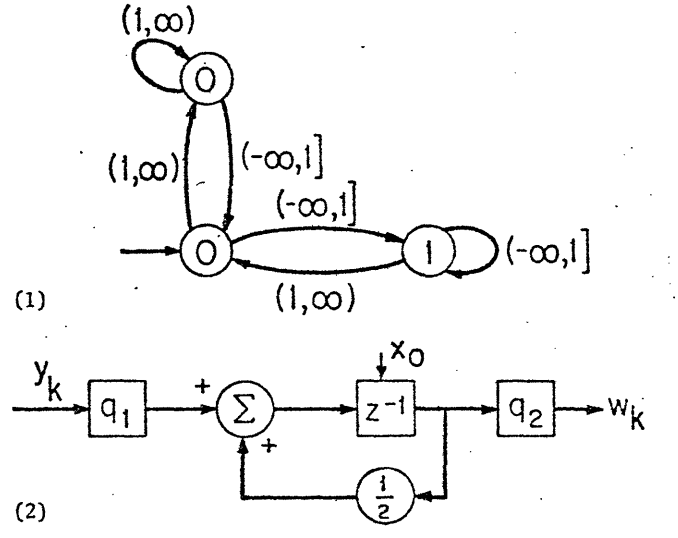


Figure 7 A finitary coder
(1) Specification as a TFA
(2) Realization in the form of Figure 3:
$q_1$: R → {-1,1} via

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

$q_2$: R → {0,1} via

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

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

ABSTRACT

This paper examines certain aspects of the realizability of input/output maps f: $R^* \to R$ by one-dimensional non-linear discrete-time systems, where $R^*$ is the free monoid on the reals R, under the operation of string concatenation. Realizations are sought which are bounded and/or Lipschitz. A natural connection between such constrained realizations and the string-processing properties of the input/output mapping is shown to exist. These results have further implications for the synthesis of general analog/digital coders and decoders which find uses in computer control system interfaces.

Appendix B

# ON INTRINSIC LIPSCHITZ CONSTANTS AND BOUNDS IN NONLINEAR ONE-DIMENSIONAL DISCRETE-TIME REALIZATIONS OF INPUT/OUTPUT MAPS

by

Martin E. Kaliski
Department of Electrical Engineering
Northeastern University
Boston, MA 02115 USA

"This work was supported in part by the United States Air Force Office of Scientific Research (AFSC); United States Air Force Contract F49620-80-C-002, Directorate of Mathematical and Information Sciences."

1982 ACC
Regular Paper

## I. Introduction

The purpose of this paper is to explore certain aspects of the realizability of maps f: $R^* \to R$ by one-dimensional discrete-time systems, where $R^*$ is the set of all finite-length sequences of real numbers, including $\lambda$ the null string. The discrete-time systems considered are of the following form:

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \ldots \tag{1}$$

$x_0$ the initial state

where F: $R^2 \to R$ is the state-transition map.

By the realizability of f we mean that f is the input/output map associated with (1), i.e. there exist F and $x_0$ such that

$$f(\lambda) = x_0$$

and, for $u_0u_1u_2\ldots u_k$ arbitrary, $k \geq 0$,

$$f(u_0\ldots u_k) = F(\ldots(F(F(x_0, u_0), u_1), u_2), \ldots, u_k)$$

We consider below several aspects of the realizability problem:

(A) Realizability by systems wherein F obeys no other constraint than being well-defined.

(B) Realizability by systems wherein F is bounded, i.e. for some $B_0 \geq 0$, $-B_0 \leq F(\underline{v}) \leq B_0$ for all $\underline{v}$ in $R^2$.

and

(C) Realizability by systems wherein, in addition to the boundedness constraint, F obeys a Lipschitz condition for some $L_0 \geq 0$, i.e. for all $\underline{v}$, $\underline{w}$ in $R^2$.

$$|F(\underline{v}) - F(\underline{w})| \leq L_0 \, \|\underline{v} - \underline{w}\|$$

We will take the norm above to be any fixed norm on $R^2$.

This paper will develop several general theorems yielding both necessary and sufficient conditions for the realizability of maps f by systems of types A, B, and C, above. Note that as no assumption is made here about f being onto all of the results below hold even if range(f) is a finite subset of the reals.

The problem of realizing input/output maps by (nonlinear) discrete-time systems is, in of itself, not a new one. Studies of this type in the mathematical systems theory literature (e.g. Kalman, Falb, and Arbib, 1969) have been available for several years. What is new, we feel, in the aspects of the realizability problem considered here, are the following two viewpoints.

First, that such natural regularity requirements as boundedness and, "Lipschitzness" lead to input/output properties expressible as string-processing requirements. Second, that this one-dimensional problem is actually sufficient to represent higher-order finite-dimensional problems as well. This is based on our earlier work (Kaliski and Lemone, 1980) dealing with the realization of real acceptors/coders by one-dimensional systems.

## II. The Basic Result: Well-Posed State Transition Functions

Let $f: R^* \to R$ be a given input/output (I/O) map.

Theorem 1: f is realizable by a system of the form (1) if and only if f obeys the following property[1]:

for all $u_0 \ldots u_\ell$, $v_0 \ldots v_m$ in $R^*$, r in R,

$$f(u_0 \ldots u_\ell) = f(v_0 \ldots v_m) \Rightarrow f(u_0 \ldots u_\ell\ r) = f(v_0 \ldots v_m\ r)$$

(Note: Either $u_0 \ldots u_\ell$ or $v_0 \ldots v_m$ may be null in the above statement.)

Proof: That any map realized by the system (1) obeys the above property is clear. Conversely, suppose that f obeys this property.

Set K = range (f). Define a map G: K x R → R as follows. To compute G(a, b) find a string $u_0 \ldots u_\ell$ in $R^*$ such that a = f $(u_0 \ldots u_\ell)$; set G(a, b) equal to $f(u_0 \ldots u_\ell\ b)$. G is well-defined by virtue of the assumed property for f. Extend G arbitrarily to a map F: $R^2 \to R$ and set $x_0 = f(\lambda)$. The system (1) with transition map F then realizes f.

QED

Thus we have characterized realizability of type A.

III. Adding Constraints of Boundedness

Adding the constraint of boundedness by $B_0$ yields the following Theorem 2: f is realizable by a system of the form (1) with $|F(\underline{v})| \leq B_0$ for all $\underline{v}$ in $R^2$ if and only if f obeys the property below:

for all $u_0 \ldots u_\ell$, $v_0 \ldots v_m$ in $R^*$, r in R

(i) $f(u_0 \ldots u_\ell) = f(v_0 \ldots v_m)$

$\Rightarrow f(u_0 \ldots u_\ell\ r) = f(v_0 \ldots v_m\ r)$

(ii) $|f(u_0 \ldots u_\ell\ r)| \leq B_0$

(Note again that either $u_0 \ldots u_\ell$ or $v_0 \ldots v_m$ or both may be null.)

Proof: Necessity is clear. Conversely let K and G be as in the proof of Theorem 1. From the given property for f, $|G(\underline{v})| \leq B_0$ for all $\underline{v}$ in K x R; extending G to all of $R^2$ so as to retain this bound is trivial. The rest of the proof is immediate.

QED

We next turn to realizability of type C.

IV. Adding the Lipschitz Condition

We begin this section with the following Lemma, whose proof appears in (Kaliski, 1971), (Czipszer and Geher, 1955). Lemma: Let

S be a given subset of $R^2$, h: S → R a map satisfying the conditions below for all $\underline{v}$, $\underline{w}$ in S, for some non-negative real constants $B_0$, $L_0$, and for some norm on $R^2$.

(i) $|h(\underline{v}) - h(\underline{w})| \leq L_0 \|\underline{v} - \underline{w}\|$

(ii) $|h(\underline{v})| \leq B_0$

Then the map H: $R^2 \to R$ defined by

$$H(\underline{z}) = \max\ (-B_0,\ \min\ (B_0,\ \underset{\underline{y}\ \text{in}\ S}{\text{glb}}\ (h(\underline{y}) + L_0\|\underline{z} - \underline{y}\|)\ )$$

is indeed well-defined, is an extension of h, and obeys for all $\underline{v}$, $\underline{w}$ in $R^2$ (i) and (ii); above, with h replaced by H.

We will use the above Lemma in proving the following result.

Theorem 3: f is realizable by a system of the form (1) with $|F(\underline{v})| \leq B_0$, and $|F(\underline{v}) - F(\underline{w})| \leq L_0 \|\underline{v} - \underline{w}\|$, for all $\underline{v}$, $\underline{w}$ in $R^2$ if and only if f obeys the property below:

for all $u_0 \ldots u_\ell$, $v_0 \ldots v_m$ in R, r, s in R

(i) $|f(u_0 \ldots u_\ell\ r) - f(v_0 \ldots v_m\ s)|$

$\leq L_0 \|(f(u_0 \ldots u_\ell), r) - (f(v_0 \ldots v_M), s)\|$

(ii) $|f(u_0 \ldots u_\ell\ r)| \leq B_0$

(Note that condition (i) of Theorem 3 implies condition (i) in Theorems 1 and 2; also $u_0 \ldots u_\ell$ or $v_0 \ldots v_M$ or both may be null.)

Proof: Again necessity is clear, as if such a system exists,

$$f(u_0 \ldots u_\ell\ r) = F\ (f(u_0 \ldots u_\ell), r)$$

$$f(v_0 \ldots v_m\ s) = F\ (f(v_0 \ldots v_m), s)$$

As for sufficiency, again define G and K as in the proof of Theorem 1. From condition (i) it is easy to see that G is well-defined, from our remark following the Theorem statement above. Setting S = K x R, and h = G it is also clear, from the assumed property for f, that the conditions of the Lemma hold. Setting F = H, and $x_0 = f(\lambda)$ completes the proof.

QED

In resolving question C, then, we have the following intuitive interpretation of the constraints needed on f (viewing the norm as being, for example, the p = 1 norm): sequences producing "similar" f-values, followed by close together real numbers, must in turn produce "similar" f-values; all such f-values must be bounded in magnitude by $B_0$. The second use of similar is taken to mean "up to $L_0$ times as different" as the first use of similar.

## V. Discussion; Limitations of the Theory

We address several issues briefly in this concluding section. First we address the obvious limitations of the theory: the systems considered are one-dimensional with the further constraint that the output is the state. It is only in the context of a constrained setting of this kind that we can obtain such a concise set of conditions for realizability. More general systems settings will imply more complex statements of constraints on f.

Nonetheless, in defense of these limitations, it should be pointed out that theoretically, at least, (Kaliski and Lemone, 1980), one-

dimensional realizations of the type considered here are sufficient for coding purposes.

An essential component in the design of finite-state compensators for discrete-time continuous-state plants is the coder, a map $C: R^* \to \{0, 1\}$ (Johnson, 1978). Such a coder, which forms an interface between the plant and the compensator, can always be realized as the composition of two maps: a map $f: R^* \to R$, and a map $g: R \to \{0, 1\}$ (a threshold device), i.e. $C = g\ f$. (Kaliski and Lemone, 1980).

The importance of finite state compensation as a design tool has thus renewed interest in coder design (Jones, 1978), (Wimpey, 1980), (Kaliski and Johnson, 1979), and spurred research in related areas of finite-state regulator theory (Gatto and Guardabassi, 1976).

It is then, in this context, that we address the questions of this paper. With the basic, theoretical, understanding achieved here we hope to, in subsequent research, bridge the gap to somewhat more practical approaches. As a theoretical result, then,Theorem 3 has considerable intrinsic interest.

## VI. Acknowledgement

The author would like to thank his research colleague, Dr. Timothy Johnson of M.I.T., for his valuable assistance in helping to formulate the ideas presented herein.

VII. References

Czipszer, J. and Geher, L. (1955) Extension of Functions Satisfying a Lipschitz Consition, Acta Math. Acad. Sci.Hungar, 6, 213-220.

Gatto, M. and Guardabassi, G. (1976) The Regulator Theory for Finite Automata, Information and Control, V. 31, N. 1.

Johnson, T. L. (1978) Finite-State Compensation of Continuous Processes Proc. IFAC Triennial World Congress, Helsinki, Finland, June 1978.

Jones, S. N. Realization of A/D Coders, M.I.T. Electronics Systems Laboratory Report ESL-TM-817, March, 1978.

Kaliski, M. E. (1971) Dynamic Systems: An Automata-Motivated Analytic Approach, M.I.T. Electronics Systems Laboratory Report ESL-R-453, July, 1971.

Kaliski, M. E. (1974) Autonomous Sequence Generation, Information and Control, V. 26, N. 3.

Kaliski, M. E. and Johnson, T. L. (1979) Binary Classification of Real Sequences by Discrete-Time Systems, Proc. 1978 IEEE Conference on Decision and Control, San Diego, CA, January, 1979.

Kaliski, M. E. and Lemone, K. L. (1980) Discrete-Codings of Continuous-Valued Signals, Proc. 1980 Conference on Information Sciences and Systems, Princeton University, Princeton, NJ, March 1980.

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

Wimpey, D.G. (1980) Towards a Structure Theory for Coders of Real-Valued Signals, M.I.T. Laboratory for Information and Decision Systems, Report LIDS-TM-1052, October, 1980.

## VIII. Footnotes

[1]This property is equivalent to the finitariness of f when f has finite range (see (Wimpey, 1980) ).

# FP5-4:00

## STABILITY OF DICED SYSTEMS*

T. L. Johnson
Department of Electrical Engineering and Computer Science
Room No. 35-210
Massachusetts Institute of Technology
Cambridge, Mass. 02139

and

Bolt, Beranek and Newman
50 Moulton Street
Cambridge, Mass. 02138

### ABSTRACT

Diced systems are defined as autonomous systems governed by ordinary differential equations having discontinuities (in $R^n$) on submanifolds where one or more of the state variables takes an integer value. Such systems may be regarded as approximations of continuous systems or as representative models of a class of discontinuous systems. Trajectories of such systems (for a given initial state) are readily calculated and may exhibit complex sliding-mode segments. Asymptotic properties of such trajectories are discussed and classified. Motivation is given in terms of observed properties of interconnected power systems.

### I. Introduction

Diced systems, as defined here, are finite-dimensional autonomous continuous-time dynamic systems governed by equations of the form $\frac{dx}{dt}(t) = f(x(t))$; $x_o(t_o) = x_o \in R^n$, $t \geq t_o$, where $f : R^n \to R^n$ is piecewise-constant with discontinuities only on the surfaces where one or more coordinates of $R^n$ take integer values. A diced system in $R^2$ is very easy to illustrate: the plane can be divided into a uniform gird, and within each square a vector representing the magnitude and direction of f is shown (Figure 1).

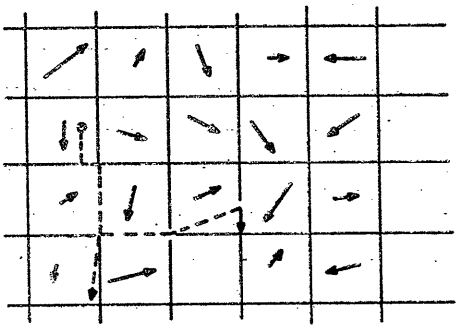

Figure 1: A Diced System in $R^2$.

Existence and uniqueness of a solution for any fixed initial state, $x_o$ can be studied using a generalization of the method introduced by Filippov [1]; trajectories may exhibit sliding mode segments and higher-order non-differentiable behavior as illustrated in Figure 1. In order to obtain existence of solutions, multivalued extensions of f onto its discontinuity surfaces are required. Every trajectory can be represented by a sequence of transition points and times, $\{x(t_i), t_i\}$.

Definitions of various types of stability and instability can be constructed from an examination of the invariant limit sets [2] of the trajectories. For diced systems, the range of asymptotic behavior of trajectories starting from different initial conditions can be exceedingly rich. The possibility of approximate global stability analysis using nondeterministic automata is examined and its limitations are indicated.

In practice, diced systems might be viewed as approximations of continuous or discontinuous systems. In the former case, for instance, we might seek the best piecewise-constant (finite-element) approximation to a continuous system. Wang [3] has presented an application of this type for solving partial differential equations. In the latter case, a state space diffeomorphism might be used first to transform the discontinuities of a system to lie along coordinate axes, and then a diced approximation could be developed which would preserve the discontinuous behavior of such systems. The potential practical advantages of diced approximations lie in a reduction of information storage required to characterize a system and the possibility of assessing its approximate asymptotic behavior without a detailed simulation.

For example, at the time of a known failure of a power system, it is often desirable to predict the long-term consequences of various control strategies so that an operator can decide among them. Yet the system is too big to store all possible consequences in advance. A practice which has thus been followed in some cases [4] is to run a simulation "faster than real-time" for each control strategy. While the issue of approximation accuracy is not treated here, the results suggest that significant economy of real-time computation might be achieved by approximating the dynamics of a diced system. However, they also suggest that the patterns of stability and instability exhibited by such discontinuous systems may be highly complex and that analytical methods are not likely to yield clear-cut predictions about global stability.

### II Preliminaries, Notation

Let $i = [i_1,\ldots,i_n] \in Z^n$ be a multi-index on the n-tuples of integers (Z). Let $b = [b_1,\ldots,b_n] \in B^n$ represent an n-tuple of binary numbers ($B \triangleq \{0,1\}$). Let $\chi_i(x): R^n \to R$ be the characteristic function of the open set $\{x = [x_1,\ldots,x_n] \in R^n \mid i_k < x_k < i_k + 1, k = 1,2,\ldots,n\}$.

*This research has been performed at the M.I.T. Laboratory for Information and Decision Systems with support provided by the U.S. Department of Energy (Contract ET-76-C-0102295) and the U.S. Air Force Office of Scientific Research (Contract F49620-80-C-0002).



0191-2216/80/0000-1110$00.75 © 1980 IEEE

Definition: A diced initial value problem (DIVP) is specified by a system of ordinary differential equations

$$\dot{x}(t) = f(x(t)) \; ; \; x(t_o) = x_o \in R^n \; ; \; t \geq t_o \qquad (2.1)$$

where $f : R^n \to R^n$ has the particular form

$$f(x) = \sum_{i \in Z^n} f_{bi} \, \chi_i(x) \; ; \; b = 0 \in B^n \qquad (2.2)$$

and $f_{oi} \in R^n$ for each multi-index i.

The surfaces of discontinuity of f may be classified by their dimension. Let $1(b) : B^n \to \{1,\ldots,n\}$ be a function denoting the number of "1"'s in the binary n-tuple b. For fixed $i \in Z^n$, consider the sets

$$S_{bi} = \{x \in R^n \mid i_k < x_k < i_{k+1} \text{ if } b_i = 0$$

$$i_k = x_k \text{ if } b_k = 1, \; k = 1,2,\ldots,n\} \qquad (2.3)$$

These may be viewed as the set of submanifolds "attached to" the point x=i.* For example $S_{oi}$ is the interior of the n-dimensional cube indexed by its vertex at x=i; $S_{1i}$ (the shorthand 1 denoting b = [1,1,..., 1]) is the single point x=i. The submanifolds of dimension p associated with x=i are

$$S_i^p = \{S_{bi} \mid 1(b) = n-p\} \qquad p = 0,1,\ldots,n. \qquad (2.4)$$

This notation provides a compact classification of all of the subsets of $R^n$ which are of interest.

In Section III, conditions for well-posedness of a DIVP are examined. This is done by extending f to its discontinuity surfaces (from $\{f_{oi}\}$, we generate $\{f_{bi}\}$, $b \neq 0 \in B^n$). Then a constructive procedure can be used to generate solutions $\dot{x}(t) = \phi(t,t_o,x_o)$ for each $x_o \in R^n$, $t_o \in R$ and hence to define the transition map $\phi$: $R \times R \times R^n \to R^n$. Let $\chi$ denote the function space in which trajectories are defined. This leads to the following

Definition: A diced system is an autonomous dynamical system $(\chi, R^n, \phi)$ (See [5]).

Stability has been viewed as a qualitative property of a dynamical system, and concerns the asymptotic behaviors of trajectories $x(\cdot) = \phi(\cdot,t_o,x_o)$ as $x_o \in X$ is varied. Stability of diced systems is discussed in Section IV. Two useful notions will be those of the positive limit set and the invariant set [2].

Definition: The set $\Omega \subseteq R^n$ is invariant with respect to the system $\dot{x}(t) = f(x(t),t)$ if for any $x_o \in \Omega$ there is a $t_o$ such that the motion $\tilde{\phi}(t,t_o,x_o)$ belongs to $\Omega$ for all $t \geq t_o$.

Definition: The set $\Pi \subset R^n$ is called the positive limit set of a bounded motion $\phi(t;t_o,x_o)$ if, for any point $p \in \Pi$, there exists a sequence of times $\{t_n\}$ tending to infinity as $n \to \infty$, so that

$$\lim_{n \to \infty} \| \phi(t_n,t_o,x_o) - p \| = 0 \qquad (2.5)$$

In applying these definitions it will be useful to recall that a function x(t) is periodic of period $T > 0$ if $x(t) = x(t+T)$ for all t; "the" period of a periodic function is defined as the least T for which this equality holds.

## III Existence and Uniqueness

Consider the DIVP (2.1), (2.2). Defining solutions within the cubes $S_{oi}$ by integration is entirely straightforward; all difficulties arise in attempting to extend solutions across the discontinuity surfaces of f; in general, there is no unique continuation. Various possibilities are

(a) To restrict the class of f so that continuations are always unique (this is very restrictive indeed, and essentially eliminates many interesting phenomena from consideration).

(b) To eliminate the non-continuable surfaces from the domain of f; however, then all points on all trajectories leading to such surfaces must also be eliminated, and a large part of the original domain of definition may ultimately be excluded.

(c) To choose an ad hoc rule for continuation of solutions; however, it proves difficult or impossible to do this in a self-consistent and unbiased manner.

A fourth alternative has been selected here:

(d) To sacrifice uniqueness and continue all solutions through a discontinuity.

In this way a viable deterministic existence theory can be developed, at the cost of considering a countable number of alternative solutions. A "physical" justification for adopting this approach is that in the presence of small perturbations of the initial conditions, a solution near to at least one alternative solution will occur.

A constructive procedure is given for defining solutions. To simplify its presentation, a multivalued continuation of f to the surfaces $S_{bi}$, $b \neq 0$, is first defined. Initially, f is specified on the submanifolds $S_i^n = \{S_{oi}\}$ of dimension n. The continuation proceeds recursively to submanifolds, $S_i^p$ of dimension n-1, n-2, ...,0. Recall that $S_i^o$ is the point set $\{x \in R^n \mid x = i_k, \; i_k \text{ an integer}\}$. Notationally, a single valued $f_{bi}$ will not be distinguished from a multivalued $f_{bi}$, the implication being that the prescribed rule is applied to each possible value of $f_{bi}$ in turn, and the set of all results is retained. Let p=n. Suppose $f_{bi}$ are known on $S_i^q$, $p \leq q \leq n$. Then $f_{bi}$ can be extended to $S_i^{p-1}$ as follows, for each $i \in Z^n$.

Suppose $S_{bi} \in S_i^{p-1}$. Let indices $j_j \ldots j_{n-(p-1)}$ denote the ordered nonzero positions of b, i.e., $b_{j_k} = 1$, $k = 1,\ldots,$ $n-(p-1)$ and $b_j = 0$ otherwise. The neighborhoods of $S_{bi}$ of dimension q, $p \leq q \leq n$, can be defined as follows. For q=n, consider all indices $\tilde{i}$ formed by decrementing

*The obvious injection of the integers into the reals is implied.

$i_{jk}$ by one for any subset of the subindices k=1,..., n-(p-1), including the null-set; then $S_{\tilde{b}\tilde{i}} \epsilon S_{\tilde{i}}^{n}$ is a neighborhood of $S_{bi}$ where $\tilde{b} \neq 0$. For q = n-1, consider all values b having a single "one" in one of the positions $j_1 \ldots j_{n-(p-1)}$ and for each $\tilde{b}$, form $\tilde{i}$ from the remaining n-(p-1)-1 indices as above; then $S_{\tilde{b}\tilde{i}} \epsilon S_{\tilde{i}}^{n-1}$ is a neighborhood of $S_{bi}$. For q = n-2, consider all values b having "ones" in any <u>two</u> of the positions $j_1,\ldots,$ $j_{n-(p-1)}$ and from each $\tilde{b}$ and $\tilde{i}$ from the remaining n-(p-1)-2 indices as above; then $S_{\tilde{b}\tilde{i}} \epsilon S_{\tilde{i}}^{n-2}$ is a neighborhood of $S_{bi}$. This procedure is continued until q=p.

The values of $f_{bi}$ on $S_{bi} \epsilon S_i^{p-1}$ are determined from the values of $f_{\tilde{b}\tilde{i}}$ on each of its neighborhoods $S_{\tilde{b}\tilde{i}} \epsilon S_{\tilde{i}}^{q}$, $p \leq q \leq n$. It is thus sufficient to give the procedure for determining $f_{bi}$, assuming that these values on higher-dimensional submanifolds are known (i.e., the values can be determined recursively). Define $S_{\tilde{b}i}$ to be an <u>input submanifold</u> to $S_{bi}$ if $(f_{\tilde{b}\tilde{i}})_\ell = 0$ for all $\ell$ such that $\tilde{b}_\ell = 1$, and for all remaining $\ell$ in the set $j_1,\ldots,j_{n-(p-1)}$, $(f_{\tilde{b}\tilde{i}}) < 0$ for those $\ell$ such that $\tilde{i}_\ell = i_\ell$, while $(f_{\tilde{b}\tilde{i}}) > 0$ for those $\ell$ such that $\tilde{i}_\ell = i_\ell - 1$. Define $S_{\tilde{b}i}$ to be an <u>output submanifold</u> if $(f_{\tilde{b}\tilde{i}})_\ell = 0$ for all $\ell$ such that $\tilde{b}_\ell = 1$, and for all remaining $\ell$ in the set $j_1,\ldots,j_{n-(p-1)}$, $(f_{\tilde{b}\tilde{i}})_\ell \geq 0$ for those $\ell$ such that $\tilde{i}_\ell = i_\ell$, while $(f_{\tilde{b}\tilde{i}})_\ell \geq$ for those such that $\tilde{i}_\ell = i_\ell - 1$. Note that those sets for which $(f_{\tilde{b}\tilde{i}})_\ell \neq 0$ when $\tilde{b}_\ell = 1$ need not be considered. So long as the set of output submanifolds of $S_{bi}$ is non-empty, $f_{bi}$ is assigned the set of all values $f_{\tilde{b}i}$ on the output submanifolds. If the set of output submanifolds is empty, $S_{bi}$ is a <u>generalized sliding surface</u>. Consider $f_{\tilde{b}\tilde{i}}$ on $S_{\tilde{b}\tilde{i}} \epsilon S_{\tilde{i}}^{p}$ in the input set. If this set is empty, set $f_{\tilde{b}\tilde{i}} = 0$. Recall that $S_{\tilde{b}\tilde{i}} \epsilon S_{\tilde{i}}^{p}$ is formed by keeping i unchanged in all but one position, say $j_k$, of b, so $\tilde{b} = [b_1,\ldots,b_{j_k-1}, 0, b_{j_k+1},\ldots,b_n]$ and either $\tilde{i} = i$ or $\tilde{i} = [i_1,\ldots,i_{j_k-1}, i_{j_k+1},i_n]$. Thus there are a maximum of 2(n-(p-1)) surfaces in this subset of the input set. These surfaces are considered in pairs to determine the admissable values of $f_{bi}$; using the example above, if $S_{\tilde{b}i}$ is in the input set then $(f_{\tilde{b}\tilde{i}})_{j_k} < 0$ and if $S_{\tilde{b}\tilde{i}}$ is in the input set $(f_{\tilde{b}\tilde{i}})_{j_k} > 0$. If both elements are members of the input set then

$$f_{bi} = [(f_{\tilde{b}\tilde{i}})_{j_k} f_{\tilde{b}i} - (f_{\tilde{b}i})_{j_k} f_{\tilde{b}\tilde{i}}]/[(f_{\tilde{b}\tilde{i}})_{j_k} - (f_{\tilde{b}i})_{j_k}] \qquad (3.1)$$

while if only one is in the input set, let

$$f_{bi} = 0$$

The set of possible values of $f_{bi}$ on a generalized sliding mode is completed by considering each $S_{\tilde{b}i} \epsilon S_i^p$ in this manner. In all such cases, $(f_{bi})_{j_k}$ ; k = 1,...,n-(p-1) are zero, so that further motion occurs on $S_{bi}$ itself.

Thus, the procedure for extending the function f to all of $R^n$ is completed. The complexity of the procedure arises from the large number of possibilities which can arise. A number of such special cases are illustrated on Figure 2. Evidently, the procedure for extending f is not the only one which could be devised. In the next step, construction of solutions, however, it will become apparent that the underlying principle has been to define f in a manner which preserves <u>all</u> trajectories that might arise from each initial condition.

Let $x_o \epsilon R^n$ be given as the initial condition of (2.1) at $t = t_o$; let $S_{bi} \epsilon S_i^p$ be the smallest submanifold containing $x_o$. Let $f_{bi}$ denote one of the extended values of f on $S_{bi}$. Define

$$\phi(\tau,t_o,x_o) = x_o + f_{bi}(\tau - t_o) \; ; \; t_o < \tau \leq t_1 \qquad (3.2)$$

The time $t_1$ is defined as follows: for each $\ell$ such that $(f_{bi})_\ell$ is nonzero, let $(t_1)_\ell$ denote the first $\tau > t_o$ such that $[\phi(\tau,t_o,x_o)]_\ell$ is an integer; then $t_1 = \min_\ell[(t_1)_\ell]$ and $x_1 = \phi(t_1,t_o,x_o)$. If $f_{bi} = 0$, then $t_1 = \infty$ and $x_1 = x_o$, and this solution terminates. Otherwise, $x_1$ defines new values of b,i, and p, and the solution process continues:

$$\phi(\tau,t_k,x_k) = x_k + f_{bi}(\tau - t_k) \; ; \; t_k < \tau \leq t_{k+1} \qquad (3.3)$$

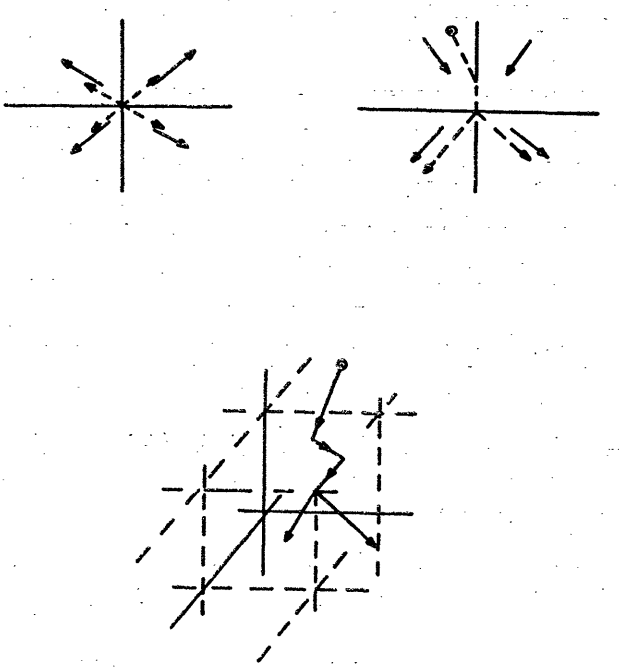

Figure 2
Illustration of Nonuniqueness of Solutions

On those surfaces where $f_{bi}$ is multivalued, each possibility must be examined in turn; in this sense, $\phi$ is also multivalued. Each trajectory pieced-together in this fashion can be summarized by a sequence $\{x_k,t_k\}$, $k = 0,1,\ldots$ in some cases, these sequences are finite and in other cases infinite. By inspection of $\{x_k\}$ alone, a corresponding sequence of regions $\{\sigma_k\}$, where $\sigma_k \epsilon \{S_{bi}\}$ is the minimal submanifold containing $x_k$, can be constructed.

A solution of (2.1), (2.2) is then defined in the obvious manner, as any $\phi(t,t_o,x_o)$ constructed by the continuation procedure (3.3). It has the property that for any finite admissable k, $\phi(t,t_o,x_o)$ is piecewise continuous on $[t_o,t_k]$. This solution by continuation is said to be asymptotic if $\lim_k t_k = \infty$. An asymptotic solution is piecewise continuous. For purposes of the present work, a solution will be said to exist if the state-space continuation is asymptotic.* Asymptotic solutions need not be unique, but the rate of growth in the number of solutions can be bounded as a function of k, since the maximum number of output submanifolds can be bounded above for any $S_{bi}$. If there is only one asymptotic solution through $(x_o,t_o)$, it is said to be unique. Continuous dependence of $\phi(t,t_o,x_o)$ with respect to $x_o$, of course, is not to be expected for $t > t_1$.

## IV Stability

The usual definitions of stability presuppose a solution which is well-posed in the sense of existence, uniqueness, and continuous dependence on the initial data. Diced systems, in general, do not possess the last two properties. One alternative is to nevertheless use the standard notions of stability, restricting their domain of application to those initial states for which the usual notions of well-posedness are (locally) satisfied. Unfortunately, the set of such initial states appears quite difficult to characterize and thus imposes an awkward restriction on the applicability of this alternative.

Another alternative, introduced here, does not impose such restrictions, but weakens the notion of stability that is employed. Stability is viewed as a qualitative property of a trajectory, and a system is then said to be stable when all of its trajectories share this property.

Definition: The motion of a diced system (2.1), (2.2) initiated at $(t_o,x_o)$ is

$$M(t_o,x_o) = \{\phi(t,t_o,x_o),\ t \geq t_o \mid \phi \text{ is a transition function initiated at } (t_o,x_o)\}$$

which is the set of all trajectories originating at $(t_o,x_o)$.

Definition: The motion $M(t_o,x_o)$ of a diced system is said to be

(a) Bounded in magnitude if there is a constant $\Phi > 0$ such that

$$\max_{\phi \epsilon M(t_o,x_o)} \{\sup_{t \geq t_o} ||\phi(t,t_o,x_o)||\} < \Phi$$

(b) Bounded in cardinality is there exists a constant N such that

$$\sup_{t \geq t_o} \{\text{cardinality of } M(t_o,x_o)\} < N$$

The concepts of boundedness in magnitude and cardinality are independent. In both cases, the only difficulties occur at $t \to \infty$, since (a) any $\phi(t,t_o,x_o)$ is by construction bounded for all finite t, and (b) the cardinality of $\phi(t,t_o,x_o)$ is finite, by construction, for all finite t. The following propositions are almost immediate.

Proposition 1: In (2.1), suppose $||f_{oi}|| < F$ for all i, then $||\phi(t,t_o,x_o) - x_o|| < F(t-t_o)$ for all $\phi \epsilon M(t_o,x_o)$.

Proof: The extension of $f_{oi}$ to $f_{bi}$ always guaranteed that $||f_{bi}|| < F$, and the construction procedure (3.3) guaranteed that the estimate of the proposition held for each t. q.e.d.

Proposition 2: Let $|i| = |i_1|+\ldots+|i_n|$. Suppose for system (2.1) there exists $B > 0$ such that for all $|i| > B$, and $k = 1,\ldots,n$, $(f_{oi})_k\ i_k < 0$. Then $M(t_o,x_o)$ is bounded in magnitude.

Proof: For any i such that $|i| > B$, every set $S_{bi}$ contains output submanifolds with the same $|i|$ or smaller $|i|$, and input submanifolds with the same $|i|$ or larger $|i|$; further more, $S_{1i}$ always outputs to $S_{o\tilde{i}}$ with $|\tilde{i}| < |i|$. Thus the construction process cannot terminate for $|i| > B$, and for such i, $|i|$ is reduced at least once every n intervals; hence every solution satisfies $|\phi(t,t_o,x_o)| < B$ for t sufficiently large. Thus $M(t_o,x_o)$ is magnitude-bounded.

Proposition 3: Suppose that for every $i \epsilon Z^n$, $b \epsilon B^n$, $S_{bi}$ has at most one output submanifold. Then the motion $M(t_o,x_o)$ of (2.1),(2.2) is bounded in cardinality.

Proof: The extension procedure of Section III shows that in this case $f_{bi}$ takes the value on its output submanifold or the value zero. If a trajectory enters $S_{bi}$, it either continues uniquely to the output submanifold, or terminates at $S_{bi}$. In either case, the cardinality of the solution cannot increase during its construction.

Thus there are two notions of instability for diced systems: solutions may become unbounded in magnitude, and/or they may become unbounded in cardinality. This second form of instability is new: a trajectory can fracture and a chain reaction of subsequent fractures may ensue--the complexity of the process grows without bound.

Next, a notion of stability is put forth. Suppose that the motion $M(t_o,x_o)$ of a diced system is bounded in magnitude and cardinality (or simply "bounded").

*Moreover if $\lim t_k \neq \infty$, solutions by time-continuation could be defined; however, their properties will not be explored here.

Then a set $S \subset R^n$ consisting of a finite union of the submanifolds $S_{bi}$ is termed a positive limit set of a (bounded) trajectory $\phi(t,t_o,x_o)$ if for any point $x \in S$, there exists a sequence of times $\{\tau_k\}$, tending to infinity as $k \to \infty$, so that

$$\lim_{k\to\infty} )\phi(\tau_k,t_o,x_o) - x( = 0 \qquad (4.1)$$

where $)(\,$ denotes the set-membership metric, i.e., if $x \in S_{bi}$,

$$)y-x( = \begin{cases} 0 & y \in S_{bi} \\ & y \notin S_{bi} \end{cases}$$

In applying this definition, it is important to recall the standing assumption from Section III, that all trajectories are asymptotic, so that such sequences $\{\tau_k\}$ exist.

Definition: A bounded motion $M(t_o,x_o)$ of a diced system is termed pointwise stable if all trajectories $\phi(t,t_o,x_o) \in M(t_o,x_o)$ have the same positive limit set. The motion is locally stable for $x_o \in S_{bi}$ if all trajectories $\phi(t,t_o,x) \in M(t_o,x)$, $x \in S_{bi}$, have the same positive limit set. The motion is globally stable if all trajectories $\phi(t,t_o,x)$ have the same positive limit set.

Concepts of uniform stability will not be discussed since only time-invariant diced systems are considered in the present account.* In fact, the evaluation of stability, according to the definitions given, can be based merely on knowledge of the sequence $\{\sigma_k\}$ of submanifolds containing $\{x_k\}$, since it is known from the construction procedure that $t_{k+1} > t_k$ and from the asymptotic assumption that $\lim_{k\to\infty} t_k = \infty$. This suggests that a way to generate the sequence $\{\sigma_k\}$ autonomously, without explicit integration and generation of $\{x_k,t_k\}$ would be particularly valuable in the assessment of stability. This has not been achieved yet.

Knowledge of the time-structure $\{t_k\}$ of individual solutions can be of further value in refining stability notions. To simplify the remaining concepts it is now assumed that the trajectories are uniquely-defined (e.g., as occurs in Proposition 3) and bounded. Suppose $\Pi$ is a positive limit set of such a solution in the conventional sense of Section II (eq. (2.5)). Then in the usual manner it can be shown that $\Pi$ is bounded, closed, non-empty and invariant, the last property being a consequence of time-invariance. In fact, as a consequence of finite-dimensionality of $R^n$, all such solutions are asymptotically almost-periodic [6]. Two cases of special interest are the asymptotically constant (equilibrium) solution and the asymptotically periodic solution. These can be identified directly from the sequence $\{x_k,t_k\}$ characterizing $\phi(t,t_o,x_o)$.

Proposition 4: If the sequence $\{x_k,t_k\}$ is finite of length N, the positive-invariant limit set consists of one point, the last value $x_N$ (for which $t_N = \infty$). If the sequence $\{x_k,t_k\}$ is jointly periodic of period m for $k > N$, then the positive-invariant limit set is a cycle (closed curve) in $R^n$.

Proof: For the first case, note that the construction procedure automatically defines $t_N = \infty$ when the sequence is finite, and this implies a constant solution for $t \geq t_N$. In the second case, note that since $\{x_k,t_k\}$ completely specify $\phi(t,t_o,x_o)$, $\phi$ must be periodic $t_{k+m}-t_k$, $k > N$, whenever $\{x_k,t_k\}$ is periodic (in fact, the solution is a linear interpolation between these points).

It is interesting to note that for diced systems, the establishment of an equilibrium or periodic solution after a finite time ($t_N$) is often to be expected (whereas this would be considered exceptional in the case of continuous differential equations); however, in some cases almost periodic solutions may also exist.

## V Discussion and Conclusions

The present account of the stability of diced systems leaves a number of questions unanswered and raises some new ones. A study of methods for temporal continuation of non-asymptotic solutions is needed; such solutions may represent a new sort of sliding mode which can arise in higher dimensional spaces, as suggested by an example of Utkin [7]. The possibility of extending the techniques developed here to time-varying systems has been mentioned; Filippov's general existence results apply to this problem. A study of the partitioning of initial states which is implied by the proposed stability definition would also be fruitful; what properties are shared by initial state sets giving rise to the same asymptotic solution? In general, it would appear that the initial states within a given region $S_{bi}$ can ultimately end up widely dispersed. The possibility of using an automaton to simplify the propagation of solutions has also been raised. The approximation of continuous systems by diced systems has not been explored, but under appropriate conditions, a bound on the approximation error should be achievable.

In spite of the questions that are unanswered, some modest progress has been made toward defining the stability properties of diced systems. First, a constructive continuation procedure for higher dimensions has been found; the problem readily evades one's intuition above n = 1,2 and even 3 as endless combinations of difficult situations may occur. Second, a compromise on the issue of uniqueness has been put forth: the number of admissable solutions at any finite time is bounded. Third, the concepts of stability have been generalized to provide meaningful criteria for discontinuous systems of diced type.

Returning to the electric power system example cited in the opening section, it would appear that the implications of the research might be very disturbing, for two primary reasons. First, a new type of instability--an unbounded growth in the number of possible solutions with time--has been identified. Second, and independently, the partitioning of the initial state--at least in worst-case situations--based on asymptotic properties, appears to be very fine and irregular; thus a small perturbation in the initial state may give rise to completely different asymptotic behavior than is

*The results could be extended in this direction for systems with continuous time-variation; however discontinuously time-varying systems may not be continuable, as Filippov pointed out.

found for the unperturbed initial state. Both of these phenomena imply that the future behavior of a diced system with a (approximately) specified initial state may be fundamentally unpredictable; if the long-term future consequences of a present control policy are unpredictable, the problem of choosing the best policy becomes more difficult and planning must be done with a shorter horizon.

## References


[1] Filippov, A.F., [1964], "Differential Equations with Discontinuous Right-Hand Sides," Amer. Math. Soc. Trans., Ser. 2, Vol. 42, pp. 199-231.

[2] Willems, J.L. [1970], Stability Theory of Dynamical Systems, Nelson and Sons, Ltd., London.

[3] Wang, P.K.C., [1968], "A Method for Approximating Dynamical Process by Finite-State Systems," Int. J. Control, Vol. 8, No. 3, pp. 285-296.

[4] Ewart, D.N., "Whys and Wherefores of Power System Blackouts," IEEE Spectrum, April 1968, pp. 36-41.

[5] Willems, J.C. and S.K. Mitter [1971], "Controllability, Pole Allocation and State Reconstruction," IEEE Trans. Auto. Control, Vol. AC-16, No. 6, pp. 582-595.

[6] Dafermos, C.M. [1974], "Semiflows Associated with Compact and Uniform Processes," Math. Sys. Theory, Vol. 8, No. 2.

[7] Utkin, V.I. [1978], Sliding Modes and Their Application in Variable Structure Systems, Mir Publishers, Moscow.

Submitted to 20th IEEE Conference on Decision and Control, San Diego, Ca. Dec. 1981

20th CDC

Appendix D LIDS-P-1085

# Analytic Models of Multitask Processes*

Timothy L. Johnson

Lecturer
Department of Electrical
Engineering & Computer Science
M.I.T., Rm. 35-205B
Cambridge, Mass. 02139

Senior Scientist
Bolt, Beranek and Newman, Inc.
50 Moulton Street
Cambridge, Mass. 02238

## Abstract

Asynchronous multitask processes occur in a wide variety of control applications ranging from industrial control to computer operating systems, yet no analytical methods are available for studying their detailed behavior. The preliminary results reported here illustrate that a very general class of such processes can be represented by discontinuous hybrid-state discrete-time systems.

*This research has been performed at the M.I.T. Laboratory for Information and Decision Systems with support provided by the U.S. Air Force Office of Scientific Research under Contract F49620-80-C-0002. The results presented here do not necessarily represent the views of the U.S. Government.

## Background and Motivation

A multitask process is characterized by a number of tasks which operate concurrently or sequentially, on an external resource or data base. The timing of the tasks is generally asynchronous in that new task execution is initiated by the completion of previous tasks. If necessary, synchrony and sequential ordering of tasks can be enforced in a number of ways through the task definitions themselves. However in this research no such constraints are imposed: rather, the general qualitative behaviors which may arise in such systems are analyzed. Only two basic assumptions are imposed: (1) a task requires a finite amount of time and storage to execute, and (2) task descriptions are fixed, in that the execution of a task cannot alter its own nature nor the number or nature of any other tasks.

The range of possible behavior of such systems is so large that the problem of conceptualizing, analyzing and "debugging" multitask processes is very common and enormously complex. Two approaches are presently in use: stochastic queueing analysis [1],[2] and simulation [3],[4]. Queueing analysis is most useful for evaluating the average performance properties of an operational multitasking system, while simulation allows certain undesirable properties of a planned system to be discovered and corrected during the design process. Neither of these methods provides very much insight about generic problems in the design of such systems, nor do they provide ideas about how to remedy or detect flaws. The results reported here constitute a modest step in that direction.

## Model Development

Let $t \in [t_o, \infty)$ denote time. Three sets of state variables will be identified:

$x^1$ - those states which vary continuously with time and take on real values.

$x^2$ - those states which are real-valued but change only at discrete instants of time

$x^3$ - those states which are discrete-valued and (necessarily) change only at discrete instants of time.

The state set is denoted $X = \{X^1, X^2, X^3\}$. For present purposes, it will be assumed that these sub sets of states are finite-dimensional and recognizable; an example will be provided below. Let the increasing sequence $\{t_k\}$ denote the set of all values of t for which changes in at least one element of $x^2$ or $x^3$ occur, and let the values of the states prior to and following $t_k$ be denoted $x_k^{i^-}$, $x_k^{i^+}$, respectively, for i = 1,2,3. In the sequel, $x_k^{i^+}$ will be identified with $x_k^i$.

The instants $\{t_k\}$ will be identified with task initiation or termination times. Let the set of tasks in the system be denoted $G \doteq \{G_1 \ldots G_n\}$. Associated with each task is an iniation function, a termination function and a state-update function[1]:

$g_j^I: X \to \{0,1\}$ - initiation function for task j

$g_j^T: X \to \{0,1\}$ - termination function for task j

$f_j: X \to X$ - state-update function for task j

Each task is either "on" or "off": let $\hat{G}$ denote those tasks which are "on" and $\check{G}$ denote those which are off, so that $G = \hat{G} \cup \check{G}$ and $\hat{G} \cup \check{G} = \phi$ (the null set). The subscript $\hat{j}$ will be used to denote tasks which are "on" and $\check{j}$

[1]To simplify this exposition, these are assumed to be time-invariant; however, this assumption may be relaxed.

to denote tasks which are "off". The task succession rule is as follows: A transition time, $t_k$, is declared whenever

(a) For some $j\varepsilon\{\check{j}\}$, $g_j^I(x^1,x^2,x^3)$ undergoes a 0→1 transition

or

(b) For some $j\varepsilon\{\hat{j}\}$, $g_j^T(x^1,x^2,x^3)$ undergoes a 0→1 transition

Between task transition times, only the states $x^1$ can change, according to a state equation

$$\dot{x}^1(t) = f(x^1(t),\ x_k^{2^+},\ x_k^{3^+}) \tag{1}$$

with $x^1(t_k) = x_k^{1^+}$. At the completion time $t_k^-$ of a task j, the transformation

$$\begin{bmatrix} x_k^{1^+} \\ x_k^{2^+} \\ x_k^{3^+} \end{bmatrix} = f_j \left( \begin{bmatrix} x_k^{1-} \\ x_k^{2-} \\ x_k^{3-} \end{bmatrix} \right) \tag{2}$$

is applied, with $x_k^{2-} = x_{k-1}^{2^+}$ and $x_k^{3-} \doteq x_{k-1}^{3^+}$.

At a transition time, it is possible that more than one task terminates and/or more than one task is initiated. This produces an inherent conflict situation which must be resolved in a consistent manner. For instance, if tasks $j_1$ and $j_2$ terminate together, it is not necessarily true that $f_{j1} \circ f_{j2} = f_{j2} \circ f_{j1}$ (functional composition may not be commutative). Or if task $j_1$ is initiated when $j_2$ terminates, then up-dating with $f_{j2}$ may turn off $j_1$, while terminating $j_1$ may turn on $j_2$ again, etc. In this preliminary abstract, it will be assumed that

• there is a fixed priority among task completions (e.g. 1 > 2 > 3 > j > ...> n)

- all completions are performed first according to priority, and then initiation functions are re-evaluated to redetermine which tasks (if any) should be initiated at the transition times.

Other conflict-resolution methods, such as imposed sequential orderings, are also possible.

Let $\hat{j}_k \in 2^n$ be the set of tasks active at $t_k^+$. Let the transition mapping of (1) be given by $\phi$: $[0,\infty) \times X^1 \to X^1$, so that the solution of

$$\dot{x}^1(t) = f(x^1, x_k^2, x_k^3) \; ; \; x^1(t_k) = x_k^1 \tag{3}$$

is

$$x^1(t) = \phi(t-t_k, x_k^1; \; x_k^2, x_k^3) \tag{4}$$

where $x_k^2, x_k^3$ are viewed as parameters. Define the function $\tau$: $2^n \times X \to R^+$ to be the first transition-time encountered with processes $\hat{j} \in 2^n$ active at $t = t_o$, with initial state $x = (x^1, x^2, x^3) \in X$. This can be tabulated by integrating (1) and applying rules (a) and (b). Let the function $\sigma$: $2^n \times X \to 2^n$ define the next set of active tasks, determined from the preceding priority rules, at the transition time defined by . In other words,

$$t_{k+1} = t_k + \tau(\hat{j}_k, x_k^1, x_k^2, x_k^3) \tag{5}$$

$$\hat{j}_{k+1} = (\hat{j}_k, x_k^1, x_k^2, x_k^3) \tag{6}$$

The important point to observe is that, in principle, it is not necessary to include the continuous-time part of the dynamics, since $\tau$ and $\sigma$ can be pre-computed from f, $\{g_j^I\}$, and $\{g_j^T\}$.

In summary, the dynamics of the asynchonous multitask system can always be represented in the form

$$\left.\begin{aligned} t_{k+1} &= t_k + \tau(\hat{j}_k, x_k) \\ x_{k+1} &= f^{\wedge}_{j_k}(x_k) \\ j_{k+1} &= \sigma(\hat{j}_k, x_k) \end{aligned}\right\} \tag{7}$$

where $f^{\wedge}_{j_k}$ is the composition, according to priority, of the transition functions (2) of the tasks completing at $t^{-}_{k+1}$. It is then clear that $t_{k+1}$ may be combined with $x^1$ and $x^2$, and that $x^3$ may be combined with $\hat{j}$ to yield a general discontinuous hybrid discrete-time system. Extensions to stochastic behavior of $f, \{f_j\}$, $\{g^I_j\}$ and $\{g^T_j\}$ are readily accomodated.

Qualitative Properties

The finite-state part of (7) may be further aggregated to produce an equivalent real-state discrete-time system with discontinuous transition function. Systems of this general class have been discussed by Johnson [5] and Kaliski and Lemone [6]. Their behavior may roughly approximate the behavior of discontinuous systems discussed in Utkin [7] and Johnson [8]. The pertinent properties of such systems will be described more fully in the final version of this paper. Here it is merely noted that problems may arise if $\lim_{k\to\infty} t_k$ is finite. A possible behavior in this situation is an approximation to sliding mode behavior, which is closely akin to the phenomenon of "thrashing" observed in heavily-loaded multitasking systems.

Examples

Realistic examples will be provided in the conference version of this paper.

Appendix E

# Multitask Control of Distributed Processes

T.L. Johnson*

## Abstract

As one class of real-time operating system, multitask systems are widely used in distributed process control applications. The evolution of tasks in such systems may depend on the dynamic response of the controlled process and on task completion times. Under certain assumptions, it is shown that the dynamic evolution of such feedback systems can be modelled by a finite-dimensional discrete-time hybrid-state dynamic system. An example of a thermal control system is given.

*Senior Scientist; Bolt Beranek and Newman, Inc.; 10 Moulton Street; Cambridge, MA 02238
Lecturer; Dept. of Electrical Engineering and Computer Science; M.I.T., Room 35-205B; Cambridge, MA 02139 (USA)

References

[1] M.G. Kienzle and K.C. Sevcik, "Survey of Analytic Queueing Network Models of Computer Systems," Proc. 1979 Conf. on Simulation, Measurement and Modelling of Computer Systems, pp. 113-129.

[2] G.K. Hutchinson and J.J. Hughes, "A Generalized Model of Flexible Manufacturing Systems", Proc. Multi-Station Digitally-Controlled Mfg. Systems Workshop, Milwaukee, Wisc. Jan. 1977; pp. 88-115.

[3] Shannon, R.E., Systems Simulation: The Art and Science, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

[4] Pritsker, A.A.B., The GASP IV Simulation Language, J. Wiley & Sons, New York, 1974.

[5] Johnson, T.L., "Finite-State Compensators for Continuuous Processes", Proc. 7th IFAC Congress, Helsinki, Finland, June 1978, pp. 1823-1828.

[6] Kaliski, M.E. and Lemone, K., "Discrete Codings of Continuous Valued Signals", Proc. 14th Ann. Conf. on Info. Sci. and Sys., Johns Hopkins Univ., March 1980.

[7] Utkin, V.I., Sliding Modes and Their Application to Valuable-Structure Systems, MIR Publishers, Moscow, 1978.

[8] Johnson, T.L., "Stability of Diced Systems", Proc. 19th IEEE Conf. on Decision and Control, Albuquerque, N.M., Dec. 1980.

## I. Introduction

The design of real-time computer control systems for distributed processes has been hampered by the lack of adequate analysis tools. Since the evolution of state in such systems depends on the timing of process events as well as computing tasks (and not just the sequence of completed tasks), conventional finite-state or differential-equation models are inadequate to predict details of task coordination and sequencing which are critical for design purposes. For multitask systems, a class of hybrid-state discrete-time models is shown to be applicable to this problem; other real-time operating systems may require more general models (Gonzalez, 1977).

Even for relatively simple systems, the analysis of timing is complex. An example of a furnace temperature controller with overheat protection is given to illustrate this fact.

Qualitative properties of hybrid systems of this class have been studied in Sontag (1981), Johnson (1980). Some design methods are described in Vidal (1969). General controller synthesis methods are not yet available for such systems.

## II. Model Development

In order to be most useful, the model employs a level of aggregation suitable for predicting events on the time-scale of the controlled process, but not on the (typically) much shorter timescales characterizing register transfers and bus transactions. The dynamics of the task scheduler are thus viewed as being instantaneous: the task scheduler polls a subset of computer memory, as well as external inputs, at a very high rate to determine which task initiation and completion criteria are satisfied. At any instant when such criteria are met, all corresponding tasks are serviced in order of priority. When a task is initiated, its input data are read instantaneously and a fixed computation is performed; any time after a fixed time allowed for the computation, and upon completion of a task termination condition, the results of the task computation are deposited in memory or used to actuate an external device or process (I/O registers are assumed to be memory-mapped). The task scheduler is assumed to keep track of task completion times (e.g., by setting a counter-timer for each task); in this model, expiration of task computation time is viewed as a precondition for task completion, but the possibility of task abortion can be allowed by removing this restriction and adopting a convention about what action is taken by the scheduler when an abort occurs. Whether the scheduler operates sequentially or in parallel, whether or not special-purpose hardware is involved, and what the detailed software implementation of the scheduler is, are all irrelevant to this model, provided only that it operates on a much faster time scale than the process.

The transition between tasks may depend on the values of continuous or discrete process variables, and it is the dynamics of the joint process that is of interest. Let $t\varepsilon(t_o,\infty)$ denote time. The state set $X=\{X^1,X^2\}$ is the product space of $X^1=$ {those states which are discrete-valued, including all discrete memory states}, and $X^2$ = {those states which vary continuously with time and take on real values}. Let the set of tasks in the system be denoted $G=\{G_1,\ldots,G_n\}$. Associated with each task is a task initiation function $g_j^I$: $X\rightarrow\{0,1\}$, a task termination function $g_j^T$: $X\rightarrow\{0,1\}$, and a state update function $f_j$: $X\times X\rightarrow X$. Each task is either "on", or "off". Let $q^1=\{q_1^1,\ldots q_n^1\}\varepsilon 2^n$ denote those tasks which are "on", i.e., $q_j^1=1$ when task j is "on", $q_j^1=0$ when it is "off". A task transition time, $t_K$, is declared whenever

$$T=[\bigvee_{j=1}^{n}(g_j^I\wedge\bar{q}_j^1)]\vee[\bigvee_{j=1}^{n}(g_j^T\wedge q_j^1)] \qquad (1)$$

makes a $0\rightarrow1$ transition, i.e., an inactive task is initiated or an active task is terminated. Between task transition times, the states in $X^1$ are assumed to be constant*, and the states in $X^2$ are assumed to evolve according to a time-invariant differential equation

$$\dot{x}^2(t) = f(x^2(t),x^1(t_K)) \qquad t\varepsilon[t_k,t_{k+1}) \qquad (2)$$

where f is assumed to be uniformly Lipschitz continuous on $X^2$ for each $x^1\varepsilon X^1$. Let $\phi$: $[0,\infty)\times X\rightarrow X$ be the transition mapping of (2), defined for each $x^1\varepsilon X^1$, such that the solution of (2) with initial data $x_k^1=x^1(t_k)$, $x_k^2=x^2(t_k)$ is

*This assumption can be relaxed by introducing additional transition times for spontaneous discrete-state transitions, if desired. The assumption is satisfied, for instance, when there are no discrete-state memory devices external to the computer memory, and when only the task scheduler is allowed to change the memory states $x^1$.

$$x^2(t) = \phi(t-t_k, x_k^1, x_k^2) \quad t\in[t_k,\infty) \tag{3}$$

For any $(x_k^1, x_k^2)\in X$, $q_k^1 \doteq q^1(t_k)\in 2^n$, (3) defines the functions

$$q_{k+1}^2 \doteq t_{k+1} = \tau(q_k^1, x_k^1, x_k^2) = \inf_{t\geq t_k} \{T(t) \mid \lim_{\tau\uparrow t} T(\tau)=0, \lim_{\tau\downarrow t} T(\tau)=1\} \tag{4}$$

$$q_{j,k+1}^1 \doteq \begin{cases} \sigma_j(q_k^1, x_k^1, x_k^2) \\ 0 \ ; \ q_{j,k}^1=1, g_j^T(x(\tau_k^-))=0, g_j^T(x(\tau_k^+))=1 \\ 1 \ ; \ q_{j,k}^1=0, g_j^I(x(\tau_k^-))=0, g_j^I(x(\tau_k^+))=1 \\ q_{j,k}^1 \ ; \ \text{otherwise} \end{cases} \tag{5}$$

for $j=1,\ldots,n$

Thus, the next set of active tasks and the next transition time can be defined from the current state; but the state-update function on X must still be defined. In order to do this, it is important that task completion priorities be defined, because if two tasks complete simultaneously, the state-update function of one may overwrite the other, and the order of overwrite will in general affect the future evolution of the system; it is assumed that tasks have been numbered in order of completion priority. Thus, let $j_{1,k},\ldots j_{m,k}$, $m\leq n$, denote the indices of the tasks completing at $t_{k+1}$ (i.e., those corresponding to the first case in (5)), arranged in increasing numerical order. At time $t_{k+1}$, $x_k$ is updated according to:

$$\begin{aligned} x_{k+1} &= \eta(q_k, x_k) \\ &= f_{jm}(x_k, f_{jm-1}(x_k, \ldots f_{j1}(x_k, \phi(t_{k+1}-t_k, x_k)))\ldots) \end{aligned} \tag{6}$$

The task state-update functions $f_j$ must thus take as their arguments the state at time of task initiation and at time of task completion; these functions are composed with one another in order of completion priority*. The convention $q_k=(q_k^1, q_k^2)'$, $x_k=(x_k^1, x_k^2)'$ has been used in (6). Rephrasing (4)-(5) and defining the state of the combined process as

$$\tilde{x} = \begin{bmatrix} x^1 \\ q^1 \\ x^2 \\ x^1 \end{bmatrix} \quad ; \quad \tilde{x}_{k+1} = \begin{bmatrix} \eta^1(\tilde{x}_k) \\ \sigma(\tilde{x}_k) \\ \eta^2(\tilde{x}_k) \\ \tau(\tilde{x}_k) \end{bmatrix} \tag{7}$$

where $\eta^1$ and $\eta^2$ denote the projection of $\eta$ onto $X^1$ and $X^2$, respectively, and the arguments of $\eta, \sigma$ and $\tau$ appear as subsets of the elements $\tilde{x}_k$, respectively.

The well-posedness of this model depends on existence and uniqueness of solutions to (2), for which sufficient conditions have been stated, and also on the property $\lim_k t_k = \infty$. In some cases, this limit may be achieved for finite k, which is acceptable; in others, $\lim_{k\to\infty} t_k$ may be finite, which is not. The latter case may be prevented by placing a positive lower bound on the range of the function (e.g., related to the cycle time of task scheduler).

## III. Example

The temperature control of a room by a furnace which is subject to overheating is considered. The room is equipped with a temperature sensor and the combustion chamber is equipped with

*The definition of "priority" is a question of semantics; the task which has highest termination priority may have its results overwritten by lower-priority tasks which complete afterward, according to the convention adopted here.

an overheat indicator. When the room temperature is below a specified setpoint, furnace ignition occurs, unless (i) the furnace has been extinguished due to room temperatures exceeding setpoint within the last $d_1$ minutes, or (ii) the furnace has been extinguished due to overheating within the last $d_2$ minutes. Imagine that a computer performs these simple logic functions and that (e.g., for reasons of available hardware), the delays $d_1$ and $d_2$ are implemented by fixed-interval analog timers which are set by the computer to logic states $T_1=1$, $T_2=1$, respectively, and reset at elapsed time to $T_1=0, T_2=0$. The operation of the furnace is controlled by the state of the bit F, where F=0 denotes "off". The output of the overheat indicator is the bit H, which is 1 when overheat occurs. Three tasks may be defined for this system:

(j=1) Turn the furnace on when it is currently off, timers have elapsed, and room temperature falls below setpoint.

(j=2) Turn the furnace off when it is currently on and the room temperature exceeds setpoint. Set $T_1$.

(j=3) Turn the furnace off when it is currently on and the combustion chamber temperature overheats, setting H. Set $T_2$.

The dead-time $d_1$ should be chosen comparable to the room time-constant and the delay $d_2$ should be chosen comparable to the combustion chamber thermal time constant for proper operation, but the model will yield correct performance predictions even if these design objectives are not satisfied. Normally, the

sequencing of tasks alternate between j=1 and j=2, but the sequences j=1,2,3 and j=1,3 are also possible, depending on the parameters of the continuous system. Since the task computations are so simple, it is assumed in this example that they are completed instantaneously.

Suppose that the dynamics of the room temperature, R, are given by

$$\dot{R}=-\alpha_1 R+\beta_1 F+\delta_1 \tag{8}$$

where

$\alpha_1$ = inverse of room thermal time constant (min.)

$\beta_1$ = heat capacity of furnace

$\delta_1$ = constant depending on ambient temperature

The dynamics of the combustion chamber temperature, C, are given by

$$\dot{C} = -\alpha_2 C+\beta_2 F+\delta_2 \tag{9}$$

where

$\alpha_2$ = inverse of combustion chamber thermal time constant (min.)

$\beta_2$ = heat capacity of combustion process

$\delta_2$ = constant depending on ambient temperature.

Obviously, more complex distributed-parameter or nonlinear radiation models could be chosen in place of (8)-(9). The room-temperature setpoint is denoted by $\bar{R}$ (implemented in the computer), while the overheat setpoint is denoted by $\bar{C}$. Define the threshold function

$$Q_{\bar{x}}(x) = \begin{cases} 1 & x \geq \bar{x} \\ 0 & x < \bar{x} \end{cases} \tag{10}$$

The dynamics of the timers are given by

$$\dot{D}_1 = -1 \tag{11}$$

$$\dot{D}_2 = -1 \tag{12}$$

Define the binary variables

$$M = q_{\bar{R}}(R) \tag{13}$$

$$H = q_{\bar{C}}(C) \tag{14}$$

$$T_1 = q_o(D_1) \tag{15}$$

$$T_2 = q_o(D_2) \tag{16}$$

These may all be considered as input bits, even though (13) might be evaluated by the task scheduler. The continuous states may then be identified as $x^2=[R,C,D_1,D_2]'$. A discrete state is $x^1=F$ representing the current furnace status. Table 1 gives initiation, termination, and state-update functions for each task.

| Task | Initiation | Termination | State-update* |
|---|---|---|---|
| j | $g_j^I(x_k)$ | $g_j^T(x_k)$ | $f_j(x_k, \bar{x}_{k+1})$ |
| 1 | $\bar{T}_1 \wedge \bar{T}_2 \wedge \bar{F} \wedge \bar{M}$ | 1 | $f^1 = 1$<br>$f^2 = x^{2-}_{k+1}$ |
| 2 | $F \wedge M$ | 1 | $f^1 = 0$<br>$f^2 = [\bar{R}_{k+1}, \bar{C}_{k+1}, d_1, \bar{D}_{2,k+1}]$ |
| 3 | $F \wedge H$ | 1 | $f^1 = 0$<br>$f^2 = [\bar{R}_{k+1}, \bar{C}_{k+1}, \bar{D}_{1,k+1}, d_2]$ |

Table 1: Task Parameters for Example

* $\bar{x}_{k+1}$ is shorthand for $\phi(t_{k+1}-t_k, x_k)$ regarded as a function of $x_k$; see (4),(6).

In order to define the functions $\tau$ and $\sigma_j$ (eq. (3), (4)) of the model, note that the transition function for (8) can be written

$$R(t) = [(\delta_1+\beta_1 F)/\alpha_1] + e^{-\alpha_1(t-t_k)}[R(t_k)-(\delta_1+\beta_1 F)/\alpha_1] \qquad (17)$$

and an analogous expression is obtained for C(t).

Since all tasks complete in zero time, it may be assumed that $q_k^1 \equiv \{000\}$ in (4). Suppose that F=0. Then only Task 1 can be initiated next, and the time for this to occur is

$$\tau(0,0,x^2) = \begin{cases} \max(D_1,D_2) & ; \quad T_1 \wedge T_2 = 1, \quad M=0 \\ \max(D_o,D_1,D_2) & ; \quad T_1 \wedge T_2 = 1, \quad M=1 \\ 0 & ; \quad T_1 \wedge T_2 = 0, \quad M=0 \\ D_o & ; \quad T_1 \wedge T_2 = 0, \quad M=1 \end{cases} \qquad (18)$$

where

$$D_o = (1/\alpha_1)\ell n \left[\frac{R-\delta_1/\alpha_1}{\bar{R}-\delta_1/\alpha_1}\right] \qquad (19)$$

is the time for room temperature, R, to cool to $\bar{R}$ if $R>\bar{R}$, found from (17).

If F=1, then either Task 2 or Task 3 may occur first, depending on the parameters of the problem and the value of the states R and C in $x^2$ (since $T_1=0, T_2=0$ are required when F=1). The time for R to reach $\bar{R}$ when $R \leq \bar{R} \leq (\delta_1+\beta_1)/\alpha_1$ is

$$D_2 = (1/\alpha_1)\ell n \left[\frac{R-(\delta_1+\beta_1)/\alpha_1}{\bar{R}-(\delta_1+\beta_1)/\alpha_1}\right] \qquad (20)$$

And the time for C to reach $\bar{C}$ when $C \leq \bar{C} \leq (\delta_2+\beta_2)/\alpha_2$ is

$$D_3 = (1/\alpha_2)\ell n \left[\frac{C-(\delta_2+\beta_2)/\alpha_2}{\bar{C}-(\delta_2+\beta_2)/\alpha_2}\right] \qquad (21)$$

It is assumed for convenience that $\bar{R},\bar{C}$ are set to satisfy the righthand inequalities, which would normally be the case. The locus of values where these are equal is given by

$$C-(\delta_2+\beta_2)/\alpha_2 = (R-(\delta_1+\beta_1)/\alpha_1)\left[\frac{\bar{C}-(\delta_2+\beta_2)/\alpha_2}{\bar{R}-(\delta_1+\beta_1)/\alpha_1}\right]^{(\alpha_1/\alpha_2)} \tag{22}$$

and it is readily verified that either time may occur sooner under reasonable assumptions. Thus

$$\tau(0,1,x^2) = \begin{cases} 0 & M\vee H=1 \\ \min(D_2,D_3) & M\vee H=0 \end{cases} \tag{23}$$

It is also apparent that (15) corresponds to

$$\begin{aligned} \sigma_1(0,0,x^2) &= 1 \\ \sigma_1(0,1,x^2) &= 0 \end{aligned} \tag{24}$$

since all conditions in (18) initiate Task 1, while all conditions in (23) preclude Task 1, and

$$\begin{aligned} \sigma_2(0,0,x^2) &= 0 \\ \sigma_2(0,1,x^2) &= (M)\vee[\bar{M}\wedge\bar{H}\wedge(D_2\leq D_3)) \end{aligned} \tag{25}$$

since Task 2 can only be initiated when F=1 and the other conditions from (23) are met, and

$$\begin{aligned} \sigma_3(0,0,x^2) &= 0 \\ \sigma_3(0,1,x^2) &= (H)\vee(\bar{M}\wedge\bar{H}\wedge(D_3\leq D_2)) \end{aligned} \tag{26}$$

for similar reasons. Note that the third condition of (5) was not invoked because the task completion times were assumed to be zero. In addition, (25) and (26) are not mutually exclusive, so that both tasks could be initiated concurrently if M=H=1 or if $D_2=D_3$.

In this case, the order of invoking the tasks is immaterial so that (6) does not depend on task priority. Mutual exclusion of the tasks could be represented, if desired, by modifying the state $x^1$ to include status information corresponding to $q^1$.

In addition to illustrating the application of this modelling procedure, the example has been concoted to illustrate the versatility of conditions which can be represented by the multitask formalism.

## IV. Conclusion

The salient assumptions of this modelling procedure are: (1) the number of tasks is fixed and finite, (2) tasks cannot modify their own computations, (3) bounds on completion times of each task are known, (4) tasks are assigned fixed priorities, (5) the time scale of the task scheduler is much faster than the process time scale of interest in the model. While many multitasking systems are conceptualized in a more general manner which violates one or more of these assumptions, it is often possible (with some inconvenience) to represent them in a form which satisfies these constraints through appropriate choice of state variables.

Finally, an important objective of this paper is to illustrate conclusively that traditional infinite-dimensional continuous process control systems are very inadequate for the representation of typical computer-controlled processes, and that substantial further research in this area is required.

## V. References

Sontag, E.D., "Nonlinear Regulation: The Piecewise Linear Approach", IEEE Trans. Auto. Contr., Vol. AC-26, No. 2, pp. 346-357 (April 1981).

Johnson, T.L., "Stability of Diced Systems", Proc. 19th IEEE Conf. on Decision & Control, pp. 1110-1115, Albuquerque, NM, (December 1980).

Gonzalez, M.J., "Deterministic Processor Scheduling", ACM Computing Surveys, Vol. 9, No. 3, pp. 173-204 (Sept. 1977).

Vidal, P., Nonlinear Sampled-Data Systems, Gordon & Breach, New York, NY, 1969.

Appendix F

# ON FEEDBACK LAWS FOR ROBOTIC SYSTEMS

T. L. Johnson
Senior Scientist, Control Systems Department
Bolt Beranek and Newman Inc., 50 Moulton Street
Cambridge, Massachusetts

and

Lecturer, Dept. of Electrical Engineering & Computer Science[1]
M.I.T. Room 35-205B
Cambridge, Massachusetts

## INTRODUCTION

A variety of control problems arising from robotics applications can be restated as optimal control problems of minimum-time state transfer in the presence of state-space constraints and constraints of incomplete-state information. The traditional approaches to solving such problems are Pontryagin's Maximum Principle (Pontryagin et.al., 1962), in the case of open-loop control, and Bellman's Dynamic Programming method (Bellman, 1957). While a number of technical difficulties exist, approximate solutions of such problems can generally be computed off-line (see Kahn and Roth, 1971). Perturbation methods for obtaining local feedback laws are also available (Whitney, 1972, Hemami, 1980).

However, no currently operational robots are known to be based on solutions of such optimal control problems, nor is it likely that this will come about. Some of the reasons for this situation can be given: (a) complete equations of motion are extremely complex, and are often not available; (b) trajectories must usually be planned in a short time-period preceding execution--there is no time for detailed design studies or numerical analysis for every motion being performed; (c) reliability and repeatability or accuracy of motion are often more important than minimizing execution time; (d) optimal control laws often require too much storage or real-time computation during execution of the motion; (e) nonlinearities are often sufficiently severe that local linearization gives poor results (even if its heavy computational requirements are overlooked).

By contrast, current practice is often to determine a feasible open-loop position trajectory by a "teaching" procedure (e.g., Unimation, Inc., 1979). The trajectory recorded during this procedure is then "played-back" as a sequence of position commands to joints which are servo-actuated; the rate of playback may be increased in a sequence of preliminary trials, until the bandwidth or power limitations of the servos are encountered. This methodology is relatively quick, intuitive, and yields reliable performance when the disturbances to the robot and workspace are relatively small. Although this state-of-the-art approach to trajectory formation is very effective, it possesses inherent limitations and is already being superceded in more demanding applications such as locomotion and manipulation. One limitation is that a human controller cannot readily communicate commands to such a robot. The robot is also unable to anticipate or accommodate unexpected changes in workspace configuration; the teaching paradigm cannot be readily extended to allow for feedback from additional sensors (e.g., touch or machine vision). The objective of the present note is to extend and affirm the suggestion of Young (1978) that discontinuous feedback laws are naturally-suited to robotics problems, to describe two further examples of discontinuous feedback laws, and to explore further notions for the synthesis of such systems.

### Rationale for Discontinuous Feedback Laws

Accepting the fact that optimal feedback laws for this class of problems generically exhibit discontinuous behavior (Athans and Falb, 1966, Kahn and Roth, 1971), one is motivated to seek simpler methods of determining loci of discontinuity. The theory of variable-structure controllers, developed originally by Emel'yanov (1967) and extended by his colleagues (see Utkin, 1978) has provided new design methods for certain classes of systems; it is a remarkable observation that the performance of such systems can be qualitatively quite robust, even though their precise trajectories may depend strongly on the initial state, disturbances, or modelling errors (Young, 1978).

[1]Portions of this research have been performed at the MIT Laboratory for Information and Decision Systems with support provided by the U.S. Air Force Office of Scientific Research under Contract No. F49620-80-C-0002.

The author has previously suggested (Johnson, 1978) that there is a close relationship of this theory and the theory of control laws described in linguistic terms, e.g., as a digital computer program (Zadeh, 1973). Example 1, in the sequel, exhibits this relationship. Discontinuous control in robotics applications can thus arise from the nature of the task description as well as from discontinuities in the mechanical system and environment, as illustrated by Example 2. A third reason for developing discontinuous controls arises from implementation considerations. Discrete sensors and actuators are usually cheaper and more reliable than continuous ones; they arise naturally in discontinuous control law synthesis. Discrete signals are also preferred for signalling task initiation, completion, or interrupts to a control computer. Finally, digital computers typically perform binary operations faster than (approximations to) real number operations.

The following two examples illustrate the use discrete feedback control in two very simplified problems arising in robotics, which lie just beyond the current state-of-the-art. Since a general design theory for such cases is not yet available, each example is solved on its own merits.

Example 1: Catching a Ball

In this example, the "hand" is idealized as a cup-shaped weight of mass M which can be acted on by vertical and horizontal forces in order to catch a (vertically) falling ball of mass m. First, it is assumed that the hand is beneath the ball and the interception dynamics are analyzed. Then, a simple control law to achieve catching from an arbitrary initial position, using remote sensing of the position of the ball (a primitive form of vision), is given in algorithmic form.

The geometry of the problem is shown in Figure 1. Suppose that $x_m(t) = x_M(t) = 0$ to analyze the catching process. According to Newtonian mechanics, the ball's motion is given approximately[2] by

$$\ddot{mz}_m = -mg \quad ; \; z_m(t_o) = z_{mo} \; ; \; \dot{z}_m(t_o)=0 \qquad (1)$$

where g is the acceleration due to gravity and $z_{mo}$ is the initial position of the ball at $t_o$, the time it is dropped. The motion of the hand is given by

$$\ddot{Mz}_M = -Mg + f_z(t); \; z_M(t_o) = z_{Mo}; \; \dot{z}_M(t_o)=0 \qquad (2)$$

where $f_z(t)$ represents the control force.

[2] In air, a viscous drag force depending on cross-sectional area is also present, and could be used in estimating the ball's mass. This digression is not pursued here.

Suppose that $t_1$ is the first time of contact between ball and hand, and let $t_1^-$ denote a time just prior to $t_1$ while $t_1^+$ denotes a time just after $t_1$. Assuming $f_M(t)$ is approximately constant on the interval $(t_1^-, t_1^+)$, it can be set to zero without affecting the conclusions of the following analysis; this is done for simplicity in the sequel. Either an elastic or inelastic collision may occur at $t_1$. If an inelastic collision occurs, and the ball is caught, the combined dynamics for $t>t_1$ are

$$\begin{aligned} (M+m)\ddot{z}_M &= -(M+m)g + f_z(t); \; z_M(t) \equiv z_m(t), \quad t \geq t_1 \\ z_M(t_1) &= z_{M1} = z_{m1}, \text{ the location of impact} \\ \dot{z}_M(t_1) &= ? \end{aligned} \qquad (3)$$

If an elastic collision occurs, then

$$\begin{aligned} \ddot{mz}_m &= -mg, \; z_m(t_1) = z_{m1}; \; \dot{z}_m(t_1) = ? \\ \ddot{Mz}_M &= -Mg + f_z(t); \; z_M(t_1) = z_{M1} = z_{m1}; \\ & \qquad \dot{z}_M(t_1) = ? \end{aligned} \qquad (4)$$

Conservation of energy and momentum can be invoked (now taking $f_z(t)=0$) in order to deduce which of these situations will occur, and to find the missing velocites at $t=t_1$. Conservation of momentum is

$$P \doteq m\dot{z}_m(t_1^-) + M\dot{z}_m(t_1^-) = m\dot{z}_m(t_1^+) + M\dot{z}_M(t_1^+) \qquad (5)$$

while conservation of energy (omitting potential energy, which is approximately constant, from both sides of the equation) is

$$E \doteq m\dot{z}_m(t_1^-)^2 + M\dot{z}_M(t_1^-)^2 = m\dot{z}_m(t_1^+)^2 + M\dot{z}_M(t_1^+)^2 \qquad (6)$$

Viewing these as simultaneous equations for $\dot{z}_m(t_1^+)$ and $\dot{z}_M(t_1^+)$, bouncing is predicted when there is a solution with $\dot{z}_m(t_1^+) > z_M(t_1^+)$[3]. A simultaneous solution yields the possibilities

$$\dot{z}_m(t_1^+) = \{P \pm [mM(E(M+m)-P^2)]^{1/2}\}/m(M+m) \qquad (7)$$

As a special case, suppose that $\dot{z}_M(t_1^-)=0$, i.e., the hand is at rest at the time of impact. Then it can be shown that a real-valued solution of (7) always exists and that

[3] Otherwise, in an inelastic collision, energy dissipation will occur at $t_1$ so that the physically realizable solution $\dot{z}_m(t_1^+) = \dot{z}_M(t_1^+)$ comes about. This is not explored further here.

$$\dot{z}_m(t_1^+) = \frac{(m-M)}{(m+M)}\ \dot{z}_m(t_1^-) \qquad (8)$$

Thus bouncing will occur whenever M>m, which is typically the case. A further analysis shows that if M>m, a finite negative velocity of the hand prior to impact ($\dot{z}_M(t_1^-)<0$) will prevent bouncing; in the limit $m\approx 0, \dot{z}_M(t_1^-)\approx \dot{z}_m(t_1^-)$, i.e., perfect tracking will be required; if the ball is very heavy (m>M), or has a very large velocity at impact, then a catch can be made even if $\dot{z}_M(t_1^-)$ is positive, i.e., if the hand comes to meet it. Typically, one expects m<M but not m<< M, so that a very small movement to produce a slightly negative hand velocity prior to impact will ensure a successful catch.

In a catch, the hand must merely intercept the ball's predicted trajectory before the ball arrives at the point of interception, and then wait to make a small final maneuver to avoid bouncing. If the ball is to be struck, (say, in the x-direction) quite a different strategy is required: The ball's trajectory must be intercepted precisely at the time the ball reaches the interception point, with a velocity which is approximately perpendicular to the trajectory.

Now suppose that the ball's position, $x_m(t)$, $z_m(t)$ can be measured, that the hand position $x_M(t)$, $z_M(t)$ is available from internal measurement, and that forces $f_x(t)$ and $f_z(t)$ can be applied independently. Assume that accurate velocity estimates can be obtained from the position measurements. At $t=t_o$, the initial time, suppose $z_m(t_o)=z_{mo}$, $x_m(t_o)=0$, while $z_m(t_o) = z_{Mo}<z_{mo}$, $x_M(t_o) = x_{Mo}$. A simple implementation of the rendezvous strategy for catching the ball is the following pseudo-Pascal algorithm:

```
PROCEDURE CATCH
BEGIN
REPEAT
        f_z(t) = 0
        e_x(t) = x_M(t)-x_m(t)
        f_x(t) = -K_x1 e_x(t)
UNTIL  |e_x(t)|<E_x
i_x = 0
REPEAT
        fz_M(t) = -K_z*(ż_M(t)-ż_m(t))
        e_x(t) = x_M(t)-x_m(t)
        i_x = i_x + Δe_x(t)
[Δ is the sampling interval]
        f_x(t) = -K_x2 i_x
UNTIL z_m(t) ≤ z_M(t) + E_z
IF  |z_M(t)-z_m(t)| ≤ E_z AND |x_M(t)-x_m(t)| ≤ E_x
    THEN RETURN
    ELSE [MISSED THE BALL, GO TO ERROR
          RECOVERY]
END
```

The first REPEAT loop uses position feedback on the x-position error (intended with a "large" gain $K_{x1}$) to bring the hand below the ball as fast as possible. The second REPEAT loop uses integral control on the x-error to more accurately position the hand below the ball, and derivative feedback on the z-velocity error (intended with a "small" gain, $K_z^*$) so that the hand has a small downward velocity when the ball strikes it. Although the details of this control law are essentially irrelevant, it is primarily intended to illustrate two points: (a) it is not necessary to explicitly predict the trajectory of the ball (i.e., to preplan the trajectory) or to know the precise mass of the ball; (b) The control strategy is discontinuous at the time between the two REPEAT loops, which is determined by the motion of the ball itself. In the second example, the control law discontinuity arises primarily from state-variable constraints rather than from the task description.

## Example 2: Converting Vertical Force to Horizontal Locomotion

A single massless link of length $\ell_0$ terminated at the upper end by a mass $m_1$ and at the lower end by a mass $m_0$, is considered in the example (Figure 2). A vertical force, F(t), may be applied to the upper mass: When this force lifts the link above a horizontal surface at z=0, it is free to swing back and forth in one direction (defined as the x-direction); when mass $m_0$ is in contact with the surface, it "sticks" unless an upward vertical force component is subsequently applied to it. This assumption approximates the effect of a friction contact between $m_0$ and the surface.

The intriguing feature of this example is that there exist simple strategies whereby the purely vertical force F(t) can be used to propel the link in a forward horizontal motion. These result from a proper combination of two motions:

F: The link falls down (like an inverted pendulum) when $m_0$ is on the surface and no vertical force is applied (F(t)=0).

S: The link swings back and forth in a stable pendulum motion when $m_0$ is off the surface and a vertical force is applied to counteract gravity ($F(t)\geq(m_0+m_1)g$).

obstacles has been met. A second goal, of varying the speed of locomotion, can be met by varying $\theta_{min}$ parametrically. The time per cycle is roughly related to the area enclosed by the periodic trajectory, while the horizontal distance is approximately $\ell_0(\theta_{max} - \theta_{min})$; the ratio of distance to time is an approximate measure of average forward velocity. The range of achievable velocities with this locomotion strategy is rather small, even though the corresponding range of step sizes (between 0 and $2\ell_0$) is rather large. The margin of stability of the larger step sizes is considerably decreased, however.

The continued forward motion of $m_1$ does not violate conservation of momentum; the initial forward momentum is conserved during motion S, but during motion F, it is augmented by momentum exchange, which occurs due to the constraint that $m_0$ remain fixed on the surface. Thus, the energy expended in lifting during motion S can in fact be converted into forward acceleration during motion F, and the forward motion will not die out (e.g., due to friction effects). No laws of physics are violated by this strategy.

## CONCLUSIONS

The examples, drawn from two different areas of robotics, illustrate that discontinuous feedback laws are readily devised for a variety of applications. In both examples, the feedback law could be viewed as a finite set of mutually exclusive continuous-control subtasks. In the first example, two different linear control laws were used, while in the second example, two different constant values of control were used. Furthermore, the transitions between tasks were closely tied to events in the (full) state space which were readily detectable, e.g. interception and rendezvous in the first example, and contact with the surface z=0 in the second. These examples illustrate that a generalization of the methods employed by Young (1978) may be useful in future robotics applications. A set of control values or continuous-control feedback laws sufficiently rich to control the motion of the system in each of its known or desired states is chosen. The trajectories of the system under these forms of feedback are computed. The switching loci between the control laws are then taken along the loci of intersection of these trajectories, or along a physical constraint locus of the system motion, in such a way that the desired combination of movements is obtained. This has the effect of assuring a well-defined mode of sliding along the discontinuities of the closed-loop system; otherwise the nature of sliding might change markedly and unpredictably within a discontinuity surface due to trigonometric-type nonlinearities of the equations of motion.

The selection of a finite number of candidate control strategies and the choice of switching loci defined by intersections of natural motions of the system under these candidate control laws appear to be primary requirements for a practical design theory of discontinuous control for robotic systems. Presently, the greatest difficulties in the development of such a theory are the relationship of linguistically-described goals to feedback law selection, the lack of analytic methods for characterizing controlled motions of the system, and the inherent difficulties of stability analysis for discontinuous systems (Johnson, 1980).

## REFERENCES


Athans, M., and P. Falb (1966). Optimal Control. McGraw-Hill Book Co., New York, NY.

Bellman, R. (1957). Dynamic Programming. Princeton University Press, Princeton, NJ.

Emel'yanov, S.V. (1967). Sistemy automaticheskogo upravkniya s peremennoi strukturoi. (Variable Structure Control Systems), Nauka, Moscow.

Hemami, H. (1980). A Feedback On-Off Model of Biped Dynamics. IEEE Trans. Syst., Man & Cybern.; Vol. SMC-10; No. 7; pp. 376-383.

Johnson, T.L. (1978). Finite State Compensators for Continuous Processes. Proc. 7th IFAC World Congress, Helsinki, Finland, pp. 1823-1831.

Johnson, T.L. (1980). Stability of Diced Systems. Proc. IEEE. CDC Albuquerque, NM.

Kahn, M.E. and B. Roth (1972). The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains. J. Dyn. Syst. Meas. & Control; pp. 164-172.

Pontryagin, L.S., V. Botyanskii, R. Gamkreldize, and E. Mishchenko (1962). The Mathematical Theory of Optimal Processes. Interscience Publishers, Inc., NY.

Unimation, Inc. (1979). A User's Guide to Val: A Robot Programming and Control System. Danbury, CT.

Utkin, V.I. (1978). Sliding Modes and Their Application in Variable Structure Systems. Mir Publishers, Moscow.

Whitney, D.E. (1972). The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators. J. Dyn. Syst. Meas. & Control. pp. 303-309.

Young, K.D. (1978). Controller Design for a Manipulation Using Theory of Variable Structure Systems. IEEE Trans. Syst., Man, & Cybern. Vol. SMC-8; No. 2; pp. 101-109.

Zadeh, L.A. (1973). Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. IEEE Trans. Syst., Man & Cybern. Vol. SMC-3, pp. 28-44.

The equations of motion are first derived in the two cases where $m_0$ is not in contact with the surface z=0 (Case S), and then when it is in contact (Case F).

Case S: Let $F_{01}$ denote the force on $m_0$ exerted through the link by $m_1$, and $F_{10}$ denote the force on $m_1$ exerted by $m_0$, defined in the direction of the link for each mass. Newton's equations for $m_0$ are

$$m_0\ddot{z}_0 = -m_0g+F_{01}\sin\theta_0 \quad (9)$$

$$m_0\ddot{x}_0 = F_{01}\cos\theta_0 \quad (10)$$

And for $m_1$ they are

$$m_1\ddot{z}_1 = F - m_1g - F_{10}\sin\theta_0 \quad (11)$$

$$m_1\ddot{x}_1 = -F_{10}\cos\theta_0 \quad (12)$$

where g denotes the acceleration due to gravity. The constraint of equal and opposite reactions (rigid link) is $F_{01}=F_{10}$. The link imposes constraints between $(x_0,z_0)$ and $(x_1,z_1)$ which are most readily expressed in terms of $\theta_0$:

$$x_1 = x_0 + \ell_0\cos\theta_0 \quad (13)$$

and

$$z_1 = z_0 + \ell_0\sin\theta_0 \quad (14)$$

The time-derivatives of the constraints are used because the constraints must hold at each instant of time. Elementary algebra and trigonometry can be used to solve for $F_{01}$ and $F_{10}$ in (9) and (10). Further algebra yields the key equation for $\theta_0$:

$$\ddot{\theta}_0 = F\cos\theta_0/m_1\ell_0 \quad (15)$$

In this example it is natural to assume that inertial measurements could be made only on $m_1$, and thus it is of interest to have equations of motion directly in terms of the inertially measured states $(x_1,z_1)$ rather than $(x_0,z_0)$. These equations are:

$$\ddot{x}_1 = -(m_0\ell_0/(m_0+m_1))\cos\theta_0\dot{\theta}_0^2 - (m_0/m_1(m_0+m_1))\sin\theta_0\cos\theta_0F \quad (16)$$

$$\ddot{z}_1 = -(m_0\ell_0/(m_0+m_1))\sin\theta_0\dot{\theta}_0^2 - g + [\cos^2\theta_0/m_1+\sin^2\theta_0/(m_0+m_1)]F \quad (17)$$

Purely algebraic constraints (13)-(14) can be used to find $(x_0,z_0)$ and to check that $z_0>0$; otherwise a transition to Case F may occur. Furthermore, note that (16), the forward acceleration of $m_1$, is driven by the vertical force F, providing the possibility of locomotion.

Case F: Let $F_{01}$ and $F_{10}$ be defined as in Case S. During Case F, it is assumed that $(x_0,z_0)$ remain fixed at their initial values, and that $z_0=0$. Newton's equations for $m_1$ are

$$m_1\ddot{z}_1 = F - m_1g - F_{10}\sin\theta_0 \quad (18)$$

$$m_1\ddot{x}_1 = -F_{10}\cos\theta_0 \quad (19)$$

In differentiating the constraints (13) and (14), $x_0$ and $z_0$ are held constant. The equation for $\theta_0$ is derived in a similar fashion to Case S:

$$\ddot{\theta}_0 = F\cos\theta_0/m_1\ell_0 - g\cos\theta_0/\ell_0 \quad (20)$$

Since $x_0$ and $z_0$ are fixed, $(x_1,z_1)$ could be found directly from the algebraic constraints once (20) was solved. However, differential expressions analogous to (16) and (17) are more useful for guidance purposes:

$$\ddot{x}_1 = (-\ell_0\dot{\theta}_0^2 + g\sin\theta_0)\cos\theta_0 - F\sin\theta_0\cos\theta_0/m_1 \quad (21)$$

$$\ddot{z}_1 = (-\ell_0\dot{\theta}_0^2 + g\sin\theta_0)\sin\theta_0 - g - F\cos^2\theta_0/m_1 \quad (22)$$

As expected, (20)-(22) do not depend on $m_0$, because $m_0$ doesn't move in Case F.

Feedback law: Only the most simple form of feedback control strategy is described here, and it is shown that feedback from only $\theta_0$ and $\dot{\theta}_0$, as illustrated by the solid feedback line of Figure 3, is sufficient to provide the features of useful locomotion described above. The discontinuous feedback law is most readily illustrated on the phase-plane plot of $\theta_0$ vs. $\dot{\theta}_0$ of Figure 4.

The feedback law is:

Whenever $(\theta_0(t),\dot{\theta}_0(t))$ in Regions A,E or F

take F(t) = 0

Whenever $(\theta_0(t),\dot{\theta}_0(t))$ in Regions B,C or D

take $F(t) = (m_0+m_1)g$

For any initial condition inside the shaded area except the point $(\pi/2,0)$[4], the motion of the system will eventually settle into a periodic motion. Initial conditions outside the shaded regions cannot be corrected by this feedback law. Disturbances such as variation in surface height, friction, etc., result in perturbations to the trajectory, which are stable if the system remains inside the shaded region. Thus, one goal of accommodating small

[4]Certain additional constraints and assumptions, which may slightly decrease the size of this area, have been intentionally ignored in this simplified analysis.

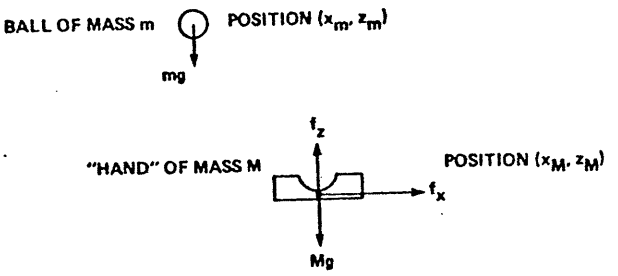


Figure 1. Geometry and Definition of Variables for Example 1

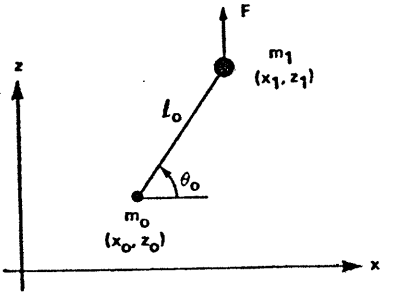


Figure 2. Geometry and Definition of Variables for Example 2

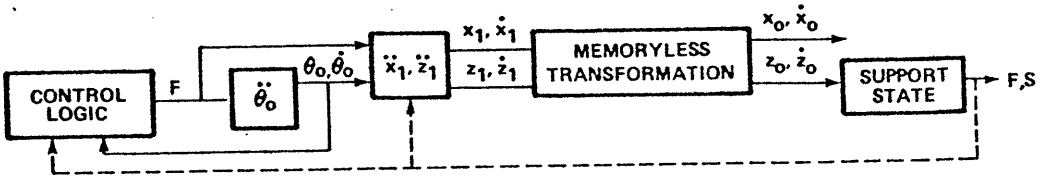


Figure 3. Block Diagram of System Dynamics and Control for Example 2

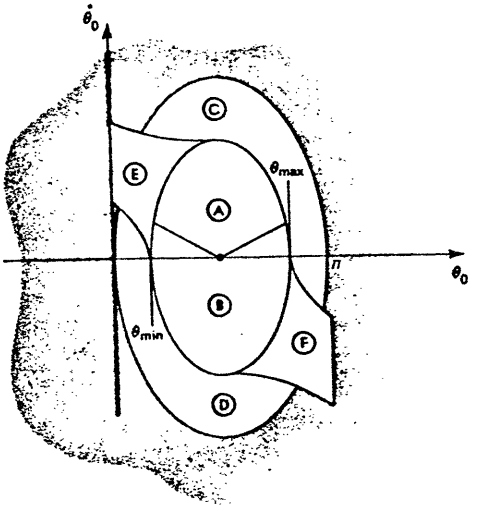


Figure 4. Phase-plane Plot of Feedback Law in Example 2

## Appendix G

## Essentially Continuous Functional Real Coders

1. Let $F_T$, $T > 0$ be the set of all piecewise uniformly continuous maps, with a finite number of pieces from $[0,T) \to R$.

For $f \in F_{T_1}$ $g \in F_{T_2}$ we define an operation * as follows

$$g*f(x) = \begin{cases} f(x) & \text{if } x \in [0,T_1) \\ g(x-T_1) & \text{if } x \in [T_1, T_1+T_2) \end{cases}$$

It is clear that $g*f$ is in $F_{T_1+T_2}$ and thus $F = \bigcup_{T>0} F_T$ is closed under *.

Let $\Phi : F \to R$ be a given map. We call $\Phi$ a <u>functional real coder</u>. In the sequel we examine certain properties of such coders.

Let use define the relation $\approx_\Phi$ on F by

$$f \in F_{T_1} \simeq_\Phi g \in F_{T_2}$$

$$\iff \forall\ h \in F_{T_3}$$

$$\Phi(h*f) = \Phi(h*g)$$

Clearly $\Phi$ is a right-congruence relation. We say that $\Phi$ is unitary if $F/\approx_\Phi$ has but one class: otherwise we call $\Phi$ finitary, if $\approx_\Phi$ has finite index, or non-finitary if it does not. We focus on unitary coders below.

2. Characterizing Unitary Coders

Suppose $\Phi$ is unitary. We would like to see if we can find a simple characterization of $\Phi$. We first begin with an elementary but, nonetheless, important theorem.