FINITE MEMORY ESTIMATION AND CONTROL

OF FINITE PROBABILISTIC SYSTEMS

by

Loren Kerry Platzman

S.B., Massachusetts Institute of Technology

(1972)

S.M., Massachusetts Institute of Technology

(1973)

E.E., Massachusetts Institute of Technology

(1974)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January, 1977

Signature of Author . . . .
Department of Electrical Engineering and
Computer Science, January 13, 1977

Certified by . . . .
Thesis Co-supervisor

Certified by . . . .
Thesis Co-supervisor

Accepted by . . . .
Chairman, Departmental Committee on Graduate Students

FINITE MEMORY ESTIMATION AND CONTROL OF FINITE PROBABILISTIC SYSTEMS

by

Loren Kerry Platzman

Submitted to the Department of Electrical Engineering and Computer Science on January 13, 1977, in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

ABSTRACT

A finite probabilistic system (FPS) is a stationary discrete-time controlled stochastic dynamical process, having finite input, output, and (internal) state sets. The partially-observable Markov decision process is an example of such a system. FPS formulations provide a convenient framework for the study of problems of state estimation, statistical decision, or control, where state information is available only through a finite memoryless channel, and observation dynamics may depend on the inputs selected.

Notions of reachability and detectability in FPS's (similar to controllability and observability in linear systems) are made precise. It is shown that every FPS can be reduced to components that are either reachable and detectable, or transient, or null-recurrent.

It is well known that the information vector (whose i-th entry is the *a posteriori* probability that the system is in state i) is a sufficient statistic (for the estimation of future dynamics given past inputs and outputs). A contraction property of the information vector transition function is exploited to obtain procedures for ε-optimal (arbitrarily close) approximation of the information vector by a deterministic time-invariant finite-memory observer. Each observer state corresponds to a particular configuration of most recent input-output pairs. The average error achieved by such an approximation is bounded by the expression $(m/m_0)^{-\tau}$, where m_0 and τ are parameters associated with the observed system, and m is the number of observer states.

Control problems, in which the average reward is maximized over a discounted or undiscounted infinite horizon, may be solved by an iterative procedure which has been given the name perceptive dynamic programming. Successively weaker assumptions that the controller "perceives" unavailable state values transform the problem into a sequence of formulations which may be solved by dynamic programming. Each solution obtained in this manner is used to construct a feasible controller formulation, taking the form of a deterministic time-invariant finite-state automaton. Monotone geometrically convergent bounds, containing both the supremum feasible performance and that of the current design, are also obtained. Computation may be terminated when these bounds become sufficiently close, or when the number of controller states becomes excessively large. Although computing a solution by perceptive dynamic programming may require considerable time and storage, both are roughly proportional to the number of controller states allowed in the final iteration; thus the cost of controller design reflects the cost of controller implementation.

This procedure was applied to idealized problems of machine maintenance and computer communication, both of which had been investigated by other researchers. The first problem was solved exactly; a design suitable close to the optimum was obtained for the second problem.

NAME AND TITLE OF THESIS CO-SUPERVISORS:

Alvin Drake
Professor of Systems Science and Engineering

Sanjoy Mitter
Professor of Electrical Engineering

TABLE OF CONTENTS

TABLE OF FIGURES

ACKNOWLEDGEMENTS

This research, initiated in my senior year, grew out of an attempt to define the concept of "control in the steady-state" in systems that are neither state-observable nor linear-quadratic-Gaussian. Starting with the simplest such system, which has two inputs, two outputs, and two states, and guided by the adage "that which can be done for two can be done for N," I found myself confronted with a finite probabilistic system. The final report clearly shows the influence of four outstanding educators at MIT who took an early interest in the work and in time formed my doctoral thesis committee, each concentrating on a distinct aspect of the research (as indicated below): co-supervisors Alvin Drake (probabilistic models in applied operations reserach) and Sanjoy Mitter (mathematical system theory), and readers Michael Athans (reduced-order compensator design) and Amedeo Odoni (bounds on suboptimal performance).

In early stages of the research, I also benefited from conversations with Dimitri Bertsekas, Harold Kushner, Georgio Picci, Alan Willsky and Hans Witsenhausen. The doctoral dissertation of Edward Sondik was of invaluable assistance to me. Adrian Segal suggested the application of FPS decision analysis to a slotted ALOHA problem; further advice was provided by Simon Lam and Eberhardt Wunderlicht. The "value of information" interpretation of certain bounds was contributed by James Yee.

I particularly wish to thank Paul Schweitzer, who read a great deal of a later draft and offered many comments that resulted in improved clarity of presentation in the final report. Additional editorial assistance was obtained from Alvin Drake, Michael Loui, and Kathleen Platzman.

Computations were performed at the MIT Information Processing Center. The illustrations were drafted by Arthur Giordani. The report was typed with superb efficiency by Annie Cooper.

This research was supported at its inception by a Research Traineeship, with funds provided by the Alfred P. Sloan foundation. I am especially grateful for that opportunity to pursue an unorthodox line of research with unqualified financial support. Further support was later provided by the Department of Electrical Engineering and Computer Science (in the form of a Teaching Assistantship), the Air Force Office of Sponsored Research (under Grant 72-2273), and my family.

Finally, I owe a great debt of gratitude to Michael Athans, who as my undergraduate advisor made it possible for me to enter graduate school at MIT, and a greater debt to Alvin Drake, who as chairman of my Graduate Area Examination Committee and advisor in many other matters made it possible for me eventually to leave. The greatest debt of gratitude goes to my wife Kathy, who advised me in all matters not covered by Al Drake, willingly shared the frustrations and exaltations of my graduate career, and afforded me the opportunity to do the same with respect to hers.

NOTATIONS

If A and B are sets, then A-B is the set of elements in A that are not contained in B. #A is the number of elements in A. B^A is the set of mappings from A to B. 2^A is the set of subsets of A. $\emptyset$ is the null set.

$\langle a,b\rangle$ is the set of integers i satisfying $a \leq i \leq b$. The sequence $\{A_a, A_{a+1}, \ldots, A_{b-1}, A_b\}$ is denoted $\{A_k\}_{k\varepsilon\langle a,b\rangle}$. $a \div b$ denotes integer quotient rounded down, i.e. the integer q of largest magnitude such that $|bq| \leq |a|$ and $\mathrm{sgn}(bq) = \mathrm{sgn}(a)$. $\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$ is the binomial coefficient for n items taken k at a time.

$[a,b]$ is the set of real numbers x satisfying $a \leq x \leq b$; similarly $[a,b) = [a,b]-\{b\}$. $(a)^+ = \max(a,0)$ and $(a)^- = \min(a,0)$; clearly $a = (a)^+ + (a)^-$.

R_N denotes the Euclidean space of column N-vectors. A row vector π is <u>substochastic</u> if its entries are all nonnegative and sum to a quantity not exceeding unity; it is <u>stochastic</u> if it is substochastic and the sum of its entries is exactly one. Π_N and $\tilde{\Pi}_N$ denote the sets of stochastic and substochastic row N-vectors, respectively. A square matrix is stochastic (substochastic) if each of its rows is a stochastic (substochastic) vector. v_i denotes the i^{th} entry of vector v; similarly, P_{ij} is the ij^{th} entry of matrix P, and $\mathrm{row}_i[P]$ is the row vector whose ij^{th} entry is P_{ij}. The superscript "T" denotes transpose. e^i is the "unit" vector whose i^{th} entry is unity and whose remaining entries equal zero; 0 is a vector of zeroes and 1 is a vector whose every entry equals unity; the dimension and inclination (row or column) of e^i, 0, and 1, are determined by context. The usual rules of matrix algebra will be observed; thus if $\pi \varepsilon \Pi_N$ and $q \varepsilon R_N$, then the quantity πq is a scalar.

If $x \varepsilon R_N$, then $|x| = \Sigma_{i=1}^{N} |x_i|$. If $x,y \varepsilon R_N$, then $x<y$ is understood to imply $x_i<y_i$, $\forall i \varepsilon <1,N>$, and $x \leq y$ implies $x_i \leq y_i$, $\forall i \varepsilon <1,N>$.

CHAPTER I

PRELIMINARIES

1. Introduction

This dissertation introduces concepts and associated computational procedures that are applicable to a mathematical problem arising in the context of Operations Research and Stochastic Control. Briefly stated, the problem is to design a strategy for real-time decision-making on the basis of imperfect (state) information and finite memory. The plant (i.e. the object to be controlled) is modelled as a finite probabilistic system (FPS) or stationary discrete-time finite-input finite-output finite-state controlled stochastic process, a generalization of the partially-observed Markov decision model initiated by Drake (1962), which itself generalizes the Markov decision model of Bellman (1957a).

An engineering problem which might be tackled by the methods espoused in this dissertation is the following:

(1.1) Machine Maintenance and Repair Problem (Scenario). A factory contains a large number of identical machines, each of which may require overhaul from time to time. A repairman maintains a "status report" for each machine and effects the overhauls. Unfortunately, a lengthy inspection procedure must be performed in order to determine whether or not a particular machine is actually in need of an overhaul. Thus it is clearly impractical and undesirable to inspect every machine daily. For example, if a certain machine is believed likely to require overhaul, it might be advisable to overhaul that machine without inspecting it at all. The problem is to determine a simple rule for the repairman to follow in making decisions for individual machines, and in recording each machine's status. A solution to this problem may be visualized as a manual in which every possible machine status is listed, along with a course of action and a new status resulting from that action. The status code must be reasonably concise, for otherwise the manual will assume mammoth proportions. Given the relative undesirability of broken machines and repair costs, as well as a set of admissible actions, the problem may be expressed as that of determining the <u>optimal</u>[+] (most desirable) <u>strategy</u> for coding machine status and repairing machines, as <u>realized</u> by the <u>policy</u> specified in the repairman's manual.

<u>Generalizations</u>: A similar scenario might involve a crowded hospital in which patients are visited by a doctor who must decide, on the basis of previous visits, how to allocate his time. The controller might also be a computer. Possible applications include: routing "packets" through a telecommunications network, controlling traffic signals at a busy intersection or along a congested freeway, and scheduling shipments from a warehouse serving several retail outlets.

Engineering problems of this type necessarily require that a tradeoff be made between accuracy of the model in depicting the "real" problem and solvability of the problem described by the model. The FPS model is

---

[+]The optimum may not exist; ε-optimal strategies are then sought.

more general than a Markov decision model; it is also more difficult to solve. The Markov decision model assumes that perfect state information is available to the decision-maker. In the Machine Maintenance and Repair Problem, this means that, in order to use a Markov decision model, it would be necessary to assume that the repairman knows at all times whether or not a particular machine is operating properly; his course of action is then obvious. The applications envisioned for an FPS decision theory are those in which the decision to seek information is crucial, and for which the Markov decision model is, consequently, inadequate.

More specifically, two possible aspects of "real" control problems are captured by the FPS formulation, but totally ignored in Markov decision theory. One aspect is the "dual control" phenomenon, where the decision-maker must decide whether to seek better state information at the expense of short-term performance, or to seek improved immediate performance at the expense of information forgone in the interim. The other aspect is the "saturation" phenomenon, in which the decision-maker is confronted with more information than may be considered in the time allotted for decision-making. Conventional linear-quadratic-Gaussian control methods, likewise, avoid "dual control" and "saturation" phenomena by requiring that observation dynamics be unaffected by the input process.

In problems such as the Machine Maintenance and Repair Problem, where information is available only at a cost, perfect state information cannot be taken for granted, and separation of input and output dynamics does not occur. At the heart of the problem is the determination of what information is important for purposes of decision-making, and what information may be disregarded. An important contribution of this research is a bound on the <u>value of information</u>. When the cost of obtaining information exceeds its value, then it is advisable to do without that information.

The elimination of "dual control" immediately leads to a "saturation" condition, since the decision whether to seek further information must be based on all information acquired thus far. Fortunately, the <u>value of information decreases geometrically with delay</u>, in most FPS's. Thus, for any $\varepsilon>0$, there is an integer ℓ such that the value of all information delayed by ℓ or more time units has value less than ε. This implies that there exists an ε-optimal strategy (a strategy whose performance lies within ε of the supremum feasible performance) for decision-making based on the most recent ℓ inputs and outputs alone. A computational method for strategy optimization, based on this result, has been given the name <u>perceptive dynamic programming</u>.

As the number of most recent input-output pairs retained by the decision-maker increases, the loss in performance from discarded information <u>decays</u> geometrically and the number of memory states (called "status codes" in (1.1)) <u>increases</u> geometrically. Thus, the performance achieved by a decision-maker acting on the basis of m memory states can be made to lie within $(m/m_0)^{-\tau}$ of the supremum feasible performance, where m_0 is the number of values in a sufficient incremental statistic, and

$$\tau = \frac{\text{information value decay rate}}{\text{memory increase rate}} \qquad (1.2)$$

The remainder of this report is devoted to making precise the concepts outlined above. The FPS model is described in detail in the following section. The Machine Maintenance and Repair Problem is formulated as an FPS control problem and solved in Section 3. A review of related work, a compendium of original contributions, and an outline of the report complete this chapter.

2. The Model

a. Representation of the Plant.

The plant will be modeled as an FPS, which is defined by (2.1), below. Conceptually, an FPS is a generalization of a Markov chain, shown in Figure 2-1. A Markov chain has the property that, for any time $k \varepsilon \langle 1,\infty \rangle$, the random variables $\{s(k')\}_{k' \varepsilon \langle 0,k-1 \rangle}$ and $\{s(k')\}_{k' \varepsilon \langle k+1,\infty \rangle}$ are conditionally independent given $s(k)$. Thus the transition probability that $s(k+1)$ will assume value j given the values of all past states $\{s(k')\}_{k' \varepsilon \langle 0,k \rangle}$ can be expressed as a function of the value of $s(k)$ alone. The broken arrow leading from $s(k)$ to $s(k+1)$, in Figure 2-1, is intended to convey a sense that $s(k+1)$ evolves probabilitically from $s(k)$ alone.

$$\bullet\bullet\bullet \longrightarrow s(k-1) \rightsquigarrow s(k) \rightsquigarrow s(k+1) - \bullet\bullet\bullet$$

Figure 2-1. A Markov Chain

In a Markov decision process, shown in Figure 2-2, the transition probabilities depend on inputs that are provided to the system by a decision-maker. Input u(k) determines the manner in which s(k+1) evolves probabilistically from s(k). If inputs are selected on the basis of the most recent state alone, then the system becomes a Markov chain.

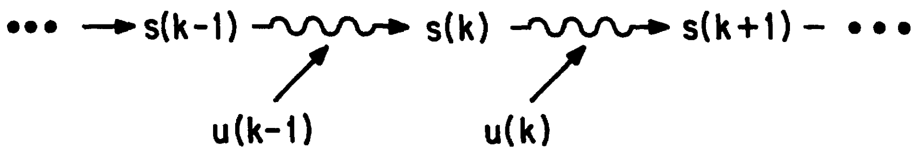


Figure 2-2. A Markov Decision Process

A partially-observable Markov decision process, shown in Figure 2-3, combines a Markov decision process with a process of noisy outputs. Output y(k) depends probabilistically on s(k) alone. It is easy to see that a partially-observable Markov decision process is entirely equivalent to a Markov decision process whose state at time k consists of the pair [s(k),y(k)]; y(k) then becomes a perfect observation of the second state component, and is referred to as an "incomplete" state observation.

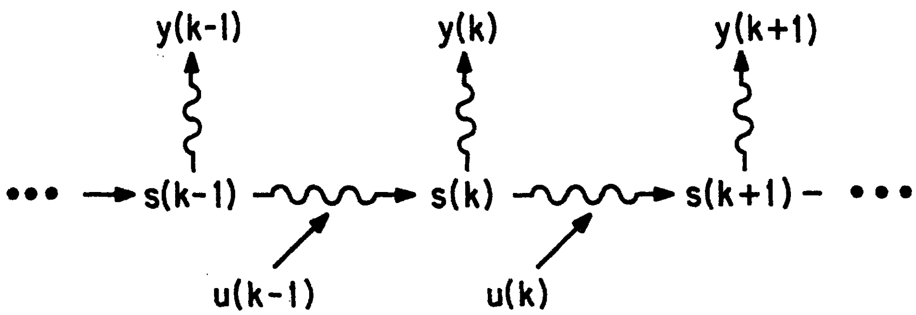


Figure 2-3. A Partially-observable Markov Decision Process

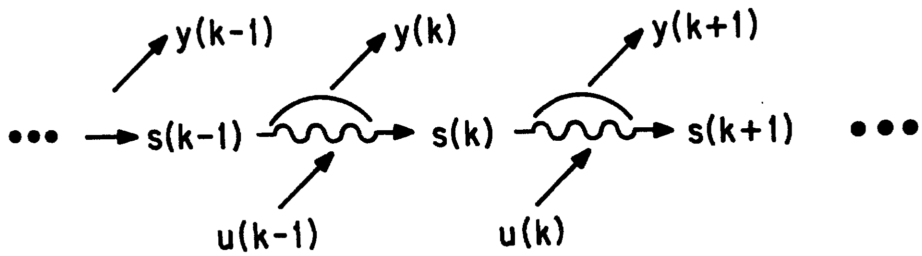


Figure 2-4. A Finite Probabilistic System

A finite probabilistic system is shown in Figure 2-4. Output y(k) now depends probabilistically on s(k-1), u(k-1), and s(k), and may be thought of as a noisy measurement of the most recent state transition. Yet, an FPS is always equivalent to a Markov decision process whose state at time k consists of the pair [s(k),y(k)]. Thus, every partially-observable Markov decision process is an FPS, and any FPS may be transformed into a partially-observable Markov decision process. The distinction between the two lies in their representations, i.e. in the notation used to describe them.

Since s(k) depends probabilistically on s(k-1) and u(k-1), the pair s(k) and y(k) may be viewed as random variables that depend jointly on s(k-1) and u(k-1). In this form, the dynamic evolution of an FPS is entirely described by an array of probabilities for the state and output, conditioned on the previous state and input. Except for the requirements that the input, output, and internal state sets be finite, and that dynamics be stationary, an FPS is totally unstructured.

The formal definition of an FPS can now be given.

(2.1) Definition. A finite probabilistic (dynamical) system (FPS) is a 5-tuple $(U,Y,S,\pi(0),\{P(y|u) : y\varepsilon Y,\ u\varepsilon U\})$ where:

(i) U is a finite nonempty set of input values (or decisions);

(ii) Y is a finite nonempty set of output values (or observations);

(iii) $S = \langle 1,N\rangle$ is a finite nonempty set of (internal) state values;

(iv) $\pi(0)$ is a stochastic N-vector of initial state probabilities;

(v) Each $P(y|u)$ is an NxN substochastic matrix of state transition probabilities, and $\Sigma_{y\varepsilon Y} P(y|u)$ is stochastic, $\forall u\varepsilon U$.

The dynamic evolution of an FPS is described in the following terminology:

1. The initial state $s(0)$ assumes value i with probability $\pi_i(0)$.
2. When a decision-maker specifies input $u(k)$, that input is said to be accepted by the FPS. Output $y(k+1)$ is subsequently emitted by the FPS.
3. Given that an FPS in state $s(k)=i$ accepts input $u(k)=u$, it will undergo a transition to state $s(k+1)=j$ and emit output $y(k+1)=y$ with probability $P_{ij}(y|u)$, conditionally independently of the "past history" $\{s(k')\}_{k\varepsilon\langle 0,k-1\rangle}$, $\{u(k')\}_{k'\varepsilon\langle 0,k-1\rangle}$,

$\{y(k')\}_{k'\varepsilon<1,k>}$.

4. The Markov decision process consisting of the internal state and input processes of an FPS is called the underlying process (of that FPS). It is described by the stochastic matrices $\{\Sigma_{y\varepsilon Y} P(y|u) : u\varepsilon U\}$.

5. The time set is $<0,K>$. The terminal time K is called the horizon.

b. Alternate Representations.

The expression "finite probabilistic system" is used in accordance with a classification of systems by Kalman, Falb, and Arbib [1969]. The notation used to specify dynamics for a particular FPS is that of Paz [1971]. It is also called the Mealy form of a FPS, in consideration of its similarity to the Mealy form of a deterministic machine. The Moore form is an alternate representation in which y(k) is expressed as a deterministic function of s(k) alone.

Yet another representation is that of Drake [1962]. Here the transition probabilities of the underlying process are provided, along with a matrix of conditional output probabilities, given internal states. A transformation to Mealy form is readily effected, although some care must be taken to insure that inputs, outputs, and time changes are defined to occur in the correct order, i.e. that y(k) is emitted before u(k) is accepted.

c. Some Important Classes of FPS's

(2.2) Definition. An FPS is state-observable if each transition probability matrix $P(y|u)$ has at most one non-zero column.

Interpretation: In a state-observable FPS, the internal state may be deduced from the most recent input-output pair alone.

Example: A Markov decision process is a state-observable FPS.

(2.3) Definition. An FPS is state-calculable if each row of a transition probability matrix has at most one non-zero entry.

Interpretation: In a state calculable FPS, knowledge of the previous internal state, along with the intervening input-output pair, is sufficient to determine the present state.

Example: Consider a queuing system, in which only the numbers of arriving and departing "customers" (over each discrete time interval) are observed. This system may be modeled as a state-calculable FPS.

(2.4) Definition. An FPS is free if its input set contains exactly one element.

Remark: A free FPS may be viewed as a "partially-observable Markov chain" (Drake [1962]) or "stochastic process of finite rank" (Paz [1971]).

d. Specification of the Input Process

A rule for the selection of inputs to an FPS will be called a (decision) strategy. A strategy γ is specified by a probability distribution for $u(k)$ conditioned on the past history $[s(0),u(0),y(1),s(1), \ldots, s(k-1),u(k-1),y(k),s(k)]$; however, this representation is cumbersome. It is far more convenient to consider the input process to be generated by a dynamical system called a controller, which is a controlled Markov process having input and state sets to be determined, and output process $\{u(k)\}$.

A particular description of a decision strategy as a dynamical system is called a realization of that strategy. Naturally some realizations are more concise then others. A decision strategy satisfies a finite-memory constraint if it has an FPS realization with input process $\{y(k-1)\}$. In this report, consideration will be limited almost exclusively to decision strategies that can be realized by deterministic time-invariant finite-state automata.

The interconnection of an FPS with decision strategy γ causes the former's input, state, and output processes to become stochastic processes; the resulting system may or may not be an FSP, depending on the size of its state set (which must include all information required to describe future inputs). This system will be called the free system induced (on the FPS) by strategy γ , or, more informally, the system under γ . If γ satisfies a finite-memory constraint, then the system under γ may be represented as a free FPS whose state is a doublet consisting of both the plant and controller states.

The output process of a free FPS is a stochastic process, since the probability distribution of system variables (states and outputs) is well-defined. Such is not the case if U contains more than one element: y(1) then depends on u(0), which is not a random variable (since no probabisistic rule describing it has been provided). The interconnection of an FPS with a decision strategy γ causes all system variables to become random variables. A probability measure, denoted Prob_γ , which describes these variables, is specified by the induction:

$$\mathrm{Prob}_\gamma\,\{s(0)=i\} = \pi_i(0).$$

$$\begin{aligned}
&\mathrm{Prob}_\gamma\,\{s(k')=s_{k'},\ u(k')=u_{k'},\ y(k'+1)=y_{k'},\quad \forall k'\varepsilon<0,k-1> \\
&\quad \text{and}\quad s(k)=i,\ u(k)=u,\ y(k+1)=y,\ s(k+1)=j\} \\
&= \mathrm{Prob}_\gamma\,\{s(k')=s_{k'},\ u(k')=u_{k'},\ y(k'+1)=y_{k'},\quad \forall k'\varepsilon<0,k-1> \\
&\quad \text{and}\quad s(k)=i\} \\
&\cdot \mathrm{Prob}\,\{\text{strategy } \gamma \text{ causes } u(k)=u \text{ to be selected} \mid \\
&\quad s(k')=s_{k'},\ u(k')=u_{k'},\ y(k'+1)=y_{k'},\quad k'\varepsilon<0,k-1> \text{ and } s(k)=i\} \\
&\cdot P_{ij}(y|u).
\end{aligned} \tag{2.5}$$

Informally, Prob_γ is called the probability under (strategy) γ.

(2.6) Definition. $E_\gamma\{\cdot\}$ denotes expectation with respect to probability measure $\text{Prob}_\gamma\{\cdot\}$, i.e. expectation given that inputs are selected according to strategy γ.

Notation: Subscript γ may be omitted in $\text{Prob}_\gamma\{\cdot\}$ and $E_\gamma\{\cdot\}$ when the probability or expectation is the same for all strategies.

e. The Information Vector

(2.7) Definition. The stochastic N-vector $\eta(k)$ having components

$$\eta_i(k) = \text{Prob}\{s(k)=i|u(0)\ldots u(k-1);\ y(1)\ldots y(k)\}$$

will be called the information vector at time k.

It is well known that $\eta(k)$ is a sufficient statistic for the estimation of future dynamics given past inputs and outputs; this is a trivial result of the Markov property of the internal state. The following result is similarly self-evident.

(2.8) Proposition. The information vector may be recursively computed according to Bayes' Rule:

$$\eta(k+1) = T(\eta(k),\ u(k),\ y(k+1)),$$

where T is the information vector transition function

$$T(\eta,u,y) = \eta P(y|u)\ /\ (\eta P(y|u)1)$$

Because $\eta(k)$ is a sufficient statistic, desirable decision strategies may be realized by a deterministic machine having state process $\{\eta(k)\}$. Such a decision strategy would be completely described by a policy on Π_N, i.e. a mapping from Π_N to U specifying the input to be applied when the information vector has a given value. This traditional approach to controller realization leads to horrendous computational difficulties which have yet to be resolved.[+] The main contributions of this research are approximation schemes for $\eta(k)$, and associated realizations which avoid the use of Π_N as an observer or controller state set.

f. Rewards and Performance Indices

It is convenient to place a mechanism for evaluation of decision strategies within the conceptual confines of the system itself. To this end, consider the process of incremental (immediate) rewards $\{r(k)\}$, each of which is determined from system variables $s(k)$, $u(k)$, $y(k+1)$, $s(k+1)$, on the basis of a given array $\{r[i,u,y,j] : i,j \in S, u \in U, y \in Y\}$, according to the rule

$$r(k) = r[s(k), u(k), y(k+1), s(k+1)]$$

[+]See the discussion, in Section 4, of previous work in this field.

(2.9) Definition. A valued finite probabilistic system (VFPS) is an FPS along with an incremental reward array, as described above.

(2.10) Definition. The performance index is a function of the decision strategy, taking one of the following forms:

(a) Finite horizon:

$$g(\{b(k)\}_{k\varepsilon<0,K>},\gamma) = E_\gamma\{\Sigma_{k=0}^{K} b(k)\ r(k)\},\ K\varepsilon<0,\infty> .$$

(b) Discounted infinite-horizon:

$$g(\beta,\gamma) = (1-\beta)E_\gamma\{\Sigma_{k=0}^{\infty} \beta^k r(k)\},\ 0\leq\beta<1.$$

(c) Undiscounted infinite-horizon:

$$g(\gamma) = \lim\inf_{\beta\uparrow 1} [g(\beta,\gamma)].$$

Remark: The undiscounted performance index $g(\cdot)$ is generally equivalent to the "time-averaged reward" $\lim\inf_{K\to\infty} E_\gamma\{\frac{1}{K}\Sigma_{k=0}^{K-1} r(k)\}$. For a discussion of the conditions under which these indices may differ, see Flynn [1974]. The definition given above is more convenient, especially when relative values are considered, since these converge as $\beta\uparrow 1$.

The incremental reward process may be replaced by a process of expected incremental rewards $\{q(k)\}$ defined by

$$q(k) = q_{s(k)}(u(k)) \tag{2.11}$$

where

$$q_i(u) = \Sigma_{j\varepsilon S} \Sigma_{y\varepsilon Y} P_{ij}(y|u) \; r[i,u,y,j] \tag{2.12}$$

denotes the expected reward given that $s(k) = i$ and $u(k) = u$.

Clearly the substitution of process $\{q(k)\}$ for $\{r(k)\}$ in (2.10) leaves the value of a performance index, for a particular decision strategy, unchanged.

Also define

$$Q_{max} = \max_{i\varepsilon S} \max_{u\varepsilon U} [q_i(u)]$$

$$Q_{min} = \min_{i\varepsilon S} \min_{u\varepsilon U} [q_i(u)]$$

$$Q = Q_{max} - Q_{min} \; . \tag{2.13}$$

g. Classification of Problems

The problems of interest fall into three categories. The first of these is given the name estimation. The finite-memory estimation problem is to learn as much as possible about the current internal state, subject to a finite-memory constraint. Note that in the absence of this constraint, the problem would be trivially solved by computing the information vector according to (2.8). This can in fact be accomplished if the set of values assumed by the information vector is finite, as occurs when the FPS is state-observable or when a finite horizon is contemplated. In general, however, the information vector cannot be exactly computed on the basis of finite memory; the greater the memory allowance, the better the approximation will be. The problem is more accurately described as that of constructing a sequence of finite-memory observers, (i.e. systems accepting plant outputs) that generate successively better approximations of the information vector. A suitable tradeoff between memory size and estimator quality can be made by the designer after this sequence has been computed, up to a maximum acceptable memory size.

The second problem is given the name statistical decision. It concerns a VFPS in which the transition probability matrices do not depend on u. The problem is to maximize a performance index of the form specified in (2.10). This problem may be solved by constructing a finite-memory observer, and using the information vector approximation as the basis for decision-making. A typical statistical decision problem is to guess the value of the internal state, according to an array of rewards (penalties) for correct (incorrect) decisions.

The third problem, that of control, is to determine a decision strategy which optimizes a performance index, necessarily taking into account the effect of current decisions on future plant behavior as well as future estimation accuracy. The Machine Maintenance and Repair Problem (1.1) falls into this category.

Since statistical decision is a special case of control, these problems are collectively referred to as FPS control problems. In such problems, as in estimation, a finite-memory optimum may not exist. The problem is then to construct a sequence of controller designs in which memory requirements increase and performance improves, approaching a supremum feasible value. Note that the problem is not to maximize performance subject to a given bound on memory size: such a formulation may lead to an artificial situation where the performance of mixed (randomized) strategies exceeds that of pure (deterministic) ones, thus defeating the main purpose of a memory constraint, which is to limit controller complexity.

3. Illustration of the Solution Procedure

The Machine Maintenance and Repair Problem, first described in (1.1), will now be precisely formulated as an undiscounted infinite-horizon FPS control problem, and solved by perceptive dynamic programming. The solution is also documented (in somewhat greater detail) in Section 23a.

a. Problem Formulation

Consider a single machine which can produce a single item, the product, during each production cycle. The machine contains two identical components, subject to failure, each of which must operate on every product. Depending on the status of the machine, the product may be defective or nondefective. There are four control alternatives (inputs) available during each production cycle. One is to manufacture an item. The second is to manufacture an item, and then to examine it, so as to determine whether or not it is defective. In the third alternative, the machine is dismantled and inspected (at a cost); any component found to be defective is replaced. The fourth alternative is to replace both components, whether or not they have failed.

Although the plant would appear to have four internal states (each of two components is operational or has failed), the number of states can be reduced to three if it is recognized that the order in which components fail is unimportant. Thus the state set is taken to be:

$$S = \begin{Bmatrix} 1 : \text{All components are operational} \\ 2 : \text{One component has failed} \\ 3 : \text{Both components have failed} \end{Bmatrix}$$

The four inputs are:

$$U = \begin{Bmatrix} 1 : \text{Manufacture} \\ 2 : \text{Examine} \\ 3 : \text{Inspect} \\ 4 : \text{Replace} \end{Bmatrix}$$

The three outputs are:

$$Y = \begin{Bmatrix} 1 : \text{No information} \\ 2 : \text{Non-defective product observed} \\ 3 : \text{Defective product observed} \end{Bmatrix}$$

Probabilistic rules governing the breakdown of machines have been modeled as follows: Both components are initially operational. There is a probability of 0.1 that an operational component will fail during the manufacture of a product, independently of the component's age and the condition of the other component. If a component fails prior to or during the manufacture of a particular item, it causes that item to be defective with probability 0.5. Thus the initial probability vector is $\pi(0) = (1, 0, 0)$, and the transition probability matrices are:

$$P(1|1) = \begin{bmatrix} 0.81 & 0.18 & 0.01 \\ 0.00 & 0.90 & 0.10 \\ 0.00 & 0.00 & 1.00 \end{bmatrix},$$

$$P(2|2) = \begin{bmatrix} 0.81 & 0.09 & 0.0025 \\ 0.00 & 0.45 & 0.0250 \\ 0.00 & 0.00 & 0.2500 \end{bmatrix},$$

$$P(2|3) = \begin{bmatrix} 0.00 & 0.09 & 0.0075 \\ 0.00 & 0.45 & 0.0750 \\ 0.00 & 0.00 & 0.7500 \end{bmatrix},$$

$$P(1|3) = P(1|4) = \begin{bmatrix} 1. & 0. & 0. \\ 1. & 0. & 0. \\ 1. & 0. & 0. \end{bmatrix}.$$

The value of an item produced is one unit if it is nondefective, zero units otherwise. The cost of examination is 0.25 units. New components cost a unit apiece, with an additional charge of 0.5 units for inspection. Hence, the expected incremental reward vectors are:

$$q(1) = \begin{bmatrix} 0.9025 \\ 0.4750 \\ 0.2500 \end{bmatrix}, \quad q(2) = \begin{bmatrix} 0.6525 \\ 0.2250 \\ 0.0000 \end{bmatrix}, \quad q(3) = \begin{bmatrix} -0.5 \\ -1.5 \\ -2.5 \end{bmatrix}, \quad q(4) = \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix}.$$

The performance index is undiscounted profit over an infinite horizon.

The Markov decision model for machine maintenance was introduced by Drake [1968]. The numbers used here were originally devised by Smallwood and Sondik [1973], to illustrate a computational algorithm that solves finite-horizon FPS control problems.

b. Solution Procedure

A solution to this problem is obtained in several iterations. In each of these, a Markov decision problem will be solved, yielding a controller design, as well as bounds that contain the performance of the optimal controller and that of the design most recently obtained. In early iterations the bounds will be loose; but as computations become more intricate, the bounds will become closer; eventually they will coincide.

In the first iteration, assume that the controller knows the true value of the internal state at all times. (The artificial assumption that a controller has the ability to "see" internal states by means other than computation based on system outputs, will be known as *perception*.) A Markov decision problem that is readily solved (e.g. by Howard's algorithm, described in Howard [1960]) results, yielding the optimal policy, relative value vector, and optimal gain:

$$\lambda^1 = \begin{pmatrix} 1 \\ 3 \\ 4 \end{pmatrix}, \quad \hat{v}^1 = \begin{bmatrix} 2.517 \\ 0.500 \\ 0.000 \end{bmatrix}, \quad g^1 = .5147.$$

This will be called a *perceptive solution*. Since the (perceptive) controller which achieved the gain .5147 had access to *more* information than will be available in reality, it follows that .5147 is an *upper bound* on feasible performance.

The strategy obtained in this iteration is called a *perceptive strategy*. It might also have been feasible if the optimal input had been the same for all states; but such is not the case; and so it cannot be applied in practice. However, a feasible controller realization might make use of the optimal perceptive strategy in the following way: a value for the current internal state is guessed and the corresponding optimal input is applied. Since this is the first iteration, the guess must be made of the basis of no real-time information whatsoever. Suppose, for example, that the guess is "state = 1" at all times. Then input 1 will be selected at all times; both machine components will eventually fail; and a gain of 0.25 results.

On the basis of these computations, it is concluded that:

1) The optimum feasible performance lies between 0.25 and .5147;

2) There is a feasible solution, requiring no memory, which achieves a performance of 0.25.

In the second iteration, a new internal state is devised, taking the form:

$$x(k) = [s(k-1), u(k-1), y(k)].$$

Clearly x(k) is the state of a controlled Markov chain, and a new FPS representation may be devised in which inputs, outputs, and rewards remain as before, but the internal state is x(k) at time k (see Brookes and Leondes [1973]). This called an augmentation of the original FPS. Since there are only four functionally distinguishable input-output pairs, these may be coded and given the representation z(k), according to the following table:

u(k-1)	y(k)	z(k)
1	1	1
2	2	2
2	3	3
3	1	4
4	1	4

Using the 12 states of the form x(k) = [s(k-1), z(k)], a new Markov decision problem is solved to obtain a new perceptive solution. However, the perception is "weaker" this time, and the optimal perceptive gain decreases to .4945. The optimal perceptive strategy is again unfeasible, and a feasible solution will be constructed by guessing the internal state delayed by one time unit, the guess being based on knowledge of z(k). For example the state guess might be $\hat{s}(k-1) = 1$ when z(k) = 1,2,4, and $\hat{s}(k-1) = 3$ when z(k) = 3. In this case input 1 will again be selected at all times, and the feasible gain is 0.25.

On the basis of these computations, it is concluded that

1) The optimum feasible performance lies between 0.25 and .4945;

2) There is a feasible solution, requiring 4 memory states states, which achieves a performance of 0.25.

In subsequent iterations, x(k) will take the form x(k) = [s(k-ℓ), z(k)] where z(k) is the memory state, a string of ℓ most recent z-coded input-output pairs. The rules by which a memory state may be constructed are rather complex, so for the moment regard the memory state during iteration n as the string of (n-1) most recent z-coded input-output pairs:

$$\underline{z}(k) = z(k+1-m)\ z(k+2-n)\ \dots\ z(k-1)\ z(k)$$

As computation proceeds, the bounds on feasible performance become closer and closer. Intuitively, this occurs because, as the memory state becomes longer, the augmented state component that is perceived or guessed is an internal state with greater delay, whose influence on the present information vector is weaker. In this particular problem, the bounds eventually coincide. On the ninth iteration, only eight memory states are "recurrent" under the optimal strategy, and for each of these, the optimal input does not depend on the delayed state component of the augmented state. The optimal inputs are in fact given by the deterministic sequence:

$$\{u(k)\} = \{1,1,1,1,1,1,1,3,\ 1,1,1,1,1,1,1,3,\ \dots\ \}$$

Eight memory states are required to realize this sequence, using a finite-state automaton. The optimal gain is $g^* = .422$.

c. Discussion.

The optimal decision-making strategy is remarkably simple; but this is merely a consequence of the peculiar rewards specified in this particular problem. For example, first-iteration computations show that the performance achievable with perfect state information is .5147, and the performance achievable on the basis of no information whatsoever is .25. Thus the value of perfect state information is no more than .2647. Examination, which costs .25 and yields little information about the state, appears unlikely to be useful; on the ninth iteration, this option will be eliminated entirely. Had the cost of examination been lower, or the information acquired through examination more useful, the solution might have been considerably more complex, requiring thousands of controller memory states. An optimal solution might not have been obtained at all.

In fact, the method described above cannot be used to generate a solution, since the final iteration would involve a $3 \cdot 4^8$-state Markov decision process! The algorithm that was actually used to solve the Machine Maintenance and Repair Problem is described in Section 22, and the solution obtained is reproduced in Section 23a, in this report.

The importance of perceptive dynamic programming as an engineering tool is derived from the outcome of early iterations, rather than the solution itself (if any is obtained). During iteration n, two quantities of interest are computed. The first of these, g^n, is an upper bound on performance that can be achieved if the (n-1) most recent inputs and outputs constitute the only available information concerning the (n-1) most recent transitions, although states further delayed might be perfectly known. The second, h^n, is a lower bound on the performance that can be achieved if decisions are made on the basis of the (n-1) most recent inputs and outputs alone, and all other information is discarded. Consequently g^n-h^n is an upper bound on the <u>value of information</u> concerning events delayed by (n-1) time units.

In a practical engineering problem, it is reasonable to assume that there exists a way to measure the internal state exactly, although the cost associated with such a measurement might be exhorbitant. When g^n-h^n remains large for large n, this indicates that greatly delayed perfect state information remains significantly useful for purposes of decision-making, which in turn suggests the option of periodically measuring the internal state exactly. If the interval separating perfect state measurements is large, then the average cost of periodic state measurements will be small, controller memory will have been reduced and performance enhanced. On the other hand, if g^n-h^n converges rapidly to zero, this indicates that information sufficiently delayed is of little value in decision-making, and that a near-optimal strategy having reasonable controller memory requirements, can be constructed.

d. Summary

Perceptive dynamic programming is a computational procedure that may be used to examine problems of decision-making, under uncertainty contraints, with perfect recall of all information previously obtained.

This is done by considering a sequence of problem approximations in which information dealing with events sufficiently delayed is either superceded by the "perception" of delayed state values, or ignored. The difference between performances achieved under these information constraints establishes a value of delayed information which may be compared with the cost of periodic state measurements, the cost of retaining greatly delayed outputs in controller memory, and the cost of continuing the design procedure. In the Machine Maintenance and Repair Problem, the value of delayed information rapidly approached zero, and an exact optimum was obtained.

4. Historical Perspective

An FPS decision theory may be associated with several disciplines. Some of these are listed below, along with representative references; this list is by no means intended to be exhaustive. Since an FPS is a probabilistic automaton, and the decision strategy is represented as a finite-state machine, the study of FPS's is closely related to probabilistic automata theory; see Paz [1971] for a summary of recent trends in this field. Since the assessment of unknown state values is involved in decision-making, a theory of FPS decisions is related to statistical decision theory in the sense of DeGroot [1970]. FPS control problems are problems of stochastic control; the introductory text of Kushner [1971] is a standard reference. Analysis of the optimization problem in an appropriate (infinite-dimensional) vector space makes use of techniques described by Luenberger [1969]. Finally, an FPS is a dynamical system; its study therefore belongs to what Kalman, Falb, and Arbib [1967] describe as the "exciting but chaotic new field of system theory."

Most of these disciplines are generally considered to be outgrowths of the pioneering work of Von Neuman and Morgenstern [1947]. A theory of statistical decisions was subsequently initiated by Wald [1950]. The importance of the concept of state in structuring sequential decision problems was enunciated by Richard Bellman [1957b]; he devised a general mathematical approach called dynamic programming, which may be applied to the optimization of sequential decisions. The finite-horizon Markov decision problem (Bellman [1957a]) is particularly well-suited to solution by dynamic programming; also see Howard [1960], Derman [1970], Mine and Osaki [1970], Ross [1970], Howard [1971], Hastings [1973], and Bertsekas [1976].

Because Markov decision problems can be solved, and because structural properties of the solution are fairly well understood, a great deal of effort has been devoted to improving the algorithms employed. Schweitzer [1973] has complied a list of hundreds of publications in this area. Among these, Brown [1965], Lanery [1967, 1968], Bather [1971] and Schweitzer and Federgruen [1977?] have studied convergence properties of value iteration, which is regarded as the most efficient form of dynamic programming; see Odoni [1967] for a comparison of convergence rates in various dynamic programming forms. The basic value iteration procedure has been supplemented and improved in many ways: D.J. White [1963] introduced a method for normalizing value functions in order to avoid divergence; Odoni [1967, 1968] generalized a result of MacQueen [1966] to obtain a method for bounding the closeness of suboptimal solutions to the optimum; Schweitzer [1971] accelerated value iteration by adding a damping term; Hastings [1976] devised a procedure for more efficient enumeration and termination when the optimum has been reached; the applicability of value iteration was extended by Platzman [1977] who introduced the concept of connected classes in Markov decision processes. Value iteration is currently feasible for problems with thousands of states (Schweitzer [1971]).

Partially-observable Markov decision problems have been studied by Drake [1962], Astrom [1965, 1969], Sawaragi and Yoshikawa [1970], and other as noted below. In each case, the problem was regarded as one of decision-making with perfect state information, considering the information vector to be the state of a transformed system. However, the number of values which may be assumed by the information vector is infinite. Thus the problem becomes one of dynamic programming on the unit simplex Π_N (an infinite state set), and describing an optimal decision-making policy, which is a finite-valued function on Π_N. Kaklik [1965] approximated the unit simplex by a finite grid of evenly spaced points; needless to say, the method failed to be practical for all but very small problems. Sondik [1971] (in research also reported by Smallwood and Sondik [1973]) established piecewise-linearity of the value function and finite-memory realizability of the optimal strategy in finite-horizon problems; however this too fails to be feasible if the number of faces on the value function is large. Existence of solutions to discounted problems was established by Sondik [1971] and by Satia and Lave [1973]. C.C. White [1976] has shown that these results are also applicable to a class of partially-observable semi-Markov decision models that are externally indistinguishable from a discrete-time partially-observable Markov decision process.

Existence of finite-memory solutions to certain infinite-horizon problems had been noted by Drake [1962, 1968]. In the context of statistical decision on a noisy Markov channel, this work has been pursued by Sulmar [1974] and Devore [1974]. Sondik [1971] provided an intuitive explanation for this phenomenon; his work inspired the definition of detectability in the present research. Similar results, regarding the near-sufficiency of a finite string of most recent observations, have been obtained by Černý [1969] and Kajser [1975]. Systems with perfect but delayed state observations were introduced by Brookes and Leondes [1973].

Finite-memory hypothesis-testing and N-armed bandit problems have been studied by Cover and Helman [1970], Hellman and Cover [1970a], Cover, Freedman, and Hellman [1976], and others noted both in these references and in DeGroot [1970]. One may observe, from the titles in subsequent correspondence between Chandresekarin [1970, 1971] and Hellman and Cover [1970b], that there is some controversy over the meaning of this problem. Chandresekarin and Lam [1971] have subsequently proposed an alternative formulation. The issue involved is the manner in which memory should be allowed to increase as performance approaches its supremum value. Similar issues arise in the solution of FPS control problems; they are discussed in Section 20 of this report.

5. Outline of Original Contributions

The aim of this research is to construct finite-memory observers, to devise a method for bounding the value of information in decision-making, and to establish a feasible computational procedure for the design of ε-optimal finite-memory controllers. Such results are meaningful only when supplemented by mathematical machinery which justifies their validity. This section provides an heuristic interpretation of concepts and intermediary results that are introduced for the first time in this report, and which contribute significantly to an understanding of the main results.

a. Ill-posedness of certain undiscounted infinite-horizon problems

Consider a "dual control" problem described by the VFPS:

$$Y = \{1,2\},$$

$$U = \{0,1,2\},$$

$$\pi(0) = (.5,\ .5),$$

$$N = 2,$$

$$P(1|0) = \begin{bmatrix} .6 & 0 \\ 0 & .4 \end{bmatrix}, \qquad P(2|0) = \begin{bmatrix} 0 & .4 \\ .6 & 0 \end{bmatrix},$$

$$P(1|1) = P(1|2) = \begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix} \quad P(2|1) = P(2|2) = \begin{bmatrix} 0 & .5 \\ .5 & 0 \end{bmatrix},$$

$$q(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; q(1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; q(2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{5.1}$$

The inputs may be assigned the meanings:

$$U = \left\{ \begin{array}{l} 0 \text{ : Obtain a measurement} \\ 1 \text{ : The state is probably } 1 \\ 2 \text{ : The state is probably } 2 \end{array} \right\}.$$

The outputs, likewise, are interpreted as:

$$Y = \left\{ \begin{array}{l} 1 \text{ : The state remained unchanged} \\ 2 \text{ : The state changed} \end{array} \right\}.$$

It is clear that use of input 0 causes the information vector to approach a unit vector, and use of inputs 1 or 2 causes the values of information vector entries to remain unchanged. Hence, when input 0 is used, information is gained, but no reward is received; when inputs 1 or 2 are used, a reward is received, but no information is gained.

If a discounted performance index is considered, then use of input 0 will eventually be discontinued. This is true because a decision-maker in information state $(1-\varepsilon,\varepsilon)$ stands to gain no more $\varepsilon/(1-\beta)$ by seeking further information, and receives an expected reward of $1-\varepsilon$ if he forgoes further information. As $\beta\to 1$, the point at which use of input 0 is discontinued becomes more and more distant. In the undiscounted case, the value of perfect state information (i.e. a unit information vector) is infinite, relative to the value of any information vector that is not a unit vector. A decision-maker confronted with an infinite horizon will therefore choose input 0 at all times. Consequently, he will receive no reward at all. E. Denardo calls this "the infinitely-delayed splurge."

The infinitely delayed splurge may be avoided in a number of ways. One way is to consider only discounted performance indices. Another is to assume that the decision-maker has access to an infinite past; he will then know the initial state exactly. However, it does not suffice to require that the underlying process be ergodic. In this problem, the internal state process consisted of independent Bernoulli trials; and yet the infinitely delayed splurge occurred.

b. Sufficient conditions for well-posedness

Two conditions which (together) are sufficient to assure well-posedness of an undiscounted infinite-horizon FPS control problem are now identified. The first, reachability, is a generalization of connectivity in Markov decision processes. In a reachable FPS, it is possible to select a finite sequence of inputs, on the basis of the information vector alone, so that the probability of entering a specified state is greater than $1-\rho$, where ρ is the reachability index. If $\rho=0$, then there are reset actions that cause the state to assume any desired value with probability one. As ρ increases to 1, it becomes more difficult to reach a desired state. If $\rho=1$, then the FPS is not reachable. Reachability is also parameterized by ℓ_ρ, an upper bound on the number of transitions required to "reach" a state.

It will be demonstrated that the state set of any FPS may be decomposed into connected classes, along with a (possibly empty) set of transient states. Within any connected class, the FPS will be reachable. The underlying process of a reachable FPS "looses memory" as it proceeds forward in time, in the sense that unconditional state probabilities in the future depend less and less on the present state.

The second condition has been given the name detectability. In a detectable FPS, the information vector is increasingly insensitive to increasingly delayed information, such as inputs, outputs, or artificially perceived states. A more precise definition of detectability is deferred to section 5d, where appropriate metrices and contractions will be introduced. Detectability is characterized by parameters $\bar{\ell}$ and $0 \leq \bar{a} < 1$, where information concerning events delayed by $\bar{\ell}$ time units causes the information vector to vary by a distance not exceeding $\bar{a}$, on the average. If $\bar{a}=0$ then information sufficiently delayed is of no value in decision-making. If $\bar{a}$ is close to 1, then information greatly delayed is important in decision making, and conversely, the present decision will affect many decisions to come. If $\bar{a}=1$, then the FPS is not detectable.

It will be demonstrated that the information state set of an FPS can be decomposed into detectable classes, along with a (possibly empty) set of null-recurrent information states. The information process of a detectable FPS thus looses information as it is viewed backward in time, in the sense that the present information vector depends less and less on state values from the increasingly distant past.

The conditions of reachability and detectability are complementary, in a manner similar to controllability and observability in linear systems.

c. A Bound on the Value of Information

A key result, Theorem (19.3), states that any infinite-horizon FPS control problem satisfying conditions of reachability and detectability has a convex relative value function $v^*(\cdot)$ satisfying:

$$\max_{\pi\in\Pi_N} \{v^*(\pi)\} - \min_{\pi\in\Pi_N} \{v^*(\pi)\} \leq \frac{(\ell_\rho+\bar{\ell})Q}{(1-\rho)(1-\bar{a})} = \Omega \qquad (5.2)$$

where Q is given by (2.13). The expression on the right of (5.2) is interpreted as the bound on the value of information. v^* may become undefined as $\rho\to 1$ or $\bar{a}\to 1$.

d. Metrics and Contractions

Consider $\delta[\pi,\pi'] = \Sigma_{i\epsilon S}(\pi_i-\pi_i')^+$, the Hajnal measure, which is extensively used (as described in Paz [1971]) to demonstrate convergence of unconditional probability vectors, in the theory of ergodic Markov chains. A more appropriate metric for the study of conditional probability vectors is

$$\Delta[\pi,\pi'] = \sup\{\delta[\pi \circ w,\pi' \circ w] : w \in \Pi_N , \; w>0\} \tag{5.3}$$

where $\pi \circ w$ is a vector in Π_N having elements $(\pi \circ w)_i = \pi_i w_i / \Sigma_{i\varepsilon S} \pi_i w_i$. It will be shown that:

$$\delta[\pi,\pi'] \leq \Delta[\pi,\pi'] \leq 1 \tag{5.4}$$

and

$$\Delta[\pi,\pi'] = \frac{1-\sqrt{c_1 c_2}}{1+\sqrt{c_1 c_2}} \tag{5.5}$$

where:

$$\begin{aligned} c_1 &= \min\{\pi_i/\pi_i') : i\varepsilon S, \; \pi_i' > 0\}, \\ c_2 &= \min\{\pi_i'/\pi_i) : i\varepsilon S, \; \pi_i > 0\}. \end{aligned} \tag{5.6}$$

The topology induced by Δ on Π_N has many interesting properties which are explored in Section 12d. For example, any convex function is continuous with respect to Δ; in particular:

$$v[\pi] - v[\pi'] \leq \; \Delta[\pi,\pi']4[\max_{\pi\varepsilon\Pi_N} \{v[\pi]\} - \min_{\pi\varepsilon\Pi_N} \{v[\pi]\}] \tag{5.7}$$

Now consider an input-output pair (u,y) such that $P(y|u)$ is <u>subrectangular</u>, i.e. $P_{ij}(y|u) > 0$ and $P_{i'j'}(y|u) > 0$ implies $P_{ij'}(y|u) > 0$ and $P_{i'j}(y|u) > 0$. Let

$$\alpha[(u,y)] = \max_{i,i'\varepsilon S} \Delta[T(e^i,u,y), \; T(e^{i'},u,y)].$$

Now $0 \leq \alpha[(u,y)] < 1$, a consequence of the subrectangularity of $P(y|u)$.

The contraction property is:

$$\Delta[T(\eta,u,y),\ T(\eta',u,y)] \leq \alpha(u,y)\ \Delta[\eta,\eta'] \tag{5.8}$$

This is illustrated in Figure 5-1. It is seen that (u,y) causes the unit simplex to be mapped into a somewhat smaller set. The greater the number of recent input-output pairs available, the smaller this set will be. Hence, the assumption that the information vector ℓ times delayed had some convenient value, allows an approximation of the information vector to be computed on the basis of the most recent ℓ input-output pairs alone. This approximation is guaranteed to be with a certain distance of the true value; that distance can be computed by measuring the contraction imposed on the information vector by the transition probability matrix corresponding to the most recent ℓ input-output pairs.

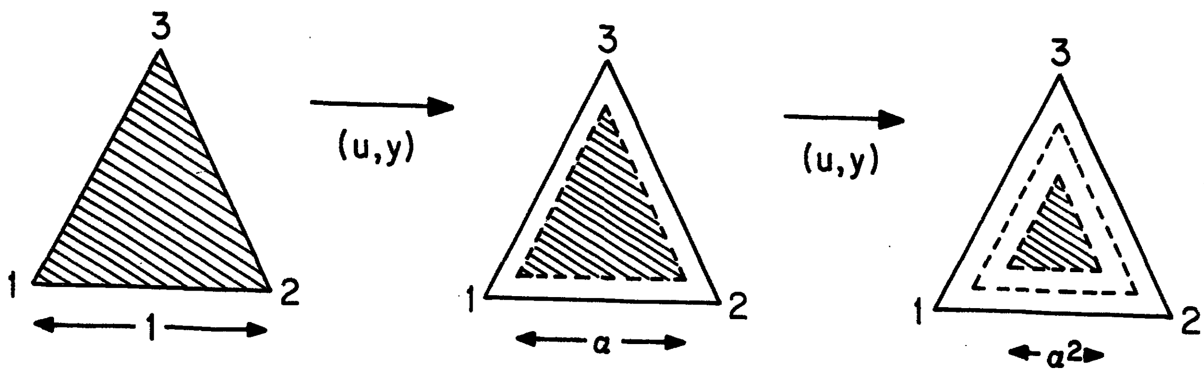


Figure 5-1. Contractions on the Unit Simplex

In the establishment of detectability, subrectangularity plays a role analogous to that of block rectangularity in the establishment of connectivity in Markov chains. An FPS satisfies a condition of strong detectability if there is an integer ℓ such that, for every possible sequence of consecutive input-output pairs $(u_1,y_1)(u_2,y_2) \ldots (u_\ell,y_\ell)$, the cumulative transition probability matrix $P(y_1|u_1) \cdot P(y_2|u_2) \cdot \ldots P(y_\ell|u_\ell)$ is subrectangular. It follows, from the contraction property stated above, that an estimate of the information vector can be made arbitrarily close (in a Δ sense) by recalling a sufficiently long string of recent input-output pairs. In particular, an estimate made on the basis of ℓ input-output pairs always lies within $\alpha^{\ell \div \overline{\ell}}$ of the true information vector, for some $\alpha<1$.

Weak detectability is a condition which implies that the expected deviation of the information vector estimate from its true value can be made arbitrarily small in an analogous way. In a weakly detectable system, $\overline{\alpha}$ denotes the average contraction induced by the most recent $\overline{\ell}$ input-output pairs. The average contraction induced by the most recent ℓ pairs is now given by $\overline{\alpha}^{\ell \div \overline{\ell}}$. $\overline{a}$ is a measure of detectability which differs slightly from $\overline{\alpha}$.

e. Existence of ε-optimal Controllers

Consider the relative value function for a reachable, detectable, FPS. It will be seen that this function spans a range of values which cannot exceed $\Omega = \dfrac{(\ell_\rho+\bar{\ell})Q}{(1-\rho)(1-\bar{a})}$. Thus, for any stochastic vectors $\pi,\pi'\varepsilon\Pi_N$,

$$|v^*[\pi] - v^*[\pi']| \leq 4\Omega .$$

When state perception is introduced, the information vector changes, at any given time, in such a way that the expected relative value of the new information vector will be greater than that of the old information vector. The difference between these quantities, called the <u>value of perception</u>, is shown in Figure 5-2. If perception of states with an ℓ time-unit delay is assumed, then the gain will increase by at most $\bar{\alpha}^{\ell\div\bar{\ell}}4\Omega = \bar{\alpha}^{\ell\div\bar{\ell}}\left[\dfrac{4(\ell_\rho+\bar{\ell})Q}{(1-\rho)(1-\bar{a})}\right]$.

The substitution of guessed state values for perceived states is called <u>pseudo-perception</u>. If a delayed state value is guessed, then the controller finds itself acting according to one information vector while actually in another information state. The value of acting according to a particular information vector is <u>linear</u> in the actual information state, because E{value of acting according to $\eta^1|\eta(k)$} $= \Sigma_{i\varepsilon S}\ \eta_i(k)$E{value of acting according to $\eta^1|s(k)=i$}. Thus the cost of pseudo-perception is as shown in Figure 5-3; this cost cannot exceed

$$\bar{\alpha}^{\ell\div\bar{\ell}}\left[\frac{4\bar{\ell}\Omega}{(1-\bar{a})}\right] = \bar{\alpha}^{\ell\div\bar{\ell}}\left[\frac{4\bar{\ell}(\ell_\rho+\bar{\ell})Q}{(1-\rho)(1-\bar{a})^2}\right].$$

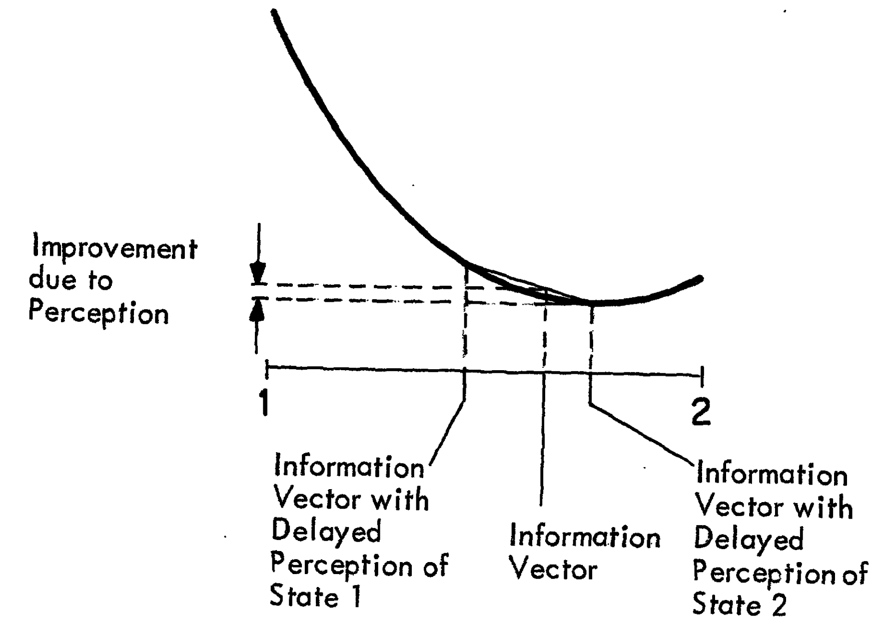


Figure 5-2. Geometric Interpretation of Performance Increase Due to Perception

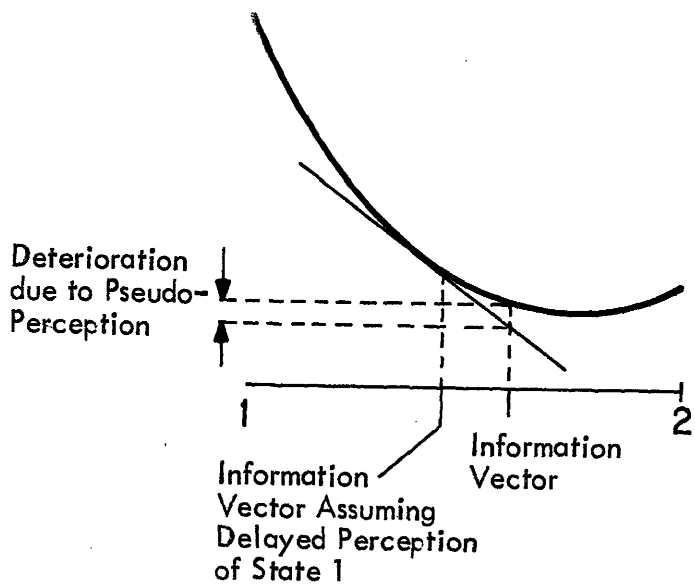


Figure 5-3. Geometric Interpretation of Performance Decrease Due to Pseudo-perception

An intuitive justification of these expression is provided by the following argument. Consider an FPS where $\ell_\rho = \bar{\ell} = 1$. Then it costs $Q/(1-\rho)$ units to reach a desired state, if it is assumed that the state is perfectly observed. This is true because Q is the cost (per unit time) of being in an undesirable state instead of being in a most desirable state, and because the expected number of transitions required to reach the most desirable state is $1/(1-\rho)$. Suppose now that state uncertainty is introduced. Then the uncertainty, caused when the most recent state perception occured ℓ time units ago, is $\bar{\alpha}^{\ell}$. Thus the value of a single perception, delayed ℓ time units, is

$$\bar{\alpha}^{\ell}[4Q/(1-\rho)] + \bar{\alpha}^{\ell+1}[4Q/(1-\rho)] + \ldots \cong \bar{\alpha}^{\ell}\left[\frac{4Q}{(1-\rho)(1-\bar{a})}\right]$$

The cost of pseudo-perception is similarly derived, resulting in an additional factor of $(1-\bar{\alpha})$ in the denominator.

f. Feedback Realization of ε-optimal Controllers

The definition of an FPS, given in Section 2a, is structural rather than functional. Much of the detail provided in the specification of a particular FPS is irrelevant to an observer who has access only to inputs and outputs. For example, the internal states of an FPS may be reordered (by means of suitable row and column manipulations on the initial state probability vector and transition probability matrices) to obtain a new system which cannot be distinguished from the first on the basis of input-output histories alone. Two or more FPS's which are indistinguishable in this sense will be called equivalent.

A valued finite probabilistic system (VFPS) was defined as an FPS, along with a reward structure which allows a performance to be assigned to any control strategy. If two or more VFPS's consist of equivalent FPS's, along with reward structures that result in identical performance indices, these VFPS's will be called equivalent.

The problem under consideration is to compute a control strategy that optimizes the performance index corresponding to a particular VFPS. The concept of equivalence is used to transform this problem into one that is more easily solved: it suffices to compute a strategy which optimizes the performance index corresponding to any particular equivalent VFPS.

A convenient equivalent VFPS is constructed by a procedure known as augmentation. Any augmented VFPS is completely described by the original VFPS from which it was obtained, and a memory set, M, which is a finite set of strings of input-output pairs. An observer is required to select, from the memory set, the element that correctly lists the largest number of most recent input-output pairs; this is called the memory state. An augmented state consists of the internal state delayed by a quantity equal to the length of the memory state, along with the memory state itself. Since the augmented state may be regarded as the state of a controlled Markov chain, an equivalent VFPS having augmented internal states in place of internal states may be constructed. This VFPS is the outcome of <u>augmentation induced by M</u>.

An example of augmentation may be found in Section 3. During the n-th iteration, a memory set containing all strings of (n-1) input-output pairs is employed. Thus the memory state consists of the (n-1) most recent input-output pairs, and the augmented state consists of the true internal state delayed by (n-1) time units, along with the string of all intervening input-output pairs.

The perceptive or feasible strategy computed during an iteration of perceptive dynamic programming determines inputs on the basis of the current augmented state alone, and thus, it may be viewed as a <u>feedback strategy</u>. This implies that the system under such a strategy is a Markov chain, a fact that is useful in evaluating feasible performances.

6. Organization of the Report

Mathematical tools for the analysis of FPS's are introduced in Chapter II. A brief outline of this chapter is given below. The notation to be used in representing strings of input-output pairs is presented in Section 7. The concepts of "memory state" and "augmentation" are made precise in Sections 8 and 9. In the computational technique of perceptive dynamic programming, it is assumed that the augmented state (induced by some memory set M) can be "perceived" by the controller; dynamic programming then yields a rule for optimal (perceptive) decision-making, expressed as a policy on the augmented state set. However the performance index is a function of <u>strategy</u>, or rule for decision-making on the basis of all past inputs, states and outputs. The relationship between a strategy and the policy which realizes it is made precise in Section 10. <u>Connectivity</u> and <u>reachability</u> are defined in Section 11. It is demonstrated that both properties are preserved when the state is augmented. Sections 12 and 13 provide the basis for definition, in Section 14, of <u>detectability</u>. This involves the development of appropriate metrics and contractions, as discussed in Section 5d. Solutions to the finite-memory estimation problem are then introduced. The final sections of Chapter II are concerned with applicability of perceptive dynamic programming. In Section 15, it is shown how any free FPS can be decomposed into detectable parts; thus perceptive dynamic programming can always be applied to each detectable component of the problem. Section 16 establishes that very few FPS's are equivalent to a state-calculable FPS; were this not so, many FPS control problems could be solved by dynamic programming alone.

Chapter III is devoted to a study of the structure of optimal controllers. The finite-horizon and state-observable cases are reviewed in Sections 17 and 18. It is then demonstrated, in Section 19, that (under suitable assumptions) an optimal strategy will exist, although it may require infinite memory. In some cases, however, the notion of an undiscounted infinite horizon is ill-defined, and the problem is meaningless. An alternate formulation, in which irregular features are constrained to finite-horizon consideration, is proposed in Section 20.

Any optimal controller which requires infinite memory cannot, in general, be described exactly. Chapter IV introduces a computational technique which allows the optimal performance to be approached as a memory constraint is weakened. This technique, called perceptive dynamic programming, approximates the problem as a Markov decision problem solvable by dynamic programming. The approximation is obtained by means of an assumption that delayed state values can be artifically "perceived." Like dynamic programming, perceptive dynamic programming is a general approach which can be realized in many ways; these are discussed in Section 21. Results obtained by implementation of a perceptive dynamic programming algorithm are then presented: a solution to the Machine Maintenance and Repair Problem, and an analysis of a computer communication problem.

Peripheral ideas, and conjectures regarding potential extentions of the theory, have been collected in Chapter V.

A symbol table and glossary are provided to assist the reader in assimilating the terminology and notation of Chapter II.

CHAPTER II

ANALYSIS OF FINITE PROBABILISTIC SYSTEMS

7. Input-output Words

Because strings of input-output words play a most important role in the analysis of FPS's, it is essential that a compact notation be developed for their representation. Such a notation is introduced in this section.

(7.1) Notation. A finite string $\underline{a} = a_1 a_2 \ldots a_\ell$ of elements in set A is called a word over A. Words are always identified by underscores. The set of all words over A is denoted A*. $\ell(\underline{a})$ is the length of word $\underline{a}$. $\underline{e}$ is the empty word (over any set). If $\underline{a} = a_1 \ldots a_\ell$ and $\underline{a}' = a'_1 \ldots a'_k$ then $\underline{a}\,\underline{a}' = a_1 \ldots a_\ell a'_1 \ldots a'_k$ is called the concatenation of $\underline{a}$ with $\underline{a}'$; clearly $\underline{a} = \underline{a}\,\underline{e} = \underline{e}\,\underline{a}$ for any word $\underline{a}$. If A and B are sets, then the concatenation AB denotes the set of words of the form $\underline{a}\,\underline{b}$ where $\underline{a} \varepsilon A$ and $\underline{b} \varepsilon B$. A^ℓ is the set of words consisting of exactly ℓ consecutive elements in A; $A^{\ell *}$ is the set of words consisting of up to ℓ consecutive elements in A.

(7.2) Definition. Z denotes the set of input-output pairs (u,y) such that $P(y|u) \neq 0$.

<u>Remark</u>: More generally, Z may be defined as the set of equivalence classes of input-output pairs corresponding to identical non-zero transition probability matrices. The tabulation of Z in Section 3 is consistent with this alternate definition.

(7.3) <u>Notation</u>. The following objects will be used interchangeably: 1) a word over Z, i.e. a string of pairs $(u_1,y_1)\ldots(u_\ell,y_\ell)$, and 2) a pair of words over U and Y, respectively, having equal length, i.e. $(\underline{u},\underline{y}) = (u_1\ldots u_\ell,\ y_1\ldots y_\ell)$. In a free FPS, the input component of an input-output pair may be omitted.

(7.4) <u>Definition</u>. For $\underline{z} = (u_1,y_1)(u_2,y_2)\ \ldots\ (u_\ell,y_\ell)\varepsilon Z^*$, define

$$P(\underline{z}) = P(y_1|u_1)\cdot P(y_2|u_2)\cdot\ldots\cdot P(y_\ell|u_\ell).$$

Also $P(\underline{e})$ is the NxN identity matrix.

<u>Interpretation</u>: $P_{ij}(\underline{z}) = P_{ij}((\underline{u},\underline{y}))$ is the probability that the FPS will emit output word $\underline{y}$ and go to state j, given that it had been in state i and that input word $\underline{u}$ was subsequently accepted.

(7.5) <u>Definition</u>. (a) $I(\underline{z}) = \{i\varepsilon S : P_{ij}(\underline{z})\neq 0,\text{ some } j\varepsilon S\}$

(b) $J(\underline{z}) = \{j\varepsilon S : P_{ij}(\underline{z})\neq 0,\text{ some } i\varepsilon S\}$

Interpretation: $I(z)$ is the set of states that may preceed the evolution of input-output word $\underline{z}$; $J(\underline{z})$ is the set of states that may follow it.

(7.6) Definition. (a) $Z^+ = \{\underline{z}\varepsilon Z^* : P(\underline{z}) \neq 0\}$

(b) $Z^+(\pi^1, \pi^2, \ldots) = \{\underline{z}\varepsilon Z^* : \pi^1 P(\underline{z}) \neq 0,\ \pi^2 P(\underline{z}) \neq 0, \ldots\}$

Interpretation: Z^+ is the set of input-output words that might eventually evolve. $Z^1(\pi^1, \pi^2, \ldots)$ is the set of input-output words that might evolve when the information vector equals π^1, and also might evolve when the information vector equals π^2, etc.

The information vector transition function was defined in (2.8) for a one-step transition, i.e. the case where the information vector is updated as soon as a single input-output pair becomes available. It is possible to generalize this transformation to the case of a multiple-step transition.

(7.7) Definition. For any $\eta\varepsilon\Pi_N$, $\underline{z}\varepsilon Z^+(\eta)$,

$$T(\eta, \underline{z}) = \eta P(\underline{z}) / (\eta P(\underline{z}) 1).$$

(7.8) Lemma. If $\underline{z}\ \underline{z}' \ \varepsilon\ Z^+(\eta)$, then

$$T(\eta, \underline{z}\ \underline{z}') = T(T(\eta, \underline{z}), \underline{z}').$$

8. Memory Sets and Memory States

This section makes precise the notion of a memory set, (a vocabulary of recent input-output pairs), and a memory state (a summary, not necessarily complete, of recent input-output pairs, lying in the memory set). Appropriate notation is first introduced.

(8.1) Definition. $\underline{z}(k_1;k_2)$ denotes the word of input-output pairs that evolved between times k_1 and k_2. Specifically:

$$\underline{z}(k_1;k_2) = ((u(k_1),y(k_1+1))\ (u(k_1+1),y(k_1+2))\ldots\ (u(k_2-1),y(k_2)))$$

(8.2) Definition. (a) "$\underline{<}$" denotes the partial order on Z^* defined by $\underline{z}' \leq \underline{z}$ if $\exists\, \underline{z}''\varepsilon Z^*$ such that $\underline{z}'\ \underline{z}'' = \underline{z}$.

(b) If M is a finite nonempty subset of Z^* that is totally ordered by "$\leq$", then max[M] denotes the unique element $\hat{\underline{z}}$ of M for which there holds $\underline{z} \leq \hat{\underline{z}}$, $\forall\, \underline{z}\varepsilon M$; min[M] is analogously defined.

(c) If $\underline{z}\varepsilon Z^*$, then trunc[$\underline{z}$] = $\{\underline{z}'\varepsilon Z^*: \underline{z}' \leq \underline{z}\}$.

(d) If $\underline{z}' \leq \underline{z}$, then $\underline{z} - \underline{z}' = \underline{z}''$ where $\underline{z}'\underline{z}'' = \underline{z}$.

Interpretation: Recall that $\underline{z}$ is a word (i.e. a string) of input-output pairs. $\underline{z}' \leq \underline{z}$ is used to indicate that $\underline{z}$ can be split into two parts so that $\underline{z}'$ matches the rightmost part. $\underline{z}$ = max[M] is a word in M having the property that all words in M are rightmost substrings of $\underline{z}$. min[M] is a word in M which is a rightmost substring of every (other) word in M. trunc[$\underline{z}$] is the set of rightmost substrings of $\underline{z}$, i.e. truncated versions of z. $\underline{z}-\underline{z}'$ is what remains when the rightmost substring $\underline{z}'$ is removed from $\underline{z}$.

(8.3) Lemma. trunc[$\underline{z}$] is a finite nonempty set which is totally ordered by "$\leq$", and $\underline{e} \in$ trunc[$\underline{z}$].

It is now possible to formulate the following definition:

(8.4) Definition. A memory set M is a finite nonempty subset of Z^* which satisfies

$$\text{(i)} \quad M = \bigcup_{\underline{z} \in M} \text{trunc}[\underline{z}]$$

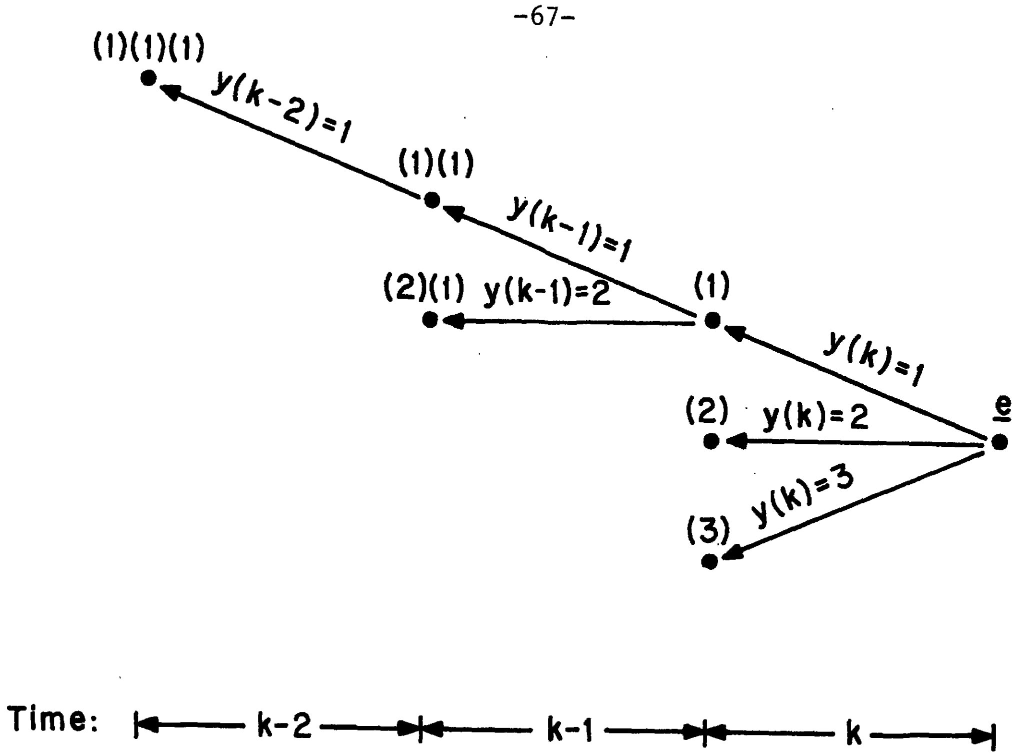
and

$$\text{(ii)} \quad M \subseteq [MZ \cap \{\underline{e}\}] .$$

The memory state induced by M at time k is

$$\underline{z}^M(k) = \max[M \cap \text{trunc}[\underline{z}(0;k)]].$$

Interpretation: The memory set may be arranged in the form of a left-handed tree, called the memory tree, as shown in Figure 8-1. An arrow from $\underline{z}'$ to $\underline{z}$ indicates that $\underline{z}' \leq \underline{z}$. The memory state at any time is the element of M that correctly summarizes the largest number of most recent input-output pairs. Following Figure 8-1, a memory state may



U = {1},

Y = {1,2,3},

M = {e,(1),(2),(3),(1)(1),(2)(1),(1)(1)(1)}.

Note: Since the FPS is free, the input component of an input-output pair may be ignored.

Figure 8-1. A Memory Tree

be constructed by following the tree, from right to left, as far as possible. The first condition which M must satisfy in (8.4) guarantees that a memory tree may be constructed, and hence that memory states will be well-defined. The second condition assures that memory states can be recursively computed, as demonstrated in (8.6) below.

Example: $Z^{\ell *} \cap Z^{+}$ is a memory set. The memory state induced by that memory set, at times $k \varepsilon <\ell, \infty>$, is the string of ℓ most recent input-output pairs.

(8.5) Definition. The memory state transition function induced by M is a mapping $T^M : M \times Z \to M$ given by

$$T^M[\underline{z}, z'] = \max[M \cap \mathrm{trunc}[\underline{z}z']],\ \underline{z} \varepsilon M,\ z' \varepsilon Z .$$

(8.6) Proposition. $\underline{z}^M(k+1) = T^M[\underline{z}^M(k),\ (u(k), y(k+1))]$.

Proof: If $\underline{z}^M(k+1) = \underline{e}$ then the result is trivial. Now assume that $\underline{z}^M(k+1) \neq \underline{e}$. Then it follows that there exists a $\underline{z}' \varepsilon Z^*$ such that $\underline{z}^M(k+1) = \underline{z}'(u(k), y(k+1))$. But, by condition (ii) of (8.4),

$$\begin{aligned} \underline{z}^M(k+1) &= \max[M \cap \mathrm{trunc}[\underline{z}(0;k+1]] \\ &\leq \max[(MZ \cup \{\underline{e}\}) \cap \mathrm{trunc}[\underline{z}(0;k+1)]] \\ &= \max[MZ \cap \mathrm{trunc}[\underline{z}(0;k+1)]] \end{aligned}$$

$$= \max[MZ \cap \mathrm{trunc}[\underline{z}(0;k)(u(k),y(k+1))]]$$

$$= \underline{z}^M(k)(u(k),y(k+1)) \; .$$

So $\underline{z}^M(k+1) \varepsilon M \cap \mathrm{trunc}[\underline{z}^M(k)(u(k),y(k+1))]$, and hence

$$\underline{z}^M(k+1) \leq \max[M \cap \mathrm{trunc}[\underline{z}^M(k)(u(k),y(k+1))]]$$

But

$$\underline{z}^M(k) \leq \underline{z}(0;k)$$

$$\Longrightarrow \underline{z}^M(k)(u(k),y(k+1)) \leq \underline{z}(0;k+1)$$

$$\Longrightarrow \mathrm{trunc}[\underline{z}^M(k)(u(k),y(k+1))] \subseteq \mathrm{trunc}[\underline{z}^M(0;k+1)]$$

$$\Longrightarrow \max[M \cap \mathrm{trunc}[\underline{z}^M(k)(u(k),y(k+1))]] \leq \max[M \cap \mathrm{trunc}[\underline{z}(0;k+1)]]$$

$$= \underline{z}^M(k+1).$$

Thus $\underline{z}^M(k+1) \leq \max[M \cap \mathrm{trunc}[\underline{z}^M(k)(u(k),y(k+1))]] \leq \underline{z}^M(k+1)$, which establishes the desired equality. †

Certain properties of memory sets are now developed for use in later sections.

(8.7) Lemma. (a) An intersection of memory sets is a memory set.

(b) A concatenation of memory sets is a memory set.

(8.8) <u>Definition</u>. If $\tilde{M}$ is a finite subset of Z^*, then mem[M] denotes the smallest memory set containing $\tilde{M}$, i.e. the intersection of all memory sets containing $\tilde{M}$.

(8.9) <u>Definition</u>. The <u>essential part</u> of memory set M is the subset:

$$\text{ess}[M] = \{\max[M \cap \text{trunc}[\underline{z}]] : \underline{z} \varepsilon (Z^+ - M)\} \subseteq M$$

<u>Interpretation</u>: There are elements of a memory set which may become memory states only during an initial transient of bounded duration. For example, in the memory set $Z^{\ell *} \cap Z^+$, the memory state at time k consists of the min(k,ℓ) most recent input-output pairs; if $k \geq \ell$, then the memory state consists of the ℓ most recent input-output pairs; in this case $\text{ess}[Z^{\ell *} \cap Z^+] = Z^{\ell} \cap Z^+$. In the memory tree interpretation of a memory set, a node in M is contained in ess[M] if it has branches in Z^+ that are not contained in M.

(8.10) <u>Lemma</u>. If M is a memory set, then mem[ess[M]] = M.

(8.11) <u>Lemma</u>. If $\underline{z} \varepsilon \text{ess}[M]$, then $T^M[\underline{z}, z'] \varepsilon \text{ess}[M]$.

<u>Interpretation</u>: Once the memory state enters ess[M], it cannot leave it.

(8.12) Definition. If M is a memory set, then

$$\ell_{max}[M] = \max\{\ell(\underline{z}) \ : \ \underline{z}\varepsilon M\}$$

$$\ell_{min}[M] = \min\{\ell(\underline{z}) \ : \ \underline{z}\varepsilon \mathrm{ess}[M]\}$$

(8.13) Lemma. For any control strategy γ,

$$\mathrm{Prob}_{\gamma}\{\underline{z}^{M}(k)\varepsilon \mathrm{ess}[M]\} = 1, \qquad k\varepsilon<\ell_{max}[M],\infty>.$$

Interpretation: The memory state enters ess[M] by the $\ell_{max}[M]$-th transition.

The notion of a memory state transition function, introduced in (8.5), may be extended to multiple-step transitions, as follows.

(8.14) Definition.

$$T^{M}[\underline{z},\underline{z}'] = \max[M\cap \mathrm{trunc}[\underline{z}\ \underline{z}']], \ \underline{z}\varepsilon M, \ \underline{z}\ \underline{z}'\varepsilon Z^{+}$$

(8.15) Lemma.

$$T^{M}[\underline{z},\underline{z}'\underline{z}''] = T^{M}[T^{M}[\underline{z},\underline{z}'],\underline{z}''], \ \underline{z}\varepsilon M, \ \underline{z}', \ \underline{z}''\varepsilon Z^{+}.$$

Interpretation: (8.15) establishes consistency of (8.14) with (8.5) and (8.6).

9. Equivalence and Augmentation

This section introduces the "augmented system induced" by a memory set, an FPS whose state consists of a delayed internal state and a memory state. The augmented system will be seen to be "equivalent" to the original system, in the sense that they are indistinguishable on the basis of inputs and outputs alone.

(9.1) Definition. The input-output relation of an FPS is a mapping $p : Z^{*} \to [0,1]$ given by $p(z) = \pi(0)P(\underline{z})1$.

Interpretation: $p(\underline{z}) = p((\underline{u},\underline{y}))$ is the probability that output word $\underline{y}$ will be emitted initially, given that the word of initial inputs was $\underline{u}$. The mapping p is a summary of all externally discernable characteristics of an FPS.

(9.2) Definition. The expected incremental reward function of a VFPS is a mapping $q : Z^{+}(\pi(0)) \times U \to R$ given by $q(\underline{z},u) = T(\pi(0),\underline{z})q(u)$.

Interpretation: $q(\underline{z},u)$ is the expected incremental reward if, immediately following the generation of input-output history $\underline{z}$, input u is selected. The mappings p and q together summarize all externally discernable characteristics of a VFPS.

(9.3) Definition. Two or more FPS's are (mutually) equivalent if their input-output relations coincide. Two or more VFPS's are (mutually) equivalent if both their input-output relations and their expected incremental reward functions respectively coincide.

The problem of constructing an FPS specification having a given input-output relation is called stochastic realization. Stochastic realization has been extensively studied by Paz (1971). Picci, in hitherto unpublished research, formulated the conjecture that almost every FPS is equivalent to a state-calculable FPS. Picci's conjecture is disproved in Section 18 of this report.

Realization of a particular input-output relation generally entails the incorporation of artificial structure into the model. The smaller the number of states used, the greater the quantity of artificial structure incorporated; consequently state calculability may be inhibited. This is illustrated below:

(9.4) Example. Consider a free state-calculable FPS with $U=\{1\}$, $Y = \{1,2,3,4\}$, $N=8$, $\pi(0) = e^{1}$, and

$$P(1|1)=\begin{bmatrix} .1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ .2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ .3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ .4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .4 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad P(2|1)=\begin{bmatrix} 0 & 0 & .4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P(3|1)=\begin{bmatrix} 0 & 0 & 0 & 0 & .1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .4 & 0 & 0 \end{bmatrix} \quad P(4|1)=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & .4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .1 \end{bmatrix}$$

This FPS is not only state-calculable; its state is uniquely determined by the most recent pair of outputs. It is equivalent to the 4-state FPS having transition probability matrices:

$$P(1|1) = \begin{bmatrix} .1 & 0 & .4 & 0 \\ .2 & 0 & .3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad P(2|1) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ .3 & 0 & .2 & 0 \\ .4 & 0 & .1 & 0 \end{bmatrix}$$

$$P(3|1) = \begin{bmatrix} 0 & .1 & 0 & .4 \\ 0 & .2 & 0 & .3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad P(4|1) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & .3 & 0 & .2 \\ 0 & .4 & 0 & .1 \end{bmatrix}$$

Equivalence is verified by using the Markov property of consecutive output pairs. The second process, though equivalent to the first, is not state calculable.

The problem of computing, for a given FPS, an equivalent system having a minimal number of states is (to the author's knowledge) unsolved, and, in any event, very intricate. It is, of course, possible to eliminate states that are overtly redundant (see Paz [1971], Section I.B.2); the elimination of such redundancy may reduce computation time in the algorithms of Chapter IV in this report. On the other hand, it is by increasing the number of states that state-calculability is enhanced, and the problem is eventually solved. This situation is notably different from that found in linear systems, where observability occurs only when the state space has been reduced to a minimal dimension.

(9.5) Definition. The augmented state set induced by memory set M is the set $X[M] = \{[i,\underline{z}] : i \in S,\ \underline{z} \in M,\ e^{i}P(\underline{z})1>0\}$. The augmented state induced by M at time k is $x^{M}(k) = [s(k-\ell(\underline{z}^{M}(k))),\ \underline{z}^{M}(k)]$.

Example: Memory set $Z^{\ell *} \cap Z^{+}$ induces augmented states consisting of the internal state delayed by ℓ time units and the memory state of ℓ most recent input-output pairs.

(9.6) Proposition. For any FPS along with a memory set M, there is a unique equivalent FPS having internal state process $\{x^{M}(k)\}$.

Proof: It is sufficient to show that the augmented underlying process is a controlled Markov chain. This occurs provided that the sequence of controlled random variables $\{k-\ell(\underline{z}^M(k))\}$ is non-decreasing, a trivial consequence of (8.6). †

(9.7) Definition. The FPS which is equivalent to a given FPS, and has internal states that are the augmented states (of the given system) induced by memory set M, is called the augmentation (of that FPS) induced by M, or, more informally, the augmented system induced by M.

A particularly efficient representation of the augmented system is obtained by recognizing that, although the augmented system has approximately N·#M states, each of these may effect a transition to at most N·#Z states. Specifically, $P^M_{\underline{z}}(i,j,z')$ may denote the probability that a transition to $[j,T^M(\underline{z},z')]$ will occur, given that the system is presently in augmented internal state $[i,\underline{z}]$ and that the input component of z' has been selected. It is given by the formula:

$$P^M_{\underline{z}}(i,j,z') = \begin{cases} \dfrac{\Sigma_{k\varepsilon S} P_{ij}{}^{T^M(\underline{z},z')}\, P_{jk}(\underline{zz'}-T^M(\underline{z},z'))}{\Sigma_{k\varepsilon S} P_{ik}(\underline{z})}, & \text{if } \underline{z}\varepsilon Z^+(e^i) \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{9.8}$$

The transformed incremental rewards are described by arrays:

$$q^M_{\underline{z}}(i,u) = \begin{cases} T(e^i,\underline{z})q(u), & \text{if } \underline{z}\varepsilon Z^+(e^i) \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{9.9}$$

Thus, the memory requirement to describe a particular augmented FPS is roughly $\#M \times [(N^2 \times \#Z) + (N \times \#U)]$ words. The fact that this quantity grows linearly in $\#M$ is particularly significant as the augmented system has $N \times \#M$ states, and the number of transition probability matrix entries might normally be expected to grow as the square of the number of augmented states.

10. Classification of Strategies

A strategy was defined, in Section 2d, as a rule for the determination of inputs, specified by probability distributions for u(k) conditioned on each past history [s(0),u(0),y(1),s(1),..., s(k-1), u(k-1),y(k),s(k)]. In such a form, however, the description of a strategy occupies an infinite tableau, and decisions must be made on the basis of infinite memory. Such difficulties are avoided by introducing a class of strategies that are totally specified by a finite tableau, called a policy.

(10.1) Definition. Let M by a memory set. Then ϕ is a feasible strategy adapted to M if there is a policy $\bar{\phi}$: M-> U such that

$$\text{Prob}_{\phi}\{u(k) = \bar{\phi}(\underline{z}^M(k))\} = 1, \qquad k\varepsilon<0,\infty> .$$

$\bar{\phi}$ is then the policy (on M) which realizes ϕ. Φ[M] denotes the set of feasible strategies adapted to M. A feasible strategy that is adapted to some memory set is called a feasible adapted strategy.

Interpretation: If $\phi\varepsilon\Phi$[M], then the inputs prescribed by ϕ can be determined by a finite memory controller whose memory set is M. Note that the input specified by ϕ and that specified by $\bar{\phi}$ need not coincide in situations which cannot occur when ϕ is used.

Remark: There exist finite-memory controllers that are not adapted (to any memory set).

(10.2) Definition. Let M be a memory set. Then ψ is a perceptive strategy adapted to M if there is a policy $\overline{\psi}$: X[M]-> U such that

$$\text{Prob}_{\psi}\{u(k)=\overline{\psi}[x^{M}(k)]\} = 1, \qquad k\varepsilon<0,\infty>$$

$\overline{\psi}$ is the policy (on X[M]) which realizes ψ. $\Psi[M]$ denotes the set of all perceptive policies adapted to M. A perceptive strategy that is adapted to some memory set is called a perceptive adapted strategy.

Interpretation: If $\psi\varepsilon\Psi[M]$ then the inputs prescribed by ψ can be computed on the basis of $x^{M}(k)$ alone. Note again that the input specified by ψ and that specified by $\overline{\psi}$ need not coincide in situations which cannot occur when ψ is used.

(10.3) Lemma. (a) $\Phi[M] \subseteq \Psi[M]$.

(b) If $M\subset M'$, then $\Phi[M]\subset\Phi[M]'$.

A (feasible or perceptive) adapted strategy induces on any FPS a free system whose underlying process is a Markov chain. Thus each augmented state may be characterized as transient or recurrent, under any particular adapted strategy. The memory state, likewise, may be given these attributes.

(10.4) Definition. Consider an adapted strategy ψ, along with a memory state $\underline{z}\epsilon M$. If there is an $i\epsilon S$ such that the augmented state $[i,\underline{z}]$ is recurrent under ψ, then $\underline{z}$ is recurrent under ψ; otherwise $\underline{z}$ is transient under ψ .

The concept of transient and recurrent memory states has the following application: Suppose that some optimal (or ϵ-optimal) strategy has been specified, by means of policy on a memory set to which that strategy is adapted. If the performance index is average gain over an undiscounted infinite horizon, then the policy may be modified in a number of ways without affecting performance. In particular, the input specified for any transient memory state may be replaced by any other value, provided that it does not cause that memory state to become recurrent. In this manner, an optimal or suboptimal strategy adapted to a smaller memory set might be obtained.

11. Connectivity

Graph properties of Markov chains have been generalized to controlled Markov chains by Platzman [1977]. These concepts are now extended to FPS's.

(11.1) Definition. State i is connected to state j if there exists an input-output word $\underline{z} \varepsilon Z^+$ such that $P_{ij}(\underline{z}) > 0$.

Interpretation: If i is connected to j, then it is possible for the system to travel from state i to state j, provided that appropriate inputs are accepted. This does not imply availability of reset inputs (which transfer the system to a given state with probability one).

(11.2) Definition. A connected class C is a set of mutually connected states, none of which is connected to a state outside C.

Clearly the state set of any FPS contains at least one connected class.

(11.3) Definition. An FPS is connected if its state set is a connected class.

(11.4) Proposition. If an FPS is connected, then there is an integer $\ell_\chi \varepsilon \langle 1,N \rangle$ and a $\chi \varepsilon [0,1)$ such that, corresponding to any $i,j \varepsilon S$, an input word $\underline{\hat{u}}\varepsilon U^*$ exists, satisfying $1 - \left[\Sigma_{\underline{y}\varepsilon Y^{\ell(\hat{u})}} P_{ij}((\underline{\hat{u}},\underline{y}))\right] \leq \chi$.

Remark: ℓ_χ and χ may be computed by enumeration on U^{N*}. A more efficient algorithm seeks a least costly path from node i to node j, where $-\log\left[\Sigma_{y\varepsilon Y} P_{i'j'}(y|u)\right]$ is the cost of a link from i' to j' labeled with input u.

In a connected FPS, it is possible to select inputs which allow the system to travel from any state to any other, provided that the initial state is known. This assumption is avoided in (11.5), below.

(11.5) Definition. An FPS is reachable if there is an integer ℓ_ρ and a $\rho\varepsilon[0,1)$ such that, corresponding to every $\pi\varepsilon\Pi_N$ and $j\varepsilon S$, an input word $\underline{\hat{u}}\varepsilon U^{\ell_\rho}$ exists satisfying:

$$1 - \left[\Sigma_{\underline{y}\varepsilon Y^{\ell(\hat{u})}} \Sigma_{i\varepsilon S}\pi_i P_{ij}((\underline{\hat{u}},\underline{y}))\right] \leq \rho$$

Interpretation: If an FPS is reachable, then for any value of the information vector, there exists a sequence of inputs, which will drive the state to a desired value with probability $1-\rho$ or more.

(11.6) Proposition. An FPS is reachable iff it is connected.

Proof: Assume connectivity and set $\ell_\rho=\ell_\chi$; $\rho=1-\frac{1}{N}(1-\chi)$. For any $\pi\varepsilon\Pi_N$, there is an $i\varepsilon S$ such that $\pi_i \geq 1/N$. Selection of $\underline{u}$ according to (11.4), for i as determined above and j as desired, satisfies the criterion in (11.5). That reachability implies connectivity is trivial. †

Remark: Although reachability is the property required to establish the existence of optimal strategies in FPS control problems, connectivity is the property that can be decided algorithmically. Reachability can be established by inspection in some systems (e.g. a network of finite queues), and the bounds thus obtained will be tighter than those obtained through connectivity arguments.

(11.7) Definition. An FPS is simply connected if its state set consists of a single connected class, along with a (possibly empty) set of states which are transient under all feasible strategies.

(11.8) Theorem. Let C be the connected class in the state set of a simply connected FPS, and let M be a memory set. Then the augmented system induced by M is simply connected, having connected class

$$\hat{X}[M] = \{[i,\underline{z}] : i\varepsilon C,\ \underline{z}\varepsilon \mathrm{ess}[M]\cap Z^+(e^i)\} \subseteq X[M]$$

Proof: Augmented states of the form $[i,\underline{z}]$ with $i\varepsilon S-C$ are clearly transient. Those of the form $[i,\underline{z}]$ with $\underline{z}\varepsilon M-\mathrm{ess}[M]$ cannot occur after the $\ell_{max}[M]$-th transition, by (8.12). To show that $[i,\underline{z}]$ and $[i',\underline{z}']\varepsilon\hat{X}[M]$ are connected, select $j\varepsilon C$ so that $P_{ij}(z) > 0$ and $\underline{z}''\varepsilon Z^+$

so that $P_{ji'}(\underline{z}'') > 0$, the existence of the latter being guaranteed by (11.1). Then the augmented system may travel from state $[i,\underline{z}]$ to state $[i',\underline{z}']$ when the intervening input-output word is $\underline{z}''\underline{z}'$. †

An algorithm which decides whether a given state-observable FPS is simply connected was introduced by Platzman [1977]. Simple connectivity of the underlying process is not necessarily implied by simple connectivity of the FPS, as is illustrated below:

(11.9) Example. Let $U=\{1,2\}$, $S=\{1,2,3\}$, $Y=\{1\}$, $\pi(0) = (1/2,1/2,0)$ and

$$P(1|1) = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/3 & 1/3 & 1/3 \\ 0 & 0 & 1 \end{bmatrix}, \qquad P(1|2) = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The single connected class is $\{3\}$; states 1 and 2 are transient under all feasible strategies. Yet there exists a perceptive strategy under which states 1 and 2 form a recurrent class: this is the strategy $u(k) = s(k)$.

The following algorithm will (in principle) determine whether or not a given FPS is simple connected. It does so by seeking to discover a strategy under which the state will never enter the connected class.

(11.10) Algorithm. Let C denote the unique connected class in the state set of a given FPS. Label each nonempty subset H of $S-C$ with a binary digit denoted $c(H)$; initially $c(H)=0$, for all $H \subset S-C$. Then perform the following step, for every $H \subset S-C$, until the $c(\cdot)$ remain invariant: set $c(H)=1$ if, for every $u \varepsilon U$, either

$$\Sigma_{i\varepsilon H}\Sigma_{y\varepsilon Y}\Sigma_{j\varepsilon C}P_{ij}(y|u) > 0$$

or

$$\Sigma_{y\varepsilon Y}\ c(\{j : P_{ij}(y|u) > 0,\ i\varepsilon H\}) > 0$$

Then the FPS is simply connected iff $c(H)\rightarrow 1$, for all nonempty subsets H of S-C.

(11.11) <u>Proposition</u>. If an FPS is simply connected, then there is an integer $\ell \leq 2^{\#(S-C)}$ such that the augmented system induced by M has a simply connected underlying process whenever $\ell_{min}[M] \leq \ell$.

<u>Proof</u>: Define $H(k) = \{i : \eta_i(k) > 0\}$ and assume that $H(0) \subseteq S-C$. Then (11.10) implies the following: for any given values of H(k-1) and u(k-1), either H(k) may contain elements in C, or there is a y(k) such that H(k) will be distinct from H(0)... H(k-1). But there are $2^{\#(S-C)}-1$ nonempty subsets of S-C, so $H(2^{\#(S-C)})$ may contain elements in C, i.e. $\text{Prob}\{H\left(2^{\#(S-C)}\right) \cap C \text{ is nonempty}\} > 0$ under any feasible strategy. Thus, internal states lying outside C are transient under any strategy adapted to M, provided that $\ell_{min}[M] \geq 2^{\#(S-C)}$. †

When S-C is a large set, the enumeration of subsets of S-C is computationally infeasible. A <u>sufficient condition</u> for simple connectivity is now derived.

(11.12) Lemma. If, in the outcome of Algorithm (11.10), $c(A)=1$ and $B \supseteq A$, then $c(B)=1$.

(11.13) Theorem. An FPS is simply connected if its underlying process is simply connected.

Proof: Simple connectivity of the underlying process implies $c(\{i\})=1$, $\forall\ i \varepsilon S-C$. Hence, by (11.12), $c(H)=1$ for all nonempty subsets H of S-C. In (11.10), this is the sufficient condition for simple connectivity.

†

12. Metrics

This section introduces metrics that are used to measure the "closeness" of approximations to the information vector. The continuity of convex functions with respect to these metrics is then established.

a. Definition of the Metrics

(12.1) Definition. Consider $\pi \varepsilon \Pi_N$, $w \varepsilon R_N$ with $w \geq 0$ and $\pi w > 0$. Then $\pi o w$ is a vector in Π_N having entries:

$$(\pi o w)_i = \frac{\pi_i w_i}{\pi w} .$$

Interpretation: This is merely Bayes' operator. For example, π might represent *a priori* probabilities of some random variable, s, on sample space S. Given an event occurring with conditional probability w_i provided that i is the true value of s, then $\pi o w$ is the vector of *a posteriori* probabilities of random variable s.

(12.2) Definition. For $\pi, \pi' \varepsilon \Pi_N$, define

(a) $\delta[\pi,\pi'] = \Sigma_{i \varepsilon S} (\pi_i - \pi'_i)^+$;

(b) $\Delta[\pi,\pi'] = \sup\{\delta[\pi o w, \pi' o w] : w \varepsilon R_N, w > 0\}$;

(c) $D[\pi,\pi] = 1 - \min[\{\pi_i/\pi'_i : \pi'_i > 0, i \varepsilon S\} \cap \{\pi'_i/\pi_i : \pi_i > 0, i \varepsilon S\}]$.

Remark: An interpretation of these functions is given in section 12b, following the derivation of certain fundamental properties.

(12.3) Lemma. (a) $0 \leq \delta[\pi,\pi'] = 1/2|\pi-\pi'| \leq \Delta[\pi,\pi'] \leq 1$;

(b) $0 \leq D[\pi,\pi'] \leq 1$.

(12.4) Lemma. $\Delta(\pi o w, \pi' o w] \leq \Delta[\pi,\pi']$, $\forall \pi,\pi' \epsilon \Pi_N$, $w \epsilon R_N$, $w \geq 0$, $\pi w>0$, $\pi' w>0$.

(12.5) Proposition. δ,Δ, and D are metrics on Π_N.

Proof: A metric satisfies

(i) $f[\pi,\pi'] \geq 0$,

(ii) $f[\pi,\pi'] = 0 \iff \pi = \pi'$,

(iii) $f[\pi,\pi'] = f[\pi',\pi]$,

(iv) $f[\pi,\pi'] + f[\pi',\pi''] \geq f[\pi,\pi'']$.

(a) Since $|\cdot|$ is norm on R_N, it defines a metric $|\pi-\pi'|$ on Π_N. By (12.3)(a), $\delta[\cdot,\cdot]$ is a metric on Π_N .

(b) Parts (i) and (ii) are trivial.

(iii) $\Delta[\pi,\pi'] = \sup\{\delta[\pi o w,\pi' o w] : w \epsilon R_N, w>0\}$

$= \sup\{\delta[\pi' o w, \pi o w] : w \epsilon R_N, w>0\}$

$= \Delta[\pi',\pi]$.

(iv) $\Delta[\pi,\pi'] + \Delta[\pi',\pi'']$

$\geq \sup\{\delta[\pi o w,\pi' o w] + \delta[\pi' o w, \pi'' o w] : w \epsilon R_N, w>0\}$

$\geq \sup\{\delta[\pi o w,\pi'' o w] : w \epsilon R_N, w>0\}$

$= \Delta[\pi,\pi'']$.

(c) Parts (i) and (ii) are trivial.

(iii) $D[\pi,\pi'] = D[\pi',\pi]$ by symmetry.

(iv) For $\pi,\pi',\pi''\varepsilon\Pi_N$, assume with no loss of generality that $\pi_1'' > 0$ and $D[\pi,\pi''] = 1 - (\pi_1/\pi_1'')$. If $\pi_1' = 0$, then $D[\pi',\pi''] = 1$ and $D[\pi,\pi'] + D[\pi',\pi''] \geq 1 \geq D[\pi,\pi'']$. If $\pi_1' > 0$, then $(\pi_1/\pi_1'') = (\pi_1/\pi_1')(\pi_1'/\pi_1'')$ and $(1-D[\pi,\pi'])(1-D[\pi',\pi'']) \leq 1 - D[\pi,\pi'']$, implying $D[\pi,\pi''] \leq D[\pi,\pi'] + D[\pi',\pi''] - D[\pi,\pi'] \cdot D[\pi',\pi''] \leq D[\pi,\pi'] + D[\pi',\pi'']$. †

(12.6) Theorem. (Evaluation of Δ). For $\pi,\pi'\varepsilon\Pi_N$, define:

$$c_1 = \min\{\pi_i'/\pi_i : \pi_i' > 0\} ,$$

$$c_2 = \min\{\pi_i/\pi_i' : \pi_i > 0\} .$$

Then

$$\Delta[\pi,\pi'] = \frac{1 - \sqrt{c_1 c_2}}{1 + \sqrt{c_1 c_2}} .$$

Proof: If $\{i : \pi_i > 0\} \neq \{i : \pi_i' > 0\}$ then $\Delta[\pi,\pi'] = 1$. To see this, assume without loss of generality that there is an $i\varepsilon S$ such that $\pi_i > 0$ and $\pi_i' = 0$. Then $\{w^m\} = \{(\frac{1}{m})1 + (1 - \frac{1}{m})e^i\}$ is a sequence in R_N for which $\lim_{m\to\infty} \delta[\pi o w^m, \pi' o w^m] = 1$, since $(\pi o w^m)_i \to 1$ and $(\pi' o w^m)_i = 0$. By (12.3)(a), the sequence $\{w^m\}$ is supremal.

It follows from (12.5) that $\Delta[\pi,\pi] = 0$. The case $\pi_i>0 \iff \pi_i'>0$, $\pi \neq \pi'$ remains. By (12.5), $\Delta[\pi,\pi'] > 0$. Assume without loss of generality that $\pi > 0$ and $\pi' > 0$. Clearly $0 < c_1 < 1$ and $0 < c_2 < 1$; hence

$$0 < c_1 < c_2^{-1} < \infty .$$

Define:

$$\Delta_\zeta[\pi,\pi'] = \sup\{\delta[\pi \circ w,\pi' \circ w] : w \in R_N, \quad w>0, \quad \pi' w/\pi w = \zeta\}$$

$$= \sup\left\{\Sigma_{i \in S} \left(\pi_i w_i - \frac{\pi'_i w_i}{\zeta}\right)^+ : w \in R_N , w \geq 0, \right.$$

$$\left. \pi w = 1, \; \pi' w = \zeta\right\}$$

which exists for all $c_1 \leq \zeta \leq c_2^{-1}$. Clearly

$$\Delta[\pi,\pi'] = \max\{\Delta_\zeta[\pi,\pi'] : c_1 \leq \zeta \leq c_2^{-1}\}.$$

Now $\Delta_\zeta[\pi,\pi']$ may be expressed as the solution of a linear program

$$\Delta_\zeta[\pi,\pi'] = \begin{bmatrix} \text{max:} & aw \\ \text{subject to:} & \pi w = 1 \\ & \tilde{\pi}' w = 1 \\ & w \geq 0 \end{bmatrix} ,$$

where

$$a_i = (\pi_i - \tilde{\pi}'_i)^+,$$

$$\tilde{\pi}'_i = \pi'_i/\zeta .$$

Any optimal basic w that solves this linear program has at most two non-zero entries; let these be denoted (i,j). Then

$$\Delta_\zeta[\pi,\pi'] = \left[\begin{array}{ll} \text{max:} & a_i w_i + a_j w_j \\ \text{subject to:} & w_i \geq 0,\ w_j \geq 0 \\ & \pi_i w_i + \pi_j w_j = 1 \\ & \tilde{\pi}'_i w_i + \tilde{\pi}'_j w_j = 1 \end{array}\right]$$

Assume without loss of generality that

$$(i,j) \in \Lambda = \{(i,j) : (\pi'_i/\pi_i) < (\pi'_j/\pi_j)\}.$$

Now $a_i > 0$ and $a_j = 0$; for otherwise one of the following must hold:

(i) $a_i = 0,\ a_j = 0 \Rightarrow \Delta_\zeta[\pi,\pi'] = 0$

(ii) $a_i > 0,\ a_j > 0 \Rightarrow \Delta_\zeta[\pi,\pi'] = a_i w_i + a_j w_j = (\pi_i - \tilde{\pi}'_i)w_i$

$+ (\pi_j - \tilde{\pi}'_j)w_j = 1 - 1 = 0.$

(iii) $a_i = 0,\ a_j > 0 \Rightarrow (i,j) \notin \Lambda.$

Hence ζ must be such that $(\pi'_i/\pi_i) \leq \zeta \leq (\pi'_j/\pi_j)$. The basic feasible solution with indices (i,j) is now seen to take the form:

$$w_i^* = \frac{\tilde{\pi}'_j - \pi_j}{\pi_i\tilde{\pi}'_j - \pi_j\tilde{\pi}_i} \geq 0$$

$$w_j^* = \frac{\pi_i - \tilde{\pi}'_i}{\pi_i\tilde{\pi}'_j - \pi_j\tilde{\pi}'_i}$$

and the corresponding expression for $\Delta_\zeta[\pi,\pi']$ is

$$\Delta_{\zeta,i,j}[\pi,\pi'] = aw^* = a_i w_i^*$$

$$= \frac{(\pi_i - \tilde{\pi}_i')\ (\tilde{\pi}_j' - \pi_j)}{\pi_i \tilde{\pi}_j' - \pi_j \tilde{\pi}_i'}$$

$$= \frac{(\zeta\pi_i - \pi_i')\ (\pi_j' - \zeta\pi_j)}{\zeta \cdot (\pi_i \pi_j' - \pi_j \pi_i')}$$

$$= \frac{\pi_i \pi_j' + \pi_j \pi_i' - \zeta\pi_i\pi_j - \zeta^{-1}\pi_i'\pi_j'}{\pi_i \pi_j' - \pi_j \pi_i'} .$$

Now

$$\Delta[\pi,\pi'] = \max\left\{\Delta_\zeta[\pi,\pi'] \ :\ c_1 \le \zeta \le c_2^{-1}\right\}$$

$$= \max\left\{\Delta_{\zeta,i,j}[\pi,\pi'] \ :\ (i,j)\varepsilon\Lambda,\ (\pi_i'/\pi_i) \le \zeta \le (\pi_j'/\pi_j)\right\}$$

$$= \max_{(i,j)\varepsilon\Lambda}\left\{\max_{(\pi_i'/\pi_i)\le\zeta\le(\pi_j'/\pi_j)}\left\{\Delta_{\zeta,i,j}\ [\pi,\pi']\right\}\right\}$$

Since $\Delta_{\zeta,i,j}[\pi,\pi']$ is <u>concave</u> in ζ, it achieves a unique maximum at

$\zeta = \zeta^* = \sqrt{\dfrac{\pi_i'\pi_j'}{\pi_i\pi_j}}$. Thus

$$\Delta[\pi,\pi'] = \max_{(i,j)\varepsilon\Lambda}\left\{\Delta_{\zeta^*,i,j}\ [\pi,\pi']\right\}$$

$$= \max_{(i,j)\varepsilon\Lambda} \left\{ \frac{1 - \sqrt{\frac{\pi_i'\pi_j}{\pi_i\pi_j'}}}{1 + \sqrt{\frac{\pi_i'\pi_j}{\pi_i\pi_j'}}} \right\}$$

$$= \frac{1 - \sqrt{c_1 c_2}}{1 + \sqrt{c_1 c_2}} \qquad \dagger$$

b. Discussion

The metric δ , also known as the Hajnal measure, has many applications in the theory of ergodic Markov chains; see Paz [1971]. Informally, $\delta[\pi,\pi']$ is the (minimal) "quantity" of probability that would have to be "reassigned" in order to transform probability distribution π into probability distribution π' . Similarly, $\Delta[\pi,\pi']$ is the minimal quantity of conditional probability by which π and π' might differ if they were supplemented by identical observations (in the sense of the interpretation following (12.1)). Consequently two information vectors that are very close in the sense of δ may be far apart in the sense of Δ. This occurs because subsequent observations might cause the two information vectors (representing similar a priori assumptions) to be transformed into radically different conclusions.

(12.7) Example. Consider an FPS in which $\pi(0) = (1-\varepsilon, \varepsilon)$, $\varepsilon<<1$, but it is desired to approximate $\pi(0)$ by $e^1 = (1,0)$. In a δ sense,

$\pi(0)$ is "near" the approximation e^1 ; this indicates that the unconditional expectation of a function of the initial state will not be significantly affected by this approximation. Suppose, however, that every input-output pair which subsequently evolves corresponds to transition probabilities $\begin{bmatrix} .1 & 0 \\ 0 & .9 \end{bmatrix}$. Given a sufficient number of input-output pairs of this form, the conditional initial state probability vector tends to (0, 1); yet if the approximation $\pi(0) \cong e^1$ is used, then the conditional initial state probability vector will remain e^1. Thus an initial error, of δ-sense magnitude $\varepsilon \ll 1$, may lead to an eventual error of δ-sense magnitude arbitrarily close to 1.

The distinction between δ and Δ is also illuminated by an examination of the topologies they induce on Π_N : the topology induced by δ is continuous, but Δ causes Π_N to be separated into faces of the form $\Pi_N(H) = \{\pi \varepsilon \Pi_N : \pi_i > 0 \iff i \varepsilon H\}$. These are exactly the subsets on which a convex function over Π_N is guaranteed to be continuous (with respect to the Euclidean metric; see Rockafellar [1970], Chapter 10).

c. Some Properties of Metric D

Metric D is introduced mainly for the purpose of making continuity of convex functions more explicit.

(12.8) Proposition. $\Delta[\pi,\pi'] \leq D[\pi,\pi'] \leq 4\Delta[\pi,\pi']$.

<u>Proof</u>: Let c_1, c_2 be as in (12.6), so

$$\Delta[\pi,\pi'] = \frac{1 - \sqrt{c_1 c_2}}{1 + \sqrt{c_1 c_2}}$$

$$D[\pi,\pi'] = 1 - \min(c_1, c_2)$$

If $c_1 = 0$ or $c_2 = 0$, then the result is trivial. However, if $c_1 \neq 0$ and $c_2 \neq 0$, then $\{i : \pi_i \neq 0\} = \{i : \pi'_i \neq 0\}$ and $c_1, c_2 \leq 1$, since the entries of π and π' (respectively) sum to one. Now:

$$\Delta[\pi,\pi'] \leq 1 - \sqrt{c_1 c_2} \leq 1 - \min(c_1, c_2) = D[\pi,\pi']$$

and

$$D[\pi,\pi'] \leq 1 - c_1 c_2 = 1 - \left(\frac{1 - \Delta[\pi,\pi]}{1 + \Delta[\pi,\pi']}\right)^2$$

$$= \frac{4\Delta[\pi,\pi']}{1 + 2\Delta[\pi,\pi'] + \Delta^2[\pi,\pi']} \leq 4\Delta[\pi,\pi']. \qquad \dagger$$

(12.9) <u>Lemma</u>. Suppose $\pi,\pi' \varepsilon \Pi_N$. Then $d \ \varepsilon [0, 1]$ satisfies $D[\pi,\pi'] \leq d$ if $\exists\ \hat{\pi}, \hat{\pi}' \varepsilon \Pi_N$ such that:

$$\pi' = (1-d)\pi + d\hat{\pi}$$

$$\pi = (1-d)\pi' + d\hat{\pi}'$$

<u>Proof</u>: If $d = 0$, the proof is trivial. Assume $d > 0$ and let $\hat{\pi} = [\pi' - (1-d)\pi]/d$, $\hat{\pi}' = [\pi - (1-d)\pi']/d$. Clearly $|\hat{\pi}| = |\hat{\pi}'| = 1$. But $d \geq D[\pi,\pi'] \iff 1-d \leq (\pi'_i/\pi_i)$, $\forall i \varepsilon S \iff \hat{\pi} \geq 0$ and similarly $\hat{\pi}' \geq 0$.

Thus $\hat{\pi},\hat{\pi}'\varepsilon\Pi_N \iff d \geq D[\pi,\pi']$. †

(12.10) <u>Corollary</u>. Let $\pi = \Sigma_{i=1}^{\infty} \lambda_i\pi(i)$, $\pi' = \Sigma_{i=1}^{\infty} \lambda_i'\pi'(i)$, λ_i, $\lambda_i' \geq 0$, $\pi(i)$, $\pi'(i)\varepsilon\Pi_N$ and $\Sigma_{i=1}^{\infty} \lambda_i = \Sigma_{i=1}^{\infty} \lambda_i' = 1$.

Then:

$$D[\pi,\pi'] \leq \sup_{i,i'} D[\pi(i), \pi'(i)]$$

<u>Proof</u>: Let $d = \sup_{i,i'}$ $D[\pi(i), \pi'(i)]$ and construct $\hat{\pi}(i,j)$, $\hat{\pi}'(i,j)\varepsilon\Pi_N$ as in (12.9) so that:

$$\pi'(j) = (1-d)\pi(i) + d\hat{\pi}(i,j),$$

$$\pi(i) = (1-d)\pi'(j) + d\hat{\pi}'(i,j).$$

Then $\hat{\pi} = \Sigma_{i=1}^{\infty} \Sigma_{j=1}^{\infty} \lambda_i\lambda'_j\hat{\pi}(i,j)$ and $\hat{\pi}' = \Sigma_{i=1}^{\infty} \Sigma_{j=1}^{\infty} \lambda_i\lambda_j'\hat{\pi}'(i,j)$ satisfy:

$$\pi' = (1-d)\pi + d\hat{\pi}$$

$$\pi = (1-d)\pi' + d\hat{\pi}'$$

and, by (12.9), $D[\pi,\pi'] \leq d$. †

d. <u>Continuity of Convex Functions</u>

(12.11) <u>Definition</u>. (a) V is the vector space of bounded real-valued continuous functions on Π_N.

(b) $||\cdot||$ is the "sup norm,"

$$||v|| = \sup_{\pi\varepsilon\Pi_N} |v(\pi)|.$$

(12.12) Definition. For any $v \varepsilon V$, $\hat{v} \varepsilon V$ denotes the White projection of v, given by

$$\hat{v}(\pi) = v(\pi) - v(e^N)$$

Remark: This projection generalizes a normalizing operation devised by D.J. White [1963], for value functions having finite domain, to avoid divergence in value iteration.

(12.13) Definition.

$$\| v \|_D = [\sup_{\pi \varepsilon \Pi_N} v(\pi)] - [\inf_{\pi \varepsilon \Pi_N} v(\pi)]$$

Interpretation: $\| \cdot \|_D$ is a norm on the subset $\hat{V}$ of V, where $\hat{V} = \{\hat{v} : v \varepsilon V\} = \{v \varepsilon V : v(e^N) = 0\}$.

(12.14) Lemma. $\| \hat{v} \| \leq \| v \|_D = \| \hat{v} \|_D \leq 2 \| v \|$.

(12.15) Theorem. If $v \varepsilon V$ is convex, then

$$|v(\pi) - v(\pi')| \leq D[\pi,\pi'] \; \| v \|_D , \quad \forall \pi,\pi' \varepsilon \Pi_N .$$

Proof: Assume without loss of generality that $v(\pi) \geq v(\pi')$. Following (12.9), construct $\hat{\pi}'$ so that $\pi = (1 - D[\pi,\pi'])\pi' + D[\pi,\pi']\hat{\pi}'$. Then $v(\pi) - v(\pi') \leq (1 - D[\pi,\pi'])v(\pi') + D[\pi,\pi']v(\hat{\pi}') - v(\pi') = D[\pi,\pi']$ $[v(\hat{\pi}') - v(\pi')] \leq D[\pi,\pi'] \; \| v \|_D$. †

(12.16) Theorem. For every convex function $v \varepsilon V$, there is a quantity $\| v \|_\Delta \leq 4 \| v \|_D$ such that

$$|v(\pi) - v(\pi')| \leq \Delta[\pi,\pi'] \; \| v \|_\Delta \, , \; \forall \pi, \pi' \varepsilon \Pi_N \; .$$

Proof: Trivial, by (12.8) and (12.15).

13. Contraction Properities of T

If P is a stochastic matrix, and

$$\tilde{\alpha}[P] = \max_{i,j\varepsilon S} \delta[\mathrm{row}_i[P], \mathrm{row}_j[P]] < 1,$$

then, for any $\pi,\pi'\varepsilon\Pi_N$,

$$\delta[\pi P, \pi' P] \leq \tilde{\alpha}[P]\ \delta[\pi,\pi'] ,$$

i.e. the transformation $f[\pi] = \pi P$ is a contraction mapping in Π_N . One consequence of this property is that $\{\pi P^k\}$ approaches a unique limit as $k\to\infty$; this is, of course, the vector of steady-state probabilities for a Markov chain having transition probability matrix P.

This section generalizes the concept of contractions in state probability vectors to the information vector transition function T [defined by (2.8) and (7.7)].

(13.1) Definition. An NxN substochastic matrix P is said to be subrectangular if, for every $i,j,i',j'\varepsilon S$,

$$P_{ij} > 0 \text{ and } P_{i'j'} > 0 .$$

$$\Longrightarrow \quad P_{ij'} > 0 \text{ and } P_{i'j} > 0$$

(13.2) Definition. If P is a substochastic matrix and $P \neq 0$, then

(a) $\alpha[P] = \max\{\Delta[\text{row}_i[P]/(\text{row}_i[P]1), \text{row}_j[P]/(\text{row}_j[P]1)] :$

$\text{row}_i[P] \neq 0, \text{row}_j[P] \neq 0\}.$

Also $\alpha[\underline{z}]$ denotes $\alpha[P(\underline{z})]$.

(b) $a[P] = \max\{D[\text{row}_i[P]/(\text{row}_i[P]1), \text{row}_j[P]/\text{row}_j[P]1)] :$

$\text{row}_i[P] \neq 0, \text{row}_j[P] \neq 0\}.$

Also $a[\underline{z}]$ denotes $a[P(\underline{z})]$.

Remark: The evaluation of $\alpha[P]$ or $a[P]$ by enumeration requires N^3 operations. This is comparable to the effort expended in multiplying two NxN matrices.

(13.3) Proposition. (a) $0 \leq \alpha[P] \leq 1$ and $0 \leq a[P] \leq 1$ for all substochastic matrices $P \neq 0$.

(b) $\alpha[P] < 1 \Longleftrightarrow a[P] < 1 \Longleftrightarrow P$ is subrectangular.

(c) $\alpha[P] = 0 \Longleftrightarrow a[P] = 0 \Longleftrightarrow P$ has rank 1.

The following lemma states a well-known property of the Hajnal measure.

(13.4) Lemma. If $w \varepsilon R_N$, and $\pi, \pi' \varepsilon \Pi_N$, then

$$|\pi w - \pi' w| \leq \delta[\pi,\pi']\{[\max_{i\varepsilon S} w_i] - [\min_{i\varepsilon S} w_i]\}.$$

Proof: Assume without loss of generality that $\pi w - \pi' w \geq 0$. Now

$$\begin{aligned}\pi w - \pi' w &= \Sigma_{i\varepsilon S}(\pi_i - \pi'_i)w_i \\ &= \Sigma_{i\varepsilon S}(\pi_i - \pi'_i)^+ w_i + \Sigma_{i\varepsilon S}(\pi_i - \pi'_i)^- w_i \\ &\leq \Sigma_{i\varepsilon S}(\pi_i - \pi'_i)^+ [\max_{i\varepsilon S} w_i] + \Sigma_{i\varepsilon S}(\pi_i - \pi'_i)^- \\ &\quad [\min_{i\varepsilon S} w_i] \\ &= \delta[\pi,\pi'][\max_{i\varepsilon S} w_i] - \delta[\pi,\pi'][\min_{i\varepsilon S} w_i] . \quad \dagger\end{aligned}$$

Remark: (13.4) may be viewed as a stronger version of (12.15), where v is constrained to be linear.

Using (13.4), it is possible to demonstrate (13.5).

(13.5) Theorem. (Contraction property of T). If $\eta,\eta'\varepsilon\Pi_N$ and $\underline{z}\varepsilon Z^+(\eta,\eta')$ then

$$\Delta[T(\eta,\underline{z}),\ T(\eta',\underline{z})] \leq \alpha[\underline{z}]\ \Delta[\eta,\eta'] .$$

Proof: Construct row vectors $\{\pi^i\}$ having elements

$$\pi^i_j = \begin{cases} P_{ij}(\underline{z})/\Sigma_{j'\varepsilon S}P_{ij'}(\underline{z}) , & \text{if } i\varepsilon I(z) \\ 0, & \text{otherwise} \end{cases}$$

and define:

$$W = \{w\varepsilon R_N : w \geq 0,\ \eta P(\underline{z})w > 0,\ \eta' P(\underline{z})w > 0\}$$

$$\hat{W} = \{w\varepsilon R_N : w \geq 0,\ \eta w > 0,\ \eta' w > 0\}$$

$$I(\underline{z},w) = \{i : \text{row}_i[P(\underline{z})]w > 0\}$$

Since 1, the N-vector of one's, is an element of each, W and $\hat{W}$ are nonempty. Also, if $\underline{z}\varepsilon Z^+(\eta,\eta')$ as required above, and $w\varepsilon W$, then $I(\underline{z},w)$ is nonempty. Finally $\alpha(\underline{z}) = \max_{i,i'\varepsilon I(\underline{z})}\{\Delta[\pi^i,\pi^{i'}]\}$ by (13.2)(a). Now

$$\Delta[T(\eta,\underline{z}),\ T(\eta',\underline{z})]$$

$$\leq \sup_{w\varepsilon W}\left\{\Sigma_{j\varepsilon S}\left(\frac{\Sigma_{i\varepsilon I(\underline{z},w)}\eta_i P_{ij}(\underline{z})w_j}{\eta P(\underline{z})w} - \frac{\Sigma_{i\varepsilon I(\underline{z},w)}\eta'_i P_{ij}(\underline{z})w_j}{\eta' P(\underline{z})w}\right)^+\right\}$$

$$= \sup_{w\varepsilon W}\max_{J\subset S}\left\{\Sigma_{j\varepsilon J}\left(\frac{\Sigma_{i\varepsilon I(\underline{z},w)}\eta_i P_{ij}(\underline{z})w_j}{\eta P(\underline{z})w} - \frac{\Sigma_{i\varepsilon I(\underline{z},w)}\eta'_i P_{ij}(\underline{z})w_j}{\eta' P(\underline{z})w}\right)\right\}$$

$$= \sup_{w\varepsilon W}\max_{J\subset S}\left\{\Sigma_{i\varepsilon I(\underline{z},w)}\left[\left(\frac{\Sigma_{j\varepsilon J}\eta_i P_{ij}(\underline{z})w_j}{\eta P(\underline{z})w} - \frac{\Sigma_{j\varepsilon J}\eta'_i P_{ij}(\underline{z})w_j}{\eta' P(\underline{z})w}\right)\left(\frac{\Sigma_{j\varepsilon S}P_{ij}(\underline{z})w_j}{\Sigma_{j\varepsilon S}P_{ij}(\underline{z})w_j}\right)\right]\right\}$$

$$= \sup_{w\varepsilon W}\max_{J\subset S}\left\{\Sigma_{i\varepsilon I(\underline{z},w)}\left[\left(\frac{\Sigma_{j\varepsilon S}\eta_i P_{ij}(\underline{z})w_j}{\eta P(\underline{z})w} - \frac{\Sigma_{j\varepsilon S}\eta'_i P_{ij}(\underline{z})w_j}{\eta' P(\underline{z})w}\right)\left(\frac{\Sigma_{j\varepsilon J}P_{ij}(\underline{z})w_j}{\Sigma_{j\varepsilon S}P_{ij}(\underline{z})w_j}\right)\right]\right\}$$

Application of (13.4) now yields

$$\Delta[T(\eta,\ \underline{z}),\ T(\eta',\ \underline{z})]$$

$$\leq \sup_{w\varepsilon W}\max_{J\subset S}\left\{\left[\Sigma_{i\varepsilon I(\underline{z},w)}\left(\frac{\Sigma_{j\varepsilon S}\eta_i P_{ij}(\underline{z})w_j}{\eta P(\underline{z})w} - \frac{\Sigma_{j\varepsilon S}\eta'_i P_{ij}(\underline{z})w_j}{\eta' P(\underline{z})w}\right)^+\right]\right.$$

$$\left.\cdot\left[\max_{i,i'\varepsilon I(\underline{z},w)}\left(\frac{\Sigma_{j\varepsilon J}\ \pi^i_j w_j}{\pi^i w} - \frac{\Sigma_{j\varepsilon J}\ \pi^{i'}_j w_j}{\pi^{i'} w}\right)\right]\right\}$$

$$\leq \left[\sup_{\hat{w}\varepsilon\hat{W}}\left\{\Sigma_{i\varepsilon I(\underline{z})}\left(\frac{\eta_i\hat{w}_i}{\eta\hat{w}} - \frac{\eta'_i\hat{w}_i}{\eta'\hat{w}}\right)^+\right\}\right]$$

$$\cdot\left[\sup_{w\varepsilon W}\max_{i,i'\varepsilon I(\underline{z},w)}\left\{\Sigma_{j\varepsilon S}\left(\frac{\pi^i_j w_j}{\pi^i w} - \frac{\pi^{i'}_j w_j}{\pi^{i'} w}\right)^+\right\}\right]$$

$$\leq\ \Delta[\eta,\eta']\cdot\alpha[\underline{z}],$$

where the last inequality follows from (12.4). †

(13.6) Corollary. $\alpha[\underline{z}\ \underline{z}'] \leq \alpha[\underline{z}]\alpha[\underline{z}']$.

Proof: By (13.2), $\alpha[\underline{z}\ \underline{z}'] = \max_{i,i' \varepsilon I(\underline{z}\ \underline{z}')} \{\Delta[T(e^i, \underline{z}\ \underline{z}'), T(e^{i'}, \underline{z}\ \underline{z}')]\}$

But, following (13.5),

$$\begin{aligned}
&\Delta[T(e^i, \underline{z}\ \underline{z}'), T(e^{i'}, \underline{z}\ \underline{z}')] \\
&= \Delta[T(T(e^i, \underline{z}), \underline{z}'), T(T(e^{i'}, \underline{z}), \underline{z}')] \\
&\leq \alpha[\underline{z}']\Delta[T(e^i, \underline{z}), T(e^{i'}, \underline{z})] \\
&\leq \alpha[\underline{z}']\alpha[\underline{z}]\Delta[e^i, e^{i'}] \\
&= \alpha[\underline{z}']\alpha[\underline{z}]. \qquad \dagger
\end{aligned}$$

The corresponding result for $a[\underline{z}]$ is considerably weaker.

(13.7) Proposition. For $\eta, \eta' \varepsilon \Pi_N$, $\underline{z} \varepsilon Z^+(\eta, \eta')$, $D[T(\pi, \underline{z}), T(\pi', \underline{z})] \leq a[\underline{z}]$.

Proof: $T(\pi, \underline{z}) = \Sigma_{i \varepsilon S}\ \lambda_i T(e^i, \underline{z})$ where $\lambda_i = \dfrac{\pi_i(e^i P(\underline{z})1)}{\pi P(\underline{z})1}$. (12.10)

completes the proof. $\dagger$

Remark: This is not a contraction.

(13.8) Corollary. $a[\underline{z}\ \underline{z}'] \leq a[\underline{z}']$.

14. Detectability

a. Preview

The intuitive notion of detectability was introduced in Section 5d; essentially, a detectable FPS has the property that the information vector is arbitrarily closely approximated on the basis of the memory state alone, if the memory set is sufficiently large. The extent to which an information vector depends on input-output pairs not contained in the memory state is given by $\alpha[\underline{z}^M(k)]$, the contraction induced on the information vector by the input-output pairs contained in the memory state. Recall that by (13.3)(b), $\alpha[\underline{z}^M(k)] < 1$ iff $P(\dot{\underline{z}}^M(k))$ is subrectangular.

Four types of detectability will be defined; these are:

(i) strong subrectangularity (SSR), a condition under which every transition probability matrix is subrectangular.

(ii) weak subrectangularity (WSR), a condition under which every transition has positive probability of generating an input-output pair to which a subrectangular transition probability matrix corresponds.

(iii) strong detectability (SDT), a condition under which there exists a memory set whose essential elements each correspond to subrectangular transition probability matrices.

(iv) weak detectability (WDT), a condition under which the memory state at any given time has positive probability of corresponding to a subrectangular transition probability matrix.

These definitions differ in the type of approximation closeness implied, and in the complexity of procedures which establish this closeness. The following implications are trivially verified:

$$\begin{tikzcd} & \mathrm{WSR} \arrow[dr, Rightarrow] & \\ \mathrm{SSR} \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] & & \mathrm{WDT} \\ & \mathrm{SDT} \arrow[ur, Rightarrow] & \end{tikzcd}$$

Each type of detectability will be investigated in turn. It will be shown, for each, that a finite-memory ε-optimal observer may be constructed, and how the estimation error and memory size interrelate.

b. Strong Subrectangularity

(14.1) Definition. An FPS satisfies the condition of strong subrectangularity (SSR) if $P(z)$ is subrectangular, $\forall z \varepsilon Z$.

(14.2) Definition. For an FPS satisfying SSR, define

$$\alpha = \max_{z \varepsilon Z}\{\alpha[z]\}$$

$$\tau = (-\log\alpha)/(\log \# Z)$$

Remark: The logarithms may be taken to any desired base.

Remark: By (14.1), SSR $\Longrightarrow \alpha<1$.

<u>Remark</u>: The definitions of τ given here and later in this section are consistent with (1.2).

(14.3) <u>Proposition</u>. If an FPS satisfies SSR then, for any $m\varepsilon<0,\infty>$, $k\varepsilon<0,m>$

$$\alpha[\underline{z}(k-m;k)] \leq \alpha^m$$

<u>Proof</u>: By (13.6), $\alpha[\underline{z}(k-m;\ k)] \leq \alpha[\underline{z}(k-m;\ k+1-m)]\ \alpha[\underline{z}(k+1-m;\ k+2-m)] \ldots$

$$\alpha[\underline{z}(k-1;\ k)] \leq \alpha^m \qquad \dagger$$

(14.4) <u>Theorem</u>. Consider an FPS satisfying SSR, along with the memory set $M = \{Z^{m*} \cap Z^+\}$. Let $\hat{\pi} : M \to \Pi_N$ be a mapping satisfying:

$$\left\{ \begin{array}{ll} \hat{\pi}(\underline{z})P(\underline{z}) \neq 0, & \underline{z}\varepsilon Z^m \cap Z^+ \\ \hat{\pi}(\underline{z}) = \pi(0), & \underline{z}\varepsilon Z^{(m-1)^*} \cap Z^+ \end{array} \right\}$$

Define $\tilde{\eta}(\underline{z}) = T(\hat{\pi}(\underline{z}),\underline{z})$. Then

$$\Delta[\eta(k),\ \tilde{\eta}(\underline{z}^M(k))] \leq \alpha^m, \qquad k\varepsilon<0,\infty>$$

<u>Proof</u>: If $k<m$, then $\eta(k) = \tilde{\eta}(\underline{z}^M(k))$. But if $k \geq m$, then $\underline{z}^M(k) = \underline{z}(k-m,\ k))$. But if $k \geq m$, then $\underline{z}^M(k) = \underline{z}(k-m,\ k)$ and, by (14.3) and (13.5),

$$\Delta[\eta^{(k)}, \tilde{\eta}(\underline{z}^M(k))]$$

$$\leq \Delta[T(\eta(k-m),\ \underline{z}(k-m;\ k)),\ T(\hat{\pi}(\underline{z}^M(k)),\ \underline{z}(k-m;\ k))]$$

$$\leq \Delta[\eta(k-m),\ \hat{\pi}(\underline{z}^M(k))]\alpha^m$$

$$\leq \alpha^m \qquad \dagger$$

Interpretation: There is a finite-memory observer requiring no more than $(\#Z)^m$ essential memory states which generates estimates of the information vector lying within α^m of its true value (in a Δ sense; (12.3)(a) determines δ and $|\cdot|$-sense bounds on this error).

Generalization: The approximate relationship between essential memory m and maximum error ε is:

$$\varepsilon \cong m^{-\tau}$$

$$m \cong \varepsilon^{-1/\tau} \tag{14.5}$$

However, the strict bounds are:

$$\varepsilon \leq (m/\#Z)^{-\tau}$$

$$m \leq (\varepsilon/\alpha)^{-1/\tau} \tag{14.6}$$

Specifically, this means that no more than $(\varepsilon/\alpha)^{-1/\tau}$ essential memory states are required to maintain a maximum error less than ε, and that m essential memory states can achieve an error bounded above by $(m/\#Z)^{-\tau}$.

c. Weak Subrectangularity

(14.7) Definition. An FPS satisfies the condition of weak subrectangularity (WSR) if, for every $i\varepsilon S$, $u\varepsilon U$, there is a $y\varepsilon Y$ such that $P(y|u)$ is subrectangular and $e^{i}P(y|u) \neq 0$.

(14.8) Definition. For a FPS satisfying WSR, define

$$\bar{\alpha} = \max_{i\varepsilon S} \max_{u\varepsilon U} \Sigma_{y\varepsilon Y} \Sigma_{j\varepsilon S} P_{ij}(y|u)\alpha[(u,y)]$$

$$\bar{\tau} = (-\log\bar{\alpha})/(\log\#Z)$$

Remark: By (14.7), WSR $\Longrightarrow \bar{\alpha} < 1$.

(14.9) Proposition. If an FPS satisfies WSR, then for any $m\varepsilon<0,\infty>$ $k\varepsilon<0,m>$, $\pi\varepsilon\Pi_N$, and any strategy γ

$$E_{\gamma}\{[\eta(k),T(\pi,\underline{z}(k-m;k))]\} \leq \alpha^{m}.$$

Proof: (By induction) If m=0 the result is trivial. But

$$E_{\gamma}\{\alpha[\underline{z}(k-m;\ k)]\}$$

$$= E_{\gamma}\{\alpha[\underline{z}(k-m;\ k-1)] \cdot E_{\gamma}\{\alpha[\underline{z}(k-1;\ k)]|\underline{z}(k-m;\ k-1)\}\}$$

$$= E_{\gamma}\{\alpha[\underline{z}(k-m,\ k-1)] \cdot E_{\gamma}\{\alpha[\underline{z}(k-1;\ k)]$$

$$|\underline{z}(k-m;\ k-1),\ s(k-1),\ u(k-1)\}$$

$$= E_\gamma\{\alpha[\underline{z}(k-m;\ k-1)] \cdot E_\gamma\{\alpha[\underline{z}(k-1;\ k)]$$

$$\Big|\ \underline{z}(k-m;\ k-1),\ s(k-1),\ u(k-1)\}$$

$$= E_\gamma\{\alpha[\underline{z}(k-m,\ k-1)]$$

$$\cdot \{\Sigma_{y\varepsilon Y}\ \Sigma_{j\varepsilon S}\ P_{s(k-1)j}(y|u(k-1))\alpha[(u(k-1),y)]\}\}$$

$$\leq E_\gamma\{\alpha[\underline{z}(k-m;\ k-1)] \cdot \overline{\alpha}\}$$

$$= \overline{\alpha} \cdot E_\gamma\{\alpha[\underline{z}(k-(m-1);\ k)]\} \qquad \dagger$$

(14.10) <u>Theorem</u>. Consider an FPS satisfying WSR, along with the memory set $M = \{Z^{M*} \cap Z^{+}\}$. Let $\hat{\pi} : M \to \Pi_N$ be a mapping satisfying:

$$\left\{\begin{array}{ll} (\underline{z})P(\underline{z}) \neq 0, & \underline{z}\varepsilon Z^{M} \cap Z^{+} \\ \hat{\pi}(\underline{z}) = \pi(0), & \underline{z}\varepsilon Z^{(m-1)*} \cap Z^{+} \end{array}\right\}$$

Define $\tilde{\eta}(\underline{z}) = T(\hat{\pi}(\underline{z}),\ \underline{z})$, then for any strategy γ,

$$E_\gamma\{\Delta[\eta(k),\ \tilde{\eta}(\underline{z}^m(k))]\} \leq \overline{\alpha}^{\,m}.$$

<u>Proof</u>: If $k<m$, then $\underline{z}^m(k)\varepsilon M^{m-1}$, and $\tilde{\eta}(k) = \eta(k)$. But if $k \geq m$, then $\ell(\underline{z}^m) = m$, and, using (13.6) and (14.3), $E_\gamma\{\Delta[\eta(k),\ \tilde{\eta}(\underline{z}^m(k))]\}$ $= E_\gamma\{\Delta[T(\eta(k-m),\ \underline{z}^m(k)),\ T(\hat{\pi}(\underline{z}^m(k)),\ \underline{z}^m(k))]\} \leq E_\gamma\{\alpha[\underline{z}^m(k)] \leq \overline{\alpha}^{\,m}$.

$\dagger$

<u>Interpretation</u>: There is a finite-memory observer requiring $(\#Z)^m$ essential memory states which generates estimates of the information vector lying <u>on the average</u> within $\bar{\alpha}^m$ of its true value (in a Δ sense).

<u>Generalization</u>: The approximate relationship between essential memory m and mean error $\bar{\varepsilon}$ is:

$$\bar{\varepsilon} \cong m^{-\tau}$$

$$m \cong \bar{\varepsilon}^{-1/\tau} \tag{14.11}$$

However, the strict bounds are

$$\bar{\varepsilon} \leq (m/\#Z)^{-\tau}$$

$$m \leq (\bar{\varepsilon}/\bar{\alpha})^{-1/\tau} \tag{14.12}$$

Specifically, this means that no more than $(\bar{\tau}/\bar{\alpha})^{-1/\tau}$ essential memory states are required to maintain a mean error below $\bar{\varepsilon}$, and that m essential memory states achieve a mean error bounded above by $(m/\#Z)^{-\tau}$.

d. Strong Detectability

(14.13) <u>Definition</u>. An FPS satisfies the condition of <u>strong detectability</u> (SDT) if there exists an integer $\hat{\ell}$ such that $P(\underline{z})$ is subrectangular, $\forall\, \underline{z} \varepsilon Z^{\hat{\ell}} \cap Z^{+}$.

(14.14) Definition. For an FPS satisfying SDT, define

$$\alpha_k = \max_{\underline{z}\varepsilon Z^k Z^+} \{\alpha[\underline{z}]\}$$

$$\hat{\ell} = \min\{k : \alpha_k < 1\}$$

$$\alpha = \alpha_{\hat{\ell}}$$

$$\tau = (-\log \alpha)/(\hat{\ell}\log \#Z)$$

Remark: By (14.13), SDT $\Longrightarrow$ $\alpha<1$.

Remark: If an FPS satisfies SDT, then definitions (14.2) and (14.14) are consistent, since $\hat{\ell}=1$.

(14.15) Proposition. If an FPS satisfies SDT, then for any $m\varepsilon<0,\infty>$, $k\varepsilon<0,m>$,

$$\alpha[\underline{z}(k-m;\ k)] \leq \alpha^{m\div\hat{\ell}}$$

Proof: By (13.7), $\alpha[\underline{z}(k-m;\ k)] \leq \alpha[\underline{z}(k-m;\ k-((m\div\hat{\ell})-1]\hat{\ell})]$

$\cdot\ \alpha[\underline{z}(k-((m\div\hat{\ell})-1)\hat{\ell};\ k-((m\div\hat{\ell})-z)\hat{\ell}] \cdot \ldots \cdot \alpha[\underline{z}(k-\hat{\ell};\ k)] \leq \alpha^{m\div\hat{\ell}}$ †

(14.16) Theorem. Consider an FPS satisfying SDT, along with the of memory set $M = \{Z^{m^*}\cap Z^+\}$. Let $\hat{\pi} : M \to \Pi_N$ be a mapping satisfying:

$$\left\{\begin{array}{ll}\hat{\pi}(\underline{z})P(\underline{z}) \neq 0, & \underline{z}\varepsilon Z^{m}\cap Z^{+} \\ \hat{\pi}(\underline{z}) = \pi(0), & \underline{z}\varepsilon Z^{(m-1)*}\cap Z^{+}\end{array}\right\}$$

Define $\tilde{\eta}(\underline{z}) = T(\hat{\pi}(\underline{z}), \underline{z})$. Then $\Delta[\eta(k), \tilde{\eta}(\underline{z}^{m}(k))] \leq \alpha^{m\div\hat{\ell}}$

Proof: If $k<m$, then $\underline{z}^{m}(k)\varepsilon M^{m-1}$, and $\tilde{\eta}(k) = \eta(k)$. But if $k \geq m$, then $\ell(\underline{z}^{m}(k)) = m$, and, using (13.6) and (14.15), $\Delta[\eta(k), \tilde{\eta}(k))]$ $= \Delta[T(\eta(k-m), \underline{z}^{m}(k)), T(\hat{\pi}(\underline{z}^{m}(k)), \underline{z}^{m}(k))] \leq \alpha[\underline{z}^{m}(k)] \leq \alpha^{\eta\div\hat{\ell}}$ †

Interpretation: There is a finite-memory observer, requiring no more than $(\#Z)^{m}$ essential memory states which generates estimates of the information vector lying within $\alpha^{m\div\hat{\ell}}$ of its true value (in a Δ sense).

Generalization: The approximate relationship between essential memory m and maximum error ε is

$$\varepsilon \cong m^{-\tau}$$

$$m \cong \varepsilon^{-1/\tau} \tag{14.17}$$

However, the strict bounds are :

$$\varepsilon \leq (m/(\#Z)^{\ell})^{-\tau}$$

$$m \leq (\varepsilon/\alpha)^{-1/\tau} \tag{14.18}$$

Specifically, this means that no more than $(\varepsilon/\alpha)^{-1/\tau}$ essential memory states are required to maintain a maximum error below ε, and that m essential memory states can achieve a maximum error bounded above by $(m/(\#Z)^{\ell})^{-\tau}$.

e. Weak Detectability

(14.19) Definition. If k is an integer and $\phi : Z^{k*} \to U$, then for any $\underline{z} = (u_1,y_1)(u_2,y_2)\ \ldots\ (u_k,y_k)\varepsilon Z^k$ define:

$$\sigma[\underline{z},\phi] = \begin{cases} 1, & \text{if } u_{j+1} = \phi[u_1,y_1)\ \ldots\ (u_j,y_j)], \quad j\varepsilon\langle 0,k-1\rangle \\ 0 & \text{otherwise} \end{cases}$$

Interpretation: $\sigma[\underline{z},\phi] = 1$ if $\underline{z}(0;k) = \underline{z}$ can evolve when inputs are selected according to the rule $u(k) = \phi[\underline{z}(0;k)]$. Thus, if $\pi(0)$ is the initial state probability vector, and inputs are selected according ϕ, then the probability distribution for random variable $\underline{z}(0;k)$ is:

$$\text{Prob}\{\underline{z}(0;\ k) = \underline{z}\} = \sigma[\underline{z},\ \phi](\pi(0)P(\underline{z})1)$$

(14.20) Definition.

$$\alpha_k = \max_{i\varepsilon S} \max_{\phi\varepsilon U^{(Z^{k*})}} \left[\Sigma_{\underline{z}\varepsilon Z^k}\ \sigma[\underline{z},\phi](e^i P(\underline{z})1)\alpha[\underline{z}]\right]$$

$$\bar{a}_k = \max_{i\varepsilon S} \max_{\phi\varepsilon U(Z^{k*})} \left[\sum_{\underline{z}\varepsilon Z^k} \sigma[\underline{z}, \phi](e^i P(\underline{z})1) a[z] \right]$$

Interpretation: $\bar{\alpha}_\ell$ is the largest possible value of $E_\gamma\{\alpha[\underline{z}(k-\ell;\ k)]\}$ where γ is a feasible strategy. $\bar{a}_\ell$ likewise is the expectation of $a[\underline{z}(k-\ell;\ k)]$.

(14.21) Definition. An FPS satisfies the condition of weak detectability (WDT) if there exists an integer $\bar{\ell}$ such that $\bar{\alpha}_{\bar{\ell}} < 1$.

(14.22) Definition. For an FPS satisfying WDT, define

- $\bar{\ell} = \min\{\ell \cdot \alpha_\ell < 1\}$
- $\bar{\alpha} = \bar{\alpha}_{\bar{\ell}}$
- $\bar{a} = \bar{a}_{\bar{\ell}}$
- $\tau = (-\log \bar{\alpha})/(\bar{\ell} \log \#Z)$

Remark: By (14.21), $\bar{\alpha} < 1$.

Remark: If an FPS satisfies WSR, then definitions (14.8) and (14.22) are consistent. If an FPS satisfies SDT then $\bar{\ell} \leq \hat{\ell}$ and if $\bar{\ell} = \hat{\ell}$ then $\bar{\alpha} \leq \alpha$.

(14.23) <u>Proposition</u>. If an FPS satisfies WDT, then for any $i\varepsilon<0,\infty>$, $k\varepsilon<0,m>$, $\pi\varepsilon\Pi_N$, and any feasible strategy γ,

$$E_\gamma\{\alpha[\underline{z}(k-m;\ k)]\} \leq \overline{\alpha}^{\overline{m\div\ell}}$$

<u>Proof</u>: Consider a transformed system in which the input is a mapping $\phi_k : Z^{(\ell-1)} \to U$, specified at intervals of $\overline{\ell}$ time units, each of which describes $u(k), u(k+1), \ldots, u(k+\overline{\ell}-1)$ as functions of $\underline{e}$, $\underline{z}(k;\ k+1)$, $\underline{z}(k, k+2)$, ... $\underline{z}(k,\ k+\overline{\ell}-1)$ respectively. The output at time k is $\underline{z}(k-\overline{\ell};\ k)$. This transformed system satisfies WSR; the desired result follows from (14.9). †

(14.24) <u>Theorem</u>. Consider an FPS satisfying WDT along with the memory set $M = \{Z^{m*}\ Z^+\}$. Let $\hat{\pi} : M \to \Pi_N$ be a mapping satisfying:

$$\begin{Bmatrix} \hat{\pi}(\underline{z})P(\underline{z}) \neq 0, & \underline{z}\varepsilon Z^m \cap Z^+ \\ \hat{\pi}(\underline{z}) = \pi(0), & \underline{z}\varepsilon Z^{(m-1)*} \cap Z^+ \end{Bmatrix}$$

Define $\tilde{\eta}(\underline{z}) = T(\hat{\pi}(\underline{z}), \underline{z})$. Then, for any feasible strategy γ,

$$E_\gamma\{\Delta[\eta(k),\ \tilde{\eta}(\underline{z}^M(k))]\} \leq \overline{\alpha}^{\overline{m\div\ell}}$$

<u>Proof</u>: If $k<m$, then $\underline{z}^m(k)\varepsilon Z^{(m-1)*}$ and $\tilde{\eta}(k) = \tilde{\eta}(\underline{z}^M(k))$. But if $k \geq m$, then $\underline{z}^m(k)\varepsilon Z^m$ and using (13.6) and (14.23),

$$E_\gamma\{\Delta[\eta(k),\ \tilde{\eta}(\underline{z}^m(k))]\} = E_\gamma\{\Delta[T(\eta(k-m),\ \underline{z}^m(k)),\ T(\hat{\pi}(\underline{z}^m(k)),\ \underline{z}^m(k)]\}$$

$$\leq E_\gamma\{\alpha[z^m(k)]\} \leq \overline{\alpha}^{m\div\overline{\ell}} \qquad \dagger$$

Interpretation: There is a finite-memory observer, requiring at most $(\#Z)^m$ essential memory states, which generates estimates of the information vector lying on the average within $\overline{\alpha}^{m\div\overline{\ell}}$ of its true value (in a Δ sense).

Generalizations: The approximate relationship between essential memory m and mean error $\overline{\varepsilon}$ is:

$$\overline{\varepsilon} \cong m^{-\tau}$$

$$m \cong \overline{\varepsilon}^{\,-1/\tau} \qquad (14.25)$$

However, the strict bounds are:

$$\overline{\varepsilon} \leq (m/\#Z^{\overline{\ell}})^{-\tau}$$

$$m \leq (\overline{\varepsilon}/\overline{\alpha})^{-1/\tau} \qquad (14.26)$$

Specifically, this means that no more than $(\overline{\varepsilon}/\overline{\alpha})^{-1/\tau}$ essential memory states are required to maintain a mean error below $\overline{\varepsilon}$, and that m essential memory states can achieve a mean error bounded above by $(m/\#Z^{\overline{\ell}})^{-\tau}$.

15. Decomposition of a Free FPS into Detectable Parts

This section is concerned with FPS's that are not detectable. An example of such a system was given in Section 5a. An FPS fails to be detectable when some function of the (internal or augmented) state may be recursively updated, but is never identified exactly. This function depends on the input process, and for this reason, the decomposition of an FPS into detectable parts is meaningful only in the case of a free FPS.

(15.1) Definition. (a) $C_i(k) = \{j : P_{ij}(\underline{z}(0;k)) > 0\} \subseteq S$

(b) $C(k) = \{C_i(k) : i\varepsilon S\} - \{\emptyset\}$

(c) $\mu(k) = \#C(k)$.

Interpretation. $C_i(k)$ is the set of possible present internal states given that $s(0)=i$. $C(k)$ is the set of possible state configurations which may result from specification of the initial state. In a detectable system, $\mu(k) \to 1$.

(15.2) Proposition. (a) $C_{i'}(k+1) = \{j : P_{ij}(y(k+1)|\, u(k))>0,$

$i\varepsilon C_{i'}(k)\}$

(b) $C(k+1) = \{\{j : P_{ij}(y(k+1)|u(k))>0,\ i\varepsilon C',$

$C'\varepsilon C(k)\} - \{\emptyset\}$

(c) $\mu(k+1) \leq \mu(k)$.

Consider a free connected FPS, i.e. one whose underlying process has an entirely recurrent state set. If pairs $[C(k), s(k)]$ are considered in place of the internal state, recurrent chains of such pairs may be determined. By (15.2)(c), $\mu(k)$ is constant within each recurrent chain. If every recurrent chain is such that $\mu(k)=1$, then the system satisfies WDT, because if $C(k)$ is at any time reset to $\{\{i\} : \eta_i(k) > 0\}$, it will tend to a value containing one element, indicating that the word of intervening input-output pairs had a subrectangular transition probability matrix. On the other hand, if $\mu(k)$ remains greater than one for all time, then subrectangular input-output words cannot occur.

If the free connected FPS is such that $\mu(k)$ need not tend to one, then the process can be described as one of at most N detectable models, which may be asymptotically identified. This decomposition is effected by allowing $\mu(k)$ to reach its minimal value, and by then assuming that the current state lies in a particular element of $C(k)$. This determines the element of $C(k)$ containing the current state at all times, and the likelihood of a particular model can be updated periodically. Since only one model is correct, its likelihood will approach one - unless some models are identical, in which case it doesn't matter which is identified.

Note that, in order to determine whether a free connected FPS is detectable, one determines whether the process $\{\{i : \eta_i(k)>0\}\}$ (which equals C(k) if $\mu(k)=1$) is simply connected in 2^S. This illustrates a duality between the notions of connectivity and detectability.

The decomposition procedure is readily extended to general free FPS's. Transient states may be ignored since information vector entries corresponding to transient states have expectation that vanishes geometrically as the number of available (most recent) input-output pairs increases, and contemplation of an infinite past eliminates transient states at time zero. If the free FPS has more than one recurrent class, then the test for detectability is performed on the system restricted to one recurrent class at a time; certain recurrent classes may be identified exactly on the basis of a particular output configuration (that eventually occurs); others may be identified on the basis of the infinite past; still others may be identical from an input-output point of view.

Since the decomposition depends crucially on a classification of states as transient or recurrent, it cannot be extended to FPS's with inputs; in practical applications, though, it often suffices to consider the free system under a particular adapted strategy.

16. Stochastic Realization of a Free FPS

The stochastic realization problem includes that of deciding whether or not a given free FPS is equivalent to a state-calculable one. Such a property would be desirable because it would indicate that after a sufficiently long initial identification procedure, the present state could be arbitrarily closely known, and the optimal strategy in the steady-state could be computed by assuming that the internal state was known exactly. This property would be equivalent to the following condition: $\{\eta(k)\}$ has a finite number of cluster points in Π_N with probability one. It will be suggested here that such is generally not the case.

(16.1) Theorem. For a given free, connected, strongly subrectangular, FPS in minimal state form (Paz [1971]), the following statements are equivalent:

(a) The FPS is equivalent to one that is state calculable.

(b) The process $\{\underline{z}(k-N(N-1)/2;\, k)\}$ is a Markov chain.

Proof: Assume first that every matrix of the form $P(\underline{z})$, $\underline{z} \varepsilon Z^{N(N-1)/2}$ has rank zero or rank one. Then (a) and (b) trivially follow.

Now assume that there is a $\underline{z} \varepsilon Z^{N(N-1)/2}$ such that $P(\underline{z})$ has rank greater then one. Then there is a $\hat{\underline{z}}\ \varepsilon Z^{+}$ and $i, j \varepsilon S$ such that

$\underline{z} = \underline{z}'\hat{\underline{z}}\ \underline{z}''$

$P_{ii}(\hat{\underline{z}}) > 0$

$P_{jj}(\hat{\underline{z}}) > 0$

and, naturally, $P(\hat{z})$ has rank greater than one, and it is subrectangular (by SSR). By Perron's theorem, $P(\hat{\underline{z}})$ has a left eigenvector $\hat{\pi}$ corresponding to the eigenvalue of largest magnitude, and satisfying $\hat{\pi}_j > 0$, $j \varepsilon J(\hat{\underline{z}})$. Consider the set $\{T(\pi, (\hat{z})^k) : k\varepsilon\langle 1,\infty\rangle\}$. Clearly this set either contains exactly one element or else it consists of an infinite number of distinct elements. Using the word $\underline{z}$ selected above, define $\hat{\eta}(\underline{z}) = T(\hat{\pi}, \underline{z}'')$. For any $\underline{z}\varepsilon Z^{N(N-1)/2}$ such that $P(\underline{z})$ has rank one, define $\hat{\eta}(\underline{z}) = T(e^i,\underline{z})$ for any $i\varepsilon I(\underline{z})$.

Now, if it is true that, for any $\underline{z}^1$, $\underline{z}^2 \varepsilon Z^{(N(N-1)/2)} \cap Z^+$,

$$T(\hat{\eta}(\underline{z}^1), \underline{z}^2) = \hat{\eta}(\underline{z}^2)$$

then (a) and (b) follow trivially. On the other hand if

$$T(\hat{\eta}/(\underline{z}^1), \underline{z}^2) \neq \hat{\eta}(\underline{z}^2)$$

for some $\underline{z}^1$, $z^2 \varepsilon Z^{(N(N-1)/1)} \cap Z^+$, then an infinite number of distinct possible information vector values exist (by decomposing $\underline{z}^2$ in the manner described above), and (a) and (b) are both false. †

An algorithm based on the proof of (16.1) decides whether or not a free FPS is equivalent to one that is state-calculable. A similar algorithm will perform the same test for an arbitrary FPS. The FPS is first decomposed into connected detectable components, following the analysis in Section 15. The possible information vector values are then enumerated. However, whenever an information vector value results from a transition having subrectangular probability matrix of rank greater than one, this information vector must coincide with the Perron eigenvector for that transition probability matrix. Since the enumerations are performed on extremely large sets, this decision algorithm is computationally infeasible in all but the trivial cases. At the same time, it should be clear that in very few cases will the FPS actually be equivalent to a state-calculable system.

A more practical approach to stochastic realization is to *approximate* the FPS by a system whose state is the memory state induced by a large memory set. This FPS is state-calculable because memory states may be recursively computed, and the closeness of the approximation may be established by detectability arguments.

CHAPTER III

STRUCTURE OF OPTIMAL CONTROLLERS

17. Finite-horizon Problems

The finite-horizon partially-observable Markov decision problem was solved by Sondik [1971]. His results are reviewed here, in slightly modified form.

Sondik showed that every finite-horizon problem has an optimal finite-memory solution. This may be demonstrated in a number of ways. One of these is to argue that the information vector assumes values of the form $\{T(\pi(0), \underline{z}) : \underline{z} \varepsilon Z^{+}(\pi(0)) \cap Z^{k^*}\}$. Since this is a finite set, the problem may be restated as a finite-horizon Markov decision problem with perfect state observation, where the memory state $\underline{z}(0; k)$ is regarded as the state variable. The optimal policy will then determine the input on the basis of this memory state. A dual argument states that at any time k, the remaining strategy (given the present time and information state) can be expressed as a policy $\phi_{[k,\eta(k)]} : Z^{(k-k)*} \rightarrow U$. Since there are only a finite number of these, the optimal input may be computed by enumeration. A computational procedure which is based on the latter argument is now described.

Consider a modification of the finite-horizon FPS control problem in which the information vector is regarded as a perfectly-observed state variable. The expected incremental reward at time k takes the form:

$$E\{r(k) \mid \eta(k), u(k)\} = \eta(k)q(u(k)) \tag{17.1}$$

The problem, consequently, is to maximize the performance index $E\{\Sigma_{k=0}^{k} b(k)\ \eta(k)q(u(k))\}$. Application of Bellman's Principle of Optimality yields

$$v^{k-1,K}[\pi] = \max_{u\epsilon U}\{b(k)\pi q(u) + \Sigma_{y\epsilon Y}(\pi P(y|u)1)v^{k,K}[T(\pi,u,y)]$$

$$v^{k,K}(\pi) = 0 \tag{17.2}$$

where $v^{k,K}$ is a real-valued function on Π_N representing the value of being in a particular information state at time k for a problem with horizon K. For ease of notation, extend the domain of $v^{k,K}$ to $\tilde{\Pi}_N$ by defining

$$v^{k,K}[\tilde{\pi}] = \begin{Bmatrix} (\tilde{\pi}1)v^{k,K}[\tilde{\pi}/(\tilde{\pi}1)], & \text{if } \tilde{\pi} \neq 0 \\ 0, & \text{if } \tilde{\pi} = 0 \end{Bmatrix} \tag{17.3}$$

Then (17.2) becomes

$$v^{k-1,K}[\pi] = \max_{u\epsilon U}\{b(k)\pi q(u) + \Sigma_{y\epsilon Y}v^{k,K}[\pi P(y|u)]\}$$

$$v^{K,K}[\pi] = 0 \tag{17.4}$$

Now define finite subsets of R_N:

$$W^{k-1,K} = \{q(u) + \Sigma_{y\varepsilon Y}\ P(y|u)w_y\ :\ u\varepsilon U,\ w_y\varepsilon W^{k,K},\ \ y\varepsilon Y\}\} \tag{17.5}$$

$$W^{K,K} = \{0\}$$

Eq. (17.4) may now be expressed as

$$v^{k,K}[\pi] = \max\{\pi w\ :\ w\varepsilon W^{k,K}\} \tag{17.6}$$

Thus, each function $v^{k,K}$ is convex and piecewise-linear with a finite number of faces. Each region of Π_N throughout which $v^{k,K}$ is linear, is a region where the strategy-to-go is constant; thus the elements of $W^{k,K}$ may be viewed as controller states. Specifically, if $v^{k,K}[\pi] = \pi\hat{w}$ and $\hat{w} = q(\hat{u}) + \Sigma_{y\varepsilon Y}\ P(y|\hat{u})\hat{w}_y$, then an optimal controller faced with information vector π at time k selects input $\hat{u}$ and is assured that $v^{k+1,K}[\eta(k+1)] = \eta(k+1)\hat{w}_{y(k+1)}$.

The size of each set $W^{k,K}$ may be reduced by eliminating elements that correspond to memory states which can never be reached. Specifically, if $w\varepsilon W^{k,K}$ is such that $\min_{\pi\varepsilon\Pi_N}\{v^{k,K}[\pi] - w\pi\} > 0$, then w can be eliminated from $W^{k,K}$ without loss of generality. This test is effected through the solution of a simple linear program.

Of course, this solution procedure is not necessarily applicable to infinite-horizon problems, because the size of $W^{0,K}$ can increase without bound as $K\to\infty$. Drake [1962, 1968] and Sondik [1971] have noted that, in certain problems, $W^{0,K}$ converges (except for a constant gain) in a finite number of iterations; a finite-memory realization of the infinite-horizon optimal controller is thus obtained. Although it is true, in the infinite-horizon problem, that existence of a finite-memory realization of the optimal controller implies that the value function is piecewise-linear with a finite number of faces, this does not in turn imply that the number of faces in the approximations $v^{0,K}$ is bounded. Thus $\#W^{0,K}$ may diverge as $K \to \infty$, although, in the limit, a piecewise-linear relative value function with a finite number of faces is approached. Furthermore, many of the faces in $W^{0,K}$ may correspond to transient memory states. In the Machine Maintenance and Repair Problem, the optimal value function is characterized by well over thirty faces, only eight of which are required to realize an optimal controller.

18. State-Observable Problems

A state-observable FPS is, of course, equivalent to a Markov decision process. This section reviews known methods for its solution; additional references are given in Section 4. Since y(k) uniquely determines s(k), $P_{ij}(u)$ will denote $\Sigma_{y\varepsilon Y}\, P_{ij}(y|u)$.

The finite-horizon problem is solved by computing value functions:

$$
\begin{aligned}
v^{k-1,K}(i) &= \max_{u\varepsilon U}\{b(k)q_i(u) + \Sigma_{j\varepsilon S}\, P_{ij}(u)v^{k,K}(j)\} \\
v^{K,K}(i) &= 0
\end{aligned}
\qquad (18.1)
$$

The optimal decision at time k-1, for a system in state i, is the input u which maximizes (18.1). Thus the optimal strategy selects inputs on the basis of current state and time alone.

If $b(k) = \beta^k$, then $v^{k,K} = \beta^k v_0^{K-k}$ where

$$
\begin{aligned}
v_0^m(i) &= \max_{u\varepsilon U}\{q_i(u) + \beta\Sigma_{j\varepsilon S}\, P_{ij}(u)v_0^{m-1}(j)\} \\
v_0^0(i) &= 0
\end{aligned}
\qquad (18.2)
$$

As $m \to \infty$, v_0^m approaches a limit v* satisfying:

$$
v^*(i) = \max_{u\varepsilon U}\{q_i(u) + \beta\Sigma_{j\varepsilon S}\, P_{ij}(u)v^*(j)\} \qquad (18.3)
$$

Thus the optimal strategy in the infinite-horizon discounted problem determines inputs on the basis of the present state alone.

Eq. (18.3) can be solved by computing the sequence $\{v_0^m\}$ according to (18.2). This computational procedure is called <u>value iteration</u>.

If β is large (i.e. near unity), then computational instability may occur. This difficulty is avoided by defining:

$$g = (1-\beta)v^*(N)$$
$$\hat{v}^*(i) = v^*(i) - v^*(N) \tag{18.4}$$

Eq. (18.3) now becomes

$$\hat{v}^*(i) = \max_{u\varepsilon U} \{q_i(u) + \beta\Sigma_{j\varepsilon S}\, P_{ij}(u)\hat{v}^*(j)\} - g \tag{18.5}$$
$$\hat{v}^*(N) = 0$$

The function $\hat{v}^*$ is called a <u>relative value function</u>, and g^* is called the <u>average gain</u>. This follows from the decomposition:

$$v^*(i) = \hat{v}^*(i) + \frac{g}{1-\beta} = \hat{v}^*(i) + \Sigma_{k=0}^{\infty}\beta^k g$$

Eq.(18.5) might be solved by <u>White's algorithm</u>

$$\tilde{v}_0^{m-1}(i) = \max_{u\varepsilon U}\{q_i(u) + \beta\Sigma_{j\varepsilon S}\, P_{ij}(u)\hat{v}_0^{m-1}(j)\}$$
$$\hat{v}_0^m(i) = \tilde{v}^{m-1}(i) - \tilde{v}^{m-1}(N) \tag{18.6}$$
$$\hat{v}_0^0(i) = 0$$

On the basis of $\tilde{v}^m$ and $\hat{v}^m$, <u>MacQueen bounds</u> on g may be computed:

$$\min_{i\varepsilon S} [\tilde{v}_0^m(i) - \hat{v}_0^m(i)] \leq g \leq \max_{i\varepsilon S}[\tilde{v}_0^m(i) - \hat{v}_0^m(i)] \qquad (18.7)$$

Eq (18.5) may also be regarded as a <u>linear program</u>:

$$\begin{aligned} &\text{min:} && g \\ &\text{subject to:} && \hat{v}^*(i) \geq q_i(u) + \beta\Sigma_{j\varepsilon S}\, P_{ij}(u)\hat{v}^*(j) - g, \quad i\varepsilon S,\ u\varepsilon U \\ & && \hat{v}^*(N) = 0 \end{aligned} \qquad (18.8)$$

As it turns out, an optimal basic solution will satisfy (18.8) with <u>strict equality for exactly one input corresponding to each state</u>. Thus, an optimal policy is obtained.

Now consider the infinite-horizon <u>undiscounted</u> problem. When $\beta=1$, $\{\hat{v}_0^m\}$ is not guaranteed to remain bounded; and even if it remains bounded, it is not guaranteed to converge. Boundedness occurs if the average gain does not depend on the initial state, and convergence occurs if the optimal system if aperiodic.

Assume first that $\{\hat{v}_0^m\}$ is bounded. Then difficulties relating to convergence are avoided by defining the problem as a limit of discounted problems as $\beta\uparrow 1$. Thus a solution to the linear program

$$\begin{aligned} &\text{min:} && g \\ &\text{subject to:} && \hat{v}^*(i) \geq q_i(u) + \Sigma_{j\varepsilon S}\, P_{ij}(u)\hat{v}^*(j) - g, \\ & && \qquad\qquad i\varepsilon S,\ u\varepsilon U \\ & && \hat{v}^*(N) = 0 \end{aligned} \qquad (18.9)$$

is sought. Computationally, convergence is assured by Schweitzer's (damped value-iteration) algorithm

$$\tilde{v}^{m-1}(i) = \max_{u\varepsilon U}\{q_i(u) + \Sigma_{j\varepsilon S}\, P_{ij}(u)\hat{v}_0^{m-1}(j)\}$$

$$\hat{v}^{m}(i) = \tilde{\beta}[\tilde{v}^{m-1}(i) - \tilde{v}^{m-1}(N)] + (1-\tilde{\beta})\hat{v}^{m-1}(i) \qquad (18.10)$$

$$\hat{v}^{0}(i) = 0\,, \qquad 0 < \tilde{\beta} < 1$$

Odoni bounds on g may be computed

$$\min_{i\varepsilon S}[\tilde{v}^{m}(i) - \hat{v}^{m}(i)] \le g \le \max_{i\varepsilon S}[\tilde{v}^{m}(i) - \hat{v}^{m}(i)] \qquad (18.11)$$

Simple connectivity is a sufficient condition for $\{\hat{v}^m\}$ to be bounded. A general Markov decision problem may be solved by decomposing it into simply connected subproblems, as described below:

(18.12) Algorithm (Solution of a Markov decision problem)

Step 1. Let $\tilde{S}$ denote the "remaining region of S" and set $\tilde{S}=S$. Let $\tilde{U}(i)$ denote the admissible input set when the system is known to be in state i, and set $\tilde{U}(i) = U$, $i\varepsilon S$. Also set $\tilde{g}(i) = Q_{min}$, $i\varepsilon S$.

Step 2. Determine a connected class C in $(\tilde{S},\tilde{U})$, the Markov decision process with state set restricted to $\tilde{S}$ and input set restricted to $\tilde{U}(i)$ when the system is in state i. Since $\tilde{S}$ is nonempty, such a connected class exists.

Step 3. Solve the Markov decision problem within $(C,\tilde{U})$ to obtain a gain g. Set $\tilde{g}(i) = g$, $\forall i\varepsilon C$.

<u>Step 4</u>. Set $\tilde{S} = \tilde{S}-C$. For every triplet $i\varepsilon\tilde{S}$, $u\varepsilon\tilde{U}(i)$, $j\varepsilon S-\tilde{S}$, that satisfies $P_{ij}(u) > 0$, set $\tilde{U}(i) = \tilde{U}(i)-\{u\}$. If $\tilde{U}(i) = \emptyset$, then set $\tilde{S} = \tilde{S}-\{i\}$. Repeat this elimination process until $\tilde{S}$, $\{\tilde{U}(i) : i\varepsilon\tilde{S}\}$ have been minimized. If $\tilde{S}$ is nonempty, then return to step 2.

<u>Step 5</u>. Solve the system of equations:

$$g(i) = \max(\tilde{g}(i), \max_{u\varepsilon U}[\Sigma_{j\varepsilon S} P_{ij}(u)g(j)]) .$$

This may be done by value iteration:

$$g^m(i) = \max(\tilde{g}(i), \max_{u\varepsilon U} [\Sigma_{j\varepsilon S} P_{ij}(u)g^{m-1}(j)])$$

$$g^0(i) = \tilde{g}(i).$$

or by solving the linear program:

$$\begin{aligned} &\text{min:} && e \\ &\text{subject:} && g(i) \geq \Sigma_{j\varepsilon S} P_{ij}(u)g(j) - e \\ & && g(i) \geq \tilde{g}(i) \end{aligned}$$

<u>Step 6</u>. Set $\tilde{U}(i) = \{u : g(i) = \Sigma_{j\varepsilon S} P_{ij}(u)g(j)\}$, and

$$\tilde{q}_i(u) = q_i(u) - g(i).$$

Now solve the Markov decision problem with incremental rewards $\tilde{q}_i(u)$ and admissible input set $\tilde{U}(i)$ while the system is in state i.

Since the average gain has been substracted from the incremental rewards, the transformed system has gain zero, and within any class of states for which g(i) is the same, the correct relative values will be obtained.

<u>Remark</u>: The policy determined in Step 5 is gain-optimal. Step 6 is necessary only if bias optimality is desired as well.

19. Existence of a Solution in General Infinite-horizon Problems

This section is concerned with well-posedness of optimization problems formulated in Chapter I. Its purpose is to establish conditions under which an optimal strategy exists. In the present analysis, the optimal strategy need not satisfy a finite-memory constraint.

A sufficient condition+ for existence of an optimal strategy is that there exist a solution to the infinite dimensional linear program:

$$v^*(\pi) = \max_{u\varepsilon U}\{\pi q(u) + \beta\Sigma_{y\varepsilon Y}(\pi P(y(u)1)\ v^*(T(\pi,u,y))\} - g^* \tag{19.1}$$

If the relative value function v^* exists, then there is a function $\bar{\phi}^*$ which describes the input maximizing (19.1) as a function of π. If $\bar{\phi}^*$ is used to select inputs on the basis of the information vector, then the optimal gain g^* will be achieved. $\bar{\phi}^*$ will be called an <u>optimal feasible policy</u>.

(19.2) <u>Definition</u>. An infinite-horizon FPS control problem is called <u>regular</u> if it is either discounted or both simply connected and detecdetectable.

+The straightforward proof parallels well-known arguments for the state-observable case; see Kushner [1971].

(19.3) Theorem. Suppose either (a) that $\beta<1$ or (b) that the FPS satisfies conditions of connectivity and (weak) detectability. If connectivity holds, then let ℓ_ρ and ρ be as in (11.5); otherwise define $\ell_\rho = \rho = 1$. If weak detectability holds, then let $\bar{\ell}$ and $\bar{a}$ be as in (14.22); otherwise define $\bar{\ell} = \bar{a} = 1$. Finally, define

$$L(\beta,\ell) = \Sigma_{k=0}^{\ell-1} \beta^k = \begin{cases} (1-\beta^\ell)/(1-\beta), & \text{if } \beta<1 \\ \ell, & \text{if } \beta=1 \end{cases} \tag{19.4}$$

and

$$\Omega = \frac{L(\beta,\ell\rho+\bar{\ell})Q}{1-\beta^{(\ell\rho+\bar{\ell})}(1-\rho)(1-\bar{a})} \tag{19.5}$$

Then there exists a solution v^* to (19.1) having the following properties:

(i) v^* is continuous throughout Π_N

(ii) v^* is convex

(iii) $\| v^* \|_D \le \Omega$

Remark. If $\beta=1$, then $\Omega = \dfrac{(\ell\rho+\bar{\ell})Q}{(1-\rho)(1-\bar{a})}$.

Heuristic Justification: Only the undiscounted ($\beta=1$), strongly connected ($\ell_\rho = 1$), weakly subrectangular ($\bar{\ell}=1$) is considered here.

A solution $v^* = \lim_{m\to\infty} \hat{v}^m$ is constructed by damped value iteration (18.10) where, following (17.3),

$$v^{m+1}(\pi) = 1/2\ v^m(\pi) + 1/2\ \max_{u\in U} \{\pi q(u) + \Sigma_{y\in Y}\ v^m(\pi P(y|u))\}$$

$$v^0(\pi) = 0 \tag{19.6}$$

Then

$$v^m = \Sigma_{k=0}^{m}\ (1/2)^k \binom{m}{k} v_0^k \tag{19.7}$$

where v_0^k is the finite-horizon value function when k decisions remain. Each v_0^k is convex by an arguments given in Section 17.

It is now demonstrated (by induction) that $\| v_0^m \|_D \leq \Omega$. Since v^m is convex, it achieves a maximum at some vertex of Π_N. Let j be the state that maximizes $v^m(e^j)$ and let u* be the input that maximizes $\{e^j q(u^*) + \Sigma_{y\in Y}\ v_0^{m-1}(e^j P(y|u^*))\}$, i.e. j is the most desirable initial state for an m-transition problem and u* is the first optimal input for such a problem when the initial state is known to be j. Then,

$$v_0^m(\pi) \geq \pi q(u^*) + \Sigma_{y\in Y}\ v_0^{m-1}(\pi P(y|u^*)), \qquad \forall\, \pi\in\Pi_N .$$

Now:

$$v_0^m(e^j - v_0^m(\pi))$$

$$\leq \{e^j q(u^*) + \Sigma_{y\in Y} v_0^{m-1}(e^j P(y|u^*))\}$$

$$- \{\pi q(u^*) + \Sigma_{y\in Y}\ v_0^{m-1}(\pi P(y|u^*))\}$$

$$= Q_{max} - Q_{min} + \Sigma_{y\varepsilon Y}[(e^j P(y|u^*)1)\ v_0^{m-1}(T(e^j,u^*,y))$$

$$- (\pi P(y|u^*)1)\ v_0^{m-1}(T(\pi,u^*,y))]$$

$$= Q + \pi_j\ \Sigma_{y\varepsilon Y}\ (e^j P(y|u^*)1)\left[v_0^{m-1}(T(e^j,u^*,y)) - v_0^{m-1}(T(\pi,u^*,y))\right]$$

$$+ (1-\pi_j)\Sigma_{y\varepsilon Y}[(e^j P(y|u^*)1)\ v_0^{m-1}(T(e^j,u^*,y))$$

$$- \left(\frac{(\pi-\pi_j e^j)}{(1-\pi_j)}\right) P(y|u^*)1\ \ v_0^{m-1}(T(\pi,u^*,y))]$$

$$\leq Q + \pi_j \Sigma_{y\varepsilon Y}\ \ (e^j P(y|u^*)1)a[(u^*,y)]\,||\,v_0^{m-1}||_D + (1-\pi_j)\,||\,v_0^{m-1}||_D$$

$$\leq Q + [1-\pi_j(1-\bar{a})]\,||\,v_0^{m-1}||_D \tag{19.8}$$

But, for any $\pi\varepsilon\Pi_N$,

$$v^{m+1}(\pi) = \max_{u\varepsilon U}\ \{\pi q(u) + \Sigma_{y\varepsilon Y}\ v^m(\pi P(y|u)) \leq Q_{max} + v^m(e^j) \tag{19.9}$$

and, letting $\hat{u}$ be the input for which $\Sigma_{i\varepsilon S}\Sigma_{y\varepsilon Y}\pi_i P_{ij}(y|\hat{u}) \leq 1-\rho$,

$$v^{m+1}(\pi) \geq \pi q(\hat{u}) + \Sigma_{y\varepsilon Y}\ v^m(\pi P(y|u))$$

$$\geq Q_{min} + v_0^m(\Sigma_{y\varepsilon Y}\ \pi P(y|\hat{u}))$$

$$\geq Q_{min} + v_0^m(e^j) - Q - [1-(1-\rho)(1-\bar{a})]\,||\,v_0^{m-1}||_D \tag{19.10}$$

where (19.8) was used to obtain the last inequality. Thus

$$\| v_0^{m+1} \|_D \le 2Q + [1-(1-\rho)(1-\bar{a})] \| v_0^{m-1} \|_D \tag{19.11}$$

Since $\| v_0^0 \|_D = 0$ and $\| v_0^1 \|_D \le Q$, it follows that

$$\| v_0^m \|_D \le \frac{2Q}{(1-\rho)(1-\bar{a})} , \quad m\varepsilon\langle 0,\infty\rangle .$$

Hence, by (19.6),

$$\| v^m \|_D = \| \hat{v}^m \|_D \le \frac{2Q}{(1-\rho)(1-\bar{a})} , \qquad m\varepsilon\langle 0,\infty\rangle \tag{19.12}$$

The damped value-iteration, (19.6), assures that, if $\{\hat{v}^m\}$ has any (pointwise) limit, then it converges uniformly to that limit; the sequence $\{\hat{v}^m\}$ has a limit because it is convex and bounded; thus v* exists and is a solution to (19.1). v* is convex and bounded, by convexity and boundedness of $\{\hat{v}^m\}$.

Continuity of v* is most readily established in strongly subrectangular systems. Here,

$$\{\pi q(u) + \Sigma_{y\varepsilon Y}(\pi P(y|u)1)v^*(T(\pi,u,y))\}$$

is continuous in π for each $u\varepsilon U$, because $\{T(\pi,u,y) : \pi\varepsilon\Pi_N\}$ lies in the interior of a face of Π_N (see Figure 5-1) and a convex function is always continuous over a relatively open subset of its domain. Thus the right-hand side of (19.1) is continuous, and v* is continuous.

<u>Proof</u>: The complete proof of (19.3) is given in Appendix A.

(19.13) <u>Corollary</u>. Let e^{j*} be the information vector of maximal value, in a connected, detectable, FPS control problem. Let ℓ_ρ, ρ be such that, for any $\pi \varepsilon \Pi_N$, there exists an input word $\underline{\hat{u}} \varepsilon U^{\ell_\rho}$ satisfying:

$$1-\Sigma_{\underline{y}\varepsilon Y^{\ell(\hat{u})}} \Sigma_{i\varepsilon S} \pi_i P_{ij*}(\underline{\hat{u}},\underline{y}) \leq \rho$$

Then $\| v^* \|_D \leq \Omega$, where Ω is given by (19.5).

<u>Interpretation</u>: $\| v^* \|_D$ may be bounded on the basis of reachability of the most valuable state alone. In a network of queues, the most valuable state is readily identified without solving the problem; (it is the state in which all queues are empty). In this manner, a tighter bound on $\| v^* \|_D$ is obtained.

(19.14) <u>Theorem</u>. Consider a regular FPS control problem. If the system is simply connected, then let ℓ_C, α_C be numbers such that the internal state enters the connected class with probability $1-\alpha_C$ or more after ℓ_C transitions and let Ω be as in (19.3) for the system restricted to C; otherwise define $\ell_C, \alpha_C=0$, and let Ω be as in (19.3) for the system as specified. Then there exists a continuous, convex, bounded, relative value function v* satisfying (19.1), such that

$$\| v^* \|_D \leq \Omega + \frac{\ell_C Q}{1-\alpha_C}$$

Proof: It is necessary only to demonstrate boundedness of values $\{\hat{v}_0^m\}$ in the proof of (19.3). Now

$$\max_{i\varepsilon S}\{v^m(e^i)\} \leq \ell_C Q + \alpha_C \max_{i\varepsilon S}\{v^m(e^i)\} + (1-\alpha_C)\max_{i\varepsilon C}\{v^m(e^i)\}$$

and so:

$$\max_{i\varepsilon S}\{v^m(e^i)\} - \max_{i\varepsilon C}\{v^m(e^i)\} \leq \frac{\ell_C Q}{1-\alpha_C}$$

Consequently, arguments given in the proof of (19.3) show that v* satisfies the desired conditions. †

20. An Alternate Formulation for Irregular Problems

Consider the following problem, to which no optimal solution exists.

(20.1) Example. $U = \{1,2\}$, $Y = \{1\}$, $N = 3$, $\pi(0) = (0,0,1)$ and

$$P(1|1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & .5 & .5 \end{bmatrix}, \qquad P(1|2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ .5 & 0 & .5 \end{bmatrix}.$$

The incremental reward vectors are:

$$q(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad q(2) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

The performance index is infinite-horizon undiscounted average reward. A suboptimal solution may be obtained by the following argument: if any reward at all is to be achieved, then the system must be made to enter state 1, through initial application of input 2. Once state 1 has been reached, input 1 should be applied at all times. Unfortunately, there is no way for the controller to learn whether state 1 has been entered. If input 2 is applied n times and input 1 is applied thereafter, the performance $1-(.5)^n$ is achieved; this may be made arbitrarily close to 1. The supremum feasible performance g can never be attained: if input 2 is applied at all times, then the gain will be zero; and if input 1 is applied once, at time k, then the system enters state 2 with probability $(.5)^k$ and the performance cannot exceed $1-(.5)^k$.

A well-known class of problems, to which no solution exists, is the finite-memory hypothesis testing problem with choice of experiments, also known as the N-armed bandit problem. In the two-armed bandit problem, a gambler is confronted with two slot machines. For each coin invested, one machine returns two coins with probability .6, none with probability .4; and the other machine returns two coins with probability .4, none with probability .6. It is not known initially which machine is the more favorable.

Failure of an optimal strategy to exist is a consequence of the infinitely-delayed splurge phenomenon discussed in section 5a. This, in turn, results from null-transitivity of certain information states in a system that is not detectable. Specifically, infinitely-delayed splurges may occur when:

(i) Under ε-optimal strategies, for ε sufficiently small, $\mu(k)>1$; i.e. there are recursively-computable functions of the state that may be interpreted as one-time hypotheses;

(ii) In the limit, where an infinite past is available, the correct hypothesis may be identified exactly, and a detectable problem results;

(iii) The cost of identifying an hypothesis is infinite.

Such problems may be solved in two steps, described below.

Step 1 (steady-state)

Under the assumption that the state was exactly known at some point in the infinitely distant past, the problem becomes detectable, and an optimal strategy exists. This strategy might not satisfy a finite-memory constraint, but its performance may be approximated, arbitrarily closely, by a finite-memory controller in the following sense: for any $\varepsilon>0$, there is a finite-memory controller whose average reward, over a given time interval of length K, lies between $g^*-\varepsilon$ and g^* with probability approaching unity as $K\to\infty$.

Step 2 (initial identification)

The correct hypothesis may be arbitrarily closely identified in a finite number of transitions. Let the terminal reward be 1 if the hypothesis is correctly identified, and 0 if it is not. Then solve the finite-horizon problem by the methods cited in Section 4, or by the algorithm of Sondik. (The initialization procedure will be described in greater detail in Section 21f).

This report is concerned with hypothesis-testing only to the extent that it occurs in problems of statistical decision and control. As long as a problem is detectable, its "dual control" aspects involve a reasonable tradeoff between information and control; otherwise the problem must be solved in two separate steps. If available memory is limited, then it must be decided how much memory is to be allocated to identification, and how much is to be allocated to steady-state performance. Note that memory allocation in this sense is indirectly determined by the discount β (when $\beta<1$), since it specifies the manner in which steady-state performance and identification costs are to be compared.

CHAPTER IV

COMPUTATION OF ε-OPTIMAL CONTROLLERS

21. Perceptive Dynamic Programming

a. The Basic Algorithm

It has been demonstrated, in Section 19, that there exist solutions to regular FPS control problems. Yet, it may be impossible to compute or to implement solutions that fail to satisfy a finite-memory constraint. This section introduces a feasible computational technique for the solution of such problems.

In the computational technique of _perceptive dynamic programming,_ an increasing sequence of memory sets, $\{M^n\}$, is used to construct approximations to the original problem. Each approximation is parameterized by a memory set; the n-th approximation depends on memory set M^n, but the iteration number n alone may be used to facilitate notation. The approximation corresponding to memory set M is the Markov decision problem that results when the augmented system induced by M is assumed to be state-observable. The solution to this problem is called a _perceptive solution_; it consists of a _perceptive value function_ $v^M : \hat{X}[M]\to R$ and a _perceptive gain_ $g[M]$, obtained by solving the system of equations:

$$v^M[i,\underline{z}] = \max_{u\varepsilon U}\{q_{\underline{z}}^M(i,\underline{u}) + \beta\Sigma_{j\varepsilon S}\Sigma_{y\varepsilon Y}P_{\underline{z}}^M(i,j,(u,y))v^M[j,T^M(\underline{z},(u,y))]\} - g[M], \qquad [i,\underline{z}]\varepsilon\hat{X}[M] \tag{21.1}$$

In (21.1), perception of delayed states is assumed only when the memory state is essential. Optimal decisions and relative values for the remaining memory states can be determined by solving:

$$\bar{v}^M[\pi(0), \underline{z}] = \max_{u \varepsilon U}\{T(\pi(0), \underline{z})q(u)$$

$$+ \beta \Sigma_{y \varepsilon Y}(T(\pi(0), \underline{z})P(y|u)1)$$

$$\left\{\begin{array}{ll} [\Sigma_{i \varepsilon S}\Sigma_{j \varepsilon S}\Sigma_{k \varepsilon S}\ \pi_i(0) \cdot P_{ij}(\underline{z}(u,y)-T^M(\underline{z},(u,y))) & \\ \cdot P_{jk}(T^M(\underline{z},\ (u,y))) \cdot v^m[j,T^M(\underline{z},\ (u,y))]] & \\ /[\pi(0)P(\underline{z})P(y|u)1], & \text{if } T^M(\underline{z},\ (u,y))\varepsilon \mathrm{ess}[M] \\ \bar{v}^M[\pi(0), \underline{z}(u,y)], & \text{otherwise} \end{array}\right\}$$

$$-g[M], \qquad \underline{z}\varepsilon Z^+(\pi(0)) \cap \mathrm{ess}[M] \tag{21.2}$$

The policy maximizing (21.1) and (21.2) is denoted $\bar{\psi}^M$.

A <u>feasible strategy</u> ϕ^M is devised by constructing a policy adapted to M which realizes it. Select any mapping $\hat{s} : \mathrm{ess}[M^m] \to S$ satisfying:

$$\hat{s}[\underline{z}]\varepsilon I(\underline{z}), \quad \forall\, \underline{z}\varepsilon \mathrm{ess}[M] \tag{21.3}$$

The substitution of a state guess for a perceived state will be called <u>pseudo-perception</u>. Define the feasible policy to be

$$\bar{\phi}^M[\underline{z}] = \begin{cases} \bar{\psi}[\underline{z}], & \text{if } \underline{z}\varepsilon M\text{-ess}[M] \\ \bar{\psi}[\hat{s}[\underline{z}],\underline{z}], & \text{if } \underline{z}\varepsilon \text{ess}[M] \end{cases} \tag{21.4}$$

h[M] will denote the performance achieved by ϕ^M. Clearly:

$$h[M] \leq g^* \leq g[M] \tag{21.5}$$

For a given sequence of memory sets $\{M^m\}$, these bounds may be denoted h^n and g^n, respectively.

A key result is the following theorem, which states that $g^n - h^n \to 0$ as $n \to \infty$.

(21.6) <u>Theorem</u>. Suppose either (a) that $\beta<1$ or (b) that the FPS satisfies conditions of <u>connectivity</u> and <u>(weak)</u> <u>detectability</u>, and let $\ell_\rho, \rho, \bar{\ell}, \bar{a}$, $L(\beta,\ell)$ and Ω be as in (19.3)-(19.5). Also let $\bar{\alpha}$ be as in (14.22) if WDT is satisfied; otherwise define $\bar{\alpha}=1$. Then:

(a) $g[M] - g^* \leq \bar{\alpha}^{\ell_{min}[M]\div\bar{\ell}} \beta^{\ell_{min}[M]} 4\Omega$

(b) $g[M] - \eta[M] \leq \bar{\alpha}^{\ell_{min}[M]\div\bar{\ell}} \beta^{\ell_{min}[M]}$

$$\cdot \left\{ L(\beta, \ell_{max}[M]-\ell_{min}[M]) + \beta^{\ell_{max}[M]-\ell_{min}[M]}\ ' \frac{L(\beta,\bar{\ell})}{1-\beta^{\bar{\ell}}\bar{\alpha}} \right)\right\} 2[Q+\beta\Omega]$$

<u>Heuristic Justification</u>: The proof follows an argument given in Section 5e.

<u>Proof</u>: The complete proof is given in Appendix B.

The generalization to systems having transient states is straightforward.

(21.7) <u>Corollary</u>. For any regular FPS control problem:

$$
\begin{aligned}
g[M] - h[M] \leq\ & \bar{\alpha}^{\ell_{\min}[M] \div \bar{\ell}}\, \beta^{\ell_{\min}[M]} \\
& \cdot \left\{ L(\beta, \ell_{\max}[M]-\ell_{\min}[M]) + \beta^{\ell_{\max}[M]-\ell_{\min}[M]} \left(\frac{L(\beta,\bar{\ell})}{1-\beta^{\bar{\ell}}\bar{\alpha}} \right) \right\} 2[Q + \beta\Omega] \\
& + \alpha_C^{\ell_{\min}[M] \div \bar{\ell}}\, \beta^{\ell_{\min}[M]} \left(\frac{\ell_C Q}{1-\alpha_C} \right)
\end{aligned}
$$

where α_C and ℓ_C are as in (19.14).

b. Discussion

The upper bounds $\{g^n\}$ are clearly nonincreasing. The lower bounds $\{h^n\}$ might decrease if an unfortunate choice of $\hat{s}(\cdot)$ is made. If $h^n < h^{n'}$, $n > n'$, then $\phi^{n'}$ may be substituted for ϕ^n, since it is adapted to M^n. Hence, the bounds $\{h^n\}$ and $\{g^n\}$ can be made <u>monotone</u>.

If the family of memory sets $\{M^n\} = \{Z^{n*} \cap Z^+\}$ is used, then the bounds will converge geometrically as well. Computational experience indicates that convergence will occur more rapidly than predicted by (21.6), but that may not be rapid enough to assure feasibility, due to the fact that the computational effort (computer time or memory) required to solve the perceptive problem increases as $n \to \infty$. Since computational effort is linearly related to the number of memory states, the effort required to place the bounds within $\bar{\varepsilon}$ of each other is proportional to $\bar{\varepsilon}^{-1/\tau}$, where τ is given by (14.22).

A more favorable rate of convergence is obtained when the memory sets are computed recursively. Memory states that are unlikely to be recurrent under the optimal perceptive policy can be ommited; those which were recurrent during the previous iteration may be extended (by the addition in the memory tree of branches from the nodes to which they correspond).

Problems of decoding a noisy Markov channel (see references listed in Section 4) are subrectangular, and lend themselves to convergence rate analysis. In most problems, however, there doesn't seem to be much use in computing the contraction indices $\bar{\alpha}$ and ρ. Execution of two or three iterations of perceptive dynamic programming yields more reliable indicators of convergence rates.

c. Pseudo-perceptive Dynamic Programming

Pseudo-perceptive dynamic programming is a computational procedure in which the delayed state is guessed and substituted into the model

<u>before</u> optimization is performed, resulting in a reduction, by a factor of N^2, in the number of augmented states considered during each optimization step. The performance obtained will be an approximation to the optimal performance: if the delayed state is optimized, and not merely guessed, then the performance obtained will be an upper bound as well. However, pseudo-perceptive dynamic programming does not then yield a <u>lower bound</u> to optimal feasible performance.

d. Recursive Computation of the Memory Sets

Experience indicates that the choice of memory sets is crucial to efficient performance of the perceptive dynamic programming algorithm. For example, computation time and storage requirements increase linearly with the number of memory states; yet, certain memory states can be shown *a priori* to occur very rarely in the optimally controlled system.

Some recommended "tricks" are:

1) Do not add branches to node $\underline{z}$ of the memory tree if, whenever the memory state is z, the optimal perceptive decision does not depend on the delayed-state component of the augmented state.

2) Do not add branches to node $\underline{z}$ of the memory tree if $\underline{z}$ is not recurrent under the optimal perceptive strategy obtained during the most recent iteration.

3) Do not add branches to node $\underline{z}$ of the memory tree if all entries of $P(\underline{z})$ are small.

e. Minimization of Memory Size by Selective Pseudo-perception

The state guess $\hat{s}(\cdot)$ may be selected according to an *ad hoc* rule which causes the feasible strategy to perform as well as possible (e.g. $\hat{s} =$ most likely state). It might instead be selected so that the number of recurrent memory states under the feasible strategy will be minimized. Such an approach assures that another iteration, with a larger memory set, might be performed, although the current feasible performance lower bound $h[M^n]$ will suffer. During the final iteration, this approach to the selection of $\hat{s}(\cdot)$ may reduce the cost of implementing the solution obtained.

f. Initialization Procedure

Suppose that a perceptive solution has been obtained, and that, from this, a feasible policy has been designed. The feasible policy determines near optimal decisions *in the steady-state*. It is also necessary to determine an *initialization procedure* to be followed by the controller.

A particularly simple way of doing so is the following: Represent the system under the feasible strategy as a Markov chain, and determine the relative values of all augmented states. Then solve a finite horizon problem, in which the input set includes the memory set as well as an input representing a memory state indicates that the feasible policy should be used thereafter, starting in the specified memory state. The value function will be monotone increasing, in the number of initialization steps allowed.

If the system under the feasible strategy is multiple chained, then the finite horizon problem should be to maximize the eventual gain. In the case of an N-armed bandit problem, the feasible (steady-state) policy is trivially computed, since the previous decision determines the optimal present decision. The initialization procedure then constitutes an identification of the correct hypothesis.

22. A Computational Algorithm

In order to assess the practicality of perceptive dynamic programming, a computer program was written to solve general FPS control problems with undiscounted infinite-horizon performance index. The program is described below. Computational results, obtained using this program, are described in the following section.

The source code, which is written entirely in PL/I, is listed in Appendix C. It has a source length of 1250 cards, and the object code occupies 110K bytes of storage on the IBM 370/168.

The program accepts the following data as input:

<u>Title</u>:

A character string of length not exceeding 32, which identifies the problem to be solved.

<u>Problem dimensions</u>

N, the number of internal states.

NU, the number of inputs.

NY, the number of outputs.

NZ, the number of input-output pairs.

FMT, the output format (1 = "long", 0 = "short").

<u>Termination specifications</u>: (conditions under which execution should be terminated)

MIN_ERR, the minimum value of $g^n - h^n$.

MAX_M, the maximum number of memory states.

MAX_ESS_M, the maximum number of essential memory states.

MAX_TIME, the maximum number of seconds to be allowed.

Transition probabilities:

Each matrix is preceeded by a list of input-output pairs and a single zero which marks the end of that list; the matrix is then listed in row-major order.

Expected incremental reward vectors:

The vector a(1), ... , q(NU) are entered in turn.

Computation then proceeds according to the following outline:

Step 1: Create a memory tree (hereafter denoted by M) containing only the empty word e; and set ERR = Q.

Step 2: Solve the perceptive problem. This is done by damped value iteration (18.10), along with the test for non-optimal actions of Hastings [1976]. The optimization is performed only on $\hat{X}[M]$, the connected class of augmented states consisting of a delayed internal state along with an essential memory state. Computation is terminated when, after k_1 steps of value iteration, the Odoni bounds (18.11) are within ERR • $(.001)(1.2)^{k_1}$ of each other.

Step 3: Flag memory states that are recurrent under the optimal perceptive strategy (indicated by a "G" in the printout). For those memory states only, determine the feasible strategy which selects the input most likely to be optimal.

Step 4: Determine h^n by value iteration without optimization of inputs. Computation is terminated when, after k_2 steps, the Odoni bounds are witbin ERR $\cdot$ $[(.001)(1.2)^{k_1}]$ $[(.01)(2)^{k_2}]$ of each other.

Step 5: Flag memory states that are recurrent under both the optimal perceptive strategy and the feasible strategy for the present iteration (indicated by an "H" in the printout).

Step 6: Set ERR = {the upper Odoni bound on g^n} - {the lower Odoni bound on h^n}. Print a report of the current iteration. If any termination specifications have been met, then stop.

Step 7: For every triplet ($\underline{z}$,u,y) satisfying

(i) $\underline{z}$ is an essential memory state that was recurrent under the most recent optimal perceptive strategy,

(ii) u is an optimal input for some augmented state of the form $[i,\underline{z}]$,

(iii) $T^M(\underline{z}, (u,y)) < \underline{z}(u,y)$,

add to M the memory state which contains the $\ell[T^M(\underline{z}, (u,y))] + 1$ rightmost input-output pairs in $\underline{z}$ (u,y). Also add whatever memory states are required to satisfy (8.4). Then return to step 2.

Further details regarding execution procedure and methodology, may be found in the source code.

The output consists of a page which lists the input data, followed by an iteration report for each iteration performed. The iteration report heading contains the following information:

Line 1: The iteration number, the number of memory states, the number of essential memory states, the time at which preparation of the memory tree for value iteration was concluded.

Line 2: The upper and lower Odoni bounds on g^n, the number of value iteration steps performed and the time at which value iteration was concluded, in Step 2.

Line 3: The upper and lower Odoni bounds on h^n, the number of value iteration steps performed and the time at which value-iteration was concluded, in Step 4.

In the long format, the iteration report heading is followed by a table in which N+1 lines are devoted to each essential memory state. The column headings and data listed are as follows:

RC Recurrent state flags "G" and "H" are listed below. "G" indicates that the memory state is recurrent under the optimal perceptive strategy; "H" indicates that the memory state is also recurrent under the feasible strategy.

I Delayed-state component of the augmented state.

U Input selected by the feasible strategy (first line), and optimal perceptive inputs (following lines). An asterisk (*) indicates that the feasible strategy always picks the optimal perceptive input.

V(G) Relative value for the perceptive problem.

V(H) Relative value for the feasible problem.

PROBS For memory state $\underline{z}$, $P(\underline{z})$ is listed.

MEMORY STATES

The memory states are listed below in the form of a left-handed tree.

In the short format, only the first line of each memory state table is printed.

23. Computational Results

a. The Machine Maintenance and Repair Problem

The Machine Maintenance and Repair Problem was formulated in Section 3, and a procedure, which in principle leads to a solution, was then described. That procedure is in fact equivalent to perceptive dynamic programming based on the fixed family of increasing memory sets $\{Z^{(n-1)*} \cap Z^+\}$.

The solution was actually obtained by perceptive dynamic programming on the basis of recursively computed memory sets, as described in Section 21d. The largest intermediary Markov decision problem solved had 93 states.

The steps that lead to this solution are briefly described. During the first six iterations, perceived states determine the optimal input, so feasible performance remains poor. Since pseudo-perception initially takes the form $\hat{s}=1$, input u=1 ("manufacture") is selected at all times. On the seventh iteration, the input u=2 ("examine") is selected whenever u=1 ("manufacture") occurred four times previously; but this is done only for the purpose of obtaining a perception free of delay. In iteration eight, the memory set is augmented by branches corresponding to input u=2 ("examine"); that input is no longer selected and feasible performance increases for the first time. A similar pattern continues until sufficient memory has been allocated to realize the optimal strategy, and to eliminate suboptimal decisions motivated by perceptive information structure.

Note that this problem is not detectable. Indeed, there are two possible decompositions into detectable parts: if the machine is never repaired, then there is only one recurrent state and the system is trivially detectable; if the machine is repaired, then all information previous to the repair is dispensable; in either case $\alpha=0$. The rate of convergence of perceptive dynamic programming is determined by the rate of absorption of transient states in the former case which is $\alpha_C = .99$, $\ell_C = 2$ (very unfavorable). The convergence rate for memory sets used in section 3 is bounded by:

$$g^n - h^n \leq (.99)^{m-1} \quad \cdot \quad \left[\frac{2 \cdot 3.4025}{1 - .99}\right]$$

The actual convergence obtained was, of course, considerably more rapid.

The input deck for this problem took the form:

```
//  EXEC  PLIXG,PROG='U.M13014.P10015.PLATZSYS.LOAD(LDMOD)'
//G.SYSIN  DD  *,DCB=BLKSIZE=2000
   'MACHINE MAINTENANCE & REPAIR', 3,4,3,4,1, 20,.01,100,100,
   1,1,0,
      .81,.18,.01,  0,.9,.1,  0,0,1,
   2,2,0,
      .81,.09,.0025,  0,.45,.025,  0,0,.25,
   2,3,0,
      0,.09,.0025,  0,.45,.075,  0,0,.75
   3,1,4,1,0,
      1,,,1,,,1,,,
   .9025,.475,.25,  .6525,.225,0,  -.5,-1.5,-2.5,  -2,-2,-2,
/*EOJ
```

The computer-generated report is given on the next 29 pages.

MACHINE MAINTENANCE & REPAIR PROBLEM SPECS

3 STATES 4 INPUTS 3 OUTPUTS 4 I/O PAIRS

TIME LIMIT: 20.00 MIN ERR: 0.010

MAX MEM: 100 MAX ESS MEM: 100

TRANSITION PROBABILITIES:

Z	(U, Y)	P		
1	1 1			
		0.8100	0.1800	0.0100
		0.0000	0.9000	0.1000
		0.0000	0.0000	1.0000
2	2 2			
		0.8100	0.0900	0.0025
		0.0000	0.4500	0.0250
		0.0000	0.0000	0.2500
3	2 3			
		0.0000	0.0900	0.0075
		0.0000	0.4500	0.0750
		0.0000	0.0000	0.7500
4	3 1			
	4 1			
		1.0000	0.0000	0.0000
		1.0000	0.0000	0.0000
		1.0000	0.0000	0.0000

INCREMENTAL REWARDS:

U	C		
1	0.9025	0.4750	0.2500
2	0.6525	0.2250	0.0000
3	-0.5000	-1.5000	-2.5000
4	-2.0000	-2.0000	-2.0000

```
MACHINE MAINTENANCE & REPAIR            PAGE  2       TABLE  1.01

+-----------------------------------------------------------------+
|   ITERATION  1     MEM =  1    ESS MEM =  1    TIME =  0.16    |
|                                                                 |
|   0.499 < G <   0.531        17  STEPS         TIME =  0.25    |
|   0.250 < H <   0.444         9  STEPS         TIME =  0.26    |
+-----------------------------------------------------------------+

RC   I  U  V(G)   V(H)   PROBS                       MEMORY STATES

GH      1                                                      <E>
     1  1  2.61   2.76  1.0000  0.0000  0.0000
     2  3  0.59   0.36  0.0000  1.0000  0.0000
     3  4  0.09  -0.79  0.0000  0.0000  1.0000
```

```
MACHINE MAINTENANCE & REPAIR          PAGE  3      TABLE  2.01

+------------------------------------------------------------+
|   ITERATION  2    MEM =  3   ESS MEM =  3   TIME =  0.36   |
|                                                            |
|  0.494 < G <   0.495        8  STEPS        TIME =  0.46   |
|  0.250 < H <   0.395       14  STEPS        TIME =  0.50   |
+------------------------------------------------------------+

RC   I  U  V(G)  V(H)  PROBS                         MEMORY STATES

        1                                                     <E>
     1  1  2.48        1.0000  0.0000  0.0000
     2  3  0.48        0.0000  1.0000  0.0000
     3  4 -0.02        0.0000  0.0000  1.0000

GH      1                                                       1
     1  1  2.07  2.63  0.8100  0.1800  0.0100
     2  3  0.38  0.43  0.0000  0.9000  0.1000
     3  4 -0.02 -0.80  0.0000  0.0000  1.0000

G       1*                                                      4
     1  1  2.48        1.0000  0.0000  0.0000
     2  1  2.48        1.0000  0.0000  0.0000
     3  1  2.48        1.0000  0.0000  0.0000
```

```
MACHINE MAINTENANCE & REPAIR          PAGE  4       TABLE  3.01

+-------------------------------------------------------------+
|   ITERATION  3    MEM =  5   ESS MEM =  5   TIME =  0.60   |
|                                                             |
|   0.477 < G <   0.478        10  STEPS      TIME =  0.81   |
|   0.250 < H <   0.385        14  STEPS      TIME =  0.89   |
+-------------------------------------------------------------+

RC   I  U  V(G)  V(H)  PROBS                         MEMORY STATES

        1                                                      <E>
     1  1  2.44        1.0000  0.0000  0.0000
     2  3  0.46        0.0000  1.0000  0.0000
     3  4 -0.04        0.0000  0.0000  1.0000

        1                                                        1
     1  1  2.01        0.8100  0.1800  0.0100
     2  3  0.36        0.0000  0.9000  0.1000
     3  4 -0.04        0.0000  0.0000  1.0000

GH      1                                                    1
     1  1  1.67  2.27  0.6561  0.3078  0.0361
     2  3  0.27  0.39  0.0000  0.8100  0.1900
     3  4 -0.04 -0.69  0.0000  0.0000  1.0000

G       1*                                                   4
     1  1  2.01        0.8100  0.1800  0.0100
     2  1  2.01        0.8100  0.1800  0.0100
     3  1  2.01        0.8100  0.1800  0.0100

G       1*                                                       4
     1  1  2.44        1.0000  0.0000  0.0000
     2  1  2.44        1.0000  0.0000  0.0000
     3  1  2.44        1.0000  0.0000  0.0000
```

ITERATION 4	MEM = 7	ESS MEM = 7	TIME = 0.99	
0.462 < G < 0.464		12 STEPS	TIME = 1.30	
0.250 < H < 0.382		13 STEPS	TIME = 1.43	

RC	I	U	V(G)	V(H)	PROBS			MEMORY STATES
		1						<E>
	1	1	2.41		1.0000	0.0000	0.0000	
	2	3	0.45		0.0000	1.0000	0.0000	
	3	4	-0.05		0.0000	0.0000	1.0000	
		1						1
	1	1	1.97		0.8100	0.1800	0.0100	
	2	3	0.35		0.0000	0.9000	0.1000	
	3	4	-0.05		0.0000	0.0000	1.0000	
		1						1
	1	1	1.62		0.6561	0.3078	0.0361	
	2	3	0.26		0.0000	0.8100	0.1900	
	3	4	-0.05		0.0000	0.0000	1.0000	
GH		1						1
	1	1	1.33	1.92	0.5314	0.3951	0.0734	
	2	3	0.18	0.34	0.0000	0.7290	0.2710	
	3	4	-0.05	-0.58	0.0000	0.0000	1.0000	
G		1*						4
	1	1	1.62		0.6561	0.3078	0.0361	
	2	1	1.62		0.6561	0.3078	0.0361	
	3	1	1.62		0.6561	0.3078	0.0361	
G		1*						4
	1	1	1.97		0.8100	0.1800	0.0100	
	2	1	1.97		0.8100	0.1800	0.0100	
	3	1	1.97		0.8100	0.1800	0.0100	
G		1*						4
	1	1	2.41		1.0000	0.0000	0.0000	
	2	1	2.41		1.0000	0.0000	0.0000	
	3	1	2.41		1.0000	0.0000	0.0000	

```
MACHINE MAINTENANCE & REPAIR          PAGE  6      TABLE  5.01

+-------------------------------------------------------------+
|   ITERATION  5    MEM =  9    ESS MEM =  9   TIME =  1.55   |
|                                                             |
|   0.449 < G <   0.452        14  STEPS       TIME =  2.01   |
|   0.250 < H <   0.374        13  STEPS       TIME =  2.18   |
+-------------------------------------------------------------+

RC   I  U  V(G)   V(H)  PROBS                       MEMORY STATES

        1                                                     <E>
     1  1  2.39         1.0000  0.0000  0.0000
     2  3  0.44         0.0000  1.0000  0.0000
     3  4 -0.06         0.0000  0.0000  1.0000

        1                                                       1
     1  1  1.93         0.8100  0.1800  0.0100
     2  3  0.34         0.0000  0.9000  0.1000
     3  4 -0.06         0.0000  0.0000  1.0000

        1                                                     1
     1  1  1.57         0.6561  0.3078  0.0361
     2  3  0.25         0.0000  0.8100  0.1900
     3  4 -0.06         0.0000  0.0000  1.0000

        1                                                   1
     1  1  1.27         0.5314  0.3951  0.0734
     2  3  0.17         0.0000  0.7290  0.2710
     3  4 -0.06         0.0000  0.0000  1.0000

GH      1                                                 1
     1  1  1.04   1.65  0.4305  0.4513  0.1183
     2  3  0.09   0.31  0.0000  0.6561  0.3439
     3  4 -0.06  -0.49  0.0000  0.0000  1.0000

G       1*                                                4
     1  1  1.27         0.5314  0.3951  0.0734
     2  1  1.27         0.5314  0.3951  0.0734
     3  1  1.27         0.5314  0.3951  0.0734

G       1*                                                  4
     1  1  1.57         0.6561  0.3078  0.0361
     2  1  1.57         0.6561  0.3078  0.0361
     3  1  1.57         0.6561  0.3078  0.0361

G       1*                                                    4
     1  1  1.93         0.8100  0.1800  0.0100
     2  1  1.93         0.8100  0.1800  0.0100
     3  1  1.93         0.8100  0.1800  0.0100
```

G 1* 4

1	1	2.39	1.0000	0.0000	0.0000
2	1	2.39	1.0000	0.0000	0.0000
3	1	2.39	1.0000	0.0000	0.0000

ITERATION 6	MEM = 11	ESS MEM = 11			TIME =	2.30
0.438 < G <	0.441	16	STEPS		TIME =	2.98
0.250 < H <	0.372	12	STEPS		TIME =	3.23

RC	I	U	V(G)	V(H)	PROBS			MEMORY STATES
		1						<E>
	1	1	2.37		1.0000	0.0000	0.0000	
	2	3	0.43		0.0000	1.0000	0.0000	
	3	4	-0.07		0.0000	0.0000	1.0000	
		1						1
	1	1	1.90		0.8100	0.1800	0.0100	
	2	3	0.33		0.0000	0.9000	0.1000	
	3	4	-0.07		0.0000	0.0000	1.0000	
		1						1
	1	1	1.52		0.6561	0.3078	0.0361	
	2	3	0.24		0.0000	0.8100	0.1900	
	3	4	-0.07		0.0000	0.0000	1.0000	
		1						1
	1	1	1.22		0.5314	0.3951	0.0734	
	2	3	0.16		0.0000	0.7290	0.2710	
	3	4	-0.07		0.0000	0.0000	1.0000	
		1						1
	1	1	0.97		0.4305	0.4513	0.1183	
	2	3	0.08		0.0000	0.6561	0.3439	
	3	4	-0.07		0.0000	0.0000	1.0000	
GH		1						1
	1	1	0.78	1.38	0.3487	0.4836	0.1677	
	2	3	0.02	0.26	0.0000	0.5905	0.4095	
	3	4	-0.07	-0.40	0.0000	0.0000	1.0000	
G		1*						4
	1	1	0.97		0.4305	0.4513	0.1183	
	2	1	0.97		0.4305	0.4513	0.1183	
	3	1	0.97		0.4305	0.4513	0.1183	
G		1*						4
	1	1	1.22		0.5314	0.3951	0.0734	
	2	1	1.22		0.5314	0.3951	0.0734	
	3	1	1.22		0.5314	0.3951	0.0734	

G		1*						4	1	1
	1	1	1.52	0.6561	0.3078	0.0361				
	2	1	1.52	0.6561	0.3078	0.0361				
	3	1	1.52	0.6561	0.3078	0.0361				
G		1*							4	
	1	1	1.90	0.8100	0.1800	0.0100				
	2	1	1.90	0.8100	0.1800	0.0100				
	3	1	1.90	0.8100	0.1800	0.0100				
G		1*								4
	1	1	2.37	1.0000	0.0000	0.0000				
	2	1	2.37	1.0000	0.0000	0.0000				
	3	1	2.37	1.0000	0.0000	0.0000				

```
+--------------------------------------------------------------+
|   ITERATION  7    MEM = 13   ESS MEM = 13   TIME =  3.37     |
|                                                              |
|   0.437 < G <   0.439       13  STEPS       TIME =  4.08     |
|   0.195 < H <   0.369       14  STEPS       TIME =  4.30     |
+--------------------------------------------------------------+

RC   I  U  V(G)  V(H)  PROBS                       MEMORY STATES

GH      1                                                    <E>
     1  1  2.36  2.94  1.0000  0.0000  0.0000
     2  3  0.43  0.43  0.0000  1.0000  0.0000
     3  4 -0.07 -0.91  0.0000  0.0000  1.0000

GH      1                                                      1
     1  1  1.90  2.40  0.8100  0.1800  0.0100
     2  3  0.33  0.24  0.0000  0.9000  0.1000
     3  4 -0.07 -0.96  0.0000  0.0000  1.0000

GH      1                                                    1
     1  1  1.52  1.93  0.6561  0.3078  0.0361
     2  3  0.24  0.07  0.0000  0.8100  0.1900
     3  4 -0.07 -1.01  0.0000  0.0000  1.0000

GH      1                                                  1
     1  1  1.21  1.51  0.5314  0.3951  0.0734
     2  3  0.16 -0.09  0.0000  0.7290  0.2710
     3  4 -0.07 -1.06  0.0000  0.0000  1.0000

GH      2                                                1
     1  2  0.96  1.15  0.4305  0.4513  0.1183
     2  3  0.08 -0.24  0.0000  0.6561  0.3439
     3  4 -0.07 -1.11  0.0000  0.0000  1.0000

        2                                              1
     1  2  0.76        0.3487  0.4836  0.1677
     2  3  0.02        0.0000  0.5905  0.4095
     3  4 -0.07        0.0000  0.0000  1.0000

        2                                            1
     1  2  0.59        0.2824  0.4980  0.2195
     2  3 -0.04        0.0000  0.5314  0.4686
     3  4 -0.07        0.0000  0.0000  1.0000

        2*                                           4
     1  2  0.76        0.3487  0.4836  0.1677
     2  2  0.76        0.3487  0.4836  0.1677
     3  2  0.76        0.3487  0.4836  0.1677
```

G		2*					4	1	1	1	1
	1	2	0.96	0.4305	0.4513	0.1183					
	2	2	0.96	0.4305	0.4513	0.1183					
	3	2	0.96	0.4305	0.4513	0.1183					
G		1*						4			
	1	1	1.21	0.5314	0.3951	0.0734					
	2	1	1.21	0.5314	0.3951	0.0734					
	3	1	1.21	0.5314	0.3951	0.0734					
G		1*							4		
	1	1	1.52	0.6561	0.3078	0.0361					
	2	1	1.52	0.6561	0.3078	0.0361					
	3	1	1.52	0.6561	0.3078	0.0361					
G		1*								4	
	1	1	1.90	0.8100	0.1800	0.0100					
	2	1	1.90	0.8100	0.1800	0.0100					
	3	1	1.90	0.8100	0.1800	0.0100					
G		1*									4
	1	1	2.36	1.0000	0.0000	0.0000					
	2	1	2.36	1.0000	0.0000	0.0000					
	3	1	2.36	1.0000	0.0000	0.0000					

```
MACHINE MAINTENANCE & REPAIR          PAGE 12      TABLE  8.01

+--------------------------------------------------------------+
|   ITERATION  8    MEM = 15   ESS MEM = 14   TIME =  4.53     |
|                                                              |
|   0.432 < G <   0.435        14  STEPS       TIME =  5.36    |
|   0.308 < H <   0.400        12  STEPS       TIME =  5.72    |
+--------------------------------------------------------------+

RC  I  U  V(G)  V(H)  PROBS                          MEMORY STATES

GH     1                                                      1
    1  1  1.76  1.89  0.8100  0.1800  0.0100
    2  3  0.19 -0.13  0.0000  0.9000  0.1000
    3  4 -0.21 -0.99  0.0000  0.0000  1.0000

GH     1                                                   1
    1  1  1.37  1.47  0.6561  0.3078  0.0361
    2  3  0.10 -0.22  0.0000  0.8100  0.1900
    3  4 -0.21 -0.88  0.0000  0.0000  1.0000

GH     1                                                1
    1  1  1.06  1.10  0.5314  0.3951  0.0734
    2  3  0.02 -0.28  0.0000  0.7290  0.2710
    3  4 -0.21 -0.76  0.0000  0.0000  1.0000

GH     2                                             1
    1  2  0.81  0.80  0.4305  0.4513  0.1183
    2  3 -0.05 -0.33  0.0000  0.6561  0.3439
    3  4 -0.21 -0.63  0.0000  0.0000  1.0000

       2                                          1
    1  2  0.61        0.3487  0.4836  0.1677
    2  3 -0.12        0.0000  0.5905  0.4095
    3  4 -0.21        0.0000  0.0000  1.0000

       2                                       1
    1  2  0.44        0.2824  0.4980  0.2195
    2  3 -0.18        0.0000  0.5314  0.4686
    3  4 -0.21        0.0000  0.0000  1.0000

       2*                                      4
    1  2  0.61        0.3487  0.4836  0.1677
    2  2  0.61        0.3487  0.4836  0.1677
    3  2  0.61        0.3487  0.4836  0.1677

GH     2*                                         4
    1  2  0.81  0.80  0.4305  0.4513  0.1183
    2  2  0.81  0.80  0.4305  0.4513  0.1183
    3  2  0.81  0.80  0.4305  0.4513  0.1183
```

MACHINE MAINTENANCE & REPAIR PAGE 13 TABLE 8.02

GH		1*						4	1	1	1
	1	1	1.06	1.10	0.5314	0.3951	0.0734				
	2	1	1.06	1.10	0.5314	0.3951	0.0734				
	3	1	1.06	1.10	0.5314	0.3951	0.0734				
GH		1*							4		
	1	1	1.37	1.47	0.6561	0.3078	0.0361				
	2	1	1.37	1.47	0.6561	0.3078	0.0361				
	3	1	1.37	1.47	0.6561	0.3078	0.0361				
GH		1*								4	
	1	1	1.76	1.89	0.8100	0.1800	0.0100				
	2	1	1.76	1.89	0.8100	0.1800	0.0100				
	3	1	1.76	1.89	0.8100	0.1800	0.0100				
GH		1									2
	1	1	2.02	2.16	0.8100	0.0900	0.0025				
	2	3	0.24	-0.06	0.0000	0.4500	0.0250				
	3	4	-0.21	-1.06	0.0000	0.0000	0.2500				
GH		4									3
	1	3	0.22	0.01	0.0000	0.0900	0.0075				
	2	3	0.15	0.01	0.0000	0.4500	0.0750				
	3	4	-0.21	0.01	0.0000	0.0000	0.7500				
GH		1*									4
	1	1	2.23	2.41	1.0000	0.0000	0.0000				
	2	1	2.23	2.41	1.0000	0.0000	0.0000				
	3	1	2.23	2.41	1.0000	0.0000	0.0000				

```
MACHINE MAINTENANCE & REPAIR          PAGE 14      TABLE  9.01

+------------------------------------------------------------+
|   ITERATION  9    MEM = 18   ESS MEM = 17   TIME =  5.96   |
|                                                            |
|   0.429 < G <   0.431       16  STEPS       TIME =  7.11   |
|   0.284 < H <   0.404       13  STEPS       TIME =  7.73   |
+------------------------------------------------------------+

RC  I  U  V(G)  V(H)  PROBS                      MEMORY STATES

G      1                                                     1
    1  1  1.75        0.8100  0.1800  0.0100
    2  3  0.19        0.0000  0.9000  0.1000
    3  4 -0.21        0.0000  0.0000  1.0000

GH     1                                                  1
    1  1  1.36 -0.84  0.6561  0.3078  0.0361
    2  3  0.10 -2.56  0.0000  0.8100  0.1900
    3  4 -0.21 -3.33  0.0000  0.0000  1.0000

GH     1                                               1
    1  1  1.05 -1.19  0.5314  0.3951  0.0734
    2  3  0.02 -2.62  0.0000  0.7290  0.2710
    3  4 -0.21 -3.22  0.0000  0.0000  1.0000

GH     1                                            1
    1  1  0.79 -1.49  0.4305  0.4513  0.1183
    2  3 -0.05 -2.66  0.0000  0.6561  0.3439
    3  4 -0.21 -3.09  0.0000  0.0000  1.0000

GH     2                                         1
    1  2  0.59 -1.74  0.3487  0.4836  0.1677
    2  3 -0.12 -2.69  0.0000  0.5905  0.4095
    3  4 -0.21 -2.97  0.0000  0.0000  1.0000

       1                                      1
    1  1  0.43        0.2824  0.4980  0.2195
    2  3 -0.18        0.0000  0.5314  0.4686
    3  4 -0.21        0.0000  0.0000  1.0000

GH     2*                                     4
    1  2  0.59 -1.74  0.3487  0.4836  0.1677
    2  2  0.59 -1.74  0.3487  0.4836  0.1677
    3  2  0.59 -1.74  0.3487  0.4836  0.1677

GH     1*                                        4
    1  1  0.79 -1.49  0.4305  0.4513  0.1183
    2  1  0.79 -1.49  0.4305  0.4513  0.1183
    3  1  0.79 -1.49  0.4305  0.4513  0.1183
```

GH		1*						4	1	1	1
	1	1	1.05	-1.19	0.5314	0.3951	0.0734				
	2	1	1.05	-1.19	0.5314	0.3951	0.0734				
	3	1	1.05	-1.19	0.5314	0.3951	0.0734				
GH		1*							4		
	1	1	1.36	-0.84	0.6561	0.3078	0.0361				
	2	1	1.36	-0.84	0.6561	0.3078	0.0361				
	3	1	1.36	-0.84	0.6561	0.3078	0.0361				
GH		1								2	
	1	1	1.58	-0.62	0.6561	0.2268	0.0196				
	2	3	0.14	-2.52	0.0000	0.4050	0.0700				
	3	4	-0.21	-3.43	0.0000	0.0000	0.2500				
GH		1*								4	
	1	1	1.75	-0.41	0.8100	0.1800	0.0100				
	2	1	1.75	-0.41	0.8100	0.1800	0.0100				
	3	1	1.75	-0.41	0.8100	0.1800	0.0100				
		1									2
	1	1	2.01		0.8100	0.0900	0.0025				
	2	3	0.24		0.0000	0.4500	0.0250				
	3	4	-0.21		0.0000	0.0000	0.2500				
GH		1								1	
	1	1	1.81	-0.40	0.6561	0.1539	0.0090				
	2	3	0.19	-2.46	0.0000	0.4050	0.0475				
	3	4	-0.21	-3.48	0.0000	0.0000	0.2500				
		4									3
	1	3	0.21		0.0000	0.0900	0.0075				
	2	3	0.15		0.0000	0.4500	0.0750				
	3	4	-0.21		0.0000	0.0000	0.7500				
GH		4								1	
	1	1	0.14	-2.30	0.0000	0.1539	0.0271				
	2	1	0.07	-2.30	0.0000	0.4050	0.1425				
	3	4	-0.21	-2.30	0.0000	0.0000	0.7500				
GH		1*									4
	1	1	2.22	0.10	1.0000	0.0000	0.0000				
	2	1	2.22	0.10	1.0000	0.0000	0.0000				
	3	1	2.22	0.10	1.0000	0.0000	0.0000				

```
MACHINE MAINTENANCE & REPAIR          PAGE 16      TABLE 10.01

+------------------------------------------------------------+
|   ITERATION 10    MEM = 23   ESS MEM = 21   TIME =  8.06   |
|                                                            |
|   0.430 < G <   0.431       12  STEPS       TIME =  9.12   |
|   0.250 < H <   0.355       14  STEPS       TIME =  9.27   |
+------------------------------------------------------------+

RC  I  U  V(G)  V(H)  PROBS                      MEMORY STATES

       1                                                  1  1
    1  1 -0.35        0.6561  0.3078  0.0361
    2  3 -1.60        0.0000  0.8100  0.1900
    3  4 -1.91        0.0000  0.0000  1.0000

       1                                               1
    1  1 -0.66        0.5314  0.3951  0.0734
    2  3 -1.69        0.0000  0.7290  0.2710
    3  4 -1.91        0.0000  0.0000  1.0000

       1                                            1
    1  1 -0.92        0.4305  0.4513  0.1183
    2  3 -1.76        0.0000  0.6561  0.3439
    3  4 -1.91        0.0000  0.0000  1.0000

       1                                         1
    1  1 -1.12        0.3487  0.4836  0.1677
    2  3 -1.82        0.0000  0.5905  0.4095
    3  4 -1.91        0.0000  0.0000  1.0000

GH     1                                      1
    1  1 -1.27  1.25  0.2824  0.4980  0.2195
    2  3 -1.88  0.26  0.0000  0.5314  0.4686
    3  4 -1.91 -0.37  0.0000  0.0000  1.0000

G      1*                                     4
    1  1 -1.12        0.3487  0.4836  0.1677
    2  1 -1.12        0.3487  0.4836  0.1677
    3  1 -1.12        0.3487  0.4836  0.1677

G      1*                                        4
    1  1 -0.92        0.4305  0.4513  0.1183
    2  1 -0.92        0.4305  0.4513  0.1183
    3  1 -0.92        0.4305  0.4513  0.1183

G      1*                                           4
    1  1 -0.66        0.5314  0.3951  0.0734
    2  1 -0.66        0.5314  0.3951  0.0734
    3  1 -0.66        0.5314  0.3951  0.0734
```

		1					2	1	1
	1	1	-0.48	0.5314	0.3222	0.0488			
	2	3	-1.65	0.0000	0.3645	0.1105			
	3	4	-1.91	0.0000	0.0000	0.2500			
G		1*					4		
	1	1	-0.35	0.6561	0.3078	0.0361			
	2	1	-0.35	0.6561	0.3078	0.0361			
	3	1	-0.35	0.6561	0.3078	0.0361			
		1						2	
	1	1	-0.13	0.6561	0.2268	0.0196			
	2	3	-1.56	0.0000	0.4050	0.0700			
	3	4	-1.91	0.0000	0.0000	0.2500			
		1					1		
	1	1	-0.30	0.5314	0.2566	0.0310			
	2	3	-1.61	0.0000	0.3645	0.0880			
	3	4	-1.91	0.0000	0.0000	0.2500			
		4						3	
	1	3	-1.58	0.0000	0.0810	0.0165			
	2	3	-1.64	0.0000	0.4050	0.1200			
	3	4	-1.91	0.0000	0.0000	0.7500			
G		1*						4	
	1	1	0.04	0.8100	0.1800	0.0100			
	2	1	0.04	0.8100	0.1800	0.0100			
	3	1	0.04	0.8100	0.1800	0.0100			
		1							2
	1	1	0.29	0.8100	0.0900	0.0025			
	2	3	-1.47	0.0000	0.4500	0.0250			
	3	4	-1.91	0.0000	0.0000	0.2500			
		1						1	
	1	1	0.09	0.6561	0.1539	0.0090			
	2	3	-1.52	0.0000	0.4050	0.0475			
	3	4	-1.91	0.0000	0.0000	0.2500			
		1					1		
	1	1	-0.10	0.5314	0.1976	0.0184			
	2	3	-1.57	0.0000	0.3645	0.0677			
	3	4	-1.91	0.0000	0.0000	0.2500			
		4							3
	1	3	-1.49	0.0000	0.0900	0.0075			
	2	3	-1.56	0.0000	0.4500	0.0750			
	3	4	-1.91	0.0000	0.0000	0.7500			

		4					1 3
	1	3	-1.56	0.0000	0.1539	0.0271	
	2	3	-1.67	0.0000	0.4050	0.1425	
	3	4	-1.91	0.0000	0.0000	0.7500	
		4					1
	1	3	-1.63	0.0000	0.1976	0.0551	
	2	1	-1.75	0.0000	0.3645	0.2032	
	3	4	-1.91	0.0000	0.0000	0.7500	
G		1*					4
	1	1	0.52	1.0000	0.0000	0.0000	
	2	1	0.52	1.0000	0.0000	0.0000	
	3	1	0.52	1.0000	0.0000	0.0000	

```
+------------------------------------------------------------------+
|   ITERATION 11    MEM = 25   ESS MEM = 23   TIME =   9.42        |
|                                                                  |
|   0.423 < G <   0.427        18  STEPS       TIME = 11.11        |
|   0.259 < H <   0.410        12  STEPS       TIME = 11.55        |
+------------------------------------------------------------------+

RC   I  U  V(G)  V(H)  PROBS                             MEMORY STATES

        1                                                        1  1
     1  1 -0.36         0.6561  0.3078  0.0361
     2  3 -1.60         0.0000  0.8100  0.1900
     3  4 -1.91         0.0000  0.0000  1.0000

GH      1                                                     1
     1  1 -0.68 -1.19   0.5314  0.3951  0.0734
     2  3 -1.69 -2.61   0.0000  0.7290  0.2710
     3  4 -1.91 -3.20   0.0000  0.0000  1.0000

GH      1                                                  1
     1  1 -0.94 -1.48   0.4305  0.4513  0.1183
     2  3 -1.76 -2.65   0.0000  0.6561  0.3439
     3  4 -1.91 -3.08   0.0000  0.0000  1.0000

GH      2                                               1
     1  2 -1.15 -1.73   0.3487  0.4836  0.1677
     2  3 -1.82 -2.67   0.0000  0.5905  0.4095
     3  4 -1.91 -2.96   0.0000  0.0000  1.0000

        2                                            1
     1  2 -1.32         0.2824  0.4980  0.2195
     2  3 -1.88         0.0000  0.5314  0.4686
     3  4 -1.91         0.0000  0.0000  1.0000

        4                                         1
     1  1 -1.46         0.2288  0.4991  0.2722
     2  4 -1.91         0.0000  0.4783  0.5217
     3  4 -1.91         0.0000  0.0000  1.0000

        2*                                        4
     1  2 -1.32         0.2824  0.4980  0.2195
     2  2 -1.32         0.2824  0.4980  0.2195
     3  2 -1.32         0.2824  0.4980  0.2195

GH      2*                                           4
     1  2 -1.15 -1.73   0.3487  0.4836  0.1677
     2  2 -1.15 -1.73   0.3487  0.4836  0.1677
     3  2 -1.15 -1.73   0.3487  0.4836  0.1677
```

```
GH      1*                                              4   1   1   1   1
     1  1 -0.94 -1.48  0.4305  0.4513  0.1183
     2  1 -0.94 -1.48  0.4305  0.4513  0.1183
     3  1 -0.94 -1.48  0.4305  0.4513  0.1183

GH      1*                                                  4
     1  1 -0.68 -1.19  0.5314  0.3951  0.0734
     2  1 -0.68 -1.19  0.5314  0.3951  0.0734
     3  1 -0.68 -1.19  0.5314  0.3951  0.0734

GH      1                                                       2
     1  1 -0.50 -1.01  0.5314  0.3222  0.0488
     2  3 -1.65 -2.58  0.0000  0.3645  0.1105
     3  4 -1.91 -3.30  0.0000  0.0000  0.2500

GH      1*                                                      4
     1  1 -0.36 -0.83  0.6561  0.3078  0.0361
     2  1 -0.36 -0.83  0.6561  0.3078  0.0361
     3  1 -0.36 -0.83  0.6561  0.3078  0.0361

        1                                                           2
     1  1 -0.14        0.6561  0.2268  0.0196
     2  3 -1.56        0.0000  0.4050  0.0700
     3  4 -1.91        0.0000  0.0000  0.2500

GH      1                                                       1
     1  1 -0.30 -0.82  0.5314  0.2566  0.0310
     2  3 -1.61 -2.54  0.0000  0.3645  0.0880
     3  4 -1.91 -3.38  0.0000  0.0000  0.2500

G       4                                                           3
     1  3 -1.58        0.0000  0.0810  0.0165
     2  3 -1.64        0.0000  0.4050  0.1200
     3  4 -1.91        0.0000  0.0000  0.7500

GH      1*                                                          4
     1  1  0.03 -0.41  0.8100  0.1800  0.0100
     2  1  0.03 -0.41  0.8100  0.1800  0.0100
     3  1  0.03 -0.41  0.8100  0.1800  0.0100

        1                                                               2
     1  1  0.29        0.8100  0.0900  0.0025
     2  3 -1.47        0.0000  0.4500  0.0250
     3  4 -1.91        0.0000  0.0000  0.2500

        1                                                           1
     1  1  0.09        0.6561  0.1539  0.0090
     2  3 -1.52        0.0000  0.4050  0.0475
     3  4 -1.91        0.0000  0.0000  0.2500
```

GH		1							1	1	2
	1	1	-0.10	-0.62	0.5314	0.1976	0.0184				
	2	3	-1.57	-2.48	0.0000	0.3645	0.0677				
	3	4	-1.91	-3.41	0.0000	0.0000	0.2500				
		4									3
	1	3	-1.49		0.0000	0.0900	0.0075				
	2	3	-1.56		0.0000	0.4500	0.0750				
	3	4	-1.91		0.0000	0.0000	0.7500				
		4								1	
	1	3	-1.56		0.0000	0.1539	0.0271				
	2	3	-1.68		0.0000	0.4050	0.1425				
	3	4	-1.91		0.0000	0.0000	0.7500				
GH		4							1		
	1	3	-1.63	-2.31	0.0000	0.1976	0.0551				
	2	1	-1.74	-2.31	0.0000	0.3645	0.2032				
	3	4	-1.91	-2.31	0.0000	0.0000	0.7500				
GH		1*									4
	1	1	0.51	0.10	1.0000	0.0000	0.0000				
	2	1	0.51	0.10	1.0000	0.0000	0.0000				
	3	1	0.51	0.10	1.0000	0.0000	0.0000				

```
+--------------------------------------------------------------+
|   ITERATION 12    MEM = 31   ESS MEM = 29   TIME = 11.94     |
|                                                              |
|   0.420 < G <   0.425       18  STEPS       TIME = 14.10     |
|   0.317 < H <   C.409       12  STEPS       TIME = 14.38     |
+--------------------------------------------------------------+

RC  I  U  V(G)  V(H)  PRCBS                       MEMORY STATES

       1                                                   1  1
    1  1 -0.37         0.6561  0.3078  C.0361
    2  3 -1.60         0.C000  0.8100  0.1900
    3  4 -1.91         0.C000  0.000C  1.0000

       1                                                1
    1  1 -0.69         0.5314  0.3951  0.0734
    2  3 -1.69         0.C000  0.729C  0.2710
    3  4 -1.91         C.C000  0.0000  1.0000

       1                                             1
    1  1 -0.95         0.4305  0.4513  0.1183
    2  3 -1.76         0.C000  0.6561  0.3439
    3  4 -1.91         0.C000  0.0000  1.0000

       1                                          1
    1  1 -1.16         0.3487  0.4836  0.1677
    2  3 -1.82         0.C000  0.5905  0.4095
    3  4 -1.91         C.C000  0.000C  1.0000

       1                                       1
    1  1 -1.32         0.2824  0.498C  0.2195
    2  3 -1.88         0.C000  0.5314  0.4686
    3  4 -1.91         0.C000  0.0000  1.0000

GH     4                                    1
    1  1 -1.45 -2.29   0.2288  0.4991  0.2722
    2  4 -1.91 -2.29   0.0000  0.4783  0.5217
    3  4 -1.91 -2.29   0.C000  0.000C  1.0000

GH     1*                                   4
    1  1 -1.32 -2.14   0.2824  0.498C  0.2195
    2  1 -1.32 -2.14   0.2824  0.4980  0.2195
    3  1 -1.32 -2.14   0.2824  0.498C  0.2195

GH     1*                                      4
    1  1 -1.16 -1.96   0.3487  0.4836  0.1677
    2  1 -1.16 -1.96   0.3487  0.4836  0.1677
    3  1 -1.16 -1.96   0.3487  0.4836  0.1677
```

GH		1*						4	1	1	1	1
	1	1	-0.95	-1.73	0.4305	0.4513	0.1183					
	2	1	-0.95	-1.73	0.4305	0.4513	0.1183					
	3	1	-0.95	-1.73	0.4305	0.4513	0.1183					
		1							2			
	1	1	-0.80		0.4305	0.3857	0.0864					
	2	3	-1.72		0.0000	0.3280	0.1469					
	3	4	-1.91		0.0000	0.0000	0.2500					
GH		1*							4			
	1	1	-0.69	-1.41	0.5314	0.3951	0.0734					
	2	1	-0.69	-1.41	0.5314	0.3951	0.0734					
	3	1	-0.69	-1.41	0.5314	0.3951	0.0734					
		1								2		
	1	1	-0.51		0.5314	0.3222	0.0488					
	2	3	-1.65		0.0000	0.3645	0.1105					
	3	4	-1.91		0.0000	0.0000	0.2500					
		1							1			
	1	1	-0.64		0.4305	0.3266	0.0620					
	2	3	-1.69		0.0000	0.3280	0.1244					
	3	4	-1.91		0.0000	0.0000	0.2500					
GH		1*								4		
	1	1	-0.37	-0.99	0.6561	0.3078	0.0361					
	2	1	-0.37	-0.99	0.6561	0.3078	0.0361					
	3	1	-0.37	-0.99	0.6561	0.3078	0.0361					
		1									2	
	1	1	-0.15		0.6561	0.2268	0.0196					
	2	3	-1.56		0.0000	0.4050	0.0700					
	3	4	-1.91		0.0000	0.0000	0.2500					
		1								1		
	1	1	-0.32		0.5314	0.2566	0.0310					
	2	3	-1.61		0.0000	0.3645	0.0880					
	3	4	-1.91		0.0000	0.0000	0.2500					
		1							1			
	1	1	-0.47		0.4305	0.2735	0.0434					
	2	3	-1.66		0.0000	0.3280	0.1042					
	3	4	-1.91		0.0000	0.0000	0.2500					
		4									3	
	1	3	-1.58		0.0000	0.0810	0.0165					
	2	3	-1.64		0.0000	0.4050	0.1200					
	3	4	-1.91		0.0000	0.0000	0.7500					

		4							1	3	1
	1	3	-1.65		0.0000	0.1385	0.0425				
	2	1	-1.71		0.0000	0.3645	0.1830				
	3	4	-1.91		0.0000	0.0000	0.7500				
GH		1*								4	
	1	1	0.03	-0.49	0.8100	0.1800	0.0100				
	2	1	0.03	-0.49	0.8100	0.1800	0.0100				
	3	1	0.03	-0.49	0.8100	0.1800	0.0100				
		1									2
	1	1	0.28		0.8100	0.0900	0.0025				
	2	3	-1.47		0.0000	0.4500	0.0250				
	3	4	-1.91		0.0000	0.0000	0.2500				
		1								1	
	1	1	0.07		0.6561	0.1539	0.0090				
	2	3	-1.52		0.0000	0.4050	0.0475				
	3	4	-1.91		0.0000	0.0000	0.2500				
		1							1		
	1	1	-0.12		0.5314	0.1976	0.0184				
	2	3	-1.57		0.0000	0.3645	0.0677				
	3	4	-1.91		0.0000	0.0000	0.2500				
		1						1			
	1	1	-0.30		0.4305	0.2256	0.0296				
	2	3	-1.62		0.0000	0.3280	0.0860				
	3	4	-1.91		0.0000	0.0000	0.2500				
		4									3
	1	3	-1.49		0.0000	0.0900	0.0075				
	2	3	-1.56		0.0000	0.4500	0.0750				
	3	4	-1.91		0.0000	0.0000	0.7500				
		4								1	
	1	3	-1.56		0.0000	0.1539	0.0271				
	2	3	-1.67		0.0000	0.4050	0.1425				
	3	4	-1.91		0.0000	0.0000	0.7500				
		4							1		
	1	3	-1.63		0.0000	0.1976	0.0551				
	2	1	-1.76		0.0000	0.3645	0.2032				
	3	4	-1.91		0.0000	0.0000	0.7500				
		4						1			
	1	3	-1.70		0.0000	0.2256	0.0887				
	2	1	-1.80		0.0000	0.3280	0.2579				
	3	4	-1.91		0.0000	0.0000	0.7500				

GH 1* 4

1	1	0.51	0.08	1.0000	0.0000	0.0000
2	1	0.51	0.08	1.0000	0.0000	0.0000
3	1	0.51	0.08	1.0000	0.0000	0.0000

```
MACHINE MAINTENANCE & REPAIR          PAGE 26      TABLE 13.01

+-------------------------------------------------------------+
|   ITERATION 13    MEM = 33   ESS MEM = 31   TIME = 14.58    |
|                                                             |
|   0.421 < G <   0.423       17  STEPS       TIME = 16.61    |
|   0.421 < H <   0.423        6  STEPS       TIME = 16.84    |
+-------------------------------------------------------------+

RC  I  U  V(G)  V(H)  PROBS                      MEMORY STATES

       1                                                  1  1
    1  1 -0.37        0.6561  0.3078  0.0361
    2  3 -1.60        0.0000  0.8100  0.1900
    3  4 -1.91        0.0000  0.0000  1.0000

       1                                               1
    1  1 -0.70        0.5314  0.3951  0.0734
    2  3 -1.69        0.0000  0.7290  0.2710
    3  4 -1.91        0.0000  0.0000  1.0000

       1                                            1
    1  1 -0.96        0.4305  0.4513  0.1183
    2  3 -1.76        0.0000  0.6561  0.3439
    3  4 -1.91        0.0000  0.0000  1.0000

       1                                         1
    1  1 -1.17        0.3487  0.4836  0.1677
    2  3 -1.82        0.0000  0.5905  0.4095
    3  4 -1.91        0.0000  0.0000  1.0000

       1                                      1
    1  1 -1.33        0.2824  0.4980  0.2195
    2  3 -1.88        0.0000  0.5314  0.4686
    3  4 -1.91        0.0000  0.0000  1.0000

       4                                   1
    1  3 -1.46        0.2288  0.4991  0.2722
    2  4 -1.91        0.0000  0.4783  0.5217
    3  4 -1.91        0.0000  0.0000  1.0000

       4                                1
    1  3 -1.55        0.1853  0.4903  0.3244
    2  4 -1.91        0.0000  0.4305  0.5695
    3  4 -1.91        0.0000  0.0000  1.0000

GH     3*                               4
    1  3 -1.46 -1.90  0.2288  0.4991  0.2722
    2  3 -1.46 -1.90  0.2288  0.4991  0.2722
    3  3 -1.46 -1.90  0.2288  0.4991  0.2722
```

GH		1*						4	1	1	1	1	1	1
	1	1	-1.33	-1.77	0.2824	0.4980	0.2195							
	2	1	-1.33	-1.77	0.2824	0.4980	0.2195							
	3	1	-1.33	-1.77	0.2824	0.4980	0.2195							
GH		1*							4					
	1	1	-1.17	-1.61	0.3487	0.4836	0.1677							
	2	1	-1.17	-1.61	0.3487	0.4836	0.1677							
	3	1	-1.17	-1.61	0.3487	0.4836	0.1677							
GH		1*								4				
	1	1	-0.96	-1.40	0.4305	0.4513	0.1183							
	2	1	-0.96	-1.40	0.4305	0.4513	0.1183							
	3	1	-0.96	-1.40	0.4305	0.4513	0.1183							
		1									2			
	1	1	-0.81		0.4305	0.3857	0.0864							
	2	3	-1.72		0.0000	0.3280	0.1469							
	3	4	-1.91		0.0000	0.0000	0.2500							
GH		1*									4			
	1	1	-0.70	-1.13	0.5314	0.3951	0.0734							
	2	1	-0.70	-1.13	0.5314	0.3951	0.0734							
	3	1	-0.70	-1.13	0.5314	0.3951	0.0734							
		1										2		
	1	1	-0.51		0.5314	0.3222	0.0488							
	2	3	-1.65		0.0000	0.3645	0.1105							
	3	4	-1.91		0.0000	0.0000	0.2500							
		1									1			
	1	1	-0.65		0.4305	0.3266	0.0620							
	2	3	-1.69		0.0000	0.3280	0.1244							
	3	4	-1.91		0.0000	0.0000	0.2500							
GH		1*										4		
	1	1	-0.37	-0.81	0.6561	0.3078	0.0361							
	2	1	-0.37	-0.81	0.6561	0.3078	0.0361							
	3	1	-0.37	-0.81	0.6561	0.3078	0.0361							
		1											2	
	1	1	-0.16		0.6561	0.2268	0.0196							
	2	3	-1.56		0.0000	0.4050	0.0700							
	3	4	-1.91		0.0000	0.0000	0.2500							
		1										1		
	1	1	-0.32		0.5314	0.2566	0.0310							
	2	3	-1.61		0.0000	0.3645	0.0880							
	3	4	-1.91		0.0000	0.0000	0.2500							

		1						1	1	2	1
	1	1	-0.48		0.4305	0.2735	0.0434				
	2	3	-1.66		0.0000	0.3280	0.1042				
	3	4	-1.91		0.0000	0.0000	0.2500				
		4								3	
	1	3	-1.58		0.0000	0.0810	0.0165				
	2	3	-1.64		0.0000	0.4050	0.1200				
	3	4	-1.91		0.0000	0.0000	0.7500				
		4							1		
	1	3	-1.65		0.0000	0.1385	0.0425				
	2	1	-1.71		0.0000	0.3645	0.1830				
	3	4	-1.91		0.0000	0.0000	0.7500				
GH		1*								4	
	1	1	0.03	-0.41	0.8100	0.1800	0.0100				
	2	1	0.03	-0.41	0.8100	0.1800	0.0100				
	3	1	0.03	-0.41	0.8100	0.1800	0.0100				
		1									2
	1	1	0.28		0.8100	0.0900	0.0025				
	2	3	-1.47		0.0000	0.4500	0.0250				
	3	4	-1.91		0.0000	0.0000	0.2500				
		1								1	
	1	1	0.07		0.6561	0.1539	0.0090				
	2	3	-1.52		0.0000	0.4050	0.0475				
	3	4	-1.91		0.0000	0.0000	0.2500				
		1							1		
	1	1	-0.13		0.5314	0.1976	0.0184				
	2	3	-1.57		0.0000	0.3645	0.0677				
	3	4	-1.91		0.0000	0.0000	0.2500				
		1						1			
	1	1	-0.30		0.4305	0.2256	0.0296				
	2	3	-1.62		0.0000	0.3280	0.0860				
	3	4	-1.91		0.0000	0.0000	0.2500				
		4									3
	1	3	-1.49		0.0000	0.0900	0.0075				
	2	3	-1.56		0.0000	0.4500	0.0750				
	3	4	-1.91		0.0000	0.0000	0.7500				
		4								1	
	1	3	-1.56		0.0000	0.1539	0.0271				
	2	3	-1.67		0.0000	0.4050	0.1425				
	3	4	-1.91		0.0000	0.0000	0.7500				

		4							1	1	3
	1	3	-1.63		0.0000	0.1976	0.0551				
	2	1	-1.76		0.0000	0.3645	0.2032				
	3	4	-1.91		0.0000	0.0000	0.7500				
		4						1			
	1	3	-1.70		0.0000	0.2256	0.0887				
	2	1	-1.80		0.0000	0.3280	0.2579				
	3	4	-1.91		0.0000	0.0000	0.7500				
GH		1*									4
	1	1	0.51	0.07	1.0000	0.0000	0.0000				
	2	1	0.51	0.07	1.0000	0.0000	0.0000				
	3	1	0.51	0.07	1.0000	0.0000	0.0000				

b. A Computer Communication Problem

The problem to be considered in this subsection concerns several units sharing a single communication channel. If any two units attempt to transmit messages simultaneously, both will fail. As the units have no means (other than the channel itself) of coordinating their efforts, the decision to transmit is made on the basis of imperfect information. A system of this type has been used to link remote terminals to a central computer at the University of Hawaii; because this system is called the ALOHA system, the problem has become known as the slotted ALOHA problem. A more familiar example of this problem is that faced by a newsman attempting to address the President of the United States at a news conference; if he asks a question while another newsman is doing the same, neither will be recognized.

The slotted ALOHA problem has been considered by Kleinrock and Lam [1975], Lam and Kleinrock [1975], and others cited in the first reference. Although the problem has been extensively studied under the assumption that the number of units seeking to transmit is known (to all units), no work known to this author considers the "dual control" aspect of the problem (characterized by the fact that clashes are useful in identifying the number of units seeking to transmit). The formulation to be considered here limits the number of units, but recognizes the "dual control" aspect of the problem. Moreover, previous work resulted in strategies sufficiently complex to preclude evaluation, even by simulation. In the present analysis, the system under an adapted feasible strategy is a Markov chain having state set SxM; exact evaluation of the controller performance is therefore possible.

In the model to be considered here, there are four units, each of which may be in idle or retransmit mode. During each time interval, a message originates at an idle unit with probability .1. The unit always attempts to broadcast a newly-originated message. The three outputs are:

$$Y = \left\{\begin{array}{l}\text{No transmissions attempted}\\ \text{One successful transmission}\\ \text{Multiple transmissions attempted}\end{array}\right\}.$$

A unit that has unsucessfully attempted to transmit subsequently enters retransmit mode. It then selects an input

$$U = \left\{\begin{array}{l}\text{Retransmit with probability .2}\\ \text{Retransmit with probability .9}\end{array}\right\}$$

Since the system, as viewed by a unit in retransmission mode, is symmetric, all units select the same input on the basis of the same input-output history. There results an FPS formulation having 5 states (corresponding to the number of units in retransmit mode), 2 inputs, and 3 outputs. The FPS is reachable and detectable. The performance measure is throughput, i.e. the average number of messages successfully transmitted per unit time.

The following results were obtained in four iterations:

h lb	h ub	g lb	g ub	essential memory	effectiveness	time (secs)
.302	.354	.330	.372	1	$\geq$ 91.2%	.39
.309	.331	.332	.336	6	$\geq$ 93.3%	1.54
.313	.329	.331	.332	26	$\geq$ 94.6%	5.46
.312	.330	.330	.331	98	$\geq$ 94.3%	23.49

"Effectiveness" was computed by comparing the lower bound on h with the final upper bound on feasible performance, .331.

These results indicate that memory is not very useful for purposes of decision-making in this problem, i.e. that the performance that may be achieved on the basis of the most recent input-output pair alone (iteration 2) is comparable to that which may be achieved on the basis of an infinite past history. This might be attributed to the small number of units involved; it is possible that a similar computation with a larger number of units might yield entirely different results.

CHAPTER V

CONCLUSIONS

The mathematical technique of dynamic programming assigns to each state a value representing the expected rewards accrued when the system is initiated in that state. A decision-maker uses these values to compare immediate rewards with potential benefits if the system is made to enter a desirable state.

Problems of decision-making under state uncertainty may, in principle, be solved by dynamic programming, if the state of information is itself considered to be a state. It may, however, be practically infeasible to assign a value to each state of information, when the number of possible states of information is sufficiently large.

The mathematical technique of perceptive dynamic programming assigns a value to certain information that might be acquired at a cost. These values may be used to compare performance achievable on the basis of existing knowledge with potential benefits if further information is sought.

In this report, perceptive dynamic programming has been developed in the context of control of finite probabilistic systems over an infinite horizon. The system is assumed to be <u>reachable</u>, so that performance will not depend on the initial state, and <u>detectable</u>, so that performance will not depend on the initial state of information. Specifically, reachability assures that the most desirable state can be reached from any other state; hence the gain achievable when the system is initiated in the most desirable state can be replicated when the system starts in any other state. Detectability assures that the information vector may be arbitrarily closely approximated on the basis of a sufficiently long string of most recent input-output pairs; hence, whatever information was available initially is irrelevant in the steady-state. Reachability and detectability also imply that a performance arbitrarily close to the supremum feasible performance may be achieved by a finite-memory controller having a sufficiently large memory set.

Reachability and detectability have many implications in FPS's that are similar to well-known properties of finite-dimensional linear systems (FDLS). For example, detectability in a FDLS implies that the observer state may be arbitrarily closely (in some suitable sense) approximated on the basis of a sufficiently long string of most recent input-output pairs. The analogous result for FPS's is given in Section 14. Moreover, any FDLS that is initiated in state zero may be expressed in a form that is controllable and observable. The assumption that a FDLS is initiated in state zero is equivalent to the assumption that it has experienced an infinite past under a stablizing control. Similarly, any FPS that has experienced an infinite past under an appropriate decision strategy may be expressed in a form that is reachable and detectable.

An algorithm for the solution of FPS control problems was implemented on a digital computer, and two simple problems were "solved" to demonstrate the efficacy of the method. It appears that more realistic (and hence more complex) problems might be solved in the same manner, but it would then be necessary that the computer implementation be problem-specific.

Possible extentions of the theory which would be beneficial in extending its applicability include the following:

1) The recursive computation of memory sets (described in Section 21d) could be explicitely optimized (e.g. by means of a branch-and-bound intepretation).

2) The computational efficiency of pseudo-perceptive dynamic programming (described in Section 21b) might be compared with that of perceptive dynamic programming. It is clear that pseudo-perceptive dynamic programming converges less rapidly than does perceptive dynamic programming, but the former requires less memory and less time to complete an iteration.

3) Perceptive dynamic programming is most effective when the index of detectability, $\bar{\alpha}$, lies near zero. In order for this to occur, outputs need not yield good reliable state information; they simply must preclude the possibility of better information being acquired from less recent input-output pairs. Thus the notion of detectability is useful in determining whether a given problem may be solved numerically. If the problem cannot be solved, then the notion of detectability might be useful in

suggesting a different observation structure, one that is more conducive to solution. In particular, the following problem might be posed: Determine outputs for a given underlying process such that, when perceptive dynamic programming is performed up to a maximum allowable memory size, feasible performance is maximized. An output that happens to equal the optimal input given the state would, of course, solve this problem.

4) The notions of reachability and detectability might be extended to systems having a large state set and a great deal of structure (e.g. routing in a network of queues). This could lead to effective rules for decision-making on the basis of imperfect state information when consideration of the exact state is physically feasible, but precluded on grounds of complexity.

5) Notions of cross-reachability and cross-detectability might be defined in decentralized systems, to indicate the extent to which various decision-makers need to coordinate their efforts.

BIBLIOGRAPHY

ASTROM, K.J. [1965],
"Optimal Control of Markov Processes with Incomplete State Information," _J. Math. Anal. Appl. 10_, pp. 174-205.

ASTROM, K.J. [1969],
"Optimal Control of Markov Processes with Incomplete State Information II: The Convexity of the Loss Function," _J. Math. Anal. Appl. 26_, pp. 403-406.

BATHER, J. [1973a],
"Optimal Decision Procedures for Finite Markov Chains. Part I: Examples," _Adv. Appl. Prob. 5_, pp. 328-339.

BATHER, J. [1973b],
"Optimal Decision Procedures for Finite Markov Chains. Part II: Communicating Systems," _Adv. Appl. Prob. 5_, pp. 521-540.

BATHER, J. [1973c],
"Optimal Decision Procedures for Finite Markov Chains. Part III: General Convex Systems," _Adv. Appl. Prob. 5_, pp. 541-553.

BELLMAN, R. [1957a],
"A Markovian Decision Process," _J. Math. and Mech. 6_, pp. 679-684.

BELLMAN, R. [1957b],
Dynamic Programming, Princeton University Press, Princeton, N.J.

BERTSEKAS, D. [1976],
Dynamic Programming and Stochastic Control, Academic, New York.

BROOKS, D.M. and LEONDES, C.T. [1973],
"Markovian Decision Processes with State-Information Lag," _Opns. Res. 21_, pp. 904-907.

BROWN, B. [1965],
"On the Iterative Method of Dynamic Programming on a Finite State Discrete Time Process," _Ann. Math. Stat. 36_, pp. 1279-1285.

CERNY, J. [1970],
"Approximation in the Space of Information Channels," _Information and Control 16_, pp. 384-395.

CHANDRESEKARIN, B. [1970],
"Finite Memory Hypothesis Testing--A Critique," _IEEE Trans. Inform. Theory_, Vol. IT-16, No. 4, pp. 496-497.

CHANDRESEKARIN, B. [1971],
"Reply to Finite Memory Hypothesis Testing--Comments on a Critique," IEEE Trans. Inform. Theory, Vol. IT-17, No. 1, pp. 104-105.

CHANDRESEKARIN, B. and LAM, C.C. [1975],
"A Finite-Memory Deterministic Algorithm for the Symmetric Hypothesis Testing Problem," IEEE Trans. Inform. Theory, Vol. IT-21, No. 1, pp. 40-44.

COVER, T.M. and HELLMAN, M.E. [1970a],
"The Two-Armed-Bandit Problem with Time-Invariant Finite Memory," IEEE Trans. Inform. Theory, Vol. IT-16, No. 2, pp. 185-195.

COVER, T.M. and HELLMAN, M.E. [1970b],
"Finite Memory Hypothesis Testing--Comments on a Critique," IEEE Trans. Inform. Theory, Vol. IT-16, No. 4, pp. 496-497.

COVER, T.M., FREEDMAN, M.Z. and HELLMAN, M.E. [1976],
"Optimal Finite-Memory Learning Algorithms for the Finite Sample Problem," Information and Control 30, pp. 49-85.

DeGROOT, M.H. [1970],
Optimal Statistical Decisions, McGraw-Hill, N.Y.

DERMAN, C. [1970],
Finite State Markovian Decision Processes, Academic, N.Y.

DRAKE, A.W. [1962],
"Observation of a Markov Process through a Noisy Channel," Sc.D. Thesis, Department of Electrical Engineering, M.I.T., Cambridge, MA.

DRAKE, A.W. [1968],
"Partially Observable Markov Models for Quality Control," Twenty-Second Technical Conference Transactions of the American Society for Quality Control, Philadelphia, PA, pp. 199-205.

DEVORE, J.L. [1974],
"A Note on the Observation of a Markov Source Through a Noisy Channel," IEEE Trans. Inform. Theory, Vol. IT-20, pp. 762-764.

FLYNN, J. [1974],
"Averaging vs. Discounting in Dynamic Programming: A Counterexample," Ann. Statist. 2, pp. 411-413.

HASTINGS, N.A.J. [1973],
Dynamic Programming with Management Applications, Butterworths, London and Crane-Russak, N.Y.

HASTINGS, N.A.J. [1976],
"A Test for Nonoptimal Actions in Undiscounted Markov Programming," Management Science 23, pp. 87-92.

HELLMAN, M.E. and COVER, T.M. [1970],
"Learning with Finite Memory," Ann. Math. Stat. 41, pp. 765-782.

HOWARD, R.A. [1960],
Dynamic Programming and Markov Processes, MIT Press, Cambridge, MA.

HOWARD, R.A. [1971],
Dynamic Probabilistic Systems, Vols. I and II, Wiley, N.Y.

KAKALIK, J.S. [1965],
"Optimum Policies for Partially Observable Markov Systems," MIT M.S. Thesis; also reported in MIT Operations Research Center Technical Report No. 18.

KALMAN, R.E., FALB, P.L. and ARBIB, M.A. [1969],
Topics in Mathematical System Theory, McGraw-Hill, N.Y.

KAJSER, T. [1975],
"A Limit Theorem for Partially-Observed Markov Chains," Ann. Prob. 3, pp. 677-696.

KLEINROCK, L. and LAM, S.S. [1975],
"Packet Switching in a Multiaccess Broadcast Channel: Performance Evaluation," IEEE Trans. Commun., Vol. COM-23, pp. 410-423.

KUSHNER, H.J. [1971],
Introduction to Stochastic Control, Holt, Rinehart and Winston, N.Y.

LAM, S.S. and KLEINROCK, L. [1975],
"Packet Switching in a Multiaccess Broadcast Channel: Dynamic Control Procedures, IEEE Trans. Commun., Vol. COM-23, pp. 891-904.

LANERY, E. [1967],
"Etude Asymptotique des Systèmes Markoviens à Commande," Revue Français D'Informatique et de Recherche Operationelle 1, pp. 3-56.

LANERY, E. [1968],
"Complements à l'Etude Asymptotique des Systèmes Markoviens à Commande," Institut de Recherche D'Informatique et D'Automatique, Rocquencourt, France.

LUENBERGER, D. [1969],
Optimization by Vector Space Methods, Wiley, New York.

MacQUEEN, J.B. [1966],
"A Modified Dynamic Programming Method for Markovian Decision Problems," J. Math. Anal. Appl. 14, pp. 38-43.

MINE, H. and OSAKI, S. [1970],
Markovian Decision Processes, Academic, N.Y.

ODONI, A.R. [1967],
"Alternative Schemes for Investigating Markov Decision Processes," M.S. Thesis, Department of Electrical Engineering; also reported in M.I.T. Operations Research Center Technical Report #28.

ODONI, A.R. [1969],
"On Finding the Maximal Gain for Markov Decision Processes," Opns. Res. 17, pp. 857-860.

PAZ, A. [1971],
Introduction to Probabilistic Automata, Academic, N.Y.

PLATZMAN, L.K. [1977],
"Improved Conditions for Convergence in Undiscounted Markov Renewal Programming," Opns. Res. 25.

ROCKAFELLAR, R.T. [1970],
Convex Analysis, Princeton University Press, Princeton, N.J.

ROSS, S.M. [1970],
Applied Probability Models with Optimization Applications, Holden-Day, San Francisco.

SATIA, J.K. and LAVE, R.E. [1973],
"Markovian Decision Processes with Probabilistic Observation of States," Mgmt. Sci. 20, pp. 1-13.

SAWARAGI, Y. and YOSHIKAWA, T. [1970],
"Discrete-Time Markovian Decision Processes with Incomplete State Observations," Ann. Math. Stat. 41, pp. 78-86.

SCHWEITZER, P.J. [1971],
"Iterative Solution of the Functional Equations of Undiscounted Markov Renewal Programming," J. Math. Anal. Appl. 34, pp. 495-501.

SCHWEITZER, P.J. [1973],
"Annotated Bibliography on Markov Decision Processes," Unpublished.

SCHWEITZER, P.J. and FEDERGRUEN, A. [1977?],
"The Asymptotic Behavior of Undiscounted Value Interation in Markov Decision Problems," to appear.

SMALLWOOD, R.D. and SONDIK, E.J. [1973],
"The Optimal Control of Partially Observable Markov Processes over a Finite Horizon," Opns. Res. 21, pp. 1071-1081.

SONDIK, E.J. [1971],
"The Optimal Control of Partially-Observable Markov Processes," Stanford Ph.D. Thesis; also reported in Stanford Information Systems Laboratory Technical Report No. 6252-4.

SULMAR, J.J. [1974],
"Observation of a Markov Source Through a Noisy Channel," S.B. Thesis, Department of Electrical Engineering, M.I.T., Cambridge, MA.

VON NEUMAN, J. and MORGENSTERN, O. [1947],
Theory of Games and Economic Behavior, Princeton University Press, Princeton, N.J.

WALD, A. [1950],
Statistical Decision Functions, Wiley, N.Y.

WHITE, C.C. [1976],
"Procedures for the Solution of a Finite-Horizon, Partially-Observed, Semi-Markov Optimization Problem," Opns. Res. 24, pp. 348-358.

WHITE, D.J. [1963],
"Dynamic Programming, Markov Chains, and the Method of Successive Approximations," J. Math. Anal. Appl. 6, pp. 373-376.

APPENDIX A

Proof of Theorem 19.3

a. Preliminaries

V has been defined, in (12.11), as the vector space of bounded, continuous, real-valued functions on Π_N. V, along with the sup norm $\|\cdot\|$, is a Banach space. It will be shown that the sequence $\{\hat{v}^m\}$, given by (19.6) or (19.7), is bounded, that it has a subsequence that converges (pointwise) to a convex function v^*, that the subsequence is Cauchy - implying $v^* \varepsilon V$, and finally that v^* satisfies (19.1). A corollary states that $\{\hat{v}^m\}$ itself is Cauchy in V, i.e. that $\hat{v}^m$ converges uniformly to v^*.

Since it cannot be shown immediately that v^* is continuous, $\{\hat{v}^m\}$ will be treated as a sequence in W, the vector space of Lebesque measurable functions on Π_N. If $v \varepsilon W$, then $\|v\|$ denotes the ess sup norm of w. Naturally $V \subset W$.

By abuse of notation, a constant (such as Q or g^*) may denote an element of V that is a constant function over Π_N. Following (17.3), $v \varepsilon W$ may be interpreted as a function on $\tilde{\Pi}_N$:

$$v \text{ is "convex" (over } \Pi_N)$$

$$\Longleftrightarrow \quad v(\tilde{\pi}) + v(\tilde{\pi}') \geq v(\tilde{\pi} + \tilde{\pi}'),$$

$$\forall\ \tilde{\pi}, \tilde{\pi}', \tilde{\pi} + \tilde{\pi}' \varepsilon \tilde{\Pi}_N . \tag{A.1}$$

W is partially ordered by "$\leq$" where:

$$v \leq v' \iff v(\pi) \leq v(\pi) \quad \forall \pi \varepsilon \Pi_N \tag{A.2}$$

It will also be necessary to consider the restriction of $v\varepsilon W$ to particular subsets of Π_N that include the range of $T(\cdot,\underline{z})$ when $P(\underline{z})$ is subrectangular. Define:

$$b(\ell) = \min\{T_j(e^i,\underline{z}) : P_{ij}(\underline{z}) > 0, \; P(\underline{z}) \text{ is subrectangular, and } \underline{z}\varepsilon Z^\ell\} \tag{A.3}$$

$$\Pi_N(b(\ell)) = \{\pi\varepsilon\Pi_N : \text{either } \pi_i = 0 \text{ or } \pi_i \geq b(\ell), \; \forall i\varepsilon S\} \tag{A.4}$$

$$\| v \|_{b(\ell)} = \sup_{\pi\varepsilon\Pi_N(b(\ell))}\{v(\pi)\} \tag{A.5}$$

b. A Transformation in W

(A.6) Definition. $f : W \to W$ is defined by:

$$fv(\pi) = \max_{u\varepsilon U} \{\pi q(u) + \beta\Sigma_{y\varepsilon Y} \, v(\pi P(y|u))\}$$

Interpretation: f is the operator of backward inductive dynamic programming.

Remark: Eq (19.1) may now be expressed as $v^* = fv^* - g^*$.

Transformation f has the following properties:

(A.7) Lemma. $v \leq v' \implies fv \leq fv'$

(A.8) Lemma. $f(v + C) = fv + \beta C$, where C is a constant.

(A.9) Proposition. f is continuous in sup norm; in particular, $\| fv - fv' \| \leq \beta \| v - v' \|$.

Proof: $fv \leq f(v' + \| v - v' \|) = fv' + \beta \| v - v' \|$ and similarly $fv \geq fv' - \beta \| v - v' \|$. †

(A.10) Proposition. $v \varepsilon V \implies fv \varepsilon V$; i.e. f preserves continuity in v.

(A.11) Proposition. If $v \varepsilon W$ is convex, then fv is convex; i.e. f preserves convexity in v.

Proof:

$$
\begin{aligned}
& fv(\tilde{\pi}) + fv(\tilde{\pi}') \\
& \quad = \max_{u \varepsilon U} \{\tilde{\pi} q(u) + \beta \Sigma_{y \varepsilon Y} w(\tilde{\pi} P(y|u))\} \\
& \quad + \max_{u \varepsilon U} \{\tilde{\pi}' q(u) + \beta \Sigma_{y \varepsilon Y} w(\tilde{\pi}' P(y|u))\} \\
& \quad \geq \max_{u \varepsilon U} \{(\tilde{\pi} + \tilde{\pi}') q(u) + \beta \Sigma_{y \varepsilon Y} [w(\tilde{\pi} P(y|u)) + w(\tilde{\pi}' P(y|u))] \\
& \quad \geq \max_{u \varepsilon U} \{(\tilde{\pi} + \tilde{\pi}') q(u) + \beta \Sigma_{y \varepsilon Y} [w((\tilde{\pi} + \tilde{\pi}') P(y|u))]\} \\
& \quad = fw(\tilde{\pi} + \tilde{\pi}').
\end{aligned}
$$

†

Adopting the notation (14.19), multiple applications of f take the form:

$$f^k v(\pi) = \max_{\phi \varepsilon U(Z^{k*})} \left\{ \left(\Sigma_{\underline{z} \varepsilon Z^{(k-1)*}} \sigma[\underline{z},\phi] \beta^{\ell(z)} \pi P(\underline{z}) q(\phi(\underline{z})) \right) \right.$$

$$\left. + \beta^k \left(\Sigma_{z \varepsilon Z^k} \sigma[\underline{z},\phi] v(\pi P(\underline{z})) \right) \right\} \qquad \text{(A.12)}$$

Continuity of f, established in (A.11), is made stronger below. This will be necessary in order to establish convergence of $\{\hat{v}^m\}$ in FPS's that satisfy only a condition of weak detectability.

(A.13) <u>Proposition</u>. $\| f^k v - f^k v' \| \leq$

$$(1 - \bar{\alpha}^{\,k \div \bar{\ell}}) \beta^k \| v - v' \|_{b(k)} + \bar{\alpha}^{\,k \div \bar{\ell}} \beta^k \| v - v' \|$$

<u>Proof</u>: For any $\varepsilon > 0$, there is a $\pi \varepsilon \Pi_N$ such that

$$\| f^k v - f^k v' \| \leq f^k v(\pi) - f^k v'(\pi) + \varepsilon$$

Let $\phi \varepsilon U^{(Z^{k*})}$ be the policy maximizing (A.12), where π is as described above. Now:

$$\| f^k v - f^k v' \| - \varepsilon \leq f^k v(\pi) - \left[\left(\Sigma_{\underline{z} \varepsilon Z^{(k-1)*}} \sigma[\underline{z},\phi] \beta^{\ell(\underline{z})} \right. \right.$$

$$\left. \left. \pi P(\underline{z}) q(\phi(z)) \right) + \beta^k \left(\Sigma_{\underline{z} \varepsilon Z^k} \sigma[\underline{z},\phi] v'(\pi P(z)) \right) \right]$$

$$= \beta^k \Sigma_{\underline{z}\varepsilon Z^k} \sigma[\underline{z},\phi][v(\pi P(\underline{z})) - v'(\pi P(\underline{z}))]$$

$$= \beta^k \Sigma_{\underline{z}\varepsilon Z^k} \sigma[\underline{z},\phi](\pi P(\underline{z})1)[v(T(\pi,\underline{z})) - v'(T(\pi,\underline{z}))]$$

$$\leq \beta^k \Sigma_{\underline{z}\varepsilon Z^k} \sigma[\underline{z},\phi](\pi P(\underline{z})1) \begin{Bmatrix} \|v-v'\|_{b(k)}, & \text{if } T(\pi,\underline{z})\varepsilon\Pi_N(b(k)) \\ \|v-v'\|, & \text{otherwise} \end{Bmatrix}$$

$$\leq \beta^k \Sigma_{\underline{z}\varepsilon Z^k} \sigma[\underline{z},\phi](\pi P(\underline{z})1) \begin{Bmatrix} \|v-v'\|_{b(k)}, & \text{if } P(\underline{z}) \text{ is subrectangular} \\ \|v-v'\|, & \text{otherwise} \end{Bmatrix}$$

$$\leq \beta^k \Sigma_{\underline{z}\varepsilon Z^k} \sigma[\underline{z},\phi](\pi P(\underline{z})1) \begin{Bmatrix} \|v-v'\|_{b(k)}, & \text{if } \alpha[\underline{z}] < 1 \\ \|v-v'\|, & \text{otherwise} \end{Bmatrix}$$

$$\leq \beta^k \Sigma_{\underline{z}\varepsilon Z^k} \sigma[\underline{z},\phi](\pi P(\underline{z})1)[(1 - \alpha[\underline{z}])\|v-v'\|_{b(k)} + \alpha[\underline{z}]\|v-v'\|]$$

$$\leq \beta^k(1 - \overline{\alpha}^{\,k\div\overline{\ell}})\|v-v'\|_{b(k)} + \overline{\alpha}^{\,k\div\overline{\ell}}\|v-v'\| .$$

Taking the limit $\varepsilon \to 0$ completes the proof. †

c. A sequence in V

(A.14) Definition. $\{v_0^m\}$ and $\{v^m\}$ are sequences in W defined by

$$v_0^{m+1} = fv_0^m ,$$

$$v^{m+1} = 1/2\, v^m + 1/2\, fv^m,$$

$$v_0^0 = v^0 = 0 \ .$$

Clearly (A.14) is consistent with (19.5). By (A.10), v_0^m and v^m lie in V, and by (A.11) they are convex. Boundedness of $\{\hat{v}^m\}$ is now established.

(A.15) <u>Lemma</u>. $v_0^m + \beta^m L(\beta,k)Q_{min} \le v_0^{m+k} \le v_0^m + \beta^m L(\beta,k)Q_{max}$

<u>Proof</u>: By (A.14), $L(\beta,k)Q_{min} \le v_0^k \le L(\beta,k)Q_{max}$. (A.7) and (A.8) complete the proof.

(A.16) <u>Lemma</u>. $\| v_0^m \|_D \le \Omega$.

<u>Proof</u>: (By induction). The result is trivial for m=0, and follows trivially from (A.15) for $m\varepsilon<0,\ell_\rho+\overline{\ell}>$.

The induction follows a plan given in the heuristic justification of (19.3). Let j be a state that maximizes $v^m(e^j)$ and let $\phi^*\varepsilon U(Z^{(\overline{\ell}-1)*})$ be a policy that maximizes (A.12) when $\pi=e^j$ and $v=v_0^m$.

Now, for any $\pi\varepsilon\Pi_N$, and any $m\varepsilon<\overline{\ell},\infty>$,

$$v_0^m(e^j) - v_0^m(\pi)$$

$$\le v_0^m(e^j) - \left(\Sigma_{\underline{z}\varepsilon Z^{(\overline{\ell}-1)*}} \sigma[\underline{z},\phi^*]\beta^{\ell(\underline{z})}\pi P(\underline{z})q(\phi^*(\underline{z}))\right)$$

$$- \beta^{\overline{\ell}}\left(\Sigma_{\underline{z}\varepsilon Z^{\overline{\ell}}} \sigma[\underline{z},\phi^*]v_0^{m-\overline{\ell}}(\pi P(\underline{z}))\right),$$

$$\leq L(\beta,\overline{\ell})Q + \beta^{\overline{\ell}} \sum_{\underline{z}\varepsilon Z^{\overline{\ell}}} \sigma[\underline{z},\phi^*][v_0^{m-\overline{\ell}}(e^j P(\underline{z})) - v_0^{m-\overline{\ell}}(\pi P(\underline{z}))]$$

$$\leq L(\beta,\overline{\ell})Q + \beta^{\overline{\ell}}\pi_j \sum_{\underline{z}\varepsilon Z^{\overline{\ell}}} \sigma[\underline{z},\phi^*](e^j P(\underline{z})1)$$

$$[v^{m-\overline{\ell}}(T(e^j,\underline{z})) - v^{m-\overline{\ell}}(T(\pi,\underline{z}))] + \beta^{\overline{\ell}}(1-\pi_j)\|v^{m-\overline{\ell}}\|_D$$

$$\leq L(\beta,\overline{\ell})Q + \beta^{\overline{\ell}}\left[\pi_j\left(\sum_{\underline{z}\varepsilon Z^{\overline{\ell}}} \sigma[\underline{z},\phi^*](e^j P(\underline{z})1)a[\underline{z}]\right) + (1-\pi_j)\right]$$

$$\|v_0^{m-\overline{\ell}}\|_D$$

$$\leq L(\beta,\overline{\ell})Q + \beta^{\overline{\ell}}[1 - \pi_j(1-\overline{a})]\|v_0^{m-\overline{\ell}}\|_D$$

But, for any $\pi\varepsilon\Pi_N$, there is an input word $\underline{\hat{u}}\varepsilon U^{\ell\rho}$ such that:

$$\sum_{i\varepsilon S}\sum_{\underline{y}\varepsilon Y^{\ell(\underline{\hat{u}})}} P_{ij}(\underline{y}|\underline{\hat{u}}) \geq 1-\rho .$$

Thus

$$v_0^{m+\ell(\underline{\hat{u}})}(\pi) \leq L(\beta,\ell(\underline{\hat{u}}))Q_{max} + \beta^{\ell(\underline{\hat{u}})} v_0^m(e^j)$$

and

$$v_0^{m+\ell(\underline{\hat{u}})}(\pi) \geq L(\beta,\ell(\underline{\hat{u}}))Q_{min} + \beta^{\ell(\underline{\hat{u}})} \sum_{\underline{y}\varepsilon Y^{\ell(\underline{\hat{u}})}} v_0^m(\pi P(\underline{y}|\underline{\hat{u}}))$$

$$\geq L(\beta,\ell(\underline{\hat{u}}))Q_{min} + \beta^{\ell(\underline{\hat{u}})} v_0^m(\pi\Sigma_{\underline{y}\varepsilon Y^{\ell(\underline{\hat{u}})}} P(\underline{y}|\hat{u}))$$

$$\geq L(\beta,\ell(\underline{\hat{u}}))Q_{min} + \beta^{\ell(\hat{u})} \left[v_0^m(e^j) - L(\beta,\overline{\ell})Q - \beta^{\overline{\ell}}[1-(1-\rho)(1-\overline{a})] \right.$$

$$\left. \| v_0^{m-\overline{\ell}} \|_D \right].$$

Using (A.15),

$$\| v_0^{m+\ell_\rho} \|_D \leq L(\beta,\ell_\rho-\ell(\hat{u}))Q + \beta^{\ell_\rho-\ell(\hat{u})} \| v_0^{m+\ell(\hat{u})} \|_D$$

$$\leq L(\beta,\ell_\rho+\overline{\ell})Q + \beta^{\ell_\rho+\overline{\ell}} [1-(1-\rho)(1-\overline{a})] \| v_0^{m-\overline{\ell}} \|_D$$

and $\| v^m \|_D \leq \Omega \Longrightarrow \| v^{m+\ell_\rho+\overline{\ell}} \|_D \leq \Omega$ †

(A.17) Proposition. $\| \hat{v}^m \| \leq \Omega$, $m\varepsilon<0,\infty>$.

Proof: By (A.14),

$$v^m = \Sigma_{k\varepsilon<0,m>} \binom{m}{k} \left(1/2\right)^k v_0^m$$

So (A.16) implies $\| v^m \|_D = \Omega$. (12.12) completes the proof. †

d. Construction of a Convergent Subsequence

(A.18) Lemma. There is a subsequence $\{\hat{v}^{m(k)}\}$ of $\{\hat{v}^m\}$ having the following properties:

(a) $\{\hat{v}^{m(k)}\}$ converges pointwise to a convex function $\hat{w}^* \varepsilon W$.

(b) $\lim_{k\to\infty} \| \hat{v}^{m(k)} - \hat{w}^* \|_{b(\ell)} = 0, \; \forall \ell \varepsilon \langle \bar{\ell}, \infty \rangle$

Proof: Theorem 10.9 of Rockafellar [1970] states that any bounded sequence of convex functions on a relatively open set has a subsequence that converges uniformly on closed subsets of its domain. $\{\hat{v}^m\}$ is bounded, by (A.17). Consider the restriction of $\{\hat{v}^m\}$ to $\Pi_N^H = \{\pi \varepsilon \Pi : \pi_i > 0 \text{ iff } i \varepsilon H\}$, for some $H \subseteq S$. One of the following must hold: Π_N^H is empty; Π_N^H contains exactly one point; or Π_N^H is relatively open (in R^m). In each case, there exists a subsequence of $\{\hat{v}^m\}$ that converges pointwise on Π_N^H and uniformly on closed subsets of Π_N^H. For any $\ell \varepsilon \langle \bar{\ell}, \infty \rangle$, $\Pi_N(b(\ell)) \cap \Pi_N^H$ is closed. Taking subsequences of $\{\hat{v}^m\}$ recursively for each $H \subseteq S$, the desired subseqence is obtained. †

(A.19) Proposition. There is a subsequence of $\{\hat{v}^m\}$ that converges in $(V, \|\cdot\|)$, i.e. uniformly on Π_N.

Proof: Define:

$$w^{m+1} = 1/2 \; w^m + 1/2 \; fw^m$$

$$w^0 = \hat{w}^* .$$

Let $\{m(k)\}_{k\varepsilon<0,\infty>}$ be the sequence of indices derived in (A.18). Then, for any $\varepsilon>o$, there is a K' such that:

$$\Sigma_{m\varepsilon< 0,m(K')>}\binom{m(K')}{m}(1/2)^m \,\overline{\alpha}^{\,m\div\overline{\ell}}\, 2\Omega \leq \varepsilon/8$$

and a K'' such that:

$$\|\hat{v}^{m(k)}-\hat{v}*\|_{b(m(K'))} \leq \varepsilon/8, \quad \forall k\varepsilon<K'',\infty>$$

By (A.9), if $m>\tilde{m}$, then

$$\|\hat{v}^{m+m(k')} - \hat{w}^{m}\| \leq \|\hat{v}^{\tilde{m}+m(k')} - \hat{w}^{\tilde{m}}\|$$

Thus, for $k,k' \geq K = \max(K',K'')$,

$$\|\hat{v}^{m(k)+m(k')} - \hat{w}^{m(k)}\|$$

$$\leq \|\hat{v}^{m(K)+m(k')} - \hat{w}^{m(K)}\|$$

$$\leq \left[1-\Sigma_{m\varepsilon<0,m(K)>}\binom{m(K)}{m}(1/2)^m \,\overline{\alpha}^{\,m\div\overline{\ell}}\right]\|\hat{v}^{m(K)}-\hat{w}^0\|_{b(m(K))}$$

$$+\left[\Sigma_{m\varepsilon<0,m(K)>}\binom{m(K)}{m}(1/2)^m \,\overline{\alpha}^{\,m\div\overline{\ell}}\right]\|\hat{v}^{m(K)}-\hat{w}^0\|$$

$$\leq \|\hat{v}^{m(K)}-\hat{w}^0\|_{b(m(K))} + \left[\Sigma_{m\varepsilon<0,m(K)>}\binom{m(K)}{m}(1/2)^m \,\overline{\alpha}^{\,m\div\overline{\ell}}\right]2\Omega$$

$$\leq \varepsilon/8 + \varepsilon/8 = \varepsilon/4$$

and

$$\| \hat{w}^{m(k)} - \hat{w}^{m(k')} \|$$

$$\leq \| \hat{w}^{m(k)} - \hat{v}^{m(k)+m(k')} \| + \| \hat{v}^{m(k)+m(k')} - \hat{w}^{m(k')} \|$$

$$\leq \varepsilon/4 + \varepsilon/4 = \varepsilon/2 .$$

But now:

$$\| \hat{v}^{2m(k)} - \hat{v}^{2m(k')} \|$$

$$\leq \| \hat{v}^{m(k)+m(k)} - \hat{w}^{m(k)} \| + \| \hat{w}^{m(k)} - \hat{w}^{m(k')} \|$$

$$+ \| \hat{w}^{m(k')} - \hat{v}^{m(k')+m(k')} \|$$

$$\leq \varepsilon/4 + \varepsilon/2 + \varepsilon/4$$

$$= \varepsilon .$$

Consequently $\{\hat{v}^{2m(k)}\}$ is a Cauchy sequence in $(V, \| \cdot \|)$. †

(A.20) <u>Proposition</u>. If $\hat{v}^* \varepsilon V$ is a limit point of $\{\hat{v}^m\}$ then $\hat{v}^*$ satisfies (19.1).

<u>Proof</u>: Define:

$$w^{m+1} = 1/2\, w^m + 1/2\, fw^m$$

$$w^0 = \hat{v}^*$$

Then, by (A.9), $\hat{v}^*$ is a limit point of $\{\hat{w}^m\}$. It will now be demonstrated that $\hat{w}^m \equiv \hat{v}^*$.

Define:

a) $t^m(\pi) = w^{m+1}(\pi) - w^m(\pi)$

b) $\dot{t}^m = \max_{\pi\in\Pi_N} \{t^m(\pi)\}$

c) $R^m = \{\pi\in\Pi_N : t^m(\pi) = \dot{t}^m\}$

Since $\hat{v}^*$ is a limit point of $\{\hat{w}^m\}$, it follows that $\dot{t}^o$ is a limit point of $\{\dot{t}^m\}$ and $t^o(\pi)$ is a limit point of $\{t^m(\pi)\}$, $\forall\pi\in\Pi_N$.

Now $t^m = w^{m+1}-w^m = 1/2[fw^m-fw^{m-1}] + 1/2[w^m-w^{m-1}] \leq 1/2[fw^m-fw^{m-1}]$

$1/2\ \dot{t}^{m-1}$. Thus, by (A.9), $\dot{t}^m \leq \dot{t}^{m-1}$. Since $\dot{t}^o$ is a limit point of $\{\dot{t}^m\}$, $\dot{t}^m \equiv \dot{t}^o$.

By the Weierstrass maximum theorem, R^m is nonempty. But $t^m = 1/2[fw^m - fw^{m-1}] + 1/2[w^m - w^{m-1}] \leq 1/2\ \dot{t}^o + 1/2\ t^{m-1}$, by (A.9). Thus $R^m \subseteq R^{m-1}$. Since $\dot{t}^m \equiv \dot{t}^o$, there is a $\pi\in\Pi_N$ such that $t^m(\pi) = \dot{t}^o$, $\forall m\varepsilon<0,\infty>$. Suppose now that there exists a $\pi'\in\Pi_N$ such that $t^o(\pi') \neq \dot{t}^o$ and define

$$\varepsilon = \dot{t}^o - t^o(\pi') > 0$$

Then $w^m(\pi) = m\dot{t}^o + \hat{v}^*(\pi)$ and $w^m(\pi') \leq (\dot{t}^o-\varepsilon) + (m-1)\dot{t}^o + \hat{v}^*(\pi')$.

Hence

$$[\hat{w}^m(\pi) - \hat{v}^*(\pi)] + [\hat{v}^*(\pi') - \hat{w}^m(\pi')]$$

$$= [w^m(\pi) - \hat{v}^*(\pi)] + [\hat{v}^*(\pi') - w^m(\pi')]$$

$$\geq m\dot{t}^o - \dot{t}^o + \varepsilon - (m-1)\dot{t}^o = \varepsilon$$

But, for some $m\varepsilon\langle 1,\infty\rangle$, $\|\hat{w}^m - \hat{v}^*\| < \varepsilon/2$, since $\hat{v}^*$ is a limit point of $\{\hat{v}^m\}$. This is a contraction; hence $t^o(\pi) = \dot{t}^o$, $\forall\pi\varepsilon\Pi_N$. Now $w^1 = w^o + \dot{t}^o$. Identify $g^* = 2\dot{t}^o$ to see that $\hat{v}^*$ satisfies (19.3).

†

e. Summary and Proof of (19.3)

By (A.19) $\{\hat{v}^m\}$ has a limit point $\hat{v}^*$ in V. By (A.20), $\hat{v}^*$ satisfies (19.3).

By (A.9), $\|\hat{v}^{m+1} - \hat{v}^*\| \leq \|\hat{v}^m - \hat{v}^*\|$, and hence $\{\hat{v}^m\}$ converges in $(V, \|\cdot\|)$, i.e. uniformly on Π_N, to $\hat{v}^*$. Thus $\hat{v}^*$ is continuous.

Since each $\hat{v}^m$ is convex, it follows that $\hat{v}^*$ is convex.

Boundedness of $\hat{v}^m$ is a consequence of (A.17).

APPENDIX B

Proof of Theorem 21.6

a. Proof of Part (a)

First consider the discounted case, $\beta<1$.

Define γ^m to be a strategy which selects inputs optimally on the basis of a finite number of delayed state perceptions, taking the form:

$$y^m(k) = \begin{cases} [s(k-\ell(\underline{z}(k))),\ y(k)], & \text{if } \underline{z}(k)\varepsilon\text{ess}[M] \text{ and } k\varepsilon<0,m-1> \\ [y(k)], & \text{otherwise} \end{cases} \tag{B.1}$$

Then the inputs prescribed by γ^m at times $k\varepsilon<m-1,\infty>$ take the form $\overline{\phi}*[\eta^m(k)]$ where $\overline{\phi}*$ is the optimal feasible policy corresponding to the solution of (19.1), and

$$\eta^m(k) = \begin{cases} T(\pi(0),\underline{z}(k)), & \text{if } k\varepsilon<0,m-1> \text{ and } \underline{z}(k)\not\varepsilon\text{ess}[M] \\ T(e^{s(k-\ell(\underline{z}(k)))},\underline{z}(k)), & \text{if } k\varepsilon<0,m-1> \text{ and } \underline{z}(k)\varepsilon\text{ess}[M] \\ T(\eta^m(k-1),u(k-1),y(k)), & \text{otherwise} \end{cases} \tag{B.2}$$

Note that $\{\eta^m(k)\}$ is the information vector process which results when the observation process is $\{y^m(k)\}$.

Also define strategy $\tilde{\gamma}^m$, which selects inputs $\{u(k)\}_{k\varepsilon<0,m-1>}$ according to γ^{m+1} and inputs $\{u(k)\}_{k\varepsilon<m,\infty>}$ according to γ^m.

Then

$$g(\beta,\tilde{\gamma}^m) \leq g(\beta,\gamma^m) \tag{B.3}$$

since γ^m maximizes $g(\beta,\cdot)$ over the set of strategies realizable on the basis of observations (B.2). Thus

$$
\begin{aligned}
& g(\beta,\gamma^{m+1}) - g(\beta,\gamma^m) \\
& \quad \leq g(\beta,\gamma^{m+1}) - g(\beta,\tilde{\gamma}^m) \\
& \quad = (1-\beta)[E_{\gamma^{m+1}}\{\Sigma_{k=0}^{\infty} \beta^k r(k)\} - E_{\tilde{\gamma}^m}\{\Sigma_{k=0}^{\infty}\beta^k r(k)\}] \\
& \quad = (1-\beta)\beta^m[E_{\gamma^{m+1}} \{\Sigma_{k=m}^{\infty} \beta^{k-m} r(k)\} - E_{\tilde{\gamma}}\{\Sigma_{k=m}^{\infty} \beta^{k-m} r(k)\} \\
& \quad = (1-\beta)\beta^m[E_{\gamma^{m+1}} \{v*(\eta^{m+1}(m))\} - E_{\tilde{\gamma}^m} \{v*(\eta^m(m))\}] \\
& \quad = (1-\beta)\beta^m E_{\gamma^{m+1}} \{v*(\eta^{m+1}(m)) - v*(\eta^m(m))\} \\
& \quad \leq (1-\beta)\beta^m E_{\gamma^{m+1}} \{\Delta[\eta^{m+1}(m),\eta(m)]\}\|v*\|_{\Delta}
\end{aligned} \tag{B.4}
$$

If $m=0$ or (with probability one) $\underline{z}(m-1)\notin ess[M]$, then $g(\beta,\gamma^m) = g(\beta,\gamma^{m+1})$. Otherwise

$$\left\{\begin{array}{lll} \eta^{m+1}(m) & = \eta^{m}(m), & \text{if } \underline{z}(m) \notin \text{ess}[M] \\ \eta^{m+1}(m) & = T(e^{s(m-\ell(\underline{z}(m)))}, \underline{z}(m)) & \text{and} \\ \eta^{m}(m) & = T\left(T\left(e^{s(m-1-\ell(\underline{z}(m-1)))}, \underline{z}(m-1-\ell(\underline{z}(m-1)); m-\ell(\underline{z}(m))\right), \underline{z}(m)\right), & \end{array}\right\}$$

$$\text{if } \underline{z}(m) \in \text{ess}[M] \tag{B.5}$$

$$\text{so} \quad \Delta[\eta^{m+1}(m), \eta^{m}(m)] \le \left\{\begin{array}{ll} \alpha[\underline{z}(m)], & \text{if } \underline{z}(m) \in \text{ess}[M] \\ 0, & \text{otherwise} \end{array}\right\} .$$

$$\le \bar{\alpha}^{\ell_{\min}[M] \div \bar{\ell}}, \text{ by (14.23)}.$$

Substitution into (B.4) yields

$$g(\beta,\gamma^{m+1}) - g(\beta,\gamma^{m}) \le (1-\beta)\beta^{m}\, \bar{\alpha}^{\ell_{\min}[M] \div \bar{\ell}} \, \| v^* \|_{\Delta} \tag{B.6}$$

Now $g[M] = g(\beta,\gamma^{\infty}]$, and $g^* = g(\beta,\gamma^{0}) = g(\beta,\gamma^{\ell_{\min}[M]})$. Moreover $\| v^* \|_{\Delta} \le 4\Omega$ by (12.16) and (19.3). Thus

$$g[M] - g^* \le \Sigma_{m=\ell_{\min}[M]}^{\infty}\, g(\beta,\gamma^{m+1}) - g(\beta,\gamma^{m})$$

$$= \beta^{\ell_{\min}[M]}\, \bar{\alpha}^{\ell_{\min}[M] \div \bar{\ell}}\, 4\Omega \tag{B.7}$$

Take the limit $\beta \uparrow 1$ to prove (21.6)(a) in the undiscounted case. †

b. A Bound on Perceptive Values

The following intermediary result will be required:

$|v^M[i,\underline{z}] - v^M[i',\underline{z}']| \leq \Omega$, $\forall[i,\underline{z}]$, $[i',\underline{z}']\varepsilon\hat{\underline{X}}[m]$. Intuitively, this must be true in the limit as $\ell_{min}[M] \to \infty$, for then $v^M[i,\underline{z}] \to v^*[T(i,\underline{z})]$ and by (19.3)(c), $|v^*[\eta] - v^*[\eta']| \leq \Omega$.

In order to bound $v^M[i,\underline{z}]$, attention will be focussed on $\bar{v}^M[\pi,\underline{z}]$, which is defined by (21.3). The pair $[\pi,\underline{z}]$ may be regarded as a <u>generalized perceptive state</u>, signifying that input-output word $\underline{z}$ has evolved since the information vector was known to equal π. Naturally

$$v^M[i,\underline{z}] = \bar{v}^M[e^i,\underline{z}] \tag{B.8}$$

The following additional properties of $\bar{v}^M$ are readily established.

(B.9) <u>Lemma</u>. $\bar{v}^M[\pi,\underline{z}]$ is convex in π, for any $\underline{z}\varepsilon M$.

(B.10) <u>Lemma</u>. $\bar{v}^M[\pi,\underline{z}] \leq \max_{j\varepsilon S}\{\bar{v}^M[e^j,\underline{e}]\}$.

(B.11) <u>Lemma</u>. $\bar{v}^M[\pi,\underline{z}] \geq \min_{j\varepsilon S}\{\bar{v}^M[e^j,\underline{e}]\}$.

<u>Proof</u>: The relative value of being in the generalized perceptive state $[\pi,\underline{z}]$ can only decrease if certain information is withdrawn. An observer in generalized perceptive state $[\pi,\underline{z}]$ at time k perceives information of the form

$$\begin{Bmatrix} [s(k-\ell(\underline{z}(k'))),y(k')], & \text{if } k'-\ell(\underline{z}(k')) \geq k-\ell(\underline{z}) \\ & \text{and } \underline{z}(k')\varepsilon ess[M] \\ [y(k')], & \text{otherwise} \end{Bmatrix} \quad k'\varepsilon\langle k,\infty\rangle$$

whereas an observer in generalized perceptive state $[T[\pi,\underline{z}],\underline{e}]$ at time k, perceives information of the form

$$\left\{\begin{array}{ll} [s(k-\ell(\underline{z}'(k'))),\ y(k')], & \text{if } k'-\ell(\underline{z}(k')) \geq k \\ & \text{and } \underline{z}(k')\varepsilon ess[M] \\ [y(k')], & \text{otherwise} \end{array}\right\}_{k'\varepsilon<k,\infty>}$$

Since, in the former case, more information (specifically, perception of states $s(k'),K'\varepsilon<k+1-\ell(\underline{z}),k>$) is available, it follows that

$$\bar{v}^M[\pi,\underline{z}] \geq \bar{v}^M[T(\pi,\underline{z}),\underline{e}] \geq \bar{v}^M[T(\pi,\underline{z}),\underline{e}]$$

$$\geq \min_{\pi'\varepsilon\Pi_N}\{\bar{v}^M[\pi',\underline{e}]\} \qquad \dagger$$

(B.12) Lemma. $||\bar{v}^M[\cdot,\underline{z}]||_D \leq ||\bar{v}^M[\cdot,\underline{e}]||_D, \quad \forall \underline{z}\varepsilon M.$

Proof: By (12.11)(d),

$$||\bar{v}^M[\cdot,\underline{z}]||_D = \max_{\pi\varepsilon\Pi_N}\{\bar{v}^M[\pi,\underline{z}]\} - \min_{\pi\varepsilon\Pi_N}\{\bar{v}^M[\pi,\underline{z}]\}$$

But (B.10) and (B.11) imply

$$\min_{\pi\varepsilon\Pi_N}\{\bar{v}^M[\pi,\underline{e}]\} \leq \min_{\pi\varepsilon\Pi_N}\{\bar{v}^M[\pi,\underline{z}]\} \leq \max_{\pi\varepsilon\Pi_N}\{\bar{v}^M[\pi,\underline{z}]\}$$

$$\leq \max_{\pi\varepsilon\Pi_N}\{\bar{v}^M[\pi,\underline{e}]\}. \qquad \dagger$$

(B.13) Proposition. $v^M[i,\underline{z}] - v^M[i',\underline{z}'] \le \Omega$,

$$\forall [i,\underline{z}],\ [i',\underline{z}]\varepsilon\hat{X}[M]$$

Proof: It suffices to show that $\| v^M[\cdot,\underline{e}] \|_D \le \Omega$. Define j to be the state which maximizes $v^M[j,\underline{e}]$, and let ψ^* denote an optimal perceptive strategy adapted to M, constructed according to (21.1) for $\pi(0)=e^j$; i.e. ψ^* selects inputs optimally on the basis of information $s(0)=j$ and $\{x^M(k)\}$. Then, by (21.2),

$$\bar{v}(e^j,\underline{e}] = E_{\psi^*}\ \{\Sigma_{k\varepsilon<0,\bar{\ell}-1>}\ \beta^k q(k)$$

$$+\ \beta^{\bar{\ell}} \begin{Bmatrix} \bar{v}^M\,[e^j,\underline{z}^M(\bar{\ell})], & \text{if } \underline{z}^M(\bar{\ell})\not\varepsilon \text{ess}[M] \\ v^M[x^M(\bar{\ell})], & \text{if } \underline{z}^M(\bar{\ell})\varepsilon \text{ess}[M] \end{Bmatrix}$$

$$|s(0)=j\} - L(\beta,\bar{\ell})g[M]$$

and, for any $\pi\varepsilon\Pi_N$,

$$\bar{v}[\pi,\underline{e}] \ge \Sigma_{i\varepsilon S}\ \pi_i\ E_{\psi^*}\ \{\Sigma_{k\varepsilon<0,\bar{\ell}-1>}\ \beta^k q(k)$$

$$+\ \beta^{\bar{\ell}} \begin{Bmatrix} \bar{v}^M\,[\pi,\underline{z}^M(\bar{\ell})], & \text{if } \underline{z}^M(\bar{\ell})\not\varepsilon \text{ess}[M] \\ v^M[x^M(\bar{\ell})], & \text{if } \underline{z}^M(\bar{\ell})\varepsilon \text{ess}[M] \end{Bmatrix}$$

$$|s(0)=i\} - L(\beta,\bar{\ell})g[M]$$

Thus

$$\bar{v}^M[e^j,\underline{e}] - \bar{v}^M[\pi,\underline{e}]$$

$$\leq L(\beta,\bar{\ell})Q$$

$$+ \beta^{\bar{\ell}} \pi_j E_{\psi*} \left\{ \begin{cases} \bar{v}^M[e^j,\underline{z}^M(\bar{\ell})] - \bar{v}^M[\pi,\underline{z}^M(\bar{\ell})], & \text{if } \underline{z}^M(\bar{\ell}) \not\in \text{ess}[M] \\ 0, & \text{otherwise} \end{cases} \middle| s(0)=j \right\}$$

$$+ \beta^{\bar{\ell}}(1-\pi_j) \max_{[i,\underline{z}]\varepsilon\hat{X}[M]} \{v^M[i,\underline{z}]\} - \min_{[i,z]\varepsilon\hat{X}[M]}\{v^M[i,\underline{z}]\}$$

$$\leq L(\beta,\bar{\ell})Q$$

$$+ \beta^{\bar{\ell}}\pi_j E_{\psi*} \left\{ \begin{cases} a[\underline{z}^M(\bar{\ell})], & \text{if } \underline{z}^M(\bar{\ell}) \not\in \text{ess}[M] \\ 0 & \text{otherwise} \end{cases} \right. \; \|\bar{v}^M[\cdot,\underline{z}^M(\bar{\ell})]\|_D \; \Big| s(0)=j \Big\}$$

$$+ \beta^{\bar{\ell}}(1-\pi_j)\|\bar{v}^M[\cdot,\underline{e}]\|_D$$

$$\leq L(\beta,\bar{\ell})Q + \beta^{\bar{\ell}} \pi_j \bar{a}\|\bar{v}^M[\cdot,\underline{e}]\|_D + \beta^{\bar{\ell}}(1-\pi_j)\|\bar{v}^M[\cdot,\underline{e}]\|_D$$

$$\leq L(\beta,\bar{\ell})Q + \beta^{\bar{\ell}}[1-\pi_j(1-\bar{a})]\|\bar{v}^M[\cdot,\underline{e}]\|_D .$$

But, for any $\pi\varepsilon\Pi_N$, there is an input word $\underline{\hat{u}}\varepsilon U^{\ell\rho}$ such that

$$\Sigma_{i\varepsilon S}\Sigma_{\underline{y}\varepsilon Y^{\ell(\underline{\hat{u}})}} \quad P_{ij}((\underline{\hat{u}},\underline{y})) \geq 1-\rho$$

Thus, for any $\pi\varepsilon\Pi_N$,

$$\bar{v}^M[\pi,\underline{e}] \geq L(\beta,\ell(\hat{u}))Q_{min}$$

$$+ \beta^{\ell(\hat{u})} \Sigma_{i\varepsilon S} \pi_i E \left\{ \begin{cases} \bar{v}^M[\pi,\underline{z}^M(\ell(\underline{\hat{u}})), & \text{if } \underline{z}^M(\ell(\hat{u}))\not\varepsilon ss[M] \\ v^M[x^M(\ell(\hat{u}))], & \text{otherwise} \end{cases} \right.$$

$$|s(0)=i,\ u(0)\ldots\ u(\ell(\hat{u}))=\underline{\hat{u}}\}\ L(\beta,\ell(\underline{\hat{u}}))g[M] - L(\beta,\ell(\hat{u}))g[M]$$

$$\geq L(\beta,\ell(\underline{\hat{u}}))Q$$

$$+ \beta^{\ell(\underline{\hat{u}})}\Sigma_{i\varepsilon S}\ E\{\bar{v}^M[\pi\Sigma_{y\varepsilon Y^{\ell(\underline{\hat{u}})}}\ P((\underline{\hat{u}},\underline{y})),\underline{e}]\}|s(0)\}$$

where (B.11) was used to obtain the second inequality. Thus:

$$||\ v^m[\cdot,\underline{e}]\ ||_D \leq \max_{k\varepsilon<0,\ell_\rho>} \{L(\beta,k)Q + \beta^k[L(\beta,\bar{\ell})Q$$

$$+ \beta^{\bar{\ell}}[1-(1-\rho)(1-\bar{a})]\ ||\ v^M[\cdot,\underline{e}]\ ||_D$$

$$\leq \max_{k\varepsilon<0,\ell_\rho>} \{L(\beta,k+\bar{\ell})Q + \beta^{k+\bar{\ell}}[1-(1-\rho)(1-\bar{a})]\ ||\ v^M[\cdot,\underline{e}]\ ||_D$$

which implies $\| v^M[\cdot,\underline{e}] \|_D \leq \Omega$. †

c. A Bound on Pseudo-perceptive Deterioration

Let $v^M[\hat{i},i,\underline{z}]$ denote the value of being in augmented state $[i,\underline{z}]$ while believing the augmented state to be $[\hat{i},\underline{z}]$, where $i,\hat{i}\varepsilon C$. Specifically,

$$v^M[\hat{i},i,\underline{z}] = q^M_{\underline{z}}(i,u^*)$$

$$+ \beta \; \Sigma_{y\varepsilon Y} \; \Sigma_{j\varepsilon S} \; P^M_{\underline{z}} \; (i,j,(u^*,y))v^M[j,T^M(\underline{z},(u^*,y))]$$

$$- \; g[M], \qquad \underline{z}\varepsilon ess[M] \cap Z^+(e^{\hat{i}},e^{i}) \tag{B.14}$$

where u^* maximizes (21.1) in the evaluation of $v^M[\hat{i},\underline{z}]$. Eqs (21.1) and (B.14) may also be written:

$$v^M[\hat{i},\underline{z}] = T(e^{\hat{i}},\underline{z}) \Bigg[q(u^*) + \Sigma_{y\varepsilon Y} \; P(y|u^*)1$$

$$\Sigma_{j\varepsilon S} \left[\frac{P^M_{\underline{z}} \; (\hat{i},j,(u^*,y))}{T(e^{\hat{i}},\underline{z})P(y|u^*)1} \right] \; \beta v^M[j,T^M(\underline{z},(u^*,y))] \Bigg] - g[M] \tag{B.15}$$

$$v^M[\hat{i},i,\underline{z}] = T(e^{i},\underline{z}) \Bigg[q(u^*) + \Sigma_{y\varepsilon Y}P(y|u^*)1$$

$$\Sigma_{j\varepsilon S}\left(\frac{P^M_{\underline{z}}(i,j,(u*y))}{T(e^i,\underline{z})P(y|u*)1}\right)\beta v^M[j,T^M(\underline{z},\ (u*,y))]\Bigg] - g[M]$$

(B.16)

Since $\delta[T(e^{\hat{i}},\underline{z}),T(e^i,\underline{z})] \leq \alpha[\underline{z}]$, application of (13.4), (2.13) and (B.11) to (B.15) yields

$$v^M[\hat{i},\underline{z}] \leq \alpha[\underline{z}][Q+\beta\Omega]$$

$$+ T(e^i,\underline{z})\Bigg[q(u*) + \Sigma_{y\varepsilon Y}P(y|u*)1$$

$$\Sigma_{j\varepsilon S}\left(\frac{P^M_{\underline{z}}(\hat{i},j,(u*,y))}{T(e^{\hat{i}},\underline{z})P(y|u*)1}\right)\beta v^M[j,T^M(\underline{z},(u*,y))]\Bigg] - g[M]$$

(B.17)

Combining (B.16) and (B.17),

$$v^M[\hat{i},\underline{z}] - v^M[\hat{i},i,\underline{z}]$$

$$\leq \alpha[\underline{z}][Q+\beta\Omega] + T(e^i,\underline{z})\Sigma_{y\varepsilon Y}P(y|u*)1$$

$$\Sigma_{j\varepsilon S}\left(\frac{P^M_{\underline{z}}(i,j,(u*,y))}{T(e^{\hat{i}},\underline{z})P(y|u*)1} - \frac{P^M_{\underline{z}}(i,j,(u*,y))}{T(e^i,\underline{z})P(y|u*)1}\right)$$

$$\beta v^M[\underline{z},T^M(\underline{z},(u*,y))] \qquad \text{(B.18)}$$

Define:

$$F(\underline{z}) = \max_{\hat{i},i\varepsilon C}\{v^M[i,\underline{z}] - v^M[\hat{i},i,\underline{z}]\} \tag{B.19}$$

Naturally

$$v^M[\hat{j},\underline{z}] - v^M[j,\underline{z}]$$

$$\leq v^M[\hat{j},\underline{z}] - v^M[\hat{j},j,\underline{z}]$$

$$\leq F(\underline{z}) \tag{B.20}$$

Substituting (B.19) and (B.20) into (B.18),

$$F(\underline{z}) \leq \max_{i\varepsilon C} \max_{u\varepsilon U}\alpha[\underline{z}][Q+\beta\Omega]$$

$$+ \Sigma_{y\varepsilon Y}(T(e^i,\underline{z})P(y|u)1)$$

$$\alpha[\underline{z}(u,y) - T^M(\underline{z},(u,y))]\beta F(T^M(\underline{z},(u,y))) \tag{B.21}$$

If M=Z and $\bar{\ell}$=1, then:

$$F(\underline{z}) \leq \alpha[\underline{z}][Q+\beta\Omega]$$

$$+ \alpha[\underline{z}]\beta\bar{\alpha}[Q+\beta\Omega]$$

$$+ \alpha[\underline{z}]\beta\bar{\alpha}\beta\bar{\alpha}[Q+\beta\Omega]$$

$$+ \ldots$$

$$= \alpha[\underline{z}] \frac{[Q+\beta\Omega]}{1-\beta\bar{\alpha}} \tag{B.22}$$

In the more general case, multiple step versions of (B.18) and (B.20) are constructed, following (14.19) and (A.13), to obtain:

$$F(\underline{z}) \leq \max_{i\epsilon C} \max_{\phi\epsilon U(Z^{(k-1)*})} \{\Sigma_{\underline{z}'\epsilon Z^{(k-1)*}} (\sigma[\underline{z}',\phi]T(e^{i},\underline{z})P(\underline{z}')1)$$

$$\beta^{\ell(\underline{z}')}\alpha[\underline{z}\ \underline{z}' - T^{M}(\underline{z},\underline{z}')]\alpha[\underline{z}'][Q+\beta\Omega] + \Sigma_{\underline{z}'\epsilon Z^{k}} (\sigma[\underline{z}',\phi]T(e^{i},\underline{z})P(\underline{z}')1)$$

$$\beta^{k}\alpha[\underline{z}\ \underline{z}' - T^{M}(\underline{z},\underline{z}')]F(T^{M}(\underline{z},\underline{z}'))\} \qquad (B.23)$$

Finally, note that:

$$v^{M}[i,\underline{z}] - v^{M}[\hat{i},i,\underline{z}]$$

$$\leq v^{M}[i,\underline{z}] - v^{M}[i,\hat{i},\underline{z}]$$

$$+ v^{M}[i,\hat{i},\underline{z}] - v^{M}[\hat{i},\underline{z}]$$

$$+ v^{M}[\hat{i},\underline{z}] - v^{M}[\hat{i},i,\underline{z}]$$

$$\leq 2F(\underline{z}) \qquad (B.24)$$

d. Proof of Part (b)

The proof of part (b) is constructed in exactly the same manner as that of part (a), except that the incremental deterioration in performance due to pseudo-perception, given by (B.23), is used in place of the incremental value of perception.

Consider first the discounted case. Define γ^m to be a strategy which selects inputs at times $<0,m-1>$ according to ϕ^M and the remaining inputs according to ψ^M. Then

$$g(\beta,\gamma^{\ell_{min}[M]}) = g(\beta,\psi^M) \tag{B.25}$$

$$g(\beta,\gamma^{\infty}) = g(\beta,\phi^M) \tag{B.26}$$

Following (B.4), and using (B.23), (14.23), and the convention $\underline{z}(b;a)=\underline{e}$ if $a<b$,

$$g(\beta,\gamma^m) - g(\beta,\gamma^{m+1})$$

$$\leq (1-\beta)\beta^m E_{\gamma^{m+1}} \{v^M[z^M(m)] - v^M[s^m(\underline{z}(m)), x^M(m)]$$

$$\leq (1-\beta)\beta^m E_{\gamma^{m+1}} \{2F(\underline{z}^M(m))\}$$

$$\leq (1-\beta)\beta^m E_{\gamma^{m+1}} \{\Sigma_{k=0}^{\infty} \beta^k\alpha[\underline{z}^M(m)\underline{z}(m;m+k)-z^M(m+k)]$$

$$\alpha[\underline{z}^M(m+k)]\}\ 2[Q+\beta\Omega]$$

$$\leq (1-\beta)\beta^m E_{\gamma^{m+1}} \{\Sigma_{k=0}^{\infty}\beta^k\alpha[\underline{z}(m-\ell_{min}[M];\ m+k-\ell_{max}[M])]$$

$$\alpha[\underline{z}(m+k-\ell_{min}[M];\ m+k)]\}\ 2[Q+\beta\Omega]$$

$$\leq (1-\beta)\beta^m E_{\gamma^{m+1}} \{\Sigma_{k=0}^{\infty} \beta^k\alpha[\underline{z}(m-\ell_{min}[M];$$

$$m+k-\ell_{max}[M])]\}\overline{\alpha}^{\ell_{min}[M]\div\overline{\ell}}\ 2[Q+\beta\Omega]$$

$$\leq (1-\beta)\beta^{m}\{L(\beta,\ell_{max}[M]-\ell_{min}[M])$$

$$+ \beta^{(\ell_{max}[M]-\ell_{min}[M])} L(\beta,\bar{\ell})(1-\bar{\alpha})^{-1}\}$$

$$\bar{\alpha}^{\ell_{min}[M]\div\ell} \quad 2[Q+\beta\Omega]$$

Summing as in (B.7) completes the proof, in the discounted case.

Take the limit $\beta\uparrow 1$ to prove (21.6) (b) in the undiscounted case.

†

APPENDIX C

Listing of the Computer Program

```
PL/I OPTIMIZING COMPILER          /* DECLARATIONS */

    STMT LEV NT

                              /****************************************/
                              /*                                      */
                              /*          MODEL PARAMETERS            */
                              /*                                      */
                              /****************************************/

      1        0      DCL 1 MODEL EXTERNAL,

                            2 N FIXED BIN,              /* NUMBER OF STATES              */
                            2 NU FIXED BIN,             /* NUMBER OF INPUTS              */
                            2 NY FIXED BIN,             /* NUMBER OF OUTPUTS             */
                            2 NZ FIXED BIN,             /* NUMBER OF SYMBOLS IN Z        */

                            2 (M,ESS_M) FIXED BIN,      /* MEMORY STATES COUNTER         */
                            2 ERR FLOAT BIN,            /* ERROR, USUALLY G.HIGH-H.LOW   */
                            2 (MAX_M,MAX_ESS_M) FIXED BIN,
                            2 MIN_ERR FLOAT BIN,        /* USER-SPECIFIED BOUNDS         */
                            2 FMT FIXED BIN,            /* OUTPUT FORMAT                 */

                            2 (P_PROBS, P_RWDS, P_ZCODE) POINTER,
                                                        /* POINTERS TO STRUCT_*, BELOW   */
                            2 P_ROOT POINTER,           /* ROOT OF MEMORY TREE           */
                            2 P_ESS_NODE_1 POINTER,     /* START OF ESS NODE CHAIN       */

                            2 G,
                              3 (HIGH,LOW) FLOAT BIN,   /* BOUNDS ON G                   */
                              3 STEPS FIXED BIN,        /* DYNAMIC PROG STEPS COUNTER    */
                            2 H LIKE MODEL.G,

                            2 P_NODE POINTER,           /* PRESENT NODE                  */
                            2 P_REL POINTER,            /* RELATIVE NODE, ARG TO SCAN    */
                            2 P_REC POINTER,            /* RECURRENT NODE                */

                            2 (LEV,MAX_LEV,LO,LOO) FIXED BIN,
                                                        /* LENGTH OF BRANCH OF P_NODE    */
                            2 (U,Y,Z) FIXED BIN;        /*  INPUT/OUTPUT/IO PAIR         */

      3    1   0      DCL 1 STRUCT_ZCODE BASED(P_ZCODE), /* TRANSLATES (U,Y) TO Z        */
                            2 (NU1,NY1) FIXED BIN,
                            2 ZCODE(NU REFER(NU1), NY REFER(NY1)) FIXED BIN;

      4    1   0      DCL 1 STRUCT_PROBS BASED(P_PROBS), /* ORIGINAL TRANS PROBS         */
                            2 (NZ2,N2) FIXED BIN,
                            2 PROBS(NZ REFER(NZ2),N REFER(N2),N REFER(N2)) FLOAT BIN;

      5    1   0      DCL 1 STRUCT_RWDS BASED(P_RWDS),   /* ORIGINAL IMM REWARDS ARRAY   */
                            2 (NU3,N3) FIXED BIN,
                            2 RWDS(NU REFER(NU3), N REFER(N3)) FLOAT BIN;
```

```
STMT LEV NT

                             /****************************************/
                             /*                                      */
                             /*      MEMORY TREE SPECIFICATION       */
                             /*                                      */
                             /****************************************/

   6   1  0       DCL 1 NODE BASED(P_NODE),
                        2 P_ESS_NODE POINTER,         /* POINTS TO ESS_NODE, BELOW   */
                        2 (P_TPM,P_BRANCHES) POINTER,/* POINT TO SUBSTRUCTS OF NODE */

                        2 P_BACK POINTER,             /* IDENTIFIES PREVIOUS NODE    */
                        2 Z_BACK FIXED BIN,           /* IDS BRANCH ON PREVIOUS NODE */

                        2 (NO,NZO) FIXED BIN,
                        2 ROWSUM(N REFER(NO)) FLOAT BIN,
                                                      /* ROWSUM(I) = SUM/J TPM(I,J)  */
                        2 TPM(N REFER(NO),N REFER(NO)) FLOAT BIN,
                                                      /* TRANS PROBABILITY MATRIX    */
                        2 BRANCHES(NZ REFER(NZO)),
                          3 P_BRANCH POINTER,         /* IDENTIFIES NODE ALONG BRANCH
                                                         Z FROM CURRENT NODE         */
                          3 E_BRANCH BIT ALIGNED;     /* IS BRANCH Z A NODE IN Z+?   */

   7   1  0       DCL 1 ESS_NODE BASED(P_ESS_NODE),
                        2 P_NEXT_ESS_NODE POINTER,    /* NEXT NODE IN ESS NODE CHAIN */
                        2 (NOO,NUOO,NZOO) FIXED BIN,

                        2 (P_VG,P_VH,P_W,P_UG,P_PZ,P_QZ) POINTER,
                                                      /* POINT TO SUBSTRUCTS, BELOW  */

                        2 REC,                        /* FLAGS WHICH ID REC MEM STS  */
                          3 (TO,FROM,G,H) BIT ALIGNED,
                        2 UH FIXED BIN,               /* INPUT - STEP H              */

                        2 P_NEXTZ(NZ REFER(NZOO)) POINTER,
                                                      /* NEXT (ESS_)NODE, IF NEXT
                                                         I/O PAIR IS THE SUBSCRIPT Z */

                        2 VG(N REFER(NOO)) FLOAT BIN,/* RELATIVE VALUE - STEP G     */
                        2 VH(N REFER(NOO)) FLOAT BIN,/* RELATIVE VALUE - STEP H     */
                        2 W(N REFER(NOO)) FLOAT BIN, /* WORKSPACE FOR LHS OF DYN PR */
                        2 UG(N REFER(NOO)) FIXED BIN,/* OPTIMAL INPUT - STEP G      */

                        2 PZ(NZ REFER(NZOO),N REFER(NOO),N REFER(NOO)) FLOAT BIN;
                                                      /* TPM OF AUGMENTED SYSTEM     */
                        2 QZ(NU REFER(NUOO),N REFER(NOO)) FLOAT BIN;
                                                      /* INCREMENTAL REWARDS FOR
                                                         AUGMENTED SYSTEM            */
```

```
    STMT LEV NT

                                         /****************************************/
                                         /*                                      */
                                         /* FAST REFERENCE OF NODAL PARAMETERS   */
                                         /*                                      */
                                         /****************************************/

      8    1   0     DCL (FP_TPM,FP_BRANCHES,FP_VG,FP_VH,FP_W,FP_UG,FP_PZ,FP_QZ,FP_FLAG)
                                   POINTER;           /* POINT TO STRUCTURES, BELOW   */

      9    1   0     DCL F_TPM(10000) BASED(FP_TPM) FLOAT BIN;

     10    1   0     DCL 1 F_BRANCHES(10000) BASED(FP_BRANCHES),
                           2 F_P_BRANCH POINTER,
                           2 F_E_BRANCH BIT ALIGNED;

     11    1   0     DCL F_VG(10000) BASED(FP_VG) FLOAT BIN;

     12    1   0     DCL F_VH(10000) BASED(FP_VH) FLOAT BIN;

     13    1   0     DCL F_W(10000) BASED(FP_W) FLOAT BIN;

     14    1   0     DCL F_UG(10000) BASED(FP_UG) FIXED BIN;

     15    1   0     DCL F_PZ(10000) BASED(FP_PZ) FLOAT BIN;

     16    1   0     DCL F_QZ(10000) BASED(FP_QZ) FLOAT BIN;

     17    1   0     DCL FLAG(10000) BASED(FP_FLAG) FIXED BIN; /* GENERALLY OVER UG(*) */

     18    1   0     DCL DP_SKIP(10000) BASED(FP_W) FIXED BIN; /* HASTINGS SKIP, OVER W*/


                                         /****************************************/
                                         /*                                      */
                                         /*          MISC DECLARATIONS           */
                                         /*                                      */
                                         /****************************************/

     19    1   0     DCL (NULL,LINENO) BUILTIN;
     20    1   0     DCL TIMING ENTRY(FIXED BIN(31,0));

     21    1   0     DCL 1 TIME EXTERNAL,              /* TIMES IN SEC/100            */
                           2 (PREP,G,H,LIMIT) FIXED BIN(31,0);
```

```
PL/I OPTIMIZING COMPILER          FPS_OPT: PROC OPTIONS(MAIN)   REORDER;

                         SOURCE LISTING

  STMT LEV NT

     1        0   FPS_OPT: PROC OPTIONS(MAIN)   REORDER;

     2    1   0  %INCLUDE DD1(DCL);
     4    1   0     DCL (PREP_G,SOLVE_G,PREP_H,SOLVE_H,REPORT) EXT ENTRY;

     5    1   0     DCL IT FIXED BIN;                   /* ITERATION NUMBER          */
     6    1   0     DCL (TOT_PAGE,IT_PAGE) FIXED BIN;  /* PAGE COUNTERS             */
     7    1   0     DCL TITLE CHAR(32), (I ,J) FIXED BIN, B BIT, P POINTER, S FLOAT BIN;
     8    1   0     DCL BAR CHAR(62) INIT(('+' || (60)'-' || '+' ));
     9    1   0     DCL TE CHAR(6) INIT('TIME =');

    10    1   0     ON ENDPAGE(SYSPRINT) BEGIN;
    11    2   0       PUT EDIT('|','|','|','|') (COL(1), A,COL(86),A,PAGE,A,COL(86),A);
    12    2   0       PUT EDIT(TITLE) (SKIP(6),COL(14),A);
    13    2   0         TOT_PAGE = TOT_PAGE+1;
    14    2   0       IF TOT_PAGE > 1
                      THEN PUT EDIT('PAGE',TOT_PAGE) (X(6),A,F(3));
    15    2   0       IF IT>0
                      THEN DO;
    16    2   1           IT_PAGE = IT_PAGE+1;
    17    2   1           PUT EDIT('TABLE', IT*100 + IT_PAGE) (X(6),A,F(6,2,-2));
    18    2   1           IF IT_PAGE=1
                          THEN DO;
    19    2   2             PUT EDIT(BAR, '|   ITERATION',IT,'MEM =',M,
                               'ESS MEM =',ESS_M,TE,TIME.PREP,'|','|','|','|',
                               G.LOW,' < G <',G.HIGH,G.STEPS,'  STEPS',TE,TIME.G,'|',
                               '|',H.LOW,' < H <',H.HIGH,H.STEPS,'  STEPS',TE,TIME.H,
                               '|',BAR)
                              (SKIP(2),2(COL(14),A),F(3),X(4),A,F(3),X(3),A,F(3),
                               X(3),A,F(6,2,-2),X(3),A,COL(14),A,COL(75),A,
                               2(COL(14),A,F(8,3),A,F(8,3),F(9), A,X(7),A,F(6,2,-2),
                               X(3),A),   COL(14),A);

    20    2   2           IF FMT=1
                          THEN PUT EDIT('RC   I   U   V(G)   V(H)   PROBS')
                                 (SKIP(2),COL(14),A);
    21    2   2           ELSE PUT EDIT('RC      U') (COL(14),A);
    22    2   2           PUT EDIT('MEMORY STATES') (COL(63),A);

    23    2   2             END;
    24    2   1           LO = 0;
    25    2   1           END;
    26    2   0       ELSE PUT EDIT('PROBLEM SPECS') (COL(63),A);
    27    2   0       END;
```

```
STMT LEV NT

                     /****************************************/
                     /*                                      */
                     /*   READ MODEL AND PRINT TITLE PAGE    */
                     /*                                      */
                     /****************************************/

  28   1  0   TITLE='';
  29   1  0   MAX_LEV, MAX_M,MAX_ESS_M,IT,FMT,TOT_PAGE = 0;
  30   1  0   M, ESS_M = 1;
  31   1  0   MIN_ERR=0.;
  32   1  0   TIME.LIMIT = 3;

  33   1  0   GET LIST(TITLE, N,NU,NY,NZ,FMT, TIME.LIMIT,MIN_ERR,MAX_M,MAX_ESS_M);
  34   1  0   TIME.LIMIT = TIME.LIMIT*100;
  35   1  0   SIGNAL ENDPAGE(SYSPRINT);
  36   1  0   PUT EDIT(N,' STATES',NU,' INPUTS',NY,' OUTPUTS',NZ,' I/O PAIRS',
                         'TIME LIMIT:',TIME.LIMIT,'MIN ERR: ',MIN_ERR,
                         'MAX MEM:',MAX_M,'MAX ESS MEM:',MAX_ESS_M)
                         (SKIP(2),COL(19),4(F(4),A),SKIP(2), COL(22),A,F(6,2 ,-2),
                         COL(53),A,F(5,3), SKIP(2),COL(22),A,F(4),COL(51),A,F(4));

  37   1  0   ALLOCATE STRUCT_ZCODE,STRUCT_PROBS,STRUCT_RWDS,NODE,ESS_NODE;

  38   1  0   ZCODE = 0;

  39   1  0   P_ROOT,P_ESS_NODE_1 = P_NODE;
  40   1  0   P_BACK,P_NEXT_ESS_NODE = NULL;
  41   1  0   P_NEXTZ = P_ROOT;
  42   1  0   P_TPM,FP_TPM = ADDR(TPM(1,1));
  43   1  0   P_BRANCHES, FP_BRANCHES = ADDR(BRANCHES(1));
  44   1  0   P_VG,FP_VG = ADDR(VG(1));
  45   1  0   P_VH,FP_VH = ADDR(VH(1));
  46   1  0   P_W = ADDR(W(1));
  47   1  0   P_UG, FP_UG = ADDR(UG(1));
  48   1  0   P_PZ = ADDR(PZ(1,1,1));
  49   1  0   P_QZ = ADDR(QZ(1,1));
  50   1  0   REC.G,REC.H = '1'B;
  51   1  0   DO I=1 TO N*N;
  52   1  1     F_TPM(I)=0;
  53   1  1     END;
  54   1  0   DO I=1 TO N;
  55   1  1     F_VG(I),F_VH(I) = 0.;
  56   1  1     F_UG(I) = 1;
  57   1  1     F_TPM((I-1)*N + I), ROWSUM(I) = 1.;
  58   1  1     END;
```

```
STMT LEV NT

                                        /****************************************/
                                        /*     PLACE INPUT PROBS IN PZ          */
                                        /****************************************/

  59    1  0      PUT EDIT('TRANSITION PROBABILITIES:','Z','(U, Y)','P')
                           (SKIP(3),COL(14),A,SKIP ,COL(15),A,X(7),A,X(9),A);
  60    1  0      DO Z=1 TO NZ;
  61    1  1        IF LINENO(SYSPRINT)+3+(N/10+1)*N > 55
                    THEN SIGNAL ENDPAGE(SYSPRINT);
  62    1  1        PUT EDIT(Z)  (F(16)) SKIP(2);
  63    1  1   GET_UY_PAIR:
                    GET LIST(U);
  64    1  1        IF U=0 THEN GOTO GET_TPM;
  65    1  1        GET LIST(Y);
  66    1  1        PUT EDIT(U,Y)  (COLUMN(22),2 F(3));
  67    1  1        ZCODE(U,Y) = Z;
  68    1  1        GO TO GET_UY_PAIR;
  69    1  1   GET_TPM:
                    IF LINENO(SYSPRINT) + (N/10+1)*N > 55
                    THEN SIGNAL ENDPAGE(SYSPRINT);
  70    1  1        B = '0'B;
  71    1  1        FP_PZ = ADDR(P_PZ->F_PZ((Z-1)*N*N + 1));
  72    1  1        DO I=1 TO N*N;
  73    1  2          F_PZ(I) = 0.;
  74    1  2          END;
  75    1  1        GET LIST((F_PZ(I) DO I=1 TO N*N));
  76    1  1        DO I=1 TO N;
  77    1  2          PUT SKIP;
  78    1  2        PUT EDIT((F_PZ(J) DO J=(I-1)*N+1 TO I*N))  (COL(36),5 F(8,4));
  79    1  2          END;
  80    1  1        DO I = 1 TO N*N;
  81    1  2          B = B | F_PZ(I)¬=0.;
  82    1  2          END;
  83    1  1        F_E_BRANCH(Z) = B;
  84    1  1        F_P_BRANCH(Z) = NULL;
  85    1  1        END;
                                                               /* COPY PZ INTO PROBS                */
  86    1  0      FP_PZ = P_PZ;
  87    1  0      P = ADDR(PROBS(1,1,1));
  88    1  0      DO I=1 TO N*N*NZ;
  89    1  1        P->F_PZ(I) = F_PZ(I);
  90    1  1        END;
```

```
STMT LEV NT

                              /****************************************/
                              /* VERIFY    SUM/Y,J/ P/I,J/(Y|U) = 1. */
                              /****************************************/

  91   1  0     B = '0'B;
  92   1  0     DO I=1 TO N;
  93   1  1       DO U=1 TO NU;
  94   1  2         S = 0.;
  95   1  2         DO Y=1 TO NY;
  96   1  3           Z = ZCODE(U,Y);
  97   1  3           IF Z¬=0
                      THEN DO J = (Z-1)*N*N+(I-1)*N+1 TO (Z-1)*N*N+I*N;
  98   1  4             S = S + F_PZ(J);
  99   1  4             END;
 100   1  3           END;
 101   1  2         IF ABS(S-1.) > 1E-4 THEN DO;
 102   1  3           PUT EDIT('ERROR: TRANS. PROBS. DO NOT SUM TO ONE FOR I =',I,
                               ', U =',U) (SKIP(2),A,F(3),A,F(3));
 103   1  3           B = '1'B;
 104   1  3           END;
 105   1  2         END;
 106   1  1       END;
 107   1  0     IF B THEN STOP;
```

```
STMT LEV NT

                             /****************************************/
                             /* PLACE INPUT REWARDS IN QZ AND RWDS  */
                             /****************************************/

 108   1  0      IF LINENO(SYSPRINT)+3+(N/10+1)*NU > 55
                 THEN SIGNAL ENDPAGE(SYSPRINT);
 109   1  0    PUT EDIT('INCREMENTAL REWARDS:','U','Q')
                          (SKIP(3),X(13),A,SKIP,COL(27),A,X(10),A);
 110   1  0    PUT SKIP(2);
 111   1  0    G.HIGH =-1E10;
 112   1  0    G.LOW= 1E10;
 113   1  0    DO U=1 TO NU;
 114   1  1      FP_QZ = ADDR( P_QZ->F_QZ((U-1)*N+1));
 115   1  1      GET LIST((F_QZ(I)  DO I=1 TO N));
 116   1  1      PUT EDIT(U)  (COL(25),F(3));
 117   1  1      PUT EDIT((F_QZ(I)  DO I=1 TO N))  (COLUMN(36), 5 F(8,4));
 118   1  1      DO I=1 TO N;
 119   1  2        G.HIGH = MAX(G.HIGH,F_QZ(I));
 120   1  2        G.LOW  = MIN(G.HIGH,F_QZ(I));
 121   1  2        END;
 122   1  1      END;

 123   1  0    FP_QZ = P_QZ;
 124   1  0    P = ADDR(RWDS(1,1));
 125   1  0    DO I=1 TO N*NU;
 126   1  1      P->F_QZ(I) = F_QZ(I);
 127   1  1      END;

                                                   /* MISC PRELIMINARIES             */
 128   1  0    ERR = G.HIGH - G.LOW;
 129   1  0    IF MAX_M<=0 THEN MAX_M=10000;
 130   1  0    IF MAX_ESS_M<=0 THEN MAX_ESS_M=1000;
 131   1  0    IF FMT¬=0 & FMT¬=1
               THEN DO;
 132   1  1      PUT EDIT('*** INCORRECT OUTPUT FORMAT',FMT,' SPECIFIED ***')
                     (SKIP,X(10),A,F(4),A);
 133   1  1      STOP;
 134   1  1      END;
```

```
PL/I OPTIMIZING COMPILER          FPS_OPT: PROC OPTIONS(MAIN)   REORDER;

    STMT LEV NT

                                 /**************************************************/
                                 /*                                                */
                                 /*          MAIN SECTION OF THE PROGRAM           */
                                 /*                                                */
                                 /**************************************************/

     135   1  0   LOOP:
                    IT = IT+1;
     136   1  0     IT_PAGE = 0;
     137   1  0     CALL TIMING(TIME.PREP);

                                 /**************************************************/
                                 /*                                                */
                                 /*   SOLVE FOR OPTIMAL GAIN  G                    */
                                 /*     AND OPTIMAL VALUE VG                       */
                                 /*   (USE VH AS INITIAL GUESS)                    */
                                 /*   (LEAVE SOLUTION IN BOTH VG AND VH)           */
                                 /*                                                */
                                 /**************************************************/

     138   1  0     CALL SOLVE_G;
     139   1  0     CALL TIMING(TIME.G);

                                 /**************************************************/
                                 /*                                                */
                                 /*   SOLVE FOR FEASIBLE GAIN  H                   */
                                 /*   AND CORRESPONDING VALUE VH                   */
                                 /*   (USING VH AS INITIAL GUESS)                  */
                                 /*                                                */
                                 /**************************************************/

     140   1  0     CALL PREP_H;
     141   1  0     CALL SOLVE_H;
     142   1  0     CALL TIMING(TIME.H);

     143   1  0     CALL REPORT;

     144   1  0     CALL PREP_G;
     145   1  0     GOTO LOOP;
     146   1  0     END;
```

```
PL/I OPTIMIZING COMPILER          PREP_G: PROC REORDER;

                         SOURCE LISTING

   STMT LEV NT

      1       0   PREP_G: PROC REORDER;

      2    1  0  %INCLUDE DD1(DCL);
                          /*********************************************************/
                          /*                                                       */
                          /*   ADD NODES AS REQUIRED (FOLLOWING REC.G)             */
                          /*   COPY V INTO V_FEAS                                  */
                          /*   PRUNE OUT NODES WHICH ARE NO LONGER ESS             */
                          /*                                                       */
                          /*********************************************************/

      4    1  0     DCL ADDNODE EXT ENTRY;
      5    1  0     DCL (BU(0:NU),BZ(0:NZ),B,BA) BIT ALIGNED, (P,P0,P1) POINTER,
                          (I,ZZ,Z_STRING(0:MAX_LEV)) FIXED BIN;

                                                  /* ADD NEW NODES              */
      6    1  0     BA = '0'B;
      7    1  0     P_NODE,P1 = P_ESS_NODE_1;

      8    1  0   A_LOOP:
                    IF ¬REC.G THEN GOTO END_A_LOOP;

      9    1  0     BU,BZ = '0'B;
     10    1  0     FP_UG = P_UG;
     11    1  0     DO I=1 TO N;
     12    1  1       BU(F_UG(I)) = '1'B;
     13    1  1       END;

     14    1  0     DO U=1 TO NU;
     15    1  1       IF BU(U)
                      THEN DO Y=1 TO NY;
     16    1  2         BZ(ZCODE(U,Y)) = '1'B;
     17    1  2         END;
     18    1  1       END;
```

```
PL/I OPTIMIZING COMPILER          PREP_G: PROC REORDER;

   STMT LEV NT

     19   1  0     DO ZZ = 1 TO NZ;
     20   1  1       IF BZ(ZZ)
                     THEN DO;
     21   1  2         P,PO = P_NODE;
     22   1  2         LEV = -1;
     23   1  2   A_LOOP1:
                       LEV = LEV+1;
     24   1  2         Z_STRING(LEV) = P-> Z_BACK;
     25   1  2         P = P-> P_BACK;
     26   1  2         IF P¬=NULL
                       THEN GOTO A_LOOP1;

     27   1  2         Z_STRING(LEV) = ZZ;
     28   1  2         P_NODE = P_ROOT;
     29   1  2   A_LOOP2:
                       FP_BRANCHES = P_BRANCHES;
     30   1  2         Z = Z_STRING(LEV);
     31   1  2         IF ¬F_E_BRANCH(Z)
                       THEN GOTO OUT_A;
     32   1  2         P = F_P_BRANCH(Z);
     33   1  2         IF P=NULL
                       THEN DO;
     34   1  3           IF ¬REC.G THEN GOTO OUT_A;
     35   1  3           FP_UG = P_UG;
     36   1  3           U = UH;
     37   1  3           DO I=1 TO N;
     38   1  4             IF F_UG(I)¬=0 & F_UG(I)¬= U
                           THEN DO;
     39   1  5               CALL ADDNODE;
     40   1  5               BA = '1'B;
     41   1  5               GOTO OUT_A;
     42   1  5               END;
     43   1  4             END;
     44   1  3           GOTO OUT_A;
     45   1  3           END;
     46   1  2         P_NODE = P;
     47   1  2         LEV = LEV-1;
     48   1  2         IF LEV>=0
                       THEN GOTO A_LOOP2;
     49   1  2   OUT_A:
                       P_NODE = PO;
     50   1  2         END;
     51   1  1       END;
     52   1  0   END_A_LOOP:
                   P_NODE = P_NEXT_ESS_NODE;
     53   1  0     IF P_NODE¬=NULL
                   THEN GOTO A_LOOP;
```

```
PL/I OPTIMIZING COMPILER          PREP_G: PROC REORDER;

    STMT LEV NT

                                                     /* CHECK: WAS ANYTHING ADDED?  */PRPG0840
      54   1  0       IF ¬BA                                                          PRPG0850
                      THEN DO;                                                        PRPG0860
      55   1  1         SIGNAL ENDPAGE(SYSPRINT);                                     PRPG0870
      56   1  1         MIN_ERR = 1.E10;                                              PRPG0880
      57   1  1         PUT EDIT('*** NO MEMORY STATES ADDED - AT MOST ONE MORE ITERATION'PRPG0890
                             ||' WILL BE ALLOWED ***')  (SKIP(2),X(10),A);            PRPG0900
      58   1  1         RETURN;                                                       PRPG0910
      59   1  1         END;                                                          PRPG0920
                                                     /* CLEAN OUT ESS NODE CHAIN    */PRPG0930
      60   1  0       P_NODE = P_ESS_NODE_1;                                          PRPG0940
      61   1  0     PRE_LOOP:                                                         PRPG0950
                      P_REL = P_NODE;                                                 PRPG0960
      62   1  0       P_NODE = P_NEXT_ESS_NODE;                                       PRPG0970
      63   1  0       IF P_NODE¬=P1 THEN GOTO PRE_LOOP;                               PRPG0980
      64   1  0       GOTO ENTER_LOOP;                                                PRPG0990
                                                                                      PRPG1000
      65   1  0     PRUNE_LOOP:                                                       PRPG1010
                      P_REL  = P_NODE;                                                PRPG1020
      66   1  0       P_NODE = P_NEXT_ESS_NODE;                                       PRPG1030
      67   1  0       IF P_NODE=NULL THEN RETURN;                                     PRPG1040
                                                                                      PRPG1050
      68   1  0     ENTER_LOOP:                                                       PRPG1060
                      FP_BRANCHES = P_BRANCHES;                                       PRPG1070
      69   1  0       DO Z=1 TO NZ;                                                   PRPG1080
      70   1  1         IF F_E_BRANCH(Z) & F_P_BRANCH(Z)=NULL                         PRPG1090
                        THEN DO;                                                      PRPG1100
      71   1  2           FP_UG = P_UG; FP_VG = P_VG; FP_VH = P_VH;                   PRPG1110
      74   1  2           DO I=1 TO N;                                                PRPG1120
      75   1  3             IF F_UG(I)¬=0 THEN F_VH(I)  = F_VG(I);                    PRPG1130
      76   1  3             END;                                                      PRPG1140
      77   1  2           GOTO PRUNE_LOOP;                                            PRPG1150
      78   1  2           END;                                                        PRPG1160
      79   1  1         END;                                                          PRPG1170
                                                                                      PRPG1180
      80   1  0       ESS_M = ESS_M-1;                                                PRPG1190
      81   1  0       P_REL -> P_ESS_NODE -> P_NEXT_ESS_NODE = P_NEXT_ESS_NODE;       PRPG1200
      82   1  0       FREE ESS_NODE;                                                  PRPG1210
      83   1  0       P_ESS_NODE = NULL;                                              PRPG1220
      84   1  0       P_NODE = P_REL;                                                 PRPG1230
      85   1  0       GOTO PRUNE_LOOP;                                                PRPG1240
      86   1  0       END;                                                            PRPG1250
```

PL/I OPTIMIZING COMPILER   ADDNODE: PROC REORDER;

SOURCE LISTING

```
STMT LEV NT

   1      0  ADDNODE: PROC REORDER;

   2   1  0 %INCLUDE DD1(DCL);

                /**********************************************************/
                /*                                                        */
                /* ADD BRANCH Z TO NODE P_NODE                            */
                /*       ALSO ADD OTHER NODES AS REQUIRED TO MAINTAIN     */
                /*       RECURSIVE PROPERTIES OF THE MEMORY SET           */
                /*                                                        */
                /**********************************************************/

   4   1  0    DCL SCAN EXT ENTRY;

   5   1  0    DCL Z_ADD FIXED BIN INIT(Z), P0 POINTER INIT(P_NODE);
                                                  /* REGISTERS TO SAVE INITIAL
                                                     VALUES OF Z AND P_NODE      */

   6   1  0    DCL R(N) FLOAT BIN;                /* ROW SUM OF NEW TPM          */
   7   1  0    DCL Z_STRING(0:MAX_LEV) FIXED BIN;
   8   1  0    DCL P_NEW(0:MAX_LEV) POINTER;

   9   1  0    DCL (S,SV,E) FLOAT BIN,(I,II,J,K,UU) FIXED BIN, (B,BB) BIT ALIGNED,
                     (P,P1,P2,FP_PROBS,FP_RWDS,FP_TPM2) POINTER;
  10   1  0    FP_RWDS = ADDR(RWDS(1,1));

                                                  /* FILL IN Z_STRING WITH
                                                     DESCRIPTION FOR P_NODE      */
  11   1  0    Z_STRING(0) = Z_ADD;
  12   1  0    DO I=1 TO MAX_LEV;
  13   1  1      P1 = P_BACK;
  14   1  1      IF P1=NULL
                 THEN GOTO OUT;
  15   1  1      Z_STRING(I) = Z_BACK;
  16   1  1      P_NODE = P1;
  17   1  1      END;
  18   1  0  OUT:
               MAX_LEV = MAX(MAX_LEV,I);
  19   1  0    LEV,L0 = I;
```

```
PL/I OPTIMIZING COMPILER         ADDNODE: PROC REORDER;

   STMT LEV NT

                                                      /* THIS LOOP ADDS BRANCH Z_ADD
                                                         TO PO & UNIQUE PRECEDENTS   */
     20   1  0   LOOP1:
                   LEV = LEV-1;

                                                      /* FIND P_NODE FOR GIVEN Z_STR */
     21   1  0     P_NODE = P_ROOT;
     22   1  0     DO I = LEV TO 0 BY -1;
     23   1  1       P = P_NODE;
     24   1  1       P_NODE = P_BRANCHES->P_P_BRANCH(Z_STRING(I));
     25   1  1       END;

     26   1  0     IF P_NODE¬=NULL
                   THEN GOTO NO_MORE_ADD;

                                                      /* ALLOCATE NEW NODE           */
     27   1  0     ALLOC NODE,ESS_NODE;

                                                      /* LINK TO OLD NODE            */
     28   1  0     Z_BACK = Z_ADD;
     29   1  0     P_BACK = P;
     30   1  0     P->P_BRANCHES->P_P_BRANCH(Z_ADD) = P_NODE;

                                                      /* PLACE NEW NODE AT START OF
                                                         ESS NODE CHAIN              */
     31   1  0     P_NEXT_ESS_NODE = P_ESS_NODE_1;
     32   1  0     P_NEW(LEV),P_ESS_NODE_1 = P_NODE;

     33   1  0     P_TPM = ADDR(TPM(1,1));
     34   1  0     FP_BRANCHES,P_BRANCHES = ADDR(BRANCHES(1));
     35   1  0     P_VG = ADDR(VG(1));
     36   1  0     FP_VH,P_VH = ADDR(VH(1));
     37   1  0     P_W = ADDR(W(1));
     38   1  0     FP_UG,P_UG = ADDR(UG(1));
     39   1  0     P_PZ = ADDR(PZ(1,1,1));
     40   1  0     FP_QZ,P_QZ = ADDR(QZ(1,1));
     41   1  0     REC.G,REC.H = '0'B;
                                                      /* UPDATE MEMORY COUNTERS      */
     42   1  0     M = M+1;
     43   1  0     ESS_M = ESS_M+1;

     44   1  0     P_NODE = P;
```

```
PL/I OPTIMIZING COMPILER         ADDNODE: PROC REORDER;

    STMT LEV NT

                                                 /* COMPUTE TPM,VH AND QZ       */ADDN0840
                                                 /* PRESET UG TO SHOW R(I) > 0? */ADDN0850
      45   1  0     FP_VG = P_VG;                                                   ADDN0860
      46   1  0     DO I = 1 TO N;                                                  ADDN0870
      47   1  1       SV,R(I) = 0.;                                                 ADDN0880
      48   1  1       FP_TPM = ADDR(P_ESS_NODE_1->P_TPM->F_TPM((I-1)*N+1));         ADDN0890
      49   1  1       FP_PZ = ADDR(PROBS(Z_ADD,I,1));                               ADDN0900
      50   1  1       DO J = 1 TO N;                                                ADDN0910
      51   1  2         S = 0.;                                                     ADDN0920
      52   1  2         II = 1;                                                     ADDN0930
      53   1  2         FP_TPM2 = ADDR(P_TPM->F_TPM(J));                            ADDN0940
      54   1  2         DO K = 1 TO N;                                              ADDN0950
      55   1  3           E = F_PZ(K)  * FP_TPM2->F_TPM(II);                        ADDN0960
      56   1  3           II = II+N;                                                ADDN0970
      57   1  3           S = S + E;                                                ADDN0980
      58   1  3           SV = SV + E * F_VG(K);                                    ADDN0990
      59   1  3           END;                                                      ADDN1000
      60   1  2         F_TPM(J) = S;                                               ADDN1010
      61   1  2         R(I) = R(I) + S;                                            ADDN1020
      62   1  2         END;                                                        ADDN1030
      63   1  1       P_ESS_NODE_1->ROWSUM(I) = R(I);                               ADDN1040
      64   1  1       IF R(I)>0                                                     ADDN1050
                      THEN DO;                                                      ADDN1060
      65   1  2         F_UG(I) = 1;                                                ADDN1070
      66   1  2         F_VH(I) = SV/R(I);                                          ADDN1080
      67   1  2         UU = 0;                                                     ADDN1090
      68   1  2         DO U=1 TO NU;                                               ADDN1100
      69   1  3           S = 0.;                                                   ADDN1110
      70   1  3           DO J=1 TO N;                                              ADDN1120
      71   1  4             S = S + F_TPM(J)  * FP_FWDS->F_QZ(UU+J);                ADDN1130
      72   1  4             END;                                                    ADDN1140
      73   1  3           F_QZ(UU+I) = S/R(I);                                      ADDN1150
      74   1  3           UU = UU+N;                                                ADDN1160
      75   1  3           END;                                                      ADDN1170
      76   1  2         END;                                                        ADDN1180
      77   1  1       ELSE F_UG(I) = 0;                                             ADDN1190
      78   1  1       END;                                                          ADDN1200
```

PL/I OPTIMIZING COMPILER   ADDNODE: PROC REORDER;

```
STMT LEV NT

                                                /* COMPUTE E_BRANCH,P_BRANCH   */
  79   1  0      DO Z=1 TO NZ;
  80   1  1        FP_PZ = ADDR(PROBS(Z,1,1));
  81   1  1        F_P_BRANCH(Z) = NULL;
  82   1  1        P_ESS_NODE_1->P_ESS_NODE->P_NEXTZ(Z) = NULL;
  83   1  1        DO I=1 TO N;
  84   1  2          IF R(I)>0
                     THEN DO J=1 TO N;
  85   1  3            IF F_PZ((J-1)*N+I) > 0
                       THEN DO;
  86   1  4              F_E_BRANCH(Z) = '1'B;
  87   1  4              GOTO NEXT_Z;
  88   1  4              END;
  89   1  3            END;
  90   1  2          END;
  91   1  1        F_E_BRANCH(Z) = '0'B;
  92   1  1    NEXT_Z:
                   END;
  93   1  0      GOTO LOOP1;
```

```
 STMT LEV NT

   94   1  0   NO_MORE_ADD:
                 P_REL = P_NODE;
   95   1  0     L00 = LEV;
   96   1  0     B = '1'B;

   97   1  0     DO LEV = L00+1 TO L0-1;
   98   1  1       Z = 1;
   99   1  1       P_NODE = P_NEW(LEV);
  100   1  1       FP_UG = P_UG; FP_TPM = P_TPM;
  102   1  1       CALL GET_R_PZ;
  103   1  1       DO Z=2 TO NZ;
  104   1  2         CALL GET_PZ;
  105   1  2         END;
  106   1  1       END;

  107   1  0     LEV = L00;
  108   1  0     P_NODE = P_REL;
  109   1  0     P_REL = P_ROOT;
  110   1  0     Y = Z_STRING(LEV+1);
  111   1  0     P2 = P_ESS_NODE_1;
  112   1  0     B = '0'B;
  113   1  0     IF P_ESS_NODE=NULL THEN GOTO NEXT_SCAN;

  114   1  0   LOOP2:
                 IF P_NEXTZ(Y)=NULL THEN GOTO NEXT_SCAN;
  115   1  0     P=P_NODE;
  116   1  0     DO I=0 TO LEV;
  117   1  1       Z_STRING(I) = P->Z_BACK;
  118   1  1       P = P-> P_BACK;
  119   1  1       END;
  120   1  0     P1 = P_REL;
  121   1  0     Z = Y;
  122   1  0     FP_UG = P_UG; FP_TPM = P_TPM;
  124   1  0     CALL GET_R_PZ;

  125   1  0   NEXT_SCAN:
                 CALL SCAN;
  126   1  0     IF P_NODE¬=NULL
                 THEN GOTO LOOP2;

  127   1  0   FINISHED:                               /* RESTORE CALLING Z,P_NODE    */
                 P_NODE = P0;
  128   1  0     Z = Z_ADD;
  129   1  0     RETURN;
```

PL/I OPTIMIZING COMPILER   ADDNODE: PROC REORDER;

```
STMT LEV NT

 130   1  0   GET_R_PZ: PROC;              /* THIS ENTRY COMPUTES R(*) FIRST      */
 131   2  0     DO I=1 TC N;
 132   2  1       R(I) = ROWSUM(I);
 133   2  1       END;

 134   2  0   GET_PZ: ENTRY;                          /* COMPUTE P_NEXTZ,PZ          */
 135   2  0     P =  P_ROOT->P_BRANCHES->F_P_BRANCH(Z);

 136   2  0     IF P=NULL
                THEN GOTO COMP2;

 137   2  0    IF B
               THEN DO;
 138   2  1     P2 = NULL;
 139   2  1     DO I = LEV TO 0 BY -1;
 140   2  2   TOP:
                    PP_BRANCHES = ADDR(P->P_BRANCHES->F_E_BRANCH(Z_STRING(I)));
 141   2  2         P1 = F_P_BRANCH(1);
 142   2  2         IF ¬F_E_BRANCH(1)
                    THEN RETURN;

 143   2  2         IF P1 = NULL
                    THEN DO;
 144   2  3           P2 = P;
 145   2  3           P = P_ROOT;
 146   2  3           GOTO TOP;
 147   2  3           END;
 148   2  2         P = P1;
 149   2  2         END;

 150   2  1      IF P2 = NULL
                 THEN DO;
 151   2  2        P2 = P1;
 152   2  2        P1 = P_ROOT;
 153   2  2        END;
 154   2  1     END;
```

```
PL/I OPTIMIZING COMPILER          ADDNODE: PROC REORDER;

   STMT LEV NT

                                                  /* COMPUTE PZ WHERE P1,P2 ARE
                                                     Z(P_NODE)||Z = Z(P1)||Z(P2)
                                                     P_NEXTZ(Z) = P2              */
    155   2  0      IF P_NEXTZ(Z)=P2
                    THEN RETURN;
    156   2  0      P_NEXTZ(Z) = P2;
    157   2  0      DO J=1 TO N;
    158   2  1        S = 0.;
    159   2  1        FP_TPM  = ADDR(P1->P_TPM->F_TPM(J));
    160   2  1        FP_PZ = ADDR(P_PZ->F_PZ((Z-1)*N*N+J));
    161   2  1        S = P2 -> ROWSUM(J);
    162   2  1        II = 1;
    163   2  1        DO I=1 TO N;
    164   2  2          IF F_UG(I)¬=0
                        THEN F_PZ(II) = F_TPM(II) * S / R(I);
    165   2  2          II = II+N;
    166   2  2          END;
    167   2  1        END;
    168   2  0      RETURN;
                                                  /* COMPUTE PZ WHERE
                                                     P_NEXTZ(Z) = P_ROOT          */
    169   2  0    COMP2:
                    BB = '0'B;
    170   2  0      IF P_NEXTZ(Z)=P_ROOT
                    THEN RETURN;
    171   2  0      P_NEXTZ(Z)=P_ROOT;
    172   2  0      DO I=1 TO N;
    173   2  1        IF F_UG(I)¬=0
                      THEN DO;
    174   2  2         FP_TPM = ADDR(P_TPM->F_TPM((I-1)*N+1));
    175   2  2         FP_PZ = ADDR(P_PZ->F_PZ((Z-1)*N*N+(I-1)*N+1));
    176   2  2         DO J=1 TC N;
    177   2  3          FP_PROBS = ADDR(PROBS(Z,1,J));
    178   2  3          S = 0.;
    179   2  3          II=1;
    180   2  3          DO K=1 TO N;
    181   2  4            S = S + F_TPM(K) * FP_PROBS->F_PZ(II);
    182   2  4            II = II+N;
    183   2  4            END;
    184   2  3          F_PZ(J) = S/R(I);
    185   2  3          BB = BB|S>0;
    186   2  3          END;
    187   2  2         END;
    188   2  1        END;
    189   2  0      IF ¬BB THEN P_NEXTZ(Z)=NULL;
    190   2  0      RETURN;
    191   2  0      END;
    192   1  0    END;
```

```
PL/I OPTIMIZING COMPILER          PREP_H: PROC REORDER;

                         SOURCE LISTING

     STMT LEV NT

        1       0   PREP_H: PROC RECRDER;

        2    1  0  %INCLUDE DD1(DCL);

                             /**************************************************/
                             /*                                                */
                             /*         CCMPUTE UH AND REC. *                  */
                             /*                                                */
                             /**************************************************/

        4    1  0     DCL (BU(0:NU),BZ(0:NZ),B,BB)  BIT ALIGNED, PP POINTER,
                            (S,T) FLOAT BIN, I FIXED BIN;

                          /* STEPO  COMPUTE UH = MOST LIKELY CPTIMAL INPUT          */
                          /*                 P_REC = LIKELY G-RECURRENT NODE        */

        5    1  0     P_NODE, P_REC = P_ESS_NODE_1;
        6    1  0   LOOPO:
                      PP_UG = P_UG;
        7    1  0     T = -1.;
        8    1  0     DC U=1 TO NU;
        9    1  1       S=0.;
       10    1  1       DO I=1 TO N;
       11    1  2         IF P_UG(I) = U
                          THEN S = S + ROWSUM(I);
       12    1  2         END;
       13    1  1       IF S>T+1E-4
                        THEN DO;
       14    1  2         UH = U;
       15    1  2         T = S;
       16    1  2         END;
       17    1  1       END;

       18    1  0     IF REC.H THEN P_REC = P_NODE;
       19    1  0     REC.G, REC.H = '0'B;

       20    1  0     P_NODE = P_NEXT_ESS_NODE;
       21    1  0     IF P_NODE ¬= NULL
                      THEN GOTO LOOPO;

       22    1  0     BB = '1'B;                        /* FIRST PASS FINDS REC.G      */
```

```
PL/I OPTIMIZING COMPILER          PREP_H: PROC REORDER;

    STMT LEV NT

                       /* STEP1  SET P_REL = LIKELY RECURRENT NODE AND SET RTO=0  */

      23   1  0    STEP1:
                     P_NODE = P_ESS_NODE_1;

      24   1  0    LOOP1:
                     P_REC = P_NODE;
      25   1  0      REC.TO = '0'B;
      26   1  0      P_NODE = P_NEXT_ESS_NODE;
      27   1  0      IF P_NODE¬=NULL
                     THEN GOTO LOOP1;

                       /* STEP2  SET REC.FROM = 0                                 */

      28   1  0    STEP2:
                     P_NODE = P_ESS_NODE_1;
      29   1  0    LOOP2:
                     REC.FROM = '0'B;
      30   1  0      P_NODE = P_NEXT_ESS_NODE;
      31   1  0      IF P_NODE ¬= NULL
                     THEN GOTO LOOP2;

      32   1  0    RPT2:
                     P_NODE = P_REC;
      33   1  0      REC.TO,REC.FROM = '1'B;
```

```
STMT LEV NT

                        /* STEP3  FILL REC.TO AND REC.FROM                                      */

  34    1   0   RPT3:
                  B = '0'B;
  35    1   0     P_NODE = P_ESS_NODE_1;
  36    1   0   LOOP3:
                  IF ((¬REC.TO|REC.FROM) & (BB|REC.G))
                  THEN DO;
  37    1   1      BZ = '0'B;
  38    1   1      IF BB THEN DO;
  39    1   2       BU = '0'B;
  40    1   2       PP_UG = P_UG;
  41    1   2       DO I=1 TO N;
  42    1   3         BU(F_UG(I)) = '1'B;
  43    1   3         END;
  44    1   2       DO U=1 TO NU;
  45    1   3         IF BU(U)
                      THEN DO Y=1 TO NY;
  46    1   4           BZ( ZCODE(U,Y)) = '1'B;
  47    1   4           END;
  48    1   3         END;
  49    1   2       END;
  50    1   1      ELSE DO Y=1 TO NY;
  51    1   2       BZ(ZCODE(UH,Y)) = '1'B;
  52    1   2       END;
  53    1   1      DO Z=1 TO NZ;
  54    1   2         IF BZ(Z)
                      THEN DO;
  55    1   3           PP = P_NEXTZ(Z);
  56    1   3           IF PP¬=NULL
                        THEN DO;
  57    1   4             PP = PP->P_ESS_NODE;
  58    1   4             IF (¬REC.TO)&PP->REC.TO
                          THEN B,REC.TO = '1'B;
  59    1   4             IF (¬PP->REC.FROM)&REC.FROM
                          THEN B,PP->REC.FROM = '1'B;
  60    1   4             END;
  61    1   3           END;
  62    1   2         END;
  63    1   1       END;
  64    1   0     P_NODE = P_NEXT_ESS_NODE;
  65    1   0     IF P_NODE ¬= NULL
                  THEN GOTO LOOP3;

  66    1   0     IF B THEN GOTO RPT3;
```

```
STMT LEV NT

                        /* STEP4 CHECK FOR CHAINS NOT CONTAINING P_REL          */

  67   1  0     PP = NULL;
  68   1  0     P_NODE = P_ESS_NODE_1;
  69   1  0   LOOP4:
                IF REC.FROM & (¬REC.TO) & (BB|REC.G)
                THEN DO;
  70   1  1       P_REC = P_NODE;
  71   1  1       GOTO STEP2;
  72   1  1       END;
  73   1  0     IF (¬REC.TO) & (¬REC.FROM) & (BB|REC.G) THEN PP=P_NODE;
  74   1  0     P_NODE = P_NEXT_ESS_NODE;

  75   1  0     IF P_NODE ¬= NULL
                THEN GOTO LOOP4;

  76   1  0     IF PP¬=NULL
                THEN DO;
  77   1  1       P_REC = PP;
  78   1  1       GOTO RPT2;
  79   1  1       END;

                        /* STEP5  FILL IN REC.G/REC.H (ACCORDING TO BB)         */

  80   1  0     P_NODE = P_ESS_NODE_1;
  81   1  0   LOOP5:
                IF BB
                THEN REC.G = REC.TO & REC.FROM;
  82   1  0     ELSE REC.H = REC.G & REC.TO & REC.FROM;
  83   1  0     P_NODE = P_NEXT_ESS_NODE;
  84   1  0     IF P_NODE ¬= NULL
                THEN GOTO LOOP5;

  85   1  0     IF BB
                THEN DO;
  86   1  1       BB = '0'B;
  87   1  1       GOTO STEP1;
  88   1  1       END;
  89   1  0     END;
```

```
PL/I OPTIMIZING COMPILER          SOLVE_G: PROC REORDER;

                    SOURCE LISTING

    STMT LEV NT

      1       0   SOLVE_G: PROC REORDER;

      2   1   0  %INCLUDE DD1(DCL);
      4   1   0     DCL (I,J) FIXED BIN, (S,SS,T,TOL) FLOAT BIN, (P,P_LHS) POINTER,
                          RT LABEL(RT_G,RT_H), (B,BB) BIT ALIGNED;
      5   1   0     DCL (WRK(N),WRK2(N)) FLOAT BIN;     /* LHS MAX AND 2ND MAX, STEP G */
      6   1   0     DCL STRUCT_FLAG(N) FIXED BIN;       /* DO DP FOR PREFIX I?         */

      7   1   0     P_LHS = ADDR(WRK(1)); FP_FLAG = ADDR(STRUCT_FLAG(1));
      9   1   0     RT = RT_G;
     10   1   0     G.STEPS,H.STEPS=0;
     11   1   0     TOL = ERR*1E-3;
     12   1   0     ERR = 1E10;

     13   1   0     P_NODE = P_ESS_NODE_1;
     14   1   0   G_LOOP0:
     15   1   0     FP_W = P_W; FP_UG = P_UG;
     16   1   0     DO I=1 TO N;
     17   1   1       DP_SKIP(I) = SIGN(F_UG(I)) - 1;
     18   1   1       END;
     19   1   0     P_NODE = P_NEXT_ESS_NODE;
     20   1   0     IF P_NODE¬=NULL
                    THEN GOTO G_LOOP0;
```

```
STMT LEV NT

  21   1  0   G_LOOP:                                                                    SOLV0250
                G.HIGH = -1E10;                                                          SOLV0260
  22   1  0     G.LOW = 1E10;                                                            SOLV0270
  23   1  0     G.STEPS = G.STEPS+1;                                                     SOLV0280
  24   1  0     TOL = TOL*1.2;                                                           SOLV0290
  25   1  0     S=0.;                                                                    SOLV0300
  26   1  0     P_NODE = P_ESS_NODE_1;                                                   SOLV0310
                                                                                         SOLV0320
  27   1  0   G_LOOP1:                  /* COMPUTE VG = MAX/U/ Q(U) + SUM/Y/ PZ VH  */SOLV0330
  28   1  0     FP_UG = P_UG; FP_VG = P_VG; FP_W = P_W;                                  SOLV0340
  30   1  0     DO I=1 TO N;                                                             SOLV0350
  31   1  1       F_VG(I) = -1.E5;                                                       SOLV0360
  32   1  1       END;                                                                   SOLV0370
                                                                                         SOLV0380
  33   1  0     DO U=1 TO NU;                                                            SOLV0390
  34   1  1       FP_QZ = ADDR(P_QZ->F_QZ((U-1)*N+1));                                   SOLV0400
                                                                                         SOLV0410
  35   1  1       BB = '0'B;                                                             SOLV0420
  36   1  1       DO I=1 TO N;                                                           SOLV0430
  37   1  2         B = DP_SKIP(I)=0 | (F_UG(I)=U&DP_SKIP(I)>0);                         SOLV0440
  38   1  2         IF B                                                                 SOLV0450
                    THEN DO;                                                             SOLV0460
  39   1  3           STRUCT_FLAG(I) = 1;                                                SOLV0470
  40   1  3           WRK(I) = F_QZ(I);                                                  SOLV0480
  41   1  3           BB = '1'B;                                                         SOLV0490
  42   1  3           END;                                                               SOLV0500
  43   1  2         ELSE STRUCT_FLAG(I) = 0;                                             SOLV0510
  44   1  2       END;                                                                   SOLV0520
  45   1  1     IF ¬BB THEN GOTO NEXT_U;                                                 SOLV0530
                                                                                         SOLV0540
  46   1  1       GOTO DP_OP;                                                            SOLV0550
  47   1  1   RT_G:                                                                      SOLV0560
                  DO I=1 TO N;                                                           SOLV0570
  48   1  2         IF DP_SKIP(I)=0                                                      SOLV0580
                    THEN DO;                                                             SOLV0590
  49   1  3          IF WRK(I)>F_VG(I)                                                   SOLV0600
                     THEN DO;                                                            SOLV0610
  50   1  4           WRK2(I) = F_VG(I);                                                 SOLV0620
  51   1  4           F_VG(I) = WRK(I);                                                  SOLV0630
  52   1  4           F_UG(I) = U;                                                       SOLV0640
  53   1  4           END;                                                               SOLV0650
  54   1  3          ELSE WRK2(I) = MAX(WRK(I),WRK2(I));                                 SOLV0660
  55   1  3          END;                                                                SOLV0670
  56   1  2         END;                                                                 SOLV0680
  57   1  1   NEXT_U:                                                                    SOLV0690
                  END;                                                                   SOLV0700
```

```
PL/I OPTIMIZING COMPILER          SOLVE_G: PROC REORDER;

    STMT LEV NT

      58   1  0      DO I=1 TO N;
      59   1  1        IF DP_SKIP(I)>0
                       THEN DO;
      60   1  2          DP_SKIP(I) = DP_SKIP(I) - 1;
      61   1  2          F_VG(I) = WRK(I);
      62   1  2          END;
      63   1  1        ELSE IF DP_SKIP(I)=0
                             THEN DP_SKIP(I) = MIN(100.,(F_VG(I)-WRK2(I))/ERR);
      64   1  1        IF F_UG(I)¬=0 THEN S = (S+F_VG(I))*.5;
      65   1  1        END;
      66   1  0      P_NODE = P_NEXT_ESS_NODE;
      67   1  0      IF P_NODE¬=NULL
                     THEN GOTO G_LOOP1;

      68   1  0      P_NODE = P_ESS_NODE_1;
      69   1  0    G_LOOP2:    /*      VH = VG - S   AND GET ODONI BOUNDS                      */
      70   1  0      FP_UG = P_UG; FP_VG = P_VG; FP_VH = P_VH;
      72   1  0      DO I=1 TO N;
      73   1  1       IF F_UG(I) ¬= 0
                      THEN DO;
      74   1  2        SS = F_VG(I) - F_VH(I);
      75   1  2        G.HIGH = MAX(SS,G.HIGH); G.LOW = MIN(SS,G.LOW);
      77   1  2        F_VH(I) = (F_VG(I)+F_VH(I)-S)*.5;
      78   1  2        END;
      79   1  1       END;
      80   1  0      P_NODE = P_NEXT_ESS_NODE;
      81   1  0      IF P_NODE ¬=NULL
                     THEN GOTO G_LOOP2;

      82   1  0      CALL TIMING(TIME.G);
      83   1  0      IF TIME.G > TIME.LIMIT THEN RETURN;

      84   1  0      ERR = G.HIGH - G.LOW;
      85   1  0      IF ERR > TOL
                     THEN GOTO G_LOOP;

      86   1  0      RETURN;
```

```
STMT LEV NT

  87   1  0   SOLVE_H: ENTRY;
  88   1  0     RT = RT_H;
  89   1  0     TOL = TOL*1E-2;

  90   1  0   H_LOOP:
  91   1  0     H.HIGH =-1E10; H.LOW = 1E10;
  92   1  0     H.STEPS = H.STEPS+1;
  93   1  0     TOL = TOL*2;
  94   1  0     S=0.;
  95   1  0     P_NODE = P_ESS_NODE_1;
  96   1  0   H_LOOP1:
                IF ¬REC.H THEN GOTO H_OUT1;
  97   1  0     FP_FLAG = P_UG; FP_W,P_LHS = P_W;
  99   1  0     U = UH;
 100   1  0     FP_QZ = ADDR(P_QZ->F_QZ((U-1)*N+1));
 101   1  0     DO I=1 TO N;
 102   1  1       IF   FLAG(I) ¬= 0
                  THEN F_W(I) = F_QZ(I);
 103   1  1       END;
 104   1  0     GOTO DP_OP;
 105   1  0   RT_H:
                DO I=1 TO N;
 106   1  1       IF   FLAG(I) ¬= 0
                  THEN S = (S+F_W(I))*.5;
 107   1  1       END;
 108   1  0   H_OUT1:
                P_NODE = P_NEXT_ESS_NODE;
 109   1  0     IF P_NODE¬=NULL THEN GOTO H_LOOP1;
 110   1  0     P_NODE = P_ESS_NODE_1;
 111   1  0   H_LOOP2:
                IF ¬REC.H THEN GOTO H_OUT2;
 112   1  0     FP_UG = P_UG; FP_W = P_W; FP_VH = P_VH;
 115   1  0     DO I=1 TO N;
 116   1  1      IF F_UG(I) ¬= 0
                 THEN DO;
 117   1  2       SS = F_W(I) - F_VH(I);
 118   1  2       H.HIGH = MAX(SS,H.HIGH); H.LOW = MIN(SS,H.LOW);
 120   1  2       F_VH(I) = (F_W(I)+F_VH(I)-S)*.5;
 121   1  2       END;
 122   1  1      END;
 123   1  0   H_OUT2:
                P_NODE = P_NEXT_ESS_NODE;
 124   1  0     IF P_NODE¬=NULL THEN GOTO H_LOOP2;
 125   1  0     CALL TIMING(TIME.H);
 126   1  0     IF TIME.H > TIME.LIMIT THEN RETURN;
 127   1  0     IF H.HIGH - H.LOW > TOL
                THEN GOTO H_LOOP;
 128   1  0     RETURN;
```

```
PL/I OPTIMIZING COMPILER          SOLVE_G: PROC REORDER;

    STMT LEV NT

     129   1  0   DP_OP:

                     /**********************************************************/
                     /* COMPUTE THE OPERATION OF DYNAMIC PROGRAMMING:           */
                     /*          W = Q(U)  + SUM/Y PZ(Z=(U,Y))*V_FEAS( T(*,Z))  */
                     /* IN NODE P_NODE WITH U AS SPECIFIED AT CALL TIME         */
                     /**********************************************************/

                   DO Y=1 TO NY;
     130   1  1      Z = ZCODE(U,Y);
     131   1  1      IF Z¬=0
                     THEN DO;
     132   1  2        P = P_NEXTZ(Z);
     133   1  2        IF P¬=NULL
                       THEN DO;
     134   1  3         FP_VH = P->P_ESS_NODE->P_VH;
     135   1  3         P = ADDR(P_PZ->F_PZ((Z-1)*N*N+1));
     136   1  3         DO I=1 TO N;
     137   1  4          IF   FLAG(I) ¬= 0
                         THEN DO;
     138   1  5            FP_PZ = ADDR(P->F_PZ((I-1)*N+1));
     139   1  5            SS = 0.;
     140   1  5            DO J=1 TO N;
     141   1  6              SS= SS+ F_PZ(J) * F_VH(J);
     142   1  6              END;
     143   1  5            P_LHS->F_W(I) = P_LHS->F_W(I) + SS;
     144   1  5            END;
     145   1  4          END;
     146   1  3         END;
     147   1  2        END;
     148   1  1      END;
     149   1  0   GOTO RT;
     150   1  0   END;
```

```
PL/I OPTIMIZING COMPILER          REPORT: PROC REORDER;

                        SOURCE LISTING

     STMT LEV NT

        1      0   REPORT: PROC REORDER;

        2    1  0 %INCLUDE DD1(DCL);
                                /****************************************/
                                /*                                      */
                                /*              PRINT RESULTS           */
                                /*                                      */
                                /****************************************/
        4    1  0    DCL (I,J) FIXED BIN, P POINTER, C CHAR(1) ALIGNED;
        5    1  0    DCL SCAN EXT ENTRY;

        6    1  0    SIGNAL ENDPAGE(SYSPRINT);

        7    1  0    ERR = G.HIGH - H.LOW + 1.E-10;
        8    1  0    P_NODE,P_REL=P_ROOT;
        9    1  0    LEV,LO,LOO = 0;
       10    1  0    IF P_ESS_NODE¬=NULL
                     THEN GOTO PD;

       11    1  0  LOOP:
                     CALL SCAN;
       12    1  0    IF P_NODE¬=NULL
                     THEN GOTO PD;

       13    1  0    IF ERR<= MIN_ERR | M >= MAX_M | ESS_M >= MAX_ESS_M
                            | TIME.G >= TIME.LIMIT
                     THEN DO;
       14    1  1      PUT EDIT('|','| *STOP*') (COL(1),A,COL(86),A);
       15    1  1      STOP;
       16    1  1      END;
       17    1  0    RETURN;
```

```
PL/I OPTIMIZING COMPILER         REPORT: PROC REORDER;

    STMT LEV NT

     18    1  0    PD:
                     IF LINENO(SYSPRINT) > 55-N*FMT
                     THEN SIGNAL ENDPAGE(SYSPRINT);
     19    1  0      PUT SKIP(2);

     20    1  0      IF REC.G THEN PUT EDIT('G') (COL(14),A);
     21    1  0      IF REC.H THEN PUT EDIT('H') (A);

     22    1  0      J = UH;
     23    1  0      PUT EDIT(J)  (COL(19),F(3));

     24    1  0      FP_UG = P_UG;
     25    1  0      C = '*';
     26    1  0      DO I=1 TO N;
     27    1  1        IF F_UG(I)¬=0 & F_UG(I)¬=J
                       THEN DO;
     28    1  2          C = ' ';
     29    1  2          GOTO STAR_OUT;
     30    1  2          END;
     31    1  1        END;
     32    1  0    STAR_OUT:
                     PUT EDIT(C) (A);

     33    1  0      IF P_NODE = P_ROOT
                     THEN PUT EDIT('<E>') (COL(73),A);
     34    1  0      ELSE DO;
     35    1  1          PUT EDIT(Z_BACK)  (COL(MAX(1,76-LEV*3)), F(3));
     36    1  1          P = P_NODE;
     37    1  1          DO I=LEV-2 TO LO BY -1;
     38    1  2            P = P->P_BACK;
     39    1  2          PUT EDIT(P->Z_BACK) (F(3));
     40    1  2            END;
     41    1  1          END;

     42    1  0      IF FMT=0
                     THEN GOTO LOOP;

     43    1  0      FP_TPM = P_TPM; FP_VG = P_VG; FP_VH = P_VH;
     46    1  0      DO I=1 TO N;
     47    1  1        IF F_UG(I)¬=0
                       THEN DO;
     48    1  2          PUT EDIT(I,F_UG(I),F_VG(I)) (COL(16),2 F(3),F(6,2));
     49    1  2          IF REC.H THEN PUT EDIT(F_VH(I))  (F(6,2));
     50    1  2          PUT EDIT((F_TPM((I-1)*N+J) DO J=1 TO N)) (COL(34),5 F(8,4));
     51    1  2          END;
     52    1  1        END;
     53    1  0      GOTO LOOP;
     54    1  0      END;
```

```
PL/I OPTIMIZING COMPILER         SCAN: PROC REORDER;

                         SOURCE LISTING

    STMT LEV NT

       1       0   SCAN: PROC REORDER;

       2    1  0  %INCLUDE DD1(DCL);
                           /* FIND NEXT ESS NODE AFTER P_NODE IN TREE ORDER     */

       4    1  0     DCL I FIXED BIN;
       5    1  0     LO = LEV;
       6    1  0   NEW_NODE:
                     I = 0;
       7    1  0   CLIMB:
                     FP_BRANCHES = P_BRANCHES;
       8    1  0     DO Z = I+1 TO NZ;
       9    1  1       IF F_E_BRANCH(Z) & F_P_BRANCH(Z)¬=NULL
                       THEN GOTO NEXT_LEV;
      10    1  1       END;
                                                       /* ALL BRANCHES HAVE BEEN
                                                          EXPLORED, GO BACK DOWN      */
      11    1  0   DOWN:
                     IF LEV=LO0
                     THEN DO;
      12    1  1       P_NODE = NULL;
      13    1  1       RETURN;
      14    1  1       END;
      15    1  0     LEV,LO = LEV-1;
      16    1  0     I = Z_BACK;
      17    1  0     P_NODE = P_BACK;
      18    1  0     P_REL = P_REL -> P_BACK;
      19    1  0     GOTO CLIMB;
                                                       /* CLIMB BRANCH Z              */
      20    1  0   NEXT_LEV:
                     LEV = LEV+1;
      21    1  0     P_NODE = F_P_BRANCH(Z);
      22    1  0     P_REL = P_REL -> P_BRANCHES -> F_P_BRANCH(Z);
      23    1  0     FP_BRANCHES = P_BRANCHES;
      24    1  0     DO Z = 1 TO NZ;
      25    1  1       IF F_E_BRANCH(Z) & F_P_BRANCH(Z)=NULL
                       THEN RETURN;
      26    1  1       END;
      27    1  0     GOTO NEW_NODE;
      28    1  0     END;
```

SYMBOL TABLE

$a[P]$, $a[\underline{z}]$,	D-sense spread of normalized range, 100
$\bar{a}$	Detectability index, 49,53,115
$\{b(k)\}$	Finite-horizon weights, 28
C	Connected class of states, 81
C(k)	Detectable classification of states at time k, 118
D	Metric on Π_N,87,94-96
e^i	Unit vector, 11
$\underline{e}$	Empty word, 62
ess[M]	Essential part of memory set, 70
$E_\gamma\{\cdot\}$	Expectation under strategy γ, 26
$g(b,\gamma)$, $g(\beta,\gamma)$, $g(\gamma)$	Performance indices, 28
$g[M]$, g^n	Perceptive gain, 145
$h[M]$, h^n	Pseudo-perceptive gain, 147
$I(\underline{z})$, $J(\underline{z})$	Possible states (preceding, following) evolution of $\underline{z}$, 63
k	Time, 21-22
K	Horizon, 22
$\ell(\underline{z})$	Length of word $\underline{z}$, 62
ℓ_ρ, $\bar{\ell}$	(Reachability, detectability) time constant, 48-49, (82,115)
$L(\beta,\ell)$	Discounted time interval, 135
M	Memory set, 66
m	Value-iteration step, 128,136
N	Number of states, 21

n	Iteration number, 37-38,145
$P_{\underline{z}}^{M}(i,j,\underline{z})$	Transition probabilities of augmented system, 76-77
$P(y\|u)$	Transition probability matrix, 21
$Prob_{\gamma}\{\ \}$	Probability under strategy γ, 25-26
$q(k)$	Expected incremental reward at time k, 28
$q(u)$	Expected incremental reward vector, 29
$q_{\underline{z}}^{M}(i,u)$	Expected incremental rewards for augmented system, 76-77
Q_{max}, Q_{min}, Q	Bounds on expected incremental rewards, 29
$r(k)$	Reward at time k,
$row_i[P]$,	Row of a matrix, 11
R_N	N-dimensional Euclidean space, 11
$s(k)$	State at time k, 21
S	State set, 21
$T(\eta,u,y)$, $T(\eta,\underline{z})$	Information vector transition function, 26,64
$T^{M}(\underline{z},\underline{z}')$	Memory state transition function, 68
$u(k)$	Input at time k, 21
U	Input set, 21
$v^{k,K}$	Finite-horizon value function, 125
v^*	Infinite-horizon relative value function, 134
V	Banach space of continuous bounded real-valued functions on Π_N, 96
$x^{M}(k)$	Augmented state at time k, 75
$X[M]$	Augmented state set, 75

$\hat{X}[M]$	Connected class of augmented states, 83
y(k)	Output at time k, 21
Y	Output set, 21
$\underline{z}^{M}(k)$	Memory state at time k, 66
Z	Set of input-output pairs, 62
Z^{+}	Set of input-output words, 64
$(\ )^{+}$	Positive part, 11
$\langle a,b\rangle$, $[a,b]$, $[a,b)$	(Integers, reals) between a and b, 11
$-$	Subtraction of rightmost part of word, 65
$\circ$	Bayes' operator, 51,82
$\lvert\cdot\rvert$	Sum of vector components, 12
$\lVert\cdot\rVert$	Sup norm, 96
$\lVert\cdot\rVert_D$, $\lVert\cdot\rVert_\Delta$	Variation of convex function, 97-98
$\alpha[P]$, $\alpha[\underline{z}]$	Δ-sense contraction, 100
α, $\overline{\alpha}$	Detectability index, 53,106,109,112,114
β	Discount, 28
γ	Decision strategy, 24,78
δ	Hajnal measure, 87,93
Δ	Metric on Π_N, 87,89
$\eta(k)$	Information vector at time k, 26
$\mu(k)$	Number of detectable classes, 118
$\pi(0)$	Initial state probability vector, 21
Π_N, $\tilde{\Pi}_N$	Unit simplex of (stochastic, substochastic) vectors, 11

ρ	Reachability index, 48,82
$\sigma[\underline{z},\phi]$	Policy compatability flag, 114
τ	Elasticity of memory effectiveness, 16-17, 106,109,112,114
ϕ, $\overline{\phi}$	Feasible strategy and the policy that realizes it, 78
ϕ^M	Pseudo-perceptive strategy derived from ψ^M, 146-147
$\overline{\phi}^*$	Optimal feasible strategy, 134
$\Phi[M]$	Set of feasible strategies adapted to M, 78
χ	Connectivity index, 81-82
ψ, $\overline{\psi}$	Perceptive strategy and the policy that realizes it, 79
ψ^M	Optimal perceptive strategy adapted to M, 146
$\Psi[M]$	Set of perceptive strategies adapted to M, 79
Ω	Value of information, 50,135

GLOSSARY

accept: The action in which a system receives an input, 21.

augmentation: Transformation of an FPS to one having augmented states, 58,76.

augmented state: Transformed state consisting of a delayed internal state and a memory state, 57,75.

concatenation: Two or more words (strings) placed end to end so as to form a single word, 62.

connectivity: A relation between states i and j indicating that the system in state i may eventually enter state j provided that suitable inputs are selected in the interim, 81.

controller: A dynamical realization of the decision strategy, 24.

control problem: The problem of designing a controller which realizes an optimal or ε-optimal strategy, 31.

decision strategy: A (possibly probabilistic) rule for the selection of plant inputs, 24.

detectability: A condition under which the information vector is increasingly insensitive to increasingly delayed information, 105-106,53.

emit: The action in which an output is generated by the system, 21.

essential memory state: A memory state that is recurrent under some policy, 70.

estimation problem: The problem of recursively computing an estimator or sufficient statistic. In the case of an FPS, the estimator is the information vector, 30.

feasible: A strategy is feasible if it can be realized on the basis of available information; otherwise it is perceptive, 78.

finite-memory constraint: The constraint that a decision strategy be realizable by a finite-state automaton, 24.

finite probabilistic system: A discrete-time, finite-input, finite output finite-state stationary controlled stochastic process, 13,20-22.

FPS: See "finite probabilistic system."

free FPS: An FPS whose input set contains but one element, i.e. an FPS whose input process may be ignored.

free system induced by a decision strategy: The system which results when a plant and its controller are considered as a single unit, 23-24.

horizon: Length of the time set, 22.

infinitely delayed splurge: Phenomenon arising in the absense of detectability, 48,142.

information vector: A vector, which may be computed by an observer, whose i-th entry is the a posteriori probability that the system is in state i, 26.

information vector transition function: The rule by which an observer updates the information vector, 26.

memory set: A vocabulary of input-output words available to the observer, 57,65-66.

memory state: The word of most recent input-output pairs retained by the observer, 57,65-66.

memory state transition function: The rule by which an observer updates the memory state, 68.

memory tree: A graphical representation of the memory set, 66-68.

observer: A system which accepts plant outputs and computes the information vector (or an approximation thereof), 30.

perception: An output which has been artificially added to the plant to facilitate computation, 35,54.

plant: The system to be observed or controlled, 13,18.

policy: A finite array which specifies the decision strategy, 14,78-79.

pseudo-perception: An approximation to a perception, obtained by guessing the value of the perception on the basis of the memory state, 54,146.

reachability: A condition under which the state of an FPS can be made to assume a desired value with probability bounded from below, for any initial state probability vector, 48,82.

realization: Specification of system components which will act according to a given rule, e.g. a controller realizes a decision strategy, 14,24.

representation: Specification according to a particular system of notation, 20,22.

reward: The component of a performance index which depends on an particular input-output pair as well as the states preceeding and following it, 27; the expected incremental reward depends only an input and the state preceeding it, 28-29.

state-calculability: A possible FPS property, given by (2.3), 23.

state-observability: A possible FPS property, given by (2.4), 23.

statistical decision problem: A control problem in which plant dynamics are unaffected by input values, 30.

strategy: See decision strategy.

subrectangularity: A property of substochastic matrices given by (13.1) 99; also, a possible property of FPS's given by (14.1) and (14.7), 105,106,109.

SDT: Strong detectability.

SSR: Strong subrectangularity.

valued finite probabilistic system: An FPS, along with a process of incremental rewards or expected incremental rewards, making possible the definition of performance indices as a function of strategy, 28.

VFPS: See "valued finite probabilistic system."

WDT: Weak detectability.

WSR: Weak subrectangularity.