A PRIMER ON PARTIALLY OBSERVABLE MARKOV PROCESSES

by

JOSEPH A. AMRAM

Submitted in Partial Fulfillment
of the Requirements for the
Degree of Bachelor of Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May, 1982

Author: ..................................................
Joseph A. Amram; Department of Electrical Engineering.

Certified by: .................................
Alvin W. Drake; Thesis Supervisor.

Accepted by: .................................
David Adler; Chairman, Departmental Theses Committee.

Archives

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
SEP 27 1982
LIBRARIES
ACKNOWLEDGEMENT

The author is grateful to his thesis supervisor, Professor Alvin W. Drake, for his guidance, advice, inspiration and critical comments.
A Primer on Partially Observable Markov Processes.

by

Joseph A. Amram

Abstract.

The objective of this work is to integrate a set of models and issues pertaining to partially observable Markov processes.

A noisy channel model for monitoring the state of an underlying discrete-state discrete-trial Markov process is formulated. The channel produces an output symbol sequence which is probabilistically related to the underlying state of the Markov source. A number of decoding schemes for estimating the actual state of the process are presented. Decoder performance issues are discussed and bounds on the average probability of error for one such decoder that considers the entire past history of the channel output are obtained. For such a decoder, operating in the two state Markov source case, the limiting steady state distribution of the state of knowledge variable is derived, allowing for exact evaluation of its average probability of error.

An optimal control problem over the finite horizon is formulated, for processes with decisions and rewards. Dynamic programming techniques are shown to be of use in solving such problems. Other solution algorithms are mentioned. Information
theory considerations are raised, and an information flow equation is presented. An extension to previous analysis for evaluating the infinite-past decoder is developed.

Thesis Supervisor: Alvin W. Drake, Professor of System's Science and Engineering
# TABLE OF CONTENTS

Acknowledgement 2
Abstract 3
Table of Contents 5
Table of Figures 7
Notational Conventions 8

Part I. The Basic Models

1. The Discrete-State Discrete-Trial Markov Process 9
2. The Partially Observable Process 11

Part II. The Decoder and Performance Issues

3. Decoding Schemes Presentation
   3.1 The Singlet Decoder 16
   3.2 Finite Memory Decoders 17
   3.3 The Infinite-Past Decoder 23
   3.4 The Infinite-Past Decoder for Symmetric Case 26

4. Evaluation of Decoding Schemes
   4.1 Conditions for Differential Performance 29
   4.2 Performance Bounds 33
   4.3 Average Probability of Error of Infinite-Past Decoder in the Two-State Case 35

Part III. The Partially Observable Markov Decision Process with Rewards

5. The Problem and Basic Models
   5.1 The Markov Process with Transition Rewards 46
   5.2 The Markov Decision Process 47
5.3 The Problem of the Partially Observable Markov Decision Process 48

6. Solution Algorithms

6.1 Solution by Dynamic Programming 51
6.2 Sondik's Algorithm 55
6.3 Platzman's Algorithm for Computing $\epsilon$-Optimal Controllers 57

Part IV. Information Theory Considerations

7. Information Flow Equation

7.1 Derivation 59
7.2 An Example 61

Part V. Comments and Extensions 63

References 67
### TABLE OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>State transition diagram of a two-state process</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2</td>
<td>The noisy channel model</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Inadequate model for the R(n) sequence generation</td>
<td>15</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Sample space of a finite memory decoder</td>
<td>19</td>
</tr>
<tr>
<td>Figure 5</td>
<td>State transition diagram for a finite memory decoder</td>
<td>21</td>
</tr>
<tr>
<td>Figure 6</td>
<td>A plot of $p_t$ for the triplet decoder</td>
<td>21</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Performance measure for the triplet decoder</td>
<td>22</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Differential performance condition</td>
<td>32</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Upper bound on performance of the infinite-past decoder</td>
<td>36</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Sample space used to obtain $P_1(u)$ and $T_1(u)$</td>
<td>36</td>
</tr>
<tr>
<td>Figure 11</td>
<td>A plot of $T_1(u)$ and $T_2(u)$</td>
<td>40</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Events that can generate $u(k+1)=v$</td>
<td>42</td>
</tr>
<tr>
<td>Figure 13</td>
<td>A plot of $h^{30}(u)$</td>
<td>43</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Average probability of error of the infinite-past decoder</td>
<td>45</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Probabilistic propagation of $u$</td>
<td>52</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Optimal total expected reward in example</td>
<td>54</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Reduction of the N-state model to a two-state one</td>
<td>65</td>
</tr>
</tbody>
</table>
Notational Conventions

\( P[A] \) \hspace{1cm} \text{Denotes the probability of event } A. \\
\( P[A / B] \) \hspace{1cm} \text{Denotes the probability of event } A, \text{ given that event } B \text{ occurred.} \\
\arg \max_y z \quad \text{Denotes the argument } y \text{ corresponding to the maximum of the expression } z \text{ e.g.} \\
\arg \max_y [- (y-1)^2] = 1 \\
\text{or, for the vector } u, \quad u = [.3 \quad .7], \text{ we have} \\
\arg \max_{S_1} [u_i] = S_2
I. The Basic Models.


The two essential concepts in the modeling of dynamical systems are those of state and state transition. The concept of state allows us to focus on the aspects of the system that are essential to the problem at hand, while the concept of state transition provides the mechanism for modeling and understanding the system's dynamic behavior. In many situations, there is an element of uncertainty in the transition of the process from one state to another, and this leads naturally to the use of the Markov process as a model of the system.

In the discrete state discrete trial Markov process, which we shall consider here, the system may be described at any time as being in any one of a set of mutually exclusive collectively exhaustive states $S_1, S_2, \ldots, S_N$. The process undergoes a series of discrete probabilistic "trials" in which the probability of transition to state $S_j$ after the next "trial", given that the system currently occupies state $S_i$, is a function of $i$ and $j$ only and not of any additional history of the system before its arrival at $S_i$. In other words, we may specify a set of conditional probabilities, $p_{ij}$, that the system, which now occupies state $S_i$, will occupy state $S_j$ after its next trial.

If we let $S_j(n)$ denote the event "Markov process is in state $S_j$ immediately after the $n$'th trial", we can formally state the above condition, known as the Markov condition, as follows:
\[ P[S_j(n) / S_a(n-1) S_b(n-2) \ldots] = P[S_j(n) / S_a(n-1)] = p_{aj} \quad \text{for all possible} \quad n,j,a,b,\ldots \]

Since the system must be in some state after its next trial, we have

\[ \sum_j p_{ij} = 1 \quad \text{for all} \quad 1 \leq i \leq N \quad . \]

As a simple example of a discrete state discrete trial Markov process, consider the following manufacturing situation. A machine is producing a product once an hour at the end of the hour. The machine is subject of some random vibrating noise at the end of the manufacturing process which may change the tuning state of one of its parts. The tuning state of the part at the beginning of the hour affects the quality of the item produced. To simplify matters we define two states in our model, \( S_1 \) (machine is properly tuned and output product is acceptable) and \( S_2 \) (machine is out of tune and output product is unacceptable).

Let us assume that when the machine is properly tuned at the beginning of the hour there is a probability of \( p_{11} = .9 \) that the machine will remain tuned at the beginning of the next hour, consequently there is a probability \( p_{12} = .1 \). When the machine is out of tune at the beginning of the hour, it may remain out of tune after an hour with probability \( p_{22} = .9 \) or may return to state \( S_1 \) with probability \( p_{21} = .1 \). In a matrix form we have,

\[
[p_{ij}] = \begin{bmatrix}
.9 & .1 \\
.1 & .9
\end{bmatrix}
\]
A corresponding transition diagram of the system showing the states and transition probabilities in a graphical form is given in Figure 1.

The above is a two state example of a more general symmetric $N$-state Markov process for which we have,

$$p_{ij} = \begin{cases} p & i=j \\ (1-p)/(N-1) & i \neq j \end{cases}.$$  

The above symmetric condition implies that at any given time, the probability that the process will remain in its present state after the next trial is $p$, and the probability of transition into any other particular state is $(1-p)/(N-1)$. In the following sections, for the sake of analytic simplicity, we shall limit our discussion to this symmetric case. For simplicity, we shall further require $p \gg 1/N$.


Unfortunately, in many practical applications of the discrete state discrete trial Markov model presented, we are not permitted exact observation of the state of the process. This may be due to a prohibitively large cost of exact observation or noise inherent in the measurement process itself. In such cases, we can often model what is observable as probabilistically related to the true state of the system. Such a model, known as the noisy channel model, will be considered here.

The noisy channel model will consider the state $S_i$, resulting from any trial of the underlying Markov process, to
Figure 1. State transition diagram for a two-state Markov process in manufacturing example.

$S_1$: Next item produced by machine will be satisfactory.

$S_2$: Next item produced by machine will be unsatisfactory.

Figure 2. The noisy channel model for the case $N=M=2$. 
serve as an "input symbol" into a noisy memoryless channel with a mutually exclusive collectively exhaustive set of output symbols \( R_1, R_2, \ldots, R_M \). The noisy channel which produces the observable output letters \( R_1, R_2, \ldots, R_M \) from the underlying states of the Markov process \( S_1, S_2, \ldots, S_N \) is described by the set of conditional probabilities \( f_{ij} \), defined for \( i=1,2,\ldots,N \) and \( j=1,2,\ldots,M \). The quantity \( f_{ij} \) defines the conditional probability that the observable output letter will be \( R_j \) when the underlying process generated the channel input symbol \( S_i \). If we let \( R_j(n) \) denote the event "channel output symbol at the end of n'th trial is \( R_j \)", the memoryless noisy channel model can be formulated as follows:

\[
P[R_j(n)/S_i(n)] = P[R_j(n)/S_i(n)S_a(n-1)R_b(n-1)S_c(n-2)R_d(n-2)\ldots] = f_{ij} \text{ for all possible } n,i,j,a,b,c,d,\ldots
\]

Since the channel must produce an output symbol at the end of every trial, we have

\[
\sum_j f_{ij} = 1 \quad \text{for all } 1 \leq i \leq N.
\]

Figure 2 presents a pictorial representation of the noisy channel model for the case \( N=M=2 \).

In our machine example, if every item produced at the end of the hour is subject to a quick imperfect inspection, we may define the output symbols \( R_1 \) ("item seems good") and \( R_2 \) ("item seems bad"). Let us suppose that when the item is in fact good and the machine is properly tuned there is a probability of \( f_{11} = .7 \) that the item will "seem good" in our imperfect test. Consequently there is a conditional probability of \( f_{12} = 1-f_{11} = .3 \)
that the item will "seem bad" when it is actually good. Assume also that when the item produced is bad there is a probability of \( f_{22} = .7 \) that it will in fact seem bad and a probability of \( f_{21} = 1 - f_{22} = .3 \) that it will seem good. (The symmetry is assumed for simplicity.)

The above is a two state example of a more general symmetric noisy channel for which

\[
f_{ij} = \begin{cases} 
  f & \text{if } i=j \\
  (1-f)/(M-1) & \text{if } i \neq j
\end{cases}
\]

In the symmetric channel case we have a probability \( f \) of a correct transmission through the channel and a \((1-f)\) chance, equally distributed among the other \((M-1)\) output symbols, of incorrect transmission.

We can proceed to compute the conditional probabilities \( P[R_i(n) / R_j(n-1) R_k(n-2)] \) and \( P[R_i(n) / R_j(n-1)] \) which are not equal in general. This can easily be verified for a numerical example, implying that the sequence of output symbols could not be generated (i.e. have the same statistical characteristics) by an "equivalent" Markov process of the form indicated in Figure 3.

The following sections of this paper will present some decoding schemes designed to estimate the state of the underlying process, given the string of output symbols. Some performance issues will be considered for the symmetric case. For simplicity, we shall further require that \( M = N \) and \( f \gg 1/M \).
Figure 3. We cannot model the generation of the $R(n)$ sequence in Figure 2 by attributing it to a source of this form.
II. The Decoder and Performance Issues.

3. Decoding Schemes Presentation.

It is the task of a decoder operating upon the channel output symbols to estimate the state of the underlying Markov process at time \( n \), given part or all of the following information:

1. The Markov source transition probability matrix \( [p_{ij}] \).
2. The channel characteristic transmission matrix \( [f_{ij}] \).
3. The string of past and future channel output symbols, \( \ldots R(n-1)R(n)R(n+1)\ldots \), which we shall denote by \( R \).

3.1 The Singlet Decoder.

One could decode each output symbol independently, using Bayes rule. This would amount to considering only the current output symbol, \( R(n) \), and ignoring all previous output symbols, when determining the input symbol most likely to have caused the present output symbol. In the steady state where the initial conditions of the process have been washed out, such decoding decisions would depend only on the steady state probabilities of the underlying process and the properties of the channel. If we denote the decoder output by \( J_1(R) \), we have
\[ J_1(R) = \arg \max_{S_i} P[S_i(n) / R_k(n)] = \]

\[ = \arg \max_{S_i} \frac{P[S_i(n)] P[R_k(n) / S_i(n)]}{P[R_k(n)]} \]

Eq 3-1 \[ J_1(R) = \arg \max_{S_i} P[S_i(n)] P[R_k(n) / S_i(n)] = \arg \max_{S_i} P_i f_{ik} \]

where we have denoted the steady state a priori probability that the underlying Markov process is in state \( S_i \) in a random instant, by \( P_i \).

For the symmetric case considered, this reduces to

Eq 3-2 \[ J_1(R) = \arg \max_{S_i} P[S_i(n) / R_k(n)] = S_k \]

Intuitively, since we require \( f > 1/N \), on reception of the symbol \( R_k \), the channel input symbol with the largest a posteriori probability is the input symbol \( S_k \).

The singlet decoding technique would produce an average probability of error of \( (1-f) \) and requires no storage, and essentially no computation. One can also note that for the case \( p = 1/N \), there is no other decoding procedure which offers any type of improvement over the singlet decoder since if \( p = 1/N \) there is no memory in the Markov process.

3.2 Finite Memory Decoders.

For the case \( p > 1/N \) the singlet decoder, previously described, did not utilize the memory properties of the underlying Markov process. To improve the performance of the decoder we may consider the past \( L \) output symbols, rather than
just the most recent one. The decoder, given a subset of length \( L \) of the string of output symbols \( R \), will select the symbol with the largest a posteriori probability of being the present input symbol. Formally

\[
J_L(R) = \arg \max_{S_1} P[S_1(n)/R(n) R(n-1) \ldots R(n-L+1)]
\]

\[
= \arg \max_{S_1} P[S_1(n) R(n) R(n-1) \ldots R(n-L+1)].
\]

Such a decoder can be considered as a finite state machine with \( N^L \) distinct states, corresponding to the different possible combinations of the last \( L \) output symbols. Each state has associated with it a decoder output symbol.

Let us consider an example of a doublet decoder where the decoder considers the last two output symbols and we have a two-state Markov source. For such a case we have \( N=L=2 \). Figure 4 presents the sample space for such a case. If we denote the last two output symbols by \( R_x(n) \) \( R_y(n-1) \), we can write the following expression for the decoder output \( J_2(R) \)

\[
J_2(R) = \arg \max_{S_1} P[S_1(n) R_x(n) R_y(n-1)]
\]

\[
= \arg \max_{S_1} \sum_j P_j f_{jy} p_{ji} f_{ix}
\]

\[
\text{Eq 3-4} \quad J_2(R) = \arg \max_{S_1} \sum_j f_{jy} p_{ji} f_{ix}
\]

where we have eliminated any constant factors from the term to be maximized. After a straightforward evaluation of Eq 3-4, under the condition \( p>1/2 \) and \( f>1/2 \), we can form the following table for the output of our finite memory decoder
Figure 4. Sample space of a finite memory decoder with $N=L=2$. 
<table>
<thead>
<tr>
<th>Received Sequence</th>
<th>Decoder Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1(n) R_1(n-1)$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>$R_1(n) R_2(n-1)$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>$R_2(n) R_1(n-1)$</td>
<td>$S_2$</td>
</tr>
<tr>
<td>$R_2(n) R_2(n-1)$</td>
<td>$S_2$</td>
</tr>
</tbody>
</table>

Figure 5 presents the state transition diagram for such a decoder. The states are labeled by the received symbol sequence and the corresponding decoder output.

From the table above we can see that this decoding technique results in all the same decoding decisions as with the singlet decoder, thus this decoder also would produce an average probability of error of $(1-f)$. This can be verified from the sample space in Figure 4. However, this doublet decoding technique would make the output observer more sure of some of the decoding decisions while less confident in others. Note also, that in our example, although the doublet decoder does not change the decoding decisions made by the singlet decoder, we might alter our policy in placing bets on the state of the underlying process.

Finite memory decoders that consider longer sequences of the channel output symbols can be shown (via gruesome algebra. Drake [1,pp25]) to make different decoding decisions than the singlet decoder, for some values of $p$ and $f$. Figure 6 plots the regions in the $p$-$f$ plan for which a triplet decoder considering $R(n) R(n-1) R(n-2)$, will perform differently (for at least two of its eight possible decoding decisions) than the singlet decoder. Figure 7 tabulates the percent reduction in the average probability of error obtained by using such a triplet decoder, as
Figure 5. State transition diagram for finite memory decoder with $N=L=2$.

Figure 6. Regions in the p-f plane in which the triplet decoder can outperform the singlet decoder. In the region above the curve drawn the triplet decoder will outperform the singlet decoder.
<table>
<thead>
<tr>
<th>p</th>
<th>f</th>
<th>0.60</th>
<th>0.70</th>
<th>0.80</th>
<th>0.85</th>
<th>0.90</th>
<th>0.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.975</td>
<td></td>
<td>10.0</td>
<td>23.0</td>
<td>38.0</td>
<td>44.0</td>
<td>50.0</td>
<td>35.0</td>
</tr>
<tr>
<td>0.95</td>
<td></td>
<td>8.8</td>
<td>19.0</td>
<td>29.0</td>
<td>31.0</td>
<td>15.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.90</td>
<td></td>
<td>5.0</td>
<td>10.0</td>
<td>12.0</td>
<td>6.4</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.85</td>
<td></td>
<td>2.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.80</td>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Figure 7. Per cent reduction in average probability of error obtained by triplet decoder considering $R(n) R(n-1) R(n-2)$ to estimate $S(n)$ (source Drake [1, pp23]).
compared to the singlet decoder.

Some finite memory decoding techniques may also consider future channel output symbols, rather than just past ones. For example, a decoder may consider the output pair \( R(n) R(n+1) \) to determine from Bayes test the most likely input symbol \( S(n) \). Such a decoder differs from the doublet decoder described in our previous example, in that it substitutes the "forward" for the "backward" data.

We might ask, how does the performance of the "forward" and the "backward" decoders compare, in regards to the average probability of error? The answer to the above question cannot be given in general and it depends on the values of the elements of the matrices \( [f_{ij}] \) and \( [p_{ij}] \). However, Drake [1, pp31] has shown that for any process for which the transition matrix of the Markov process is symmetric \( (p_{ji} = p_{ij}) \), the backward and forward data are equivalent regarding the information about \( S(n) \), and thus otherwise identical "forward" and "backward" decoders would have the same average probability of error. However, one can easily suggest other cases where this would not be the case, for example see Drake [1].

3.3 The Infinite-Past Decoder.

Drake [1] has proved the obvious, namely that one can never increase the average probability of error by intelligently taking additional channel output symbols into consideration. The proof just shows that if we work in a sample space of finer
grain, we are able to decompose each term in the expression for the probability of correct decoding to be maximized, into a sum of other terms. Thus we are then able to maximize the new terms individually rather than with a single decision.

In the previous section, we have extended the singlet decoder, which considered only the last output symbol, to the finite memory decoders that considered longer sequences of output symbols. In this section we present a decoding technique that considers all of the infinite past history of the channel output symbols in making the decoding decisions.

The output observer may summarize his state of knowledge about the underlying Markov source at the end of the n'th trial, in his state of knowledge vector, \( \mathbf{s}(n) \), whose \( k \)'th component is equal to the probability that the source is in state \( S_k \) after the n'th trial. We have

\[
\text{Eq 3-5} \quad s_k(n) = P[S_k(n) / R(n) R(n-1) R(n-2) \ldots ] .
\]

The output observer may now update his state of knowledge of the present state of the Markov source, each time an output symbol \( R_i \) is generated, in a simple manner. The \( k \)'th component of his updated state of knowledge vector, \( s_k(n) \), is given by

\[
\text{Eq 3-6} \quad s_k(n) = P[S_k(n) / s(n-1) R_i(n)] = \frac{\prod_{l=1}^{s_1(n-1)} p_{lk} f_{ki}}{\sum_{l=1}^{s_1(n-1)} \sum_j p_{lj} f_{ji}}
\]

Equation 3-6 gives the transformation of the old state of knowledge vector into the updated one at time \( n \), given that the output symbol received at time \( n \) was \( R_i \). The new state of knowledge vector \( s(n) \), is a function of the old state of
knowledge vector, \( s(n-1) \), and the current output symbol received, \( R_1(n) \). Since this transformation plays an important role in further developments, it is useful to define a set of transformation functions, denoted by \( T_i [s(n-1)] \), \( i = 1, \ldots, N \). \( T_i [s(n-1)] \) gives the new state of knowledge vector, \( s(n) \), from the old state of knowledge vector, \( s(n-1) \), given that the current channel output symbol is \( R_1 \).

The output observer starts this updating process with an initial condition specified by \( s(0) \), his state of knowledge prior to the first symbol reception. If he has no additional a priori information he will set the \( j \)'th component of \( s(0) \) to be equal to the steady state probability of the underlying Markov process to be in state \( S_j \), which is \( P_j \).

The infinite-past decoder output, \( J_\infty (R) \), according to Bayes test, will be the input symbol that has the largest a posteriori probability of being the present input symbol. This will obviously be the one corresponding to the largest component of \( s(n) \). We have

\[ Eq \ 3-7 \quad J_\infty (R) = \arg \max_{S_1} s_1(n) . \]

The state of knowledge vector \( s(n) \) summarizes all that the observer knows about the process. He may calculate the function \( P_k[s(n)] \), which specifies the probability that the next observed output symbol, on the \( (n+1) \)'st trial, will be \( R_k \). We obtain

\[ Eq \ 3-8 \quad P_k[s(n)] = P[R_k(n+1) / s(n)] = \sum_j s_j(n) \sum_i p_{ji} f_{ik} . \]
3.4 The Infinite-Past Decoder for Symmetric Case.

As an example, let us now analyze the behavior of the infinite-past decoder in the N-state symmetric case. We have

\[ N \gg 2 \]

\[ p_{ij} = \begin{cases} 
  p & p > 1/N \quad i = j \\
  (1-p)/(N-1) & i \neq j
\end{cases} \]

\[ f_{ij} = \begin{cases} 
  f & f > 1/N \quad i = j \\
  (1-f)/(N-1) & i \neq j
\end{cases} . \]

The following transformation equation can be derived [8] for such a case:

\[ Eq \ 3-9 \quad s_k(n) = P[S_k(n) / s(n-1) R_j(n)] = \sum_{k} A s_k(n-1) + B C s_j(n-1) + D \]

where:

\[ A = (pN-1)/(N-1) ; \quad B = (1-p)/(N-1) \]

\[ C = pf + \frac{(1-f)(1-p)}{(N-1)^2} - \frac{p(1-f) + f(1-p)}{N-1} \]

\[ D = \frac{p(1-f) + f(1-p)}{N-1} + \frac{(N-2)(1-f)(1-p)}{(N-1)^2} . \]

The following properties of the transformation equation (Eq 3-9) have been proven by Sulmar [8]:

**P1)** The four constants defined above \((A,B,C,D)\) satisfy the following constraints:

\[ (i) \quad 0 < A < 1 \]
\[ (ii) \quad 0 < B < 1/N \]
(iii) \(0 < C\)
(iv) \(0 < D\)
(v) \(AD < BC\)

P2) Consider a \(s(n-1)\) whose largest component is \(s_k(n-1)\). The arrival of channel output symbol \(R_k(n)\), corresponding to this largest component, will yield a posterior vector components \(\{s_i(n) ; i \neq k\}\) all of which are smaller than the corresponding components that would result from the arrival of any other symbol \(R_j(n)\), \(j \neq k\).

P3) A string of repeated channel output symbols \(R_k\), drives the corresponding state vector component, \(s_k\), asymptotically toward the equilibrium value \(Q\), given by:

\[
Q = \frac{Af - D}{2C} + \frac{\sqrt{(Af-D)^2 + 4fBC}}{4C^2}
\]

At the same time, all other components of the state probability vector asymptotically approach the value \(Q'=(1-Q)/(N-1)\). This asymptotic value of \(Q\), will always be less than 1 for \(0 < p, f < 1\).

P4) Consider a state probability vector \(s\) which has been conditioned by a long sequence of identical channel output symbols, \(R_k\), and has reached the asymptotic equilibrium defined in P3, i.e. \(s_k = Q\) and \(s_j = Q'\) for
all j≠k. Also consider a different vector s1 which is not at equilibrium, so that Q'< s1'<Q for all j. Upon arrival of the first symbol Rj≠Rk at time n, both vectors will be transformed so that the following relations hold:

1) \( s_1(n) > s_j(n) \) for all j≠i,k
2) \( s_k(n) > s'_k(n) \)

The properties presented above will be useful in the evaluation of the infinite-past decoder in the next chapter.

In this chapter, we look at the average probability of error of a decoder as a measure of its performance, for some of the decoding schemes presented earlier. In evaluating a decoder for a particular application, other considerations might include the amount of computation and memory required. We would also want to consider the delays involved in the case of future-looking decoders as well as allowing for performance measures different from the average probability of error used here.

4.1 Conditions for Differential Performance.

We begin by asking the question: Under what conditions will the infinite-past decoder yield the same average probability of error as the singlet decoder? Clearly, if both devices can be shown to make identical decisions for all possible received symbol sequences, then the average probability of error will be the same for both of them. The following discussion demonstrates that for some combinations of $p$ and $f$ both decoders will indeed perform identically for all possible sequences of channel output symbols. Obviously, for those values of $p$, $f$ and $N$, all other "backward" looking finite memory decoders will perform the same as the singlet decoder.

In the event that the channel output symbol $R_i$ arrives at a given time, the singlet decoder, operating on the channel
output will select the source symbol $S_i$. At the same time, the infinite-past receiver would select a different symbol, $S_k'$, (k$\neq i$), if and only if the largest component of the a posteriori state probability vector was $s_k(n)$. Property P4-2 tells us that if it is ever possible for a vector component not corresponding to the most recent channel output symbol to be the largest, it will certainly occur immediately upon breaking of the long homogeneous sequence of repeated output symbols. If the two receivers in question do not make different decoding decisions in this case, they will always make identical decisions, and thus have the same average probability of error. Conversely, if they do make a different decision in such a case, there will be at least one output sequence of non-zero probability which will be decoded differently by the two decoders, so we would expect the average probability of error to be better for the infinite-past decoder than for the singlet decoder.

A received long sequence of identical symbols, $R_k$, will yield the asymptotic equilibrium conditions as defined by property P3, with $s_k=Q$ and the other vector components equal to $Q'=(1-Q)/(N-1)$. The arrival of $R_{i}\neq R_k$ will cause the value of $s_k$ to decrease and the values of all other vector components to increase (by property P2), however, $s_k$ will still be larger than any of the other components (exclusive possibly of $s_i$). We must now determine under which conditions $s_k$ is also greater than $s_i$ at the time of breaking of a homogeneous long sequence of $R_k$'s by the arrival of $R_i$. The necessary and sufficient condition for identical performance is $s_i \gg s_k$, or substituting the
transformation of Eq 3-9 we obtain
\[
f \frac{A s_1(n-1) + B}{C s_1(n-1) + D} \geq \frac{1-f}{N-1} \Rightarrow \frac{A s_k(n-1) + B}{C s_1(n-1) + D} .
\]

If we put in the asymptotic values for \( s_1(n-1) = (1-Q)/(N-1) \) and \( s_k = Q \), we have
\[
f \left[ A \frac{1-Q}{N-1} + B \right] \geq \frac{1-f}{N-1} (AQ + B)
\]
rearranging terms,
\[
Eq 4-1 \quad Q \leq \frac{(p-1) + f(N-1)}{(pN-1)}
\]
Substituting the value of \( Q \) as defined by P3, we get
\[
Eq 4-2 \quad \frac{Af - D}{2C} + \sqrt{\frac{(Af-D)^2}{4BC}} \leq \frac{(p-1) + f(N-1)}{pN - 1}
\]
The above inequality is a necessary and sufficient condition for both decoders to perform identically. In Figure 8 boundaries in the \( p-f \) plane have been plotted between regions where the infinite-past decoder can and cannot outperform the singlet decoder.

From the above inequality, Eq 4-1, which is a sufficient condition for identical performance, we can obtain a weaker sufficient condition for identical performance. By noting that we always have the condition \( Q \leq 1 \) satisfied, we reason that if we satisfy
\[
\frac{(p-1) + f(N-1)}{pN - 1} \geq 1
\]
Figure 8. Differential performance threshold for N=2, 4, 16. Above the $p_t$ curve are regions where the infinite-past decoder will make decoding decisions differently, for some possible channel output sequences, and thus will outperform the singlet decoder.
we are guaranteed to satisfy Eq 4-1 and we would have identical performance. Rearranging terms in the above inequality, we get the condition $f > p$ implies identical performance of the infinite-past and the singlet decoders. We see that for models with the property $f > p$ there is not enough memory in the underlying Markov source for any decoder which considers the past history of the channel output symbols, to make any decoding decisions differently from the singlet decoder.

4.2 Performance Bounds.

In the previous sections, we mentioned several methods of reducing the average probability of incorrect decoding, by taking advantage of the memory properties of the Markov source and considering longer sequences of the channel output symbols. It is our purpose now to obtain some performance bounds for the infinite-past decoder, in the symmetric two-state case.

The infinite-past decoder considers $R(n)R(n-1)R(n-2)\ldots$ and estimates $S(n)$. We first notice that the output observer would gladly swap the sequence $R(n-L)R(n-L-1)\ldots$ for the actual state $S(n-L)$. We denote the average probability of error obtained by using the sequence $S(n-L)R(n-L+1)\ldots R(n-1) R(n)$ as the decoder input by $P_L^*(\mathcal{E})$, we also denote by $P_L(\mathcal{E})$, the average probability of error for a finite memory decoder operating with the sequence $R(n-L+1)\ldots R(n-1)R(n)$. Similarly, we denote the average probability of error for the infinite-past decoder by $P_\infty(\mathcal{E})$. We obtain
\[ P_L(\varepsilon) \geq P_\omega(\varepsilon) \geq P^*_L(\varepsilon) \quad \text{for } L=1,2,3,\ldots \]

Also, since the further removed the state about which we have exact information, \(S(n-L)\), from the state which we are trying to decide on, \(S(n)\), the less helpful that information is, we can write

\[ P_{L+1}(\varepsilon) > P_L(\varepsilon) \quad \text{for } L=1,2,3,\ldots \]

However, \(P_{L+1}(\varepsilon) \leq P_L(\varepsilon)\) for \(L=1,2,\ldots\), since considering further removed channel output symbols cannot make our average probability of error larger. Putting the above inequalities together we obtain

\[ \text{Eq 4-3 } P_{L-1}(\varepsilon) \geq P_L(\varepsilon) \geq P_\omega(\varepsilon) \geq P^*_L(\varepsilon) \geq P^*_{L-1}(\varepsilon) \quad L=2,3,\ldots \]

The inequality, Eq 4-3, can be used to obtain tighter and tighter bounds on the average probability of error for the infinite-past decoder, by evaluating \(P^*_L(\varepsilon)\) and \(P_L(\varepsilon)\) for larger and larger values of \(L\).

As an example, let us obtain bounds on \(P_\omega(\varepsilon)\), by considering the finite memory decoder of length \(L=2\). This decoder was analyzed in our example in section 3.2. We found there that this doublet decoder made all the same decoding decisions as the singlet decoder, and thus achieved an average probability of error of \((1-f)\). This gives us our desired upper bound on \(P_\omega(\varepsilon)\)

\[ P_\omega(\varepsilon) \leq P_2(\varepsilon) = P_1(\varepsilon) = 1-f \]

The above result is not very informative, however, considering longer sequences would often, of course, provide more informative
results. For our lower bound, we have $P_\infty (\varepsilon) \geq P_2^\prime (\varepsilon)$ where

$$\text{Eq 4-4} \quad P_2^\prime (\varepsilon) = 1 - \sum g \sum w \sum_{x} \max_i P[S_{(n-2)}R_{w}(n-1)R_{x}(n)S_{1}(n)]$$

$$= 1 - \sum g \sum w \sum_{x} \max_i \sum h P[S_{(n-2)}S_{h}(n-1)S_{1}(n)R_{w}(n-1)R_{x}(n)]$$

Rearranging terms, we obtain

$$\text{Eq 4-5} \quad P_2^\prime (\varepsilon) = 1 - \sum g \sum w \sum_{x} \max_i \sum h P_{gh} P_{h1} f_{hw}$$

We can evaluate Eq 4-5, and Figure 9 tabulates the percent reduction in average probability of error attainable with a decoder which produces $P_2^\prime (\varepsilon)$, this is an upper bound on the performance of the infinite-past decoder (i.e. a lower bound on its average probability of error).

A method similar to the one used above to obtain performance bounds for the infinite-past decoder can be used to obtain performance bounds for a decoder that demands maximum accuracy. Such an infinite-patience decoder would consider the entire $R(n)$ history from $R(n-\infty)$ up through $R(n+\infty)$ in order to estimate $S(n)$. For such a decoder we would obtain our bounds by considering a decoder that has knowledge of $S(n-L)$ instead of $R(n-L)R(n-L-1)...$ and $S(n+L)$ instead of $R(n+L)R(n+L+1)...$

4.3 Average Probability of Error of Infinite-Past Decoder in the Two-State Case.

We are still unable to answer the question: "Exactly what average probability of error is achievable by the infinite-past decoder?" Attempting to form an average over all possible infinite past histories of output symbols, as we would
<table>
<thead>
<tr>
<th></th>
<th>.60</th>
<th>.70</th>
<th>.80</th>
<th>.85</th>
<th>.90</th>
<th>.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>.975</td>
<td>88.</td>
<td>84.</td>
<td>76.</td>
<td>67.</td>
<td>64</td>
<td>44</td>
</tr>
<tr>
<td>.95</td>
<td>76.</td>
<td>68.</td>
<td>53.</td>
<td>50.</td>
<td>39.</td>
<td>0</td>
</tr>
<tr>
<td>.90</td>
<td>55.</td>
<td>40.</td>
<td>29.</td>
<td>18.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>.85</td>
<td>36.</td>
<td>22.</td>
<td>7.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>.80</td>
<td>20.</td>
<td>8.7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>.75</td>
<td>8.6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 9. Upper bound on the percent reduction in the average probability of error (relative to the singlet decoder) achievable with the infinite-past decoder. This upper bound was obtained from considering a decoder that has the values of $S(n-2)$ $R(n-1)$ $R(n)$.
(source Drake [1, pp36])
do for the finite memory decoders, is clearly infeasible. In this section we try to find the average probability of error for the infinite-past decoder in the two state symmetric case, by deriving the steady state statistics of the state of knowledge variable.

To simplify our notation for the two state case, since there is only one independent component in \( s(n) \), (because \( s_1(n) + s_2(n) = 1 \)), we define \( u = s_1(n) \). Thus, \( u \) is the probability that our source is in state \( S_1 \), given the entire past history of the channel output up to and including \( R(n) \).

Figure 10 presents the sample space that is used to derive the transformation equations for our case. The functions \( P_i(u) \), \( i=1,2 \), specify the probability that the next output symbol will be \( R_i \), given the current state of knowledge, and are given by

\[
P_1(u) = u[pf + (1-p)(1-f)] + (1-u)[p(1-f) + (1-p)f]
\]

or,

\[
P_1(u) = c_1 u + d_1
\]

where \( c_1 = (1-2p)(1-2f) \)

\[
d_1 = f + p - 2pf
\]

and

\[
P_2(u) = c_2 u + d_2
\]

where \( c_2 = (1-2p)(2f-1) = -c \)

\[
d_2 = 1 - f - p + 2pf = 1 - d
\]

A quick check shows, as it must, that \( P_1(u) + P_2(u) = 1 \). We shall also define the two transformations \( T_i(u) \), \( i=1,2 \), that give the value of \( u \) after the next trial, given that the next output symbol is \( R_i \). Collecting the appropriate branches from the sample space, and a little algebra yields

\[
T_1(u) = \frac{a_1 u + b_1}{c_1 u + d_1}
\]

with \( a_1 = f(2p-1) \)

\[
b_1 = f(1-p)
\]
Figure 10. Sample space used to obtain the transformation of the state of knowledge vector, $s$, in the two-state case.
and \[ T_2(u) = \frac{a_2u + b_2}{c_2u + d_2} \]\[ \text{with } a_2 = (1-f)(2p-1) \]
[\text{and } b_2 = (1-f)(1-p) .]

The derivatives of \( T_1(u) \) and \( T_2(u) \) show that for \( p>1/2 \) they are monotonically increasing in their argument. It is also interesting to note that for \( u>Q \) (where \( Q \) is the asymptotic value defined in P3) we have \( T_1(u)<u \) and for \( u<Q \) \( T_1(u)>u \). Similarly we have \( T_2(u)>u \) for \( u<Q' \) and \( T_2(u)<u \) for \( u>Q' \). Figure 11 plots \( T_1(u) \) and \( T_2(u) \) for the case \( p=0.9, f=0.7 \).

Let us now consider an infinite ensemble of systems of the form of Figure 10 operating simultaneously. We can think of the state of knowledge variable, \( u \), as a random variable over the ensemble, with a probability density function \( h(u) \). If such a function did exist and we could find it, it would allow us to average over all possible past histories of the channel output and calculate the average probability of error.

Drake [1] proposed an iterative method for calculating \( h(u) \). If we begin with an incorrect estimate of \( h[u(n)] \) at time \( n=0 \), which we shall call \( h^0[u(0)] \), we can calculate the probability density function of the random variable \( u(1) \), \( h^1[u(1)] \), which would result from the initial condition \( h^0[u(0)] \). If a steady state density function \( h(u) \) does exist, one would expect that \( h^k[u(k)] \) to converge to it as \( k \to \infty \), since the effects of the initial conditions will vanish in all but pathological cases (such as periodic sources, sources with trapping states or \( h^0(u) \) that is zero in places where it should not be).

Given a value of \( u \) immediately after receipt of the channel output symbol at time \( (k+1) \), one can specify the values
Figure 11. A plot of the functions $T_1(u)$ and $T_2(u)$ for the case $p=0.9, f=0.7$. 
that u could have had at time k. Specifically, it could have been \( T_1(u) = R_1 \) or \( T_2(u) = R_2 \). If we seek to find the probability that the random variable \( u(k+1) \) takes on a value less than or equal to \( v \), we can add up the probabilities of the two mutually exclusive events that can generate such an event (see Figure 12):

1. \( R_1(k+1) \) occurred and \( 0 \leq u(k) \leq T_1(v) \)
2. \( R_2(k+1) \) occurred and \( 0 \leq u(k) \leq T_2(v) \).

The above information can be expressed in the following equation:

\[
\text{Eq 4-6} \quad \int_0^y h^{k+1}(x) \, dx = \int_0^{T_1(v)} P_1(y) \, h^k(y) \, dy + \int_0^{T_2(v)} P_2(z) \, h^k(z) \, dz
\]

Drake [1] and Shoem [9] discuss algorithms that use a slightly different form of Eq 4-6, which exploits some symmetry properties of \( h(u) \) and its corresponding cumulative density function, \( F[u] \). For example, we can reason that due to the symmetry of the source process we must have \( h(u) = h(1-u) \) for \( 0 \leq u \leq 1 \).

This iterative procedure seems in actual computation to converge to a steady state function \( h^\infty(u) \). Figure 13 plots \( h^{30}(u) \) in the range \( 0 \leq u \leq .5 \) for \( p = .9 \), \( f = .7 \). As expected, we note that for \( u < .0679 \), \( h(u) = 0 \). The \( h^{30}(u) \) curve is very jagged, as expected from the fact that runs of identical R's are quite probable and would tend to drive u to near the asymptotic values Q and Q'. Simple combinations of R's that transform u from these likely "saturation" values produce other peaks in the density function. We can note those peaks in the density function. The first, next to Q', corresponds to successive receptions of the
Figure 12. Events that can generate $u(k+1) = v$. 
Figure 13. A plot of $h^{30}(u)$ for $p=.9$, $f=.7$, in the range $0 < u < 5$. 

$h^{30}(u)$
same output symbol $R_2$ that drive $u$ close to this asymptotic value. Also, we observe other peaks corresponding to simple combinations of transformations of $Q'$. For example,

\[ T_1(Q') = .297 \]
\[ T_2(T_1(Q')) = .18 \]
\[ T_1(T_2(T_1(Q'))) = .43 \]
\[ T_2(T_1(T_1(Q')))) = .33 \]

Having derived the steady state distribution of $h(u)$, we can now evaluate the average probability of error. We note that for the infinite-past decoder, the decoder output is $S_1$ if $u \geq 1/2$ and $S_2$ otherwise. Thus, if $u \geq 1/2$ the probability of error is $(1-u)$. Similarly, if $u \leq 1/2$ the probability of error is $u$. This gives us the following average probability of error for our decoder,

\[
P_\infty(\varepsilon) = \int_0^{.5} u \ h(u) \ du + \int_{.5}^1 (1-u) \ h(u) \ du
\]

Eq 4-7 \[ P_\infty(\varepsilon) = 2 \int_{.5}^1 (1-u) \ h(u) \ du \]

Where we have used the symmetry properties of $h(u)$. Figure 14 (taken from Shoen [9]) tabulates the average probability of error achieved with the infinite-past decoder as a function of $p$ and $f$, evaluated from Eq 4-7.
<table>
<thead>
<tr>
<th>p</th>
<th>.80</th>
<th>.85</th>
<th>.90</th>
<th>.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>.90</td>
<td>.10000</td>
<td>.6909</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.85</td>
<td>.15000</td>
<td>.13455</td>
<td>.09190</td>
<td></td>
</tr>
<tr>
<td>.80</td>
<td>.20000</td>
<td>.19684</td>
<td>.16851</td>
<td>.12310</td>
</tr>
<tr>
<td>.75</td>
<td>.24397</td>
<td>.23794</td>
<td>.20922</td>
<td>.16117</td>
</tr>
<tr>
<td>.70</td>
<td>.29881</td>
<td>.28314</td>
<td>.25615</td>
<td>.20485</td>
</tr>
<tr>
<td>.65</td>
<td>.34660</td>
<td>.33290</td>
<td>.30962</td>
<td>.26000</td>
</tr>
<tr>
<td>.60</td>
<td>.39334</td>
<td>.38622</td>
<td>.36840</td>
<td>.32737</td>
</tr>
<tr>
<td>.55</td>
<td>.44786</td>
<td>.44227</td>
<td>.43243</td>
<td>.49578</td>
</tr>
</tbody>
</table>

Figure 14. Average probability of error of infinite-past decoder operating in the two-state case, as a function of p and f. Entries were obtained using the steady state statistics of the state of knowledge variable u, from Eq 4-7.
III. The Partially Observable Markov Decision Process with Rewards.

5. The Problem and Basic Models.

Associated with most real life systems are decisions and rewards. In this chapter we extend our models of the partially observable discrete-state, discrete-trial Markov process to account for a decision and reward structures.

5.1 The Markov Process with Transition Rewards.

Suppose that the $N$-state Markov process earns $w_{ij}$ dollars when it makes a transition from state $S_i$ to state $S_j$. We call $w_{ij}$ the "reward associated with the transition from $S_i$ to $S_j". The Markov process now generates a sequence of rewards as it makes a transition from state to state.

Recalling our previous machine example, we can picture a reward mechanism, where our machine earns an amount of money $w_{ij}$ in making a transition from $S_i$ to $S_j$. For example, we might have $w_{11} = w_{12} = 1$ (reward for good item produced) and $w_{22} = w_{21} = -0.5$ (cost of bad item produced).

One question we might ask is: what is the machine's expected total reward, denoted by $V^n(S_i)$, if the machine is in state $S_i$ and there are $n$ trials remaining (in this section we change the use of the time variable $n$ to denote the number of remaining trials). Some reflection allows us to write the
following recurrence relation

\[ V^n(S_i) = \sum_j p_{ij} [w_{ij} + V^{n-1}(S_j)] \quad i=1,2,\ldots,N \]

The recurrence relation in Eq 5-1 along with the boundary values \( V^0(S_i) \) for all \( i \), can be used to obtain an answer to the above question in a recursive manner, working backwards in time. The quantities \( V^0(S_i) \), are the terminal values of ending the process in state \( S_i \) and are used to start our algorithm. Closed form solutions to the above question can be obtained with the use of the z-transform techniques (Howard [5]).

5.2 The Markov Decision Process with Rewards.

The discussion of Markov processes with rewards has been our first step on the way to our topic of interest, a discussion of Markov processes with sequential decision making.

Suppose that the decision maker in our process has a number of alternative actions from which he must choose one, at each stage, after observing the state of Markov process. We identify each of these alternatives with a numerical value of index \( k \). Conditioned upon his choosing action \( k \) and the process being in state \( S_i \), the process will make a transition on the next trial to state \( S_j \) and earn a reward \( w_{ij}^k \) with probability \( p_{ij}^k \).

The decision maker's problem is to choose the alternative \( k \) so as to maximize his expected total reward over the remaining \( n \) trials. We may ask now, what is the optimal
expected total reward of a decision maker over the remaining n trials, if the system is currently in state $S_1$, denoted by $V^n(S_1)$. We can write a recursive relation for $V^n(S_1)$ similar to Eq 5-1

$$V^n(S_1) = \max_k \sum_j p_{1j}^k [w_{1j}^k + V^{n-1}(S_j)]$$

The use of the dynamic programing recursive relation (Eq 5-2) will tell the decision maker which alternative to use in each state at each stage of the process and will also provide him with his expected total future reward. To apply this relation, we must specify the $V^n(S_1)$, the boundary condition for our process, which is the reward received for terminating the process in state $S_1$. For a more complete discussion of the solution of the sequential Markov Decision Process, see Howard [5].

5.3 The Problem of the Partially Observable Markov Decision Process.

In the previous section we presented the sequential Markov Decision Process in the observable discrete-state discrete-trial case and its solution by value iteration. We now present the problem of the Partially Observable Markov Decision Process (POMDP).

Recalling the noisy channel model, our decision maker does not have exact information about the state of the underlying process while making his decisions. In our machine example, each item produced was subject to a quick imperfect test. This test
results in a correct identification of the quality of any item with probability $f$, and incorrect identification with probability $(1-f)$. The test result was represented by the output symbols $R_1$ (item seems good) and $R_2$ (item seems bad). Suppose now that as soon as the operator of our machine observes each output symbol, he must select one of three actions. We identify each of these alternatives with a numerical value of the index $k$.

$k=1$ Assume item is acceptable, machine is properly tuned and continue manufacturing.

$k=2$ Assume item is unacceptable, machine is not properly tuned, but still continue manufacturing.

$k=3$ Stop the manufacturing process, inspect machine and item in a perfect test, and tune the machine if necessary.

The transition probabilities and reward matrices associated with these alternatives are

$$[p_{ij}^1] = [p_{ij}^2] = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix} ; \quad [p_{ij}^3] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$[w_{ij}^1] = \begin{bmatrix} B & B \\ -C & -C \end{bmatrix} ; \quad [w_{ij}^2] = \begin{bmatrix} -C & -C \\ B & B \end{bmatrix}$$

$$[w_{ij}^3] = \begin{bmatrix} -A & -A \\ -A & -A \end{bmatrix}$$

For our example, we have purposely chosen a particularly simple reward structure. If we choose alternative $k=1$ and the system occupied state $S_1$ before the trial (i.e. item inspected is in fact good) we gain $B$ dollars, and otherwise lose $C$ dollars. If we choose alternative $k=2$ and the system occupied state $S_2$ (item
inspected is in fact bad) we gain B dollars, and otherwise lose C dollars. Thus, we have a cash reward of B dollars, B>0, for correctly identifying the state of the process, whether in S₁ or in S₂. We have a cost of +C dollars, C>0, for incorrectly identifying the state of the process. We also have a +A dollars, A>0, cost for the use of the perfect inspection and repair option. Note that from the elements of [p₁³₃], we see that our operator has the option of resetting the machine to state S₁ and gaining perfect state information at the cost of A dollars, at any time. Otherwise, the process continues unperturbed, i.e. it will remain in its previous state with probability p and would change state with probability (1-p).

Given the reward and dynamic structure of the example, our problem now consists of finding the optimal decision of the operator among his alternatives at each stage and calculating the expected total reward for this optimal policy.

In this chapter we discuss solution algorithms for the POMDP problem formulated in the last chapter. We begin by presenting the dynamic programing solution technique.

6.1 Solution by Dynamic Programming.

Smallwood and Sondik [3], present a proof that if the controller of our process has available to him the past observation of the channel outputs, than at any time the state of knowledge vector $s$, is a sufficient statistic for his past sequence of observations. From this result it follows that the dynamic behavior of the information vector $s$, is itself an observable discrete-time continuous-state Markov process.

Figure 15 shows the continuous-state Markov process, whose state variable $s(n)=u$ changes probabilistically as trials occur. We can now use the dynamic programing technique to find the optimal control policies, for our new observable, continuous-state, discrete-trial Markov process. We denote by $V^n(s)$ the maximum expected reward that the system can accrue during the remaining $n$ trials, if the current information vector is $s$. Expanding over all possible next transitions and observations yields the recursive equation

$$V_{i+1}^n(s) = \max_k \sum_i \sum_j p_{ij}^k \{ w_{ij}^k + \sum_l p_{ij}^l (s) V^n[T_l(s)] \}$$
Figure 15. The continuous-state Markov process whose state variable $s_1(n)=u$, changes probabilistically as trials occur.
where \( P_1(s) \) specifies the probability that the next output symbol will be \( R_1 \), given the current information vector \( s \), and \( T_1(s) \) gives our new information vector, given that the channel output symbol is \( R_1 \). If we specify \( V^0(s) \), the terminal reward of the system for terminating the process in state \( s \), we can solve our problem recursively starting with \( n=0 \) and going backwards in time.

For our two state machine example, we replace the state of knowledge vector \( s \) by a scalar state of knowledge variable \( u=s_1 \). For the cost and decision alternatives presented in section 5.3, we have

\[
V^{n+1}(u) = \max_k \begin{cases} 
  k=1 & uB - (1-u)C + P_1(u)V^n[T_1(u)] \\
  + P_2(u)V^n[T_2(u)] \\
  k=2 & (1-u)B + uC + P_1(u)V^n[T_1(u)] \\
  + P_2(u)V^n[T_2(u)] \\
  k=3 & -A + P_1(1)V^n[T_1(1)] + P_2(1)V^n[T_2(1)]
\end{cases}
\]

The boundary condition in our example will be \( V^0(u)=0 \) for all \( 0 \leq u \leq 1 \). Beginning with \( n=1 \) and determining the best action for each possible value of \( u \), we can solve the above \( Eq \ 6-2 \) for \( V^n(u) \) iteratively. For the case \( B=C=$1.00, A=$0.50, p=0.9 \) and \( f=0.7 \), the results are plotted in Figure 16. As we would have expected from the reward structure and symmetry of the problem, when there are many future trials remaining, the decision rule takes the form: for \( u \geq 0.5 \) choose alternative \( k=1 \) if \( u \) is greater than some critical value \( u_a \); applying symmetry, the best action would be \( k=2 \) if \( u < 0.5 \) and \( u < (1-u_a) \); for \( u \) in the range \((1-u_a) \leq u \leq u_a \), the best action is \( k=3 \). The value of \( u_a \) will, of course, depend on the number of trials remaining and the particular parameter.
Figure 16. Total expected earnings using optimal policy and optimal stationary decision regions for machine example with $B=C=\$1.0$, $A=\$0.5$, $p=.9$, $f=.1$. 
values of the reward and probability matrices. Note that for our example, we shall never use the perfect inspection and tuning option, k=3, if there are only one or two trials remaining, since we would not have the adequate future opportunity to amortize the immediate cost of the option.

For a more complete discussion of the dynamic programming algorithm as applied to solving the Partially Observable Markov Decision Process, and analysis of similar examples, see Drake [2] and Kakalik [7].

6.2 Sondik's Algorithm

In the previous section we presented the dynamic programming algorithm for finding the optimal control policies for the POMDP. This algorithm, when implemented on a digital computer, requires quantization of the continuous-state space of information vectors and essentially converts the continuous-state Markov process (discussed in 6.1) into a finite-state Markov process. However, the number of states in this approximate finite-state Markov process becomes prohibitively large for any but the smallest problems. For example, if a quantization interval of 0.05 is used for the components of the probability state vector, then a seven-state underlying Markov process will require approximately 60 million states in the quantized process. This would be, of course, a completely impractical solution technique from a computational point of view.

Sondik [3] developed an algorithm that does not require
this quantization and thus provides a significant increase in the size of the problems for which an optimal control policy can be calculated. The algorithm exploits the structure of the finite-horizon optimal value function $V^n(s)$, given in Eq 6-1. It is straightforward to show by induction (Sondik [3]) that $V^n(\cdot)$ is piecewise-linear and convex in its argument i.e.

$$V^n(s) = \max_k \sum_a a^k(n) s_k$$

for some set of $a$ vectors. Using this property, we can divide the space of information vectors into regions that have the same optimal control policy (recall the solution to our example in section 6.1). The algorithm uses a linear programing technique to find the defining hyperplanes, in the continuous-space of information vectors, that separate the optimal policy regions. Thus, it does not require a separate selection of the optimal alternative for each point in those regions. The computation time for this algorithm is dependent not only on the number of states, but also upon the number of distinct policy-regions.

Extensions of this algorithm have been developed by Sondik [4] to find the optimal stationary control policies for the discounted infinite horizon case.
6.3 Platzman's Algorithm for Computing $\varepsilon$-Optimal Controllers.

In section 4.2 we obtained tighter and tighter performance bounds on the performance of the infinite-past decoder, by considering finite memory decoders that used longer and longer sequences of the channel output symbol string. We saw in Eq 4-3, that by making the length of the output symbol sequence used by a finite memory decoder, longer and longer, we could make the average probability of error closer and closer to that of the infinite-past decoder. This notion, that finite memory policies may be adequate in many decision making contexts, is the prime motivation in Platzman's algorithm (Platzman [6]). The algorithm exploits a contraction property of the infinite-past decoder information vector transformation function (Eq 3-6), to obtain procedures for constructing $\varepsilon$-optimal (arbitrarily close) approximations to the information vector by a finite-memory observer.

Control problems, in which we try to maximize the expected reward over the infinite horizon, can be solved by an iterative procedure, called "perceptive dynamic programing". Successively weaker assumptions that the controller "perceives" unavailable state values (in a manner similar to that used in section 4.2) transfer the problem into a sequence of formulations which may be solved by dynamic programing. At each iteration, the solution obtained is used to construct a feasible finite-state controller. Converging performance bounds may be obtained at each iteration. The computation may be terminated when these
bounds indicate that the current design is sufficiently close to the global optimum.
IV. Information Theory Considerations.

7. Information Flow Equation

For our purposes, it is sufficient to define the "information" carried by the occurrence of event A about event B, as the logarithm to the base 2 of the ratio of the a posteriori probability of event B, given that A has occurred, to the a priori probability of event A.

\[
I(A;B) = \log_2 \frac{P[A/B]}{P[B]} = \log_2 \frac{P[AB]}{P[A]P[B]} = \log_2 \frac{P[A/B]}{P[A]} = I(B;A)
\]

In this chapter we derive an information flow equation for the general problem of monitoring a Markov process through a noisy channel. In the last section, we give a simple numerical example. For a more complete discussion with a more intuitive interpretation of the various information quantities defined see Drake [1].

7.1 Information Flow Equation: Derivation.

We may investigate the relevant information quantities which arise when the source makes a transition from \(S_i\) to \(S_j\) and the channel produces output symbol \(R_k\).

\(I(\text{source})\) is the actual self-information generated by the Markov source. It corresponds to the amount of information which would have to be delivered to a person who knew
the state of the source at time $n$, to allow him to determine which transition actually occurred on the following trial. $I(\text{source})$ is not related to the observer's state of knowledge.

$$I(\text{source}) = -\log_2 p_{ij}$$

$I(\text{out})$ is the negative logarithm to the base 2 of the probability of $R(n+1)=R_k$ to an observer at the channel output. $I(\text{out})$ depends on the state of knowledge vector $s$, of the output observer.

$$I(\text{out}) = -\log_2 P[R_k(n+1)/R(n)R(n-1)\ldots] = -\log_2 P_k(s)$$

$I(\text{stored})$ is a measure of the information which would have to be supplied to the output observer after he has observed $R(n)$ to allow him to be certain of the actual state of the underlying process at time $n$.

$$I(\text{stored}) = -\log_2 s_i(n)$$

$\Delta I(\text{stored})$ is the increase in $I(\text{stored})$ due to the transition from $S_i$ to $S_j$ and the reception of $R(n+1)=R_k$.

$$\Delta I(\text{stored}) = -\log_2 [s_j(n+1)] + \log_2 [s_i(n)]$$

$I(\text{noise})$ is the actual self-information generated by the channel.

$$I(\text{noise}) = -\log_2 f_{jk}$$

$I(\text{lost})$ is the negative logarithm of the probability to the output observer that the new output symbol, $R(n+1)$, was caused by the transition which actually occurred, rather than another transition from $S_i(n)$ to $S_j(n+1)$. 

-60-
Eq 7-7  \[ I(\text{lost}) = -\log_2 \frac{s_i(n) p_{lj} f_{jk}}{s_1(n) p_{lj} f_{jk}} = -\log_2 \frac{s_i(n) p_{lj}}{s_1(n) p_{lj}}. \]

By direct substitution, we can establish

Eq 7-8  \[ I(\text{source}) + I(\text{noise}) = I(\text{out}) + I(\text{stored}) + I(\text{lost}). \]

This equation is called "equation of information flow". This equation can be used to establish bounds on the average informational uncertainty at the decoder (see Drake [1]).

7.2 Information Flow Equation: An Example.

The result of the previous section may be now demonstrated in terms of a numerical example for the system of Figure 2 with \( p = .9 \) and \( f = .7 \). Let us consider a situation in which the source makes a transition from \( S_1(n) \) to \( S_2(n+1) \) and the channel produces \( R_1(n+1) \). Let us assume also, that the output observer's state of knowledge vector is given by \( s_1(n) = .6 \) and consequently \( s_2(n) = .4 \).

If we plug in these numerical values into Eq 7-2 through 7-7, we get

\[
\begin{align*}
I(\text{source}) &= -\log_2 (.1) = 3.322 \text{ bits} \\
I(\text{noise}) &= -\log_2 (.3) = 1.737 \text{ bits} \\
I(\text{out}) &= -\log_2 p_1(u) = -\log_2 (.6*.32 + .4*.34) = .910 \text{ bits} \\
\Delta I(\text{stored}) &= -\log_2 [1-T_1(.6)] + \log_2 (.6) = 1.340 \text{ bits} \\
I(\text{lost}) &= -\log_2 (.6*.1)/[.6*.1 + .1*.9] = 2.807 \text{ bits}
\end{align*}
\]

one can now check
\[ I(\text{source}) + I(\text{noise}) = I(\text{out}) + \Delta I(\text{stored}) + I(\text{lost}) \]

\[
3.322 + 1.737 = 0.910 + 1.340 + 2.807 \\
5.06 = 5.06
\]
V. Comments and Extensions.

The discussion in the previous sections has left a number of questions unanswered. In this chapter one extension of our previous analysis of the infinite-past decoder performance is presented. An iterative procedure similar to the one used for finding the steady state statistics of the state of knowledge variable \( u \) in the two-state case is developed for the \( N \)-state case. It produces the marginal density function of any one component of the state of knowledge vector in this more general \( N \)-state symmetric case. This density can be used to obtain an upper bound on the average probability of error of the decoder in the \( N \)-state case.

In our previous notation, \( S_1(n) \) denoted the event "Markov process is in state \( S_1 \) immediately after the \( n \)'th trial". We now let \( S_1(n) \) denote the complementary event i.e. "Markov process is in a state other than \( S_1 \) immediately after the \( n \)'th trial". Similarly we let \( R_1(n) \) denote the event "channel output symbol at the end of the \( n \)'th trial is some symbol other than \( R_1 \)".

It can be easily shown, using some algebra that

\[
\begin{align*}
P[S_1(n+1)/S_1(n), S(n-1), \ldots] &= P[S_1(n+1)/S_1(n)] \\
&= 1 - p \\
P[S_1(n+1)/S_1(n), S(n-1), \ldots] &= P[S_1(n+1)/S_1(n)] \\
&= p \\
P[S_1(n+1)/S_1(n), S(n-1), \ldots] &= P[S_1(n+1)/S_1(n)] \\
&= (1 - p)/(N - 1) \\
P[S_1(n+1)/S_1(n), S(n-1), \ldots] &= P[S_1(n+1)/S_1(n)] \\
&= 1 - (1-p)/(N-1)
\end{align*}
\]
Similarly, we can show that

\[
P[R_i(n)/S_i(n), S(n-1), R(n-1), \ldots] = P[R_i(n)/S_i(n)] = f
\]

\[
P[\overline{R}_i(n)/S_i(n), S(n-1), R(n-1), \ldots] = P[\overline{R}_i(n)/S_i(n)] = 1 - f
\]

\[
P[R_i(n)/\overline{S}_i(n), S(n-1), R(n-1), \ldots] = P[R_i(n)/\overline{S}_i(n)] = (1 - f)/(N - 1)
\]

\[
P[\overline{R}_i(n)/\overline{S}_i(n), S(n-1), R(n-1), \ldots] = P[\overline{R}_i(n)/\overline{S}_i(n)] = 1 - (1-f)/(N-1)
\]

From the above equations, since the state transition probabilities satisfy the Markov condition, we can formulate a new Markov process model that has the two states $S_i$ and $\overline{S}_i$. Similarly, we can formulate a memoryless noisy channel model that has the two output symbols $R_i$ and $\overline{R}_i$, through which our two-state process is observed. This model is presented in Figure 17.

Our state of knowledge of this aggregated model is summarized by a state of knowledge vector $s$ with the two components $s_i$ and $\overline{s}_i = 1 - s_i$. We can calculate the function $P_i(s_i)$, which specifies the probability that the next output symbol will be $R_i$. Similarly, $P[\overline{R}_i(s_i)]$ specifies the probability that the next observed output symbol will be $\overline{R}_i$. Applying Eq 3-8, we find

\[
P_i(s_i) = cs_i + d
\]

\[
P[\overline{R}_i(s_i)] = \overline{cs}_i + \overline{d}
\]

where,

\[
c = [(n-1)(1-2f)-Np(1-Np)-(N-2)(1-f)]/(N-1)
\]

\[
d = [f(N-1)+p-Npf-(n-2)(1-f)]/(N-1)
\]

\[
\overline{c} = -c
\]

\[
\overline{d} = 1 - d
\]

Similarly, we can define the transformations $T_i(s)$ and $T[\overline{R}_i(s_i)]$. $T_i(s_i)$ gives the next value of $s$ given that the next output
Figure 17. The reduction of the N-state model into an aggregated two-state noisy channel model.
symbol is $R_1$. $T_i(s_i)$ gives the next value of $s_i$ given that the next output symbol is $R_1$. Applying Eq 3-6, we get

$$T_i(s_i) = \frac{as_i + b}{cs_i + d} \quad \text{where} \quad a = f(N_{p-1})/(N-1) \quad b = f(1-p)/(N-1)$$

$$T_i^*(s_i) = \frac{as_i + b}{cs_i + d} \quad \text{where} \quad \tilde{a} = (1-f)(N_{p-1})/(N-1) \quad \tilde{b} = (1-p)(1-f)/(N-1)$$

A quick check shows that for $N=2$ these functions reduce to the ones derived in section 4.3.

We can now find the marginal density of the $i$'th component of $s$ in our $N$-state model, $h_i(s_i)$. Using a similar iterative procedure to the one defined by Eq 4-6. We can write the corresponding equation for our aggregated two state model

$$\int_0^V h_i^{k+1}(x) \, dx = \int_0^{T_i} h_i^k(y) \, dy + \int_0^{T_i^*} h_i^k(z) \, dz$$

Having the steady state distribution of $s_i$, $h_i(s_i)$, we can now obtain a bound on the average probability of error. If we denote the largest component of $s$ by $s^*$, we can reason that $s^*$ will always satisfy $s^* \gg s_1$. Hence, the probability of error of the decoder always satisfies $P(\text{error}) = 1 - s^* \ll 1 - s_1$. Also, for $s_i \ll 1/N$, we can argue that $s^* \gg (1-s_1)/(N-1)$, since at worst the remaining probability will be equally distributed among the other $(N-1)$ components of $s$. So, we have for $s_i \ll 1/N$ $P(\text{error}) = 1 - s^* \ll 1 - (1-s_1)/(N-1)$. These inequalities on the probability of error can be used to obtain the following bound on the average probability of error:
\[ P(\varepsilon) = \int_0^1 (1-s^*) \, h_1(u) \, du \]
\[ = \int_0^{1/N} (1-s^*) \, h_1(u) \, du + \int_{1/N}^1 (1-s^*) \, h_1(u) \, du \]

hence,
\[ P(\varepsilon) < \int_0^{1/N} [1-(1-u)/(N-1)] \, h_1(u) \, du + \int_{1/N}^1 (1-u) \, h_1(u) \, du \]

The procedure introduced above for finding the marginal density function of any one component of \( \mathbf{s} \) in the \( N \)-state symmetric case might be extended for finding the joint density function of any two components of \( \mathbf{s} \). This joint density can then in turn be used to obtain a better bound on the average probability of error of the decoder. This can possibly be extended for finding the joint density function of a larger number of components of \( \mathbf{s} \). However, the computation time required by such extensions might be prohibitively large.

In summary, this paper has formulated a class of models that are useful in the analysis of partially observable Markov processes and demonstrated the type of analysis that is possible. In cases where the underlying Markov process has more than two or three states, computational limitations may prevent exact solutions of the type considered here. The models, however, still prove useful for analysis by simulation in those cases. Application areas for such models are found in quality control, human learning and instruction, medical diagnosis, and search of moving objects.
References


9. Schoen, David C.; "Decoding a Markov Signal Observed Through a Noisy Channel"; S.B. thesis, Department of Electrical