LA-6015-T

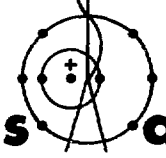
Thesis

UC-32
Reporting Date: June 1975
Issued: August 1975

Investigation of Pattern Recognition Techniques for the Identification of Splitting Surfaces in Monte Carlo Particle Transport Calculations

by

James L. Macdonald



los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

UNITED STATES
ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION
CONTRACT W-7405-ENG. 36

This report is derived from a dissertation submitted to the Graduate School, University of Texas, Austin, TX, in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

This report represents the independent work of the author and has not been edited by the Technical Information staff.

Printed in the United States of America. Available from
National Technical Information Service
U S Department of Commerce
5285 Port Royal Road
Springfield, VA 22151
Price: Printed Copy $7.60 Microfiche $2.25

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INVESTIGATION OF PATTERN RECOGNITION TECHNIQUES FOR THE IDENTIFICATION OF SPLITTING SURFACES IN MONTE CARLO PARTICLE TRANSPORT CALCULATIONS

by

James L. Macdonald

Abstract

Statistical and deterministic pattern recognition systems are designed to classify the state space of a Monte Carlo transport problem into importance regions. The surfaces separating the regions can be used for particle splitting and Russian roulette in state space in order to reduce the variance of the Monte Carlo tally.

Computer experiments are performed to evaluate the performance of the technique using one and two dimensional Monte Carlo problems. Additional experiments are performed to determine the sensitivity of the technique to various pattern recognition and Monte Carlo problem dependent parameters.

A system for applying the technique to a general purpose Monte Carlo code is described. An estimate of the computer time required by the technique is made in order to determine its effectiveness as a variance reduction device. It is recommended that the technique be further investigated in a general purpose Monte Carlo code.

I. Introduction

The Monte Carlo method of particle transport was originally developed by Fermi, Ulam, and von Neumann during the mid 1940's.[1] As digital computers became larger the method became more practical. Because of the large running times encountered on early computers, the method gained the reputation of being extremely time consuming. As a result, the Monte Carlo method has often been referred to as "a method of last resort."[2] The current generation of digital computers, such as the CDC 7600, has reduced the running time of problems previously taking in the hours or days to only a few minutes. As a result, for the Monte Carlo problems run today the method is hardly considered a "method of last resort" and in many cases is the "only method of resort." A description of Monte Carlo code development and the current state of the art at the Los Alamos Scientific Laboratory is given in Section 1.1.

When the Monte Carlo method is used to solve a transport problem, the calculated answer is based on the sampling of statistical processes. Because of this, the answer has associated with it a probabilistic error based on the statistical behavior of the answer. The purpose of a Monte Carlo calculation is to calculate an answer that has a variance below some acceptable level. As a result there have been numerous techniques devised to accelerate the reduction of the variance. These techniques and their corresponding problems are discussed in Section 1.2.

The purpose of this thesis is to allow the computer to assist in the reduction of the variance by using pattern recognition techniques. The field of pattern recognition is very new, beginning with the introduction of large computers in the 1960's. Pattern recognition is discussed briefly in Section 1.3.

Sections 1.4, 1.5, and 1.6 describe the purpose, scope, and outline of the dissertation respectively.

1.1) Monte Carlo Development

The development of the Monte Carlo method as an accepted discipline and research tool began during the second World War from weapons development work.[1] These early applications are usually attributed to the work of Fermi, von Neumann, and Ulam[1] and involved the simulation of neutron diffusion in fissile material. Even at this early stage the techniques of "splitting" and "Russian roulette" were being used for variance reduction;[1] however, the more rigorous development of importance sampling was performed by Harris and Herman Kahn in 1948.[1]

The first open discussions of Monte Carlo applications work took place in 1949 at a symposium sponsored by the RAND Corporation.[3] Since computer machinery did not exist at that time, calculations were usually performed by hand. In the course of describing the usefulness of an alignment chart for making calculations, Spinrad[4] states

> ...it also enables the computer to work completely on one sheet of paper, only interrupting his vision when a new random number is required...

where the term "computer" refers to the person performing the calculation and the random numbers were provided from tables by RAND Corporation[5]. Desk top calculators helped speed up calculations some; however, these early calculators could only add, subtract, multiply, and divide.

At Los Alamos, the first semi-useful equipment for performing Monte Carlo calculations were IBM accounting machines.[6] The development of MANIAC I resulted in the first computer application of Monte Carlo at LASL.[6] However, each problem had to be programmed separately, in machine language. Examples of some of these early problems are given in reference 7.

As Monte Carlo developed in the 50's, it quickly became a "fad" as is described in reference 1:

> ...There was an understandable attempt to solve every problem in sight by Monte Carlo, but not enough attention paid to which of these problems it could solve efficiently and which it could only handle inefficiently; and proponents of conventional numerical methods were not above pointing to those problems where Monte Carlo methods were markedly inferior to numerical analysis...

However, the same author[1] notes when referring to the 60's:

> ...In the last few years Monte Carlo methods have come back into favor. This is mainly due to better recognition of those problems in which it is the best, and sometimes the only, available technique...

The problems for which Monte Carlo is best suited have increased in number for the following reasons:

(1) Improved variance reduction techniques have made Monte Carlo more efficient where before it was very inefficient.

(2) Computer machinery has improved so as to make previously unsolvable problems solvable in a reasonable amount of time.

(3) Increasing demands for details to be included in a problem have in some cases eliminated solution by numerical techniques which required many simplifying assumptions. Examples of this are particle transport involving mixed diffusion and streaming effects, three-dimensional complex geometries, and requirements for non-group energy treatments.

As a result of the demands for Monte Carlo calculations, group TD-6 of the Los Alamos Scientific Laboratory has developed a number of particle transport codes [8]. Although these codes are primarily intended for weapons development, they are often used in many other programs at LASL. These computer codes are used on CDC 7600 computers and include the following:

(1) MCN[19] A neutron transport code

(2) MCG[33] A gamma ray transport code

(3) MCP[33] A general photon transport code (includes lower energy treatment than MCG)

(4) MCNG - A combined neutron-gamma transport code

(5) MCNA[21] A neutron adjoint code

(6) MCK - A criticality code

(7) MCMG - A neutron-gamma multi-group transport code

(8) MCGE[34] A coupled electron-photon transport code

(9) MCGB[34] A gamma code with Bremsstrahlung

In the case of the neutron related codes (except MCMG) the cross-sections are provided as pointwise data that is read into the codes in considerable detail. Although this greatly reduces the number of approximations and distortions caused by cross-section reduction, it does place a considerable burden on the computer. For example, the national Evaluated Nuclear Data File (ENDF) version of iron requires 50,000 words of storage. The codes handle three-dimensional geometry involving first, second, and some fourth (elliptical tori) degree surfaces. All codes are programmed in FORTRAN IV.

Although the research of this dissertation is applicable to Monte Carlo codes in general, it is the above codes that are of particular interest. Thus some of the research is directly related to functions as they are performed in these codes.

1.2) Incentive for this Research

The overwhelming majority of Monte Carlo improvements reported in the literature are related to the reduction of the probabilistic error or variance associated with the Monte Carlo answer or tally. The use of these "variance reduction" techniques varies in proportion to the difficulty encountered in their implementation. The difficulty of implementing many of these techniques is due to:

(1) The complexities involved in applying the technique to real problems. For example, many of the techniques are theoretically based on very simple geometries, etc.; whereas, actual Monte Carlo problems usually involve complex three-dimensional geometries. Many of the techniques do not "scale up" to real applications.

(2) Some of the variance reduction techniques proposed are "unsafe" in that they can distort the calculations resulting in the wrong value for the tally.

(3) Most of the techniques require that a priori information be provided by the user. This information is usually quantitative in nature and depends on the intuition and experience of the user. Furthermore, if the user provides the wrong information, some variance reduction techniques can actually consume more computer time than they save.

(4) Because of the diversity of Monte Carlo problems, different problems require different techniques. As a result, the

user is not sure when to use one technique as opposed to another.

A fundamental problem of all variance reduction techniques is that if one were going to use a technique optimally, he would have to know all the characteristics (including the answer) of the Monte Carlo problem being investigated before applying the technique. Thus what is needed is a technique that instead of requiring information from the user, obtains most of the necessary information during the Monte Carlo calculation.

Such learning* techniques have been proposed in the literature and are discussed in Chapter II. The methods of Spanier[9] and MacMillan[10] involve learning of an optimum parameter for use with the exponential transform technique[11] (see Section 2.3.2). This approach has the disadvantage that it is based on a technique, the exponential transform, which can be unsafe if used improperly. Furthermore, the technique is primarily concerned with directional variables and is difficult to apply to problems involving complex geometries. Another learning method[12] has been proposed which learns optimum spatial quantities. Besides being limited to spatial variables, this technique has difficulties with problems involving small probabilities.

The most successful techniques used in the Los Alamos Monte Carlo codes are geometry splitting[14] and Russian roulette[14] (see Section 2.3.1).

---

* A learning technique in this dissertation refers to the ability of a computer program to improve its performance in solving a problem based on its own experience[13]. This is achieved by a preplanned strategy wherein the program modifies itself based on information gained through experience and evaluation of its previous operations.

These techniques are popular because they are safe and simple to implement. Even when not used optimally, they still yield large savings in computer time. These methods have the disadvantage that they are confined to spatial coordinates (energy splitting can be used, but only independently of geometry splitting) and do require that quantitative information be supplied by the user prior to implementation.

What is needed is a technique which is as safe and simple to use as splitting and Russian roulette, involves all variables of the Monte Carlo problem (spatial, directional, energy, and time) as a whole instead of independently, and relieves the user of the task of providing quantitative information. The development of such a technique using pattern recognition is the subject of this research.

1.3) Pattern Recognition

Before the introduction of large digital computers pattern recognition could only be described as being a human function. Examples of human pattern recognition are:

- recognition of a man from a woman
- recognition of handwritten characters
- recognition of speech
- recognition of a dog from a cat

Pattern recognition is frequently referred to as an "artificial intelligence" technique since it performs an operation on a computer which is usually considered to require intelligence. The pattern recognition

process consists of these basic functions:

(1) identifying which features of the problem being analyzed are important

(2) finding a correlation between these features and various categories (or classes) into which the input can be sorted

(3) sorting future input into classes according to the correlation determined in (2).

These operations can be performed by mathematical transforms that usually require machine learning of some of the parameters involved. These functions will be discussed in more detail in Chapter III.

1.4) Purpose of Dissertation

The purpose of this dissertation is to establish a "proof of principle" for the application of pattern recognition techniques to the identification of splitting surfaces in Monte Carlo particle transport calculations. This is done by:

(1) Developing a pattern recognition system that can be used to learn splitting surfaces in Monte Carlo transport calculations.

(2) Investigating the performance of statistical and deterministic classifiers when used to recognize splitting surfaces. This investigation includes a sensitivity study of the pattern recognition parameters involved.

(3) Proposing a system for applying pattern recognition to a general purpose Monte Carlo code.

(4) Analyzing the effectiveness that can be expected by using pattern recognition as a variance reduction technique.

Thus the purpose of this dissertation is not to apply the technique to a general purpose Monte Carlo code but to establish that such an application would be profitable.

1.5) Scope of Dissertation

The scope of this dissertation is limited in two areas: (1) the selection of a pattern recognition system and (2) the selection of Monte Carlo problems used for demonstration.

There have been many pattern recognition systems developed for a large range of problems. This research investigates two basic techniques (one statistical and one deterministic) which are suitable for the type of information generated in a Monte Carlo calculation. These techniques are used with as little modification as possible from the basic algorithms found in the literature[15,16]. Thus the purpose of this research is not to design an optimum pattern recognition system.

The Monte Carlo problems used in this research have been chosen so as to minimize computer time while still being useful models for demonstrating the operations of the pattern recognition system. Since many Monte Carlo runs are necessary in research of this type (600 to 700 runs were performed), the computer time would be prohibitive (at least 10

times as great) using a general purpose code with complex problems.

The majority of parameter tests and classifier evaluation experiments were performed using a one-dimensional, one-region, homogeneous slab Monte Carlo problem. Although such a model is simplified, it still exhibits the characteristics necessary for the application of pattern recognition. Computational experiments are also performed using a one-dimensional multi-region slab and a two-dimensional (distance and direction) multi-region slab. It is found that the only modification to the pattern recognition system necessary for increasing the dimensionality of the problem is to increase the dimensionality of the various vectors involved.

1.6) Outline of Dissertation

It is assumed in this dissertation that the reader is familiar with statistical terminology (i.e., mean, variance, probability distribution, etc.) but is not familiar with either the Monte Carlo method or pattern recognition theory. The next two chapters are intended to introduce the reader to these topics.

Chapter II introduces the basic principles of Monte Carlo and how statistical errors are calculated. In addition variance reduction techniques are described and the incentives for state space splitting are presented. Finally, a means for measuring the success of a variance reduction technique is described.

Chapter III discusses the general operations of a pattern recognition system. Particular attention is given to the classification

techniques that are used in this research. The problem of feature selection is only described as it relates to the Monte Carlo problem.

In Chapter IV a pattern recognition system is developed for identifying splitting surfaces and performing various parameter tests (items (1) and (2) of Section 1.4). In this chapter a scheme is presented for learning a single splitting surface and is implemented on a one-dimensional one-region slab, a one-dimensional multi-region slab, and a two-dimensional slab Monte Carlo problem. Several parameter tests are made in this chapter and comparisons are made between the different classifiers used. The computer programs used in Chapter IV are given in the Appendices. Although these programs are not implemented on a general purpose Monte Carlo code, several of the timing parameters involved are approximated.

Chapter V considers the practical problem of implementing the technique for full scale applications. A system suitable for general applications is designed and required user input is noted. Finally, an analysis is performed to determine the effectiveness of the technique. Thus Chapter V treats items (3) and (4) of Section 1.4.

Chapter VI states the conclusions of this research and makes recommendations for implementing this research on a full scale in a general purpose Monte Carlo code.

II. The Monte Carlo Method and Variance Reduction

Although the Monte Carlo method is applied to a wide range of problems, the emphasis of this research will be on particle transport calculations. It is the purpose of this chapter to describe in very general terms how Monte Carlo sampling is performed and how statistical errors are determined (Section 2.1). For further details of the sampling used for neutron transport, the reader is referred to Appendix A. Section 2.2 contains a description of variance reduction in general and Section 2.3 surveys some of the more common variance reduction techniques including those which involve learning. A method is then proposed which requires learning during the Monte Carlo calculation in order to reduce the variance (Section 2.4). Finally, in Section 2.5 effectiveness of Monte Carlo calculations will be defined so that the effect of variance reduction can be measured.

2.1) The Monte Carlo Method for Particle Transport

2.1.1) Basic Principles

The transport problem in this research consists of estimating the probability that particles leaving a source and undergoing various processes (capture, escape, etc.) will finally terminate in a specified category or tally. Decisions as to which processes occur are made by sampling the appropriate "probability distribution" functions as described below.

The probability that a variable s lies between s and s+ds is given by p(s)ds where p(s) is defined as the probability density function [17] An example of such a function is shown in Figure 2.1 for s ranging from 0 to 3. In this research it is always assumed that p(s) has been normalized so that

$$\int p(s)ds = 1$$

where the integral is over all possible s.

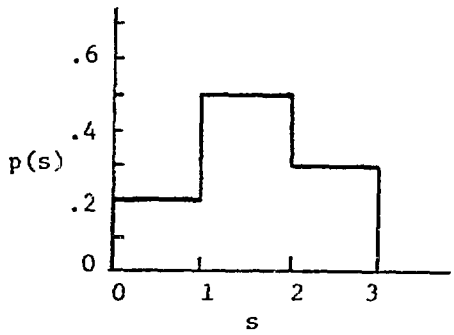


Figure 2.1 A Probability Density Function

The integral of p(s) is defined as the probability distribution function: [19]

$$P(s) = \int_0^s p(s')ds' \qquad (2.1)$$

The distribution function of the example in Figure 2.1 is shown in Figure 2.2.

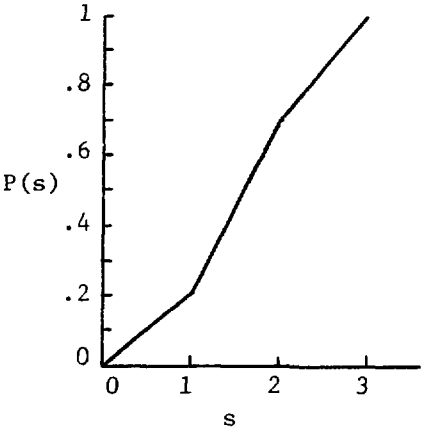


Figure 2.2 A Probability Distribution Function

The probability distribution function is sampled by choosing a random number, r, between 0 and 1 and setting r equal to P(s) as given by

$$r = \int_0^s p(s')ds' \qquad (2.2)$$

The s that satisfies this equality is used as the sampled value. The values of s sampled in this manner can be shown to have the probability density p(s).[6]

As an example of this sampling process, consider the case of a neutron emitted from one side of a slab (see Figure 2.3) in the +x direction.

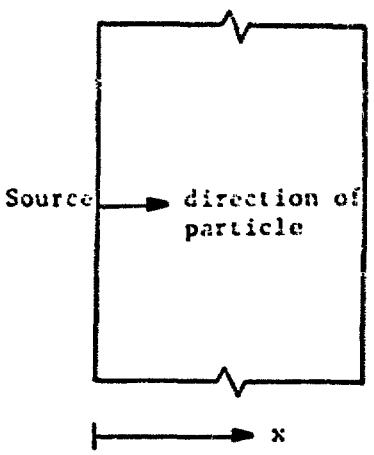


Figure 2.3 Sampling Distance to Collision

The probability that this neutron has a collision between x and x+dx is given by [16]

$$p(x)dx = \Sigma_t e^{-\Sigma_t x} dx$$

where Σ_t = the total macroscopic cross section

p(x) = the probability density function for a collision at x.

The probability distribution function for this process is given by

$$P(x) = \int_0^x p(x')dx'$$

$$= \int_0^x \Sigma_t e^{-\Sigma_t x'} dx'$$

$$P(x) = 1 - e^{-\Sigma_t x}$$

Note that if $x = \infty$, $P(x)=1$ and if $x=0$, $P(x)=0$. Setting a random number, r, $(0 \leq r \leq 1)$ equal to the probability distribution function gives

$$r = P(x) = 1 - e^{-\Sigma_t x}$$

$$x = \frac{\ln(r)}{\Sigma_t}$$

where (1-r) has been replaced by r.

The value of x for a given r is the sampled value for the distance to a collision. In Monte Carlo transport calculations many processes are sampled similar to the example above. Appendix A describes the sampling process in more detail for the case of neutron transport.

Eventually, after undergoing numerous events as determined by the appropriate probability distributions, a particle is lost to the system. This occurs when the particle is either captured, leaves the system being considered, falls below the energy range of interest, etc. At this point the contribution, x_i, of the i'th particle to the tally under study is calculated. Thus, for N particles the average contribution to the tally is

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{2.3}$$

This $\bar{x}$ is the statistical approximation used to estimate the physical quantity of interest.

2.1.2) Estimate of Statistical Errors

The mean value of N samples, $\bar{x}$, is given by Equation 2.3. The variance[17] of the N samples with respect to the sample mean $\bar{x}$ is given by

$$\bar{\sigma}^2(\bar{x}) = \frac{1}{N-1}\left[\sum_{i=1}^{N}(x_i - \bar{x})^2\right] \tag{2.4a}$$

$$\bar{\sigma}^2(\bar{x}) = \frac{N}{N-1}\left[\overline{x^2} - \bar{x}^2\right] \tag{2.4b}$$

The true mean[17] of x is given by

$$\langle x \rangle = \int_{-\infty}^{\infty} x(s)p(s)ds \tag{2.5}$$

where p(s) is the probability density function of s. The mean $\langle x \rangle$ is often referred to as the "expected"[17] value of x(s). For an unbiased[17] estimate $\bar{x}$,

$$\langle \bar{x} \rangle = \langle x \rangle \tag{2.6}$$

The <u>true variance</u>[17] of x(s) is defined as the second moment of x(s) about $\langle x \rangle$ as given by

$$\sigma^2(x) = \int_{-\infty}^{\infty} \Big(x(s) - \langle x \rangle\Big)^2 \; p(s)ds \tag{2.7}$$

It can be shown (see Appendix B) that the variance of the samples x_i about the true mean $\langle x \rangle$ is given by

$$\sigma^2(\bar{x}) = \frac{\sigma^2(x)}{N} \tag{2.8}$$

Since in practice neither $\langle x \rangle$ or $\sigma^2(x)$ is known, they are approximated by $\bar{x}$ and $\bar{\sigma}^2(\bar{x})$. Making these substitutions and assuming large N results in

$$\sigma^2(\bar{x}) \approx \frac{\bar{\sigma}^2(\bar{x})}{N} \approx \frac{\overline{x^2} - \bar{x}^2}{N} \tag{2.9}$$

as an estimate of the variance of the sample mean.

The Central Limit Theorem[17] (see Appendix B) states that

$$\text{Prob}\left\{\alpha\bar{\sigma}(x) < (\bar{x} - \langle x\rangle) < \beta\bar{\sigma}(x)\right\} \approx \frac{1}{\sqrt{2\pi}}\int_{\alpha}^{\beta} e^{-t^2/2}dt \ . \qquad (2.10)$$

For $\alpha = -1$ and $\beta = 1$, Equation 2.10 means that there is a 68.3% probability that the estimated mean is within $\pm\bar{\sigma}(x)$ of the true mean.

Frequently in Monte Carlo calculations it is helpful to express the error in terms of relative error[19] as given by

$$\text{Re} = \frac{\sigma(\bar{x})}{\bar{x}} \approx \frac{1}{\sqrt{N}}\ \frac{\bar{\sigma}(\bar{x})}{\bar{x}} \ . \qquad (2.11)$$

Equation 2.11 exhibits a very important characteristic of Monte Carlo calculations -- that the error of the sample mean varies as $1/\sqrt{N}$.

2.2) Variance Reduction

In order to decrease the error of a Monte Carlo calculation (see Equation 2.11), one must either increase N or decrease $\overline{\sigma}(\overline{x})$. The effectiveness of increasing N to reduce the error is illustrated in Figure 2.4. From this figure it is apparent that as N increases, the decrease in Re, ΔRe, for a given increase in N, ΔN, decreases. For example, increasing the number of particles from N=100 to N=10,000 reduces the error by a factor of 10; however, increasing N from 10,000 to 20,000 reduces the error by a factor of only $\sqrt{2}$. Although computer time spent per particle history is an extremely problem dependent parameter, in many cases running time becomes prohibitive after a sample of 10,000 to 100,000 particles. If the relative error is still unacceptably large after several 10,000 particles, additional histories are far too costly for the small amount of error reduction gained.

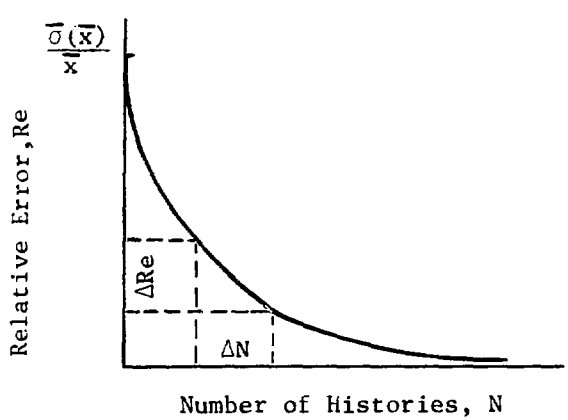


Figure 2.4 Reducing Error by Increasing the Number of Histories

Because of this problem variance reduction techniques are often required to reduce the variance of the sample mean. As seen by Equation 2.4a, *the only way to do this is to sample values of* x_i *which are closer to* $\bar{x}$.

One technique frequently used for doing this is the implementation of particle "weights"[7,11] with the elimination of capture. The weight of a particle can be thought of as representing a fraction of a particle. For example, a weight of 1.0 represents an entire particle whereas a weight of 0.5 represents only half a particle. When a particle undergoes a reaction, it is never "killed" by a capture, but instead its weight is multiplied by the factor Σ_{na}/Σ_t (Σ_{na} = non-absorption cross section, Σ_t = total cross section) and the particle history is continued with reduced weight.

Example. Consider a Monte Carlo problem in which 100 neutrons are started from a source. Of these neutrons 30% leak out of the system without a collision and 30% are captured at their first collision. The remaining neutrons undergo one scattering collision after which they are tallied with a value of x_i=1. Using no weights and assuming neutrons behave exactly as the above percentages indicate, one arrives at:

$$\bar{x} = \frac{\sum_{i=1}^{100} x_i}{100} = \frac{40x1.0 + 60x0.0}{100} = .4$$

$$\bar{\sigma}^2(\bar{x}) = \frac{1}{100}\sum_{i=1}^{100}(x_i-\bar{x})^2 = \frac{1}{100}\left[60x(.4)^2 + 40x(.6)^2\right] = .24$$

Using weights and assuming that each neutron that does not escape undergoes one collision and is then tallied results in:

$$\bar{x} = \frac{\sum_{i=1}^{100} x_i w_i}{100} = \frac{30x0.0 + 70x4/7}{100} = .4$$

$$\bar{\sigma}^2(\bar{x}) = \frac{1}{100}\sum_{i=1}^{100}(x_i-\bar{x})^2 = \frac{1}{100}\left[30x(.4)^2 + 70x(.17)^2\right] = .0686$$

A name applied to a family of variance reduction techniques is "importance sampling"[14,11]. In transport problems, importance sampling refers to preferentially sampling those particles which are more likely to contribute to the tally being investigated. From a probability density function p(x) the mean value of a function f(x) is given by

$$\left\langle f(x)\right\rangle = \int f(x)p(x)dx \qquad (2.12)$$

In importance sampling an alternate distribution $p_A(x)$ is used and the function f(x) is multiplied by w(x) where

$$w(x) = \frac{p(x)}{p_A(x)} \quad . \tag{2.13}$$

Using this alternate distribution gives a mean of

$$\left\langle f_A(x) \right\rangle = \int f_A(x) p_A(x) dx = \int f(x) \left[\frac{p(x)}{p_A(x)} \right] p_A(x) dx = \left\langle f(x) \right\rangle \tag{2.14}$$

Thus the mean is unchanged. However, the second moment of $f_A(x)$ is given by

$$\begin{aligned} \left\langle f_A^2(x) \right\rangle &= \int f_A^2(x) p_A(x) dx = \int \left[\frac{p(x)}{p_A(x)} \right] f^2(x) p(x) dx \\ &= \int I(x) f^2(x) p(x) dx, \end{aligned} \tag{2.15}$$

$$\text{where } I(x) = \left[\frac{p(x)}{p_A(x)} \right] \quad .$$

This is not the same as the second moment of the unaltered distribution which is

$$\left\langle f^2(x) \right\rangle = \int f^2(x) p(x) dx \qquad (2.16)$$

In Equation 2.15 $< f_A^2(x) >$ can be reduced by decreasing I(x) where p(x) is large. This requires that I(x) be increased where p(x) is small. In the ideal case $p_A(x) = \frac{f(x)p(x)}{< f(x) >}$ resulting in

$$\sigma^2(f_A(x)) = \left\langle f_A^2(x) \right\rangle - \left\langle f(x) \right\rangle^2$$

$$= \int \left[f(x) \frac{p(x)}{p_A(x)} - \left\langle f(x) \right\rangle \right]^2 p_A(x) dx = 0 \qquad (2.17)$$

However, before this zero variance $p_A(x)$ can be found, the mean $< f(x) >$ must be known, which of course is never the case. In the following section several techniques based on importance sampling will be discussed.

2.3) Survey of Variance Reduction Techniques

During the development of Monte Carlo, there have been numerous techniques proposed to reduce the variance of the Monte Carlo tally. However, when one looks at the major Monte Carlo codes, he finds that only very few of these techniques are used. One of the reasons for this is that many methods are "unsafe" to use because they may bias the answer or may

actually consume more computer time than they save. Other methods are rarely used because they are too complex to implement. The majority of this section will be devoted to the description of splitting[14] and Russian roulette[14] which are perhaps the most widely used variance reduction techniques. A brief description of some other popular techniques will also be given. In Section 2.3.3 techniques which require "learning" during the calculations will be discussed.

2.3.1) Splitting and Russian Roulette

Splitting accompanied by Russian roulette is one of the most commonly used variance reduction techniques.[6] It consists of dividing the geometry of the problem into regions and assigning an importance to each region. This "importance" is selected so that particles in a region of high importance have a higher probability of contributing to the tally. A particle going from a region of low importance to one of greater importance is split at the boundary between the regions into two or more particles (the number depends on the ratio of the importances) with each new particle having a reduced weight. A particle entering a region of lesser importance is terminated or "killed" with a probability determined by the ratio of importances. If the particle survives the Russian roulette, its weight is increased proportionately.

Figure 2.5 shows an example of splitting planes and importance regions used with an infinite slab of thickness T. For problems in which T is many mean free paths, splitting and Russian roulette can be very effective and often lead to several orders of magnitude reduction in

computer time.

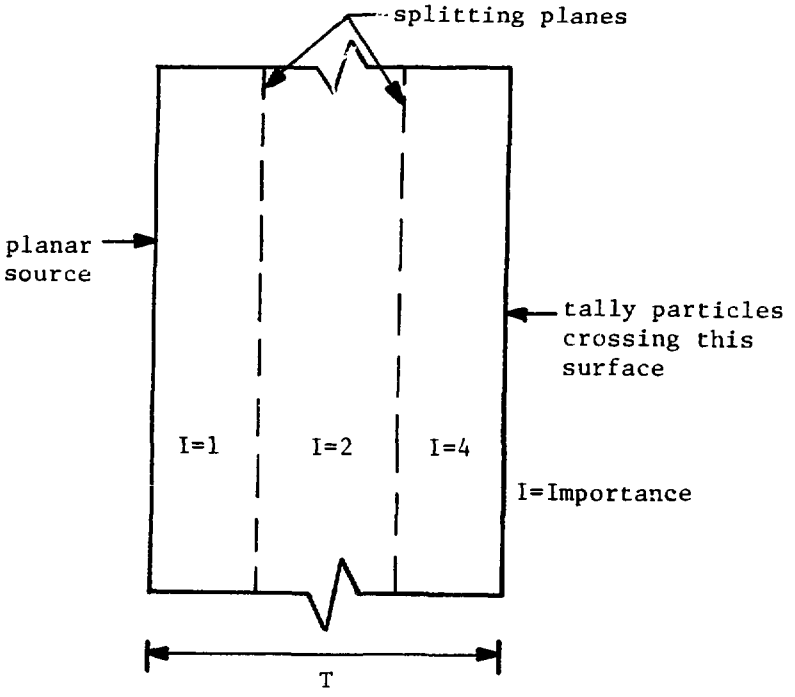


Figure 2.5 Splitting Planes in One Dimension

Splitting and Russian roulette can also be used in energy space for problems in which particular energy regions are more important than others. An example of "energy splitting"[14] is the tallying of U^{235} thermal fission. In this case, one would separate energy space into regions which increase in importance as thermal energies are approached as shown in Figure 2.6.

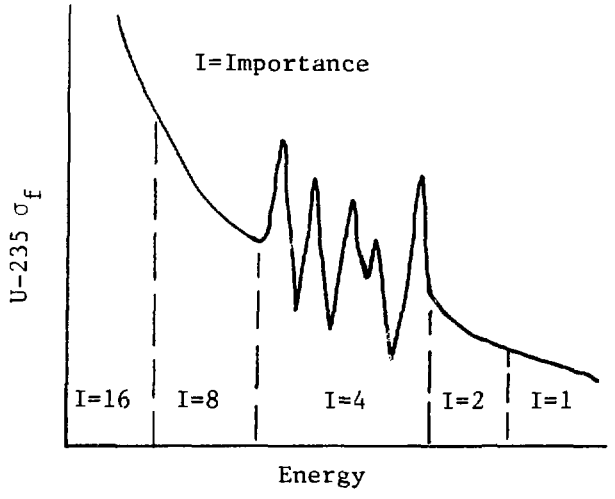


Figure 2.6 Energy Splitting

The popularity of the above techniques can be attributed primarily to the ease of their implementation. In most cases only a very rough guess based on intuition will lead to a large savings in computer time. Usually the importance regions specified are already geometrically defined by the problem (different materials, densities, and shapes) and the user only has to provide the importances.

2.3.2) Other Techniques

A very simple technique commonly used is source biasing[7,14]. In source biasing important particles are produced more frequently but with reduced weights. An example of source biasing is shown in Figure 2.7.

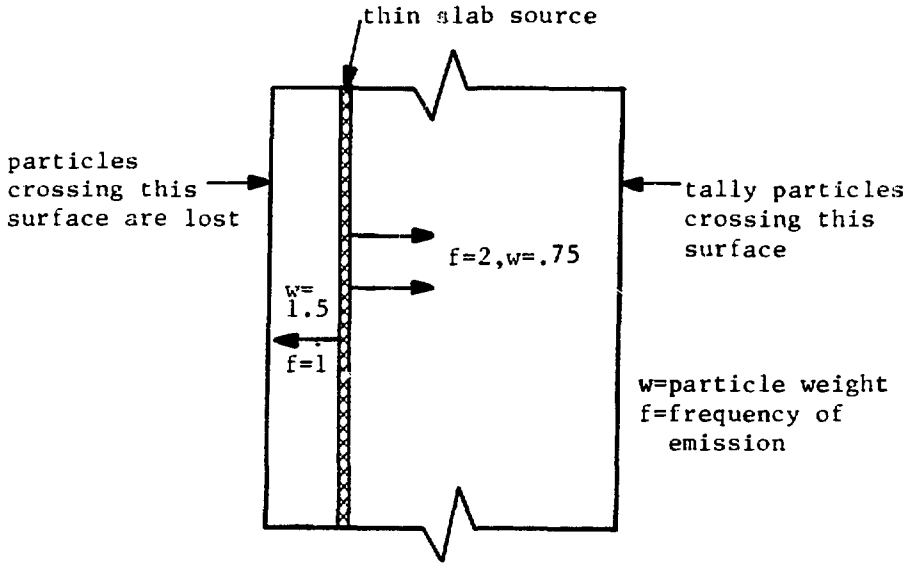


Figure 2.7 Source Biasing

In this example twice as many particles are started to the right as to the left. However particles to the left have twice as much weight.

Another method used to increase the number of particles in important regions is the exponential transform[11,14,30]. This technique transforms the transport equation, resulting in the replacement of Σ_t by $\Sigma_t - w$ where w is the direction cosine of the line of flight of the particle with the preferred line of flight. Figure 2.5 shows the case where the desired line of flight is the x-axis. The weight of a particle entering a collision is multiplied by

$$\frac{\Sigma_t}{\Sigma_t - \alpha w} e^{-\alpha w s} , \qquad (2.18)$$

where s is the distance traveled before collision.

A problem in using many importance sampling techniques is that of choosing near optimum biasing parameters. This is frequently done by rough calculations or maybe even a few preliminary Monte Carlo calculations. Another method is to use the solution to the adjoint[14,20] of the problem to estimate these parameters. The computation of importance sampling functions has also been automated by other means.

Other variance reduction techniques include stratified sampling[1], antithetic variates[11], scattering angle biasing[21], method of expected values[11], correlated sampling[11,29], and others[32].

2.3.3) Variance Reduction Through Learning

In all of the previously mentioned techniques, importance sampling parameters had to be provided prior to the execution of the Monte Carlo calculations. In this section techniques will be described which allow the variance reduction technique to improve during operation by learning from early histories of the calculations.

Spanier[9] applies a learning technique to the exponential transform using a one-dimensional slab as an example. The parameter α (see Section 2.3.2) is optimized by making estimates of $\langle f_A^2(x) \rangle$ for several values of α while histories are being generated on the basis of the

parameter value $\hat{\alpha}$. An α which minimizes $\langle f_A^2(x) \rangle$ can then be used in another iteration as the next $\hat{\alpha}$. This process continues until satisfactory agreement is reached between two stages. In the examples given three iterations were sufficient and led to a greatly reduced variance. MacMillan[10] suggests a refinement on Spanier's method involving estimates of the first and second derivatives of $\langle f_A^2(x) \rangle$ with respect to α and using these estimates to improve the approximation of $\hat{\alpha}$ in going from one iteration to another.

A multistage self-improving Monte Carlo method[12] has been described which divides space into volumes V_i and assigns each volume a weight p_i where p_i determines the amount of sampling for associated V_i. The Monte Carlo calculation then proceeds in stages after which p_i and V_i are altered in such a way as to reduce the variance. This method is analogous to learning the optimum importances for different geometry regions only in this case the extent of the regions is variable. For small probability problems the range of the tally is enlarged to increase the probability until suitable V_i and p_i are learned after which the tally is reduced to its original specifications. Running times have been reduced as much as a factor of 100 using this method over the crude Monte Carlo[12].

The Spanier and MacMillan techniques are primarily concerned with the directional variables of a Monte Carlo problem and since they are based on the exponential transform, they can be unsafe to use. The multistage technique is concerned primarily with spatial variables. Furthermore, this technique has disadvantages when used with Monte Carlo problems with low probability.

2.4) Splitting and Russian Roulette in State Space

Consider the general Monte Carlo problem in which particles are characterized by the following state variables:

- Spatial coordinates-x,y,and z
- Angular coordinates-u,v, and w, where these values are the cosines of the particle line of flight with the x,y, and z axis respectively.
- Energy-E
- Time-t

In Section 2.3.1 splitting and Russian Roulette were described primarily as applied to the spatial coordinates. Independent application was also mentioned with respect to the energy variable.

Theoretically,it would be quite effective if splitting could be used in the entire state space. In other words all variables would be considered to determine which regions in state space are more important than others. A practical problem arises in determining the importances of these state space regions. Users have trouble enough with the three spatial coordinates; the complexity involved in determining splitting surfaces in eight dimensions would certainly confuse even the most experienced user.

As has been seen in the previous sections, there is a considerable amount of information generated during a Monte Carlo calculation which can be used to accelerate the calculation. However, utilization of this information can become costly in terms of computer time and storage. In this research, pattern recognition techniques are used to

learn these splitting surfaces during the calculations.

Such a technique would be an improvement over the learning techniques described in Section 2.3.3 for two reasons:

(1) All state space parameters would be considered, not just directional or spatial quantities.

(2) The technique is based on the splitting and Russian roulette techniques which have proven to be the most popular and useful techniques.

2.5) Effectiveness of Variance Reduction Techniques

Although it is certainly useful in Monte Carlo calculations to reduce the variance, the primary goal is to reduce the amount of computer time spent on a calculation. It is quite possible to use variance reduction to decrease σ for a given N but in so doing to increase the time spent per particle to such an extent that it would be cheaper just to run more particles. Therefore, the parameter to minimize is the time required to obtain the desired relative error as given by

$$t_c = N_e \Delta t_e \tag{2.19}$$

where: N_e = number of histories required to obtain the desired Re (see Equation 2.11)

Δt_e = time spent per neutron history

Variance reduction techniques decrease N_e but increase Δt_e. If t_c^*, N_e^*, and Δt_e^* are the parameters of a calculation without using variance reduction, then the relative effectiveness of a technique can be defined as

$$E_R = \frac{t_c}{t_c^*} \quad \frac{N_e \Delta t_e}{N_e^* \Delta t_e^*} \tag{2.20}$$

It should be remembered that E_R is relative to the following factors:

(1) the computing machinery being used,

(2) the efficiency of the Monte Carlo calculation without variance reduction,

(3) the characteristics of the particular problem under study and,

(4) the programming efficiency used to implement the technique (i.e., assembly language vs. FORTRAN etc.).

Of the above, point three is the most important since the effectiveness of a technique is strongly dependent on the problem to which it is being applied.

Computer time spent during a calculation is not the only measure of performance for evaluating variance reduction techniques. Another parameter is the amount of human effort (and sometimes additional computer time) required to implement a technique. The majority of techniques in use require a certain amount of *a priori* information. Thus the

implementation of some techniques is an art in itself requiring considerable ingenuity and experience on the part of the user. In addition some techniques require that calculations be performed in order to determine importance parameters, etc. Figure 2.8 illustrates the operations and times required in implementing a variance reduction technique. The importance attributed to time spent on each of these operations is an extremely subjective function yet it certainly influences the overall acceptability of a technique.

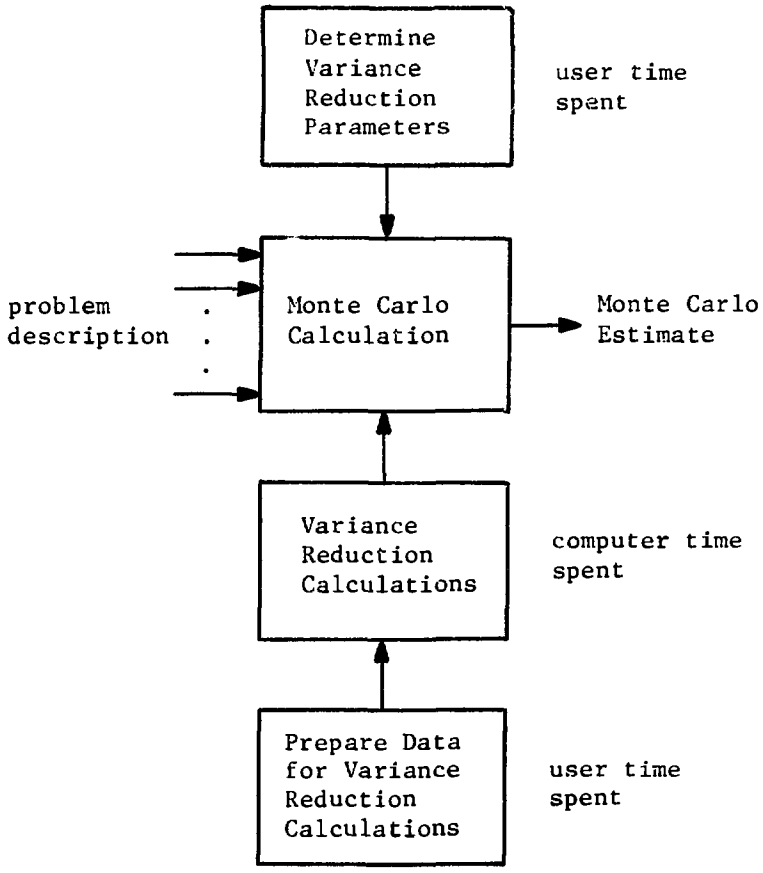


Figure 2.8 Implementation of Variance Reduction Techniques

III. Pattern Recognition

The field of pattern recognition includes an extremely broad range of topics including engineering applications, artificial intelligence studies, biological systems, and others. Because of its diffuse application, a general theory of pattern recognition is difficult to separate from its applications. To confuse matters further, it appears that even the introductory texts[15,16,22,23,24,25,26] on pattern recognition do not agree on a unified framework for describing pattern recognition systems. As a result, a novice in the field frequently encounters a variety of new vocabulary words describing types of pattern recognition systems including such terminology as statistical, parametric, non-parametric, sequential, distribution free, stochastic, nonsupervised, supervised, error-correcting, Bayesian, etc.

The purpose of this chapter is not to explain all facets of pattern recognition to the reader, but only to provide him with the tools necessary to understand how pattern recognition is to be used in this research.

A general pattern recognition system will be explained in Section 3.1 in terms of the basic operations performed. The different types of pattern classification algorithms will then be classified according to the type of input data they require . Sections 3.2 and 3.3 describe in more detail the type of pattern classification algorithms to be used in this research. Section 3.4 discusses the problem of feature selection with emphasis on the Monte Carlo transport problem. Finally, Section 3.5 expands the previous explanations to multiclass problems.

3.1) The General Pattern Recognition Problem

3.1.1) Fundamentals

The primary difficulty with understanding the fundamentals of a pattern recognition system is due to the terminology used. This section introduces pattern recognition terminology by relating the concepts to the simple example of weather prediction as given below:

Example. Given the following information -

(1) barometric pressure,

(2) temperature, and

(3) percent cloud coverage,

predict whether it will

(1) rain, or

(2) not rain.

The input - output relationship of a system to perform this task is shown in Figure 3.1.

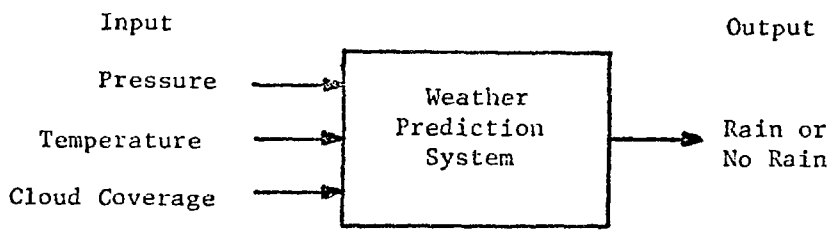


Figure 3.1 Input-Output Model of Weather Prediction

A coordinate system defined by the input variables of Figure 3.1 is shown in Figure 3.2

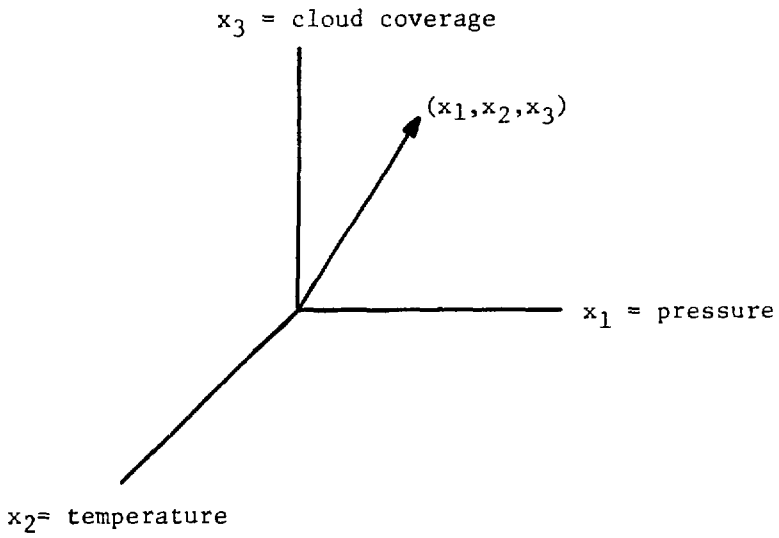


Figure 3.2 Pattern Space

and is referred to as pattern space. The vector drawn from the origin of pattern space to the point (x_1,x_2,x_3) in Figure 3.2 is called the pattern vector and in this dissertation will be designated by

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_R \end{bmatrix}$$

where R is the dimension of pattern space (R=3 for the example shown in Figure 3.2). The purpose of the pattern recognition system in this example is to divide pattern space into two regions: (1) those X which indicate rain and (2) those X which indicate no rain. The options rain and no rain are called classes and are referred to as C_1 and C_2 where

$$C_1 = \text{rain and}$$

$$C_2 = \text{no rain.}$$

In general, the purpose of a pattern recognition system is to classify pattern vectors into their appropriate classes $C_1, C_2, \ldots, C_K$ where K is the number of classes (see Figure 3.3).

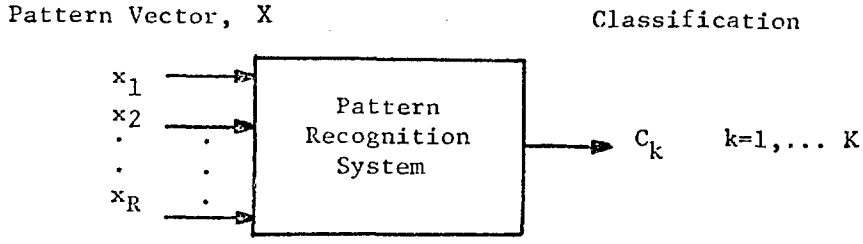


Figure 3.3 Input-Output Model of a Pattern Recognition System

The structure of a pattern recognition system can often be simplified if pattern space is transformed into a more efficient configuration.

For instance, if in the weather prediction example, it is found that the temperature is of no value for predicting rain (i.e. there is no correlation between temperature and rain) and that the probability of rain increases in proportion to the square of the percent cloud cover, then it would be more efficient to use the coordinate system shown in Figure 3.4.

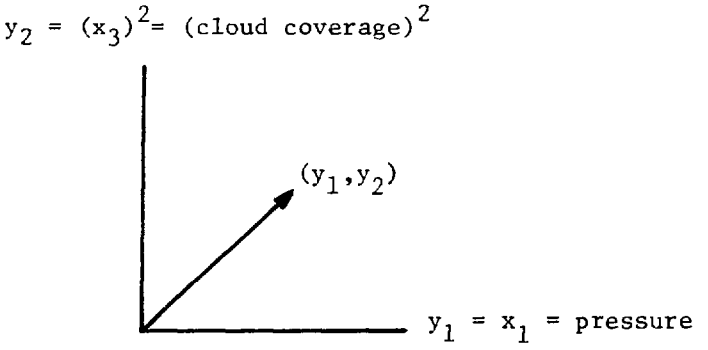


Figure 3.4 Feature Space

This new coordinate system is referred to as <u>feature space</u>. The vector from the origin of feature space to the point (y_1,y_2) is called the <u>feature vector</u> and in this dissertation is designated by

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix}$$

where N is the dimensionality of feature space. The process of transforming a pattern vector into a feature vector is called feature selection*. The input - output relationship of a feature selector is shown in Figure 3.5 where in general $N<R$.

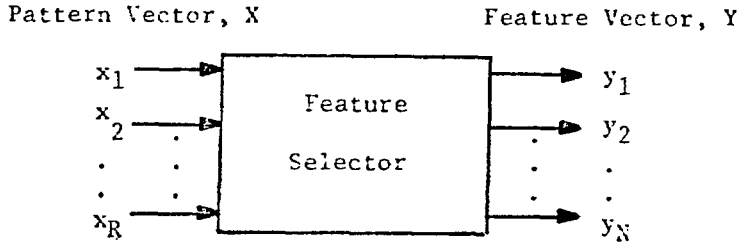


Figure 3.5 Feature Selection

The feature selection operation is highly problem dependent and will be discussed further in Section 3.4.

The operation of classifying the feature vector into classes $C_1, C_2, \ldots, C_K$ is called pattern classification. Thus the pattern recognition system consists of two major components: feature selection and pattern classification (see Figure 3.6).

---

* In this dissertation any operation performed on the pattern vector prior to classification is considered to be a feature selection operation.

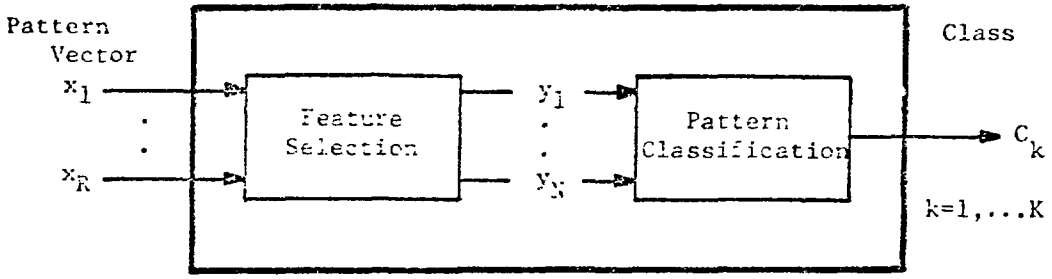


Figure 3.6 A Pattern Recognition System

Given functions $g_k(Y)$, k=1,2,...,K, of the feature vector, Y, such that

$$\text{if } g_k(Y) > g_i(Y) \qquad i=1,2,\ldots,K \quad i \neq k \tag{3.1}$$

then Y is placed in class C_k by the pattern classifier. The function $g_k(Y)$ is called the discriminant function of C_k. If K=2, as in the weather forecasting example, a discriminant function $g_{1,2}(Y)$ can be defined such that

$$g_{1,2}(Y) = g_2(Y) - g_1(Y) \quad . \tag{3.2}$$

In this case,

$$\text{if } g_{1,2}(Y) > 0, \text{ then Y belongs to } C_2 \tag{3.3}$$

$$\text{if } g_{1,2}(Y) < 0, \text{ then Y belongs to } C_1 \quad .$$

The surface for which

$$g_k(Y) = g_i(Y) \qquad \begin{matrix} k=1,2,....K \\ i=1,2,....K \\ i \neq k \end{matrix} \tag{3.4}$$

is called the decision surface between C_k and C_i. The decision surfaces separate feature space into K regions. The Y's in each region belong to the same class. For a two class problem the decision surface is given by

$$g_{1,2}(Y) = 0 \quad . \tag{3.5}$$

Such a surface (a line in this case) is shown in Figure 3.7 for the weather forecasting example.

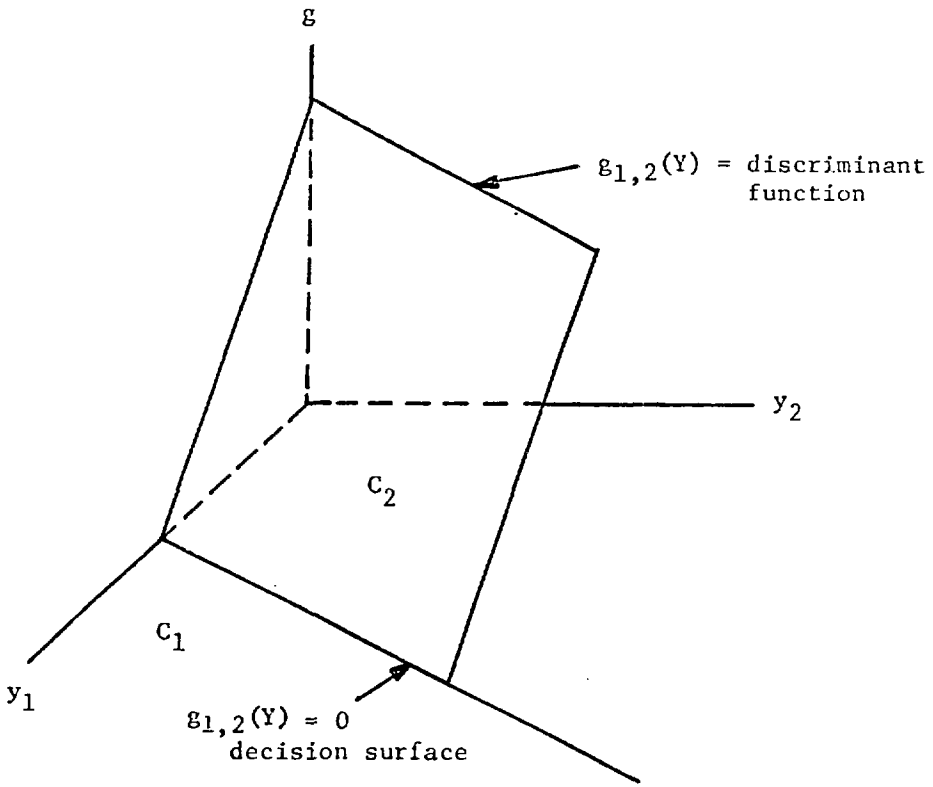


Figure 3.7 Decision Surface

The previous discussion is concerned with how features are classified and is true for pattern classifiers in general. However, before the classifier can operate, the form of the discriminant functions, $g_k(Y)$, must be known. How the $g_k(Y)$ are arrived

at for different pattern classifiers is the subject of the next section.

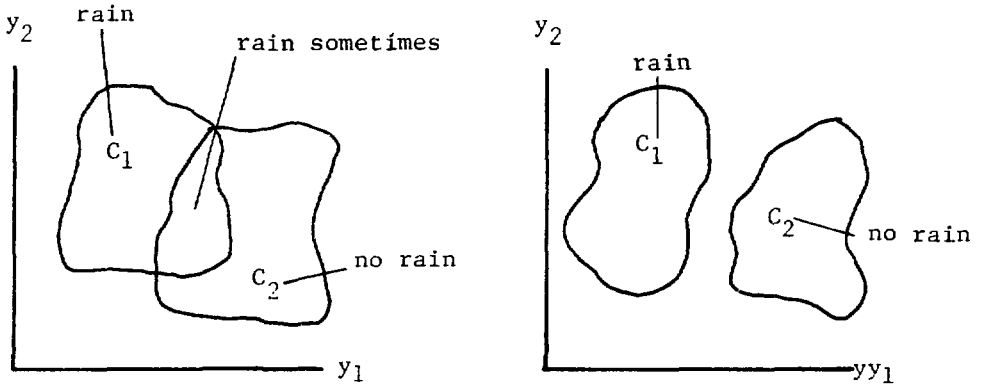
3.1.2) Types of Pattern Classifiers

Every feature vector has associated with it a probability of belonging to a given class. This probability will be denoted by $p(C_i|Y)$ which is the probability that feature vector Y belongs to class C_i. For a two-class problem, if one class can be uniquely associated with each pattern such that

$$\text{if } p(C_1|Y) > 0 \text{ then } p(C_2|Y) = 0 \qquad (3.6)$$

$$\text{if } p(C_2|Y) > 0 \text{ then } p(C_1|Y) = 0,$$

then the classes are said to be non-overlapping. If patterns can belong to one class sometimes and the other class at other times, the classes are said to be overlapping. Examples of these two different types of distributions are shown in Figure 3.8 using the weather prediction example.



(a) Overlapping (b) Non-Overlapping

Figure 3.8 Overlapping and Non-Overlapping Classes

Classifier input data and the schemes they require can be separated into different cases depending on what is known about $p(C_i|Y)$ and the input data[27]. There are basically four types of information of which one or more may be available to the classifier. These information types are[27]:

(1) Functional form of $p(C_i|Y)$ is known. For example it may be known that both $p(C_1|Y)$ and $p(C_2|Y)$ are Gaussian but with unknown means and variances.

(2) Parameters of $p(C_i|Y)$ are known. Parameters include the mean, variance, etc.

(3) Sample pattern vectors with known classification are given. Each pattern vector with its classification is called a <u>prototype</u>. These prototypes serve as a training set for the classifier.

(4) Sample pattern vectors of unknown classification are given.

Depending on which of the above information is available, six major kinds of pattern classifiers can be defined[27].

(1) Case A: Information types 1 and 2 are given

(2) Case B: Information types 1 and 3 are given

(3) Case C: Information types 1 and 4 are given

(4) Case D: Information type 3 is given,Deterministic methods are used

(5) Case E: Information type 3 is given, Statistical methods are used

(6) Case F: Information type 4 is given

In Case A all the information required to make an analytical solution for g(Y) is known. In Case B the classified Y's must be used to make an approximation of the required parameters after which the classifier becomes a Case A. Cases C and F are often referred to as "learning without a teacher" or unsupervised learning[16] and usually consist of a type of clustering technique.[16]

In Case D the basic idea is to find a g(Y) which operates "satisfactorily" on the samples of known classification. This type of approach, sometimes referred to as "distribution free"[16,15], makes no assumptions concerning the $p(C_i|Y)$. Instead the data is assumed to be separable by a given form of g(Y), i.e., linear, quadratic, etc. One drawback of such an approach is that it places an additional burden on the feature selector in order to produce feature vectors which satisfy the assumptions made on g(Y).

Case E consists of using statistical techniques to minimize classification errors. These techniques are theoretically more useful for overlapping data since they allow for the existence of error. Frequently in the statistical approach $p(C_i|Y)$ is expanded in a series[26]

$$p(C_i|Y) = \sum_{j=1}^{J} \alpha_{ij} \phi_j(Y), \qquad i=1,2 \tag{3.7}$$

where the α_{ij}'s are approximated by using prototypes. A simpler approach is similar to the deterministic approach and consists of assuming a form for $g(Y)$.[28] However, unlike the deterministic classifier, the input data need not conform to the assumptions made on $g(Y)$ since in this case $g(Y)$ is approximated by the statistical behavior of the data in order to minimize the number of misclassifications.

Classification techniques for Cases B through F can be further characterized as sequential[23] or non-sequential techniques[16]. In sequential techniques the prototypes are presented one at a time and approximations are made concerning $g(Y)$ or $p(C_i|Y)$ as each prototype is presented. In non-sequential techniques a finite number of prototypes is presented at once to the classifier and an optimum $g(Y)$ or $p(C_i|Y)$ is fitted to these prototypes.

In summary, the selection of a pattern classification scheme depends upon the information available (Cases A-F), whether the $p(C_i|Y)$ are overlapping, and the manner in which the prototypes are presented (sequential or non-sequential). Because of the characteristics

of Monte Carlo problems, only Case D and E classifiers will be investigated in this study. A more detailed description of these classifiers using sequential learning methods is presented in Sections 3.2 and 3.3.

3.2) Sequential Deterministic Classification Techniques (Case D)

In this section several deterministic techniques are described for classifying patterns when prototypes of known classification are presented sequentially. This approach consists of assuming a form for g(Y) and using the prototypes to learn the necessary parameters. Two class problems (K=2) are assumed resulting in a single discriminant function $g(Y) \equiv g_{1,2}(Y)$ (see Equation 3.2).

3.2.1) Linear Discriminant Functions

The general form of a linear discriminant function for N dimensional feature space is given by

$$g(Y) = w_1 y_1 + w_2 y_2 + \ldots\ldots w_N y_N + w_{N+1} = W^T Y^* \qquad (3.8)$$

where $Y^* = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_N \\ 1 \end{bmatrix}$, $W = \begin{bmatrix} w_1 \\ 2 \\ \cdot \\ \cdot \\ w_{N+1} \end{bmatrix}$

W^T = the transpose of W

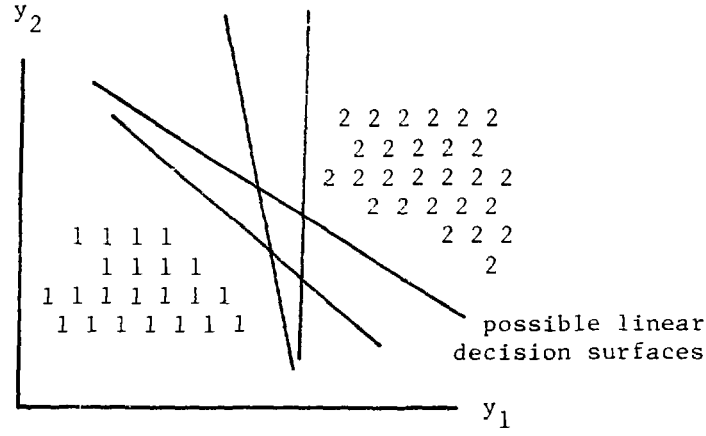
and the vector W (the weight vector) must be obtained from information contained in the prototypes. The vector Y* is called the augmented[16] feature vector and is of dimension N+1. The use of g(Y) as given by Equation 3.8 assumes that feature space is linearly separable. For two-dimensional feature space this means that all feature vectors belonging to C_1 can be separated by a linear decision surface (a straight line for N=2) from all feature vectors belonging to C_2. Figure 3.9 illustrates linearly and non-linearly separable feature vectors. Note that for the linearly separable data shown (Figure 3.9a), there is an infinite number of decision surfaces which satisfactorily separate feature space. For the data of Figure 3.9b there is no linear g(Y)=0 that will separate the classes.

The coordinate system created by the components of the weight vector, W (see Equation 3.8), is referred to as weight space[15,16] and is frequently used to explain the behavior of deterministic classifiers. Weight space for two-dimensional feature space is shown in Figure 3.10, where the vector from the origin to the point (w_1,w_2,w_3) is called the weight vector. It should be noted that weight space is of dimension N+1 when feature space is of dimension N.

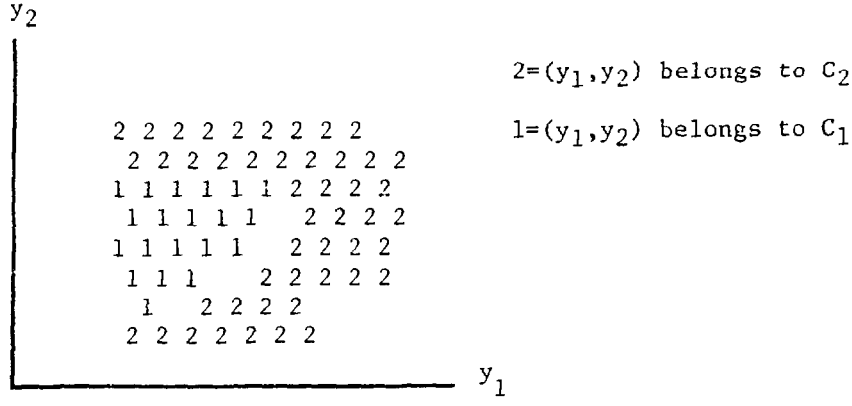
Consider the one-dimensional feature space shown in Figure 3.11 where

$$\text{if } -5 \leq y \leq 3 \quad y \text{ belongs to } C_1$$

$$\text{if } 10 \leq y \leq 15 \quad y \text{ belongs to } C_2 \ .$$



(a) Linearly Separable



(b) Non-Linearly Separable

Figure 3.9 Linearly and Non-Linearly Separable Classes

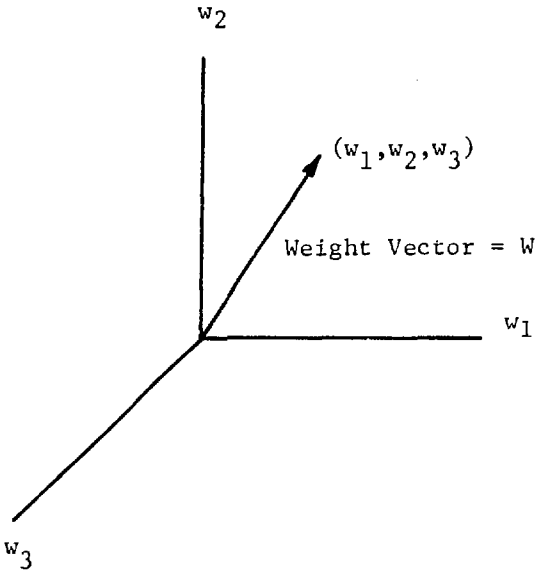


Figure 3.10 Weight Space

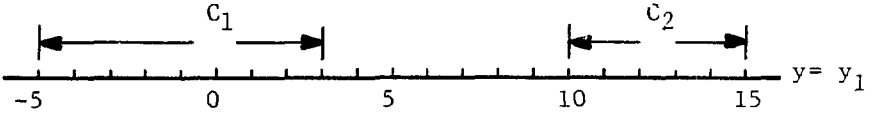


Figure 3.11 One Dimensional Feature Space

If a decision surface is located at point y', then for this example

$$g(Y) = 0 = w_1 y' + w_2$$

and

$$w_2 = -w_1 y' \quad . \tag{3.9}$$

The surface given by Equation 3.9 is called a pattern hyperplane[15,16] in weight space and divides weight space into two regions: (1) that region for which $g=(w_1y'+w_2)> 0$ and (2) that region for which $g=(w_1y'+w_2)< 0$. Pattern hyperplanes for y'=2,3, and 10 are shown in Figure 3.12 where the + and - signs indicate the sign of g on the different sides of the hyperplane. The shaded region of Figure 3.12 is that region of weight space for which

$$g > 0 \quad \text{if } y' \geq 10$$

and

$$g < 0 \quad \text{if } y' \leq 3$$

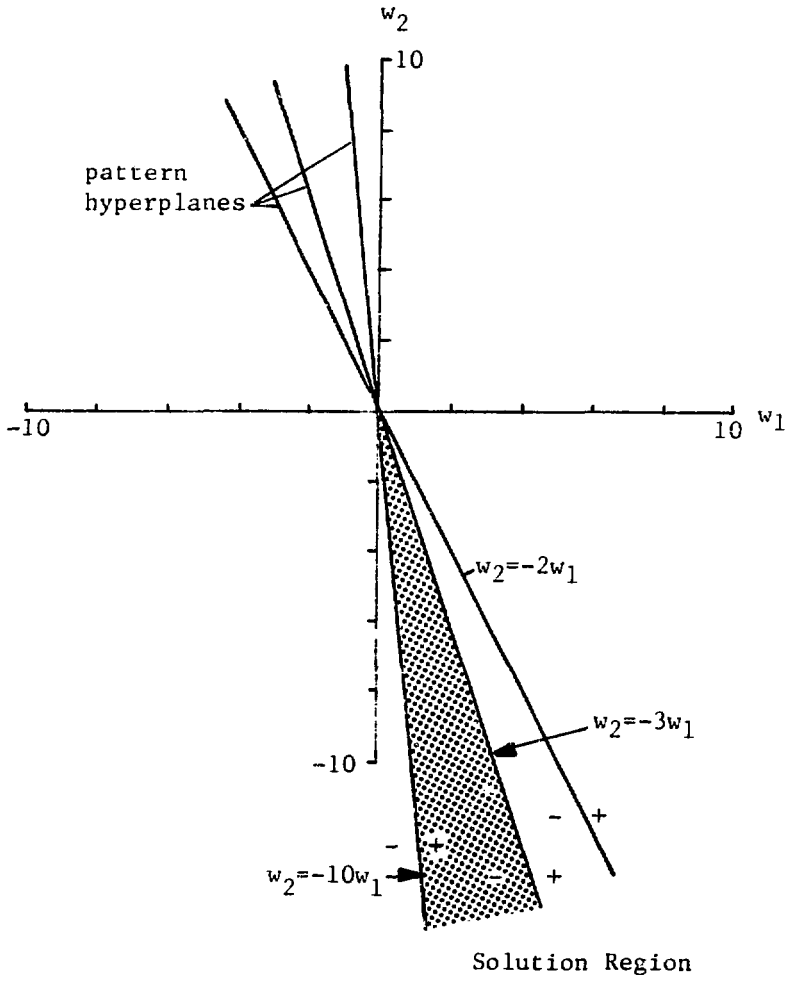


Figure 3.12 Pattern Hyperplanes in Weight Space

and is called the solution region. Any W in the solution region results in a discriminant function which satisfies Equation 3.3.

The training of a linear classifier consists of first guessing an initial weight vector, W_1. The classifier is then presented with prototypes, Y_i, of known classification. If $g(Y_i)$ gives the correct classification for Y_i, W is unchanged. If $g(Y_i)$ gives the incorrect classification then W is corrected as follows:[15]

$$\text{if } g(Y_i) > 0 \quad \text{and } Y_i \text{ belongs to } C_1 \qquad (3.10)$$

$$W_{i+1} = W_i - cY_i^*$$

$$\text{if } g(Y_i) < 0 \quad \text{and } Y_i \text{ belongs to } C_2$$

$$W_{i+1} = W_i + cY_i^*$$

$$\text{where } c>0$$

The effect of the above procedure may be varied depending on the value of c, the correction increment. The above scheme will always move W in a direction normal to the pattern hyperplane*. The size of c determines how far the W is moved. Three rules[15,16] commonly used to determine the value of c are:

*W is moved along the direction of the vector $(W_{i+1}-W_i)=\pm cY_i^*$. The equation for all hyperplanes perpendicular to the vector $(\pm cY_i^*)$ is $W\cdot(\pm cY_i^*)=r_1$ or $W\cdot Y_i^*=r_2$ where r_1 and r_2 can be any scalar values[35]. However, the equation of the hyperplane corresponding to the prototype Y_i is $W\cdot Y_i^*=0$. Therefore, using $r_2=0$, W is moved normal to the pattern hyperplane in weight space.

(1) Fixed Increment Rule: c is taken to be any fixed increment greater than zero. In this case the weight adjustment may or may not correct the misclassification of the prototype, depending on the value of $W \cdot Y^*$ in relation to c.

(2) Absolute Correction Rule: c is the smallest integer greater than $|W \cdot Y^*| / Y^* \cdot Y^*$. Thus after one adjustment with this rule W will be on the correct side of the pattern hyperplane.

(3) Fractional Correction Rule: c is chosen such that W is moved a fractional distance, λ, towards the pattern hyperplane. The distance from the weight vector W to the pattern hyperplane defined by Y is given by

$$D = \frac{|W \cdot Y^*|}{|Y^*|} = \frac{|g(Y)|}{|Y^*|} \tag{3.11}$$

Therefore, using

$$|W_{i+1} - W_i| = c|Y^*| = \lambda D$$

c is given by

$$c = \frac{\lambda D}{|Y^*|} = \frac{\lambda |g(Y)|}{|Y^* \cdot Y^*|} \tag{3.12}$$

If $\lambda > 1$, W_{i+1} will be on the correct side of the hyperplane. Throughout this study λ will be referred to as the learning parameter since it controls the rate of learning by the classifier. Figure 3.13 illustrates the behavior of these three rules using the problem illustrated in Figure 3.11 and the following prototypes (prototypes are presented to the classifier in the order presented below)

(1) y = -2 and belongs to C_1

(2) y = 10 and belongs to C_2

(3) y = 2 and belongs to C_1

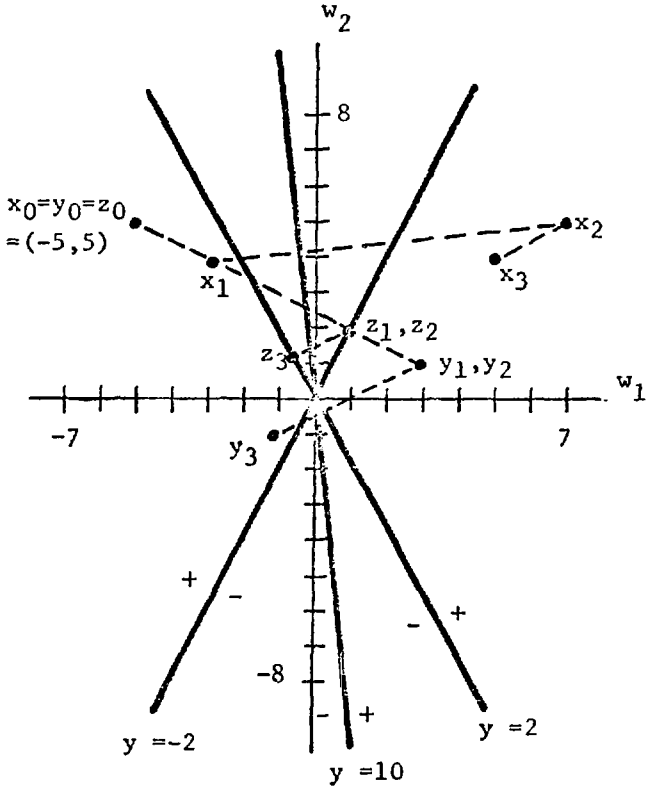


Figure 3.13 Example of the Fixed Increment, Absolute Correction, and Fractional Correction Rules

The x_i, y_i and z_i are the positions of W after the i'th prototype has been presented using the fixed increment (c=1), absolute correction, and fractional correction rules (λ=1) respectively.

3.2.2) Quadratic Discriminant Functions

The general form of a quadratic discriminant function for a two class problem with N dimensional feature vectors is given by

$$g(Y) = \sum_{n=1}^{N} w_{n,n} y_n^2 + \sum_{n=1}^{N-1} \sum_{k=n+1}^{N} w_{n,k} y_n y_k + \sum_{n=1}^{N} w_n y_n + w_{N+1}$$

A quadratic discriminant function has M=(N+1)(N+2)/2 weights. This type of g(Y) can be treated in exactly the same manner as the linear g(Y) if the feature vector is first operated on by a "quadric processor"[15] as shown in Figure 3.14. The quadric processor behaves as a feature selector except that the dimensionality of the data is increased from N to M instead of decreased. The same techniques described in Section 3.2.1 can be used to learn the W vector corresponding to g(F). This same procedure can also be performed for any g(Y) which depends linearly on the $w_1, w_2 \ldots w_M$ resulting in

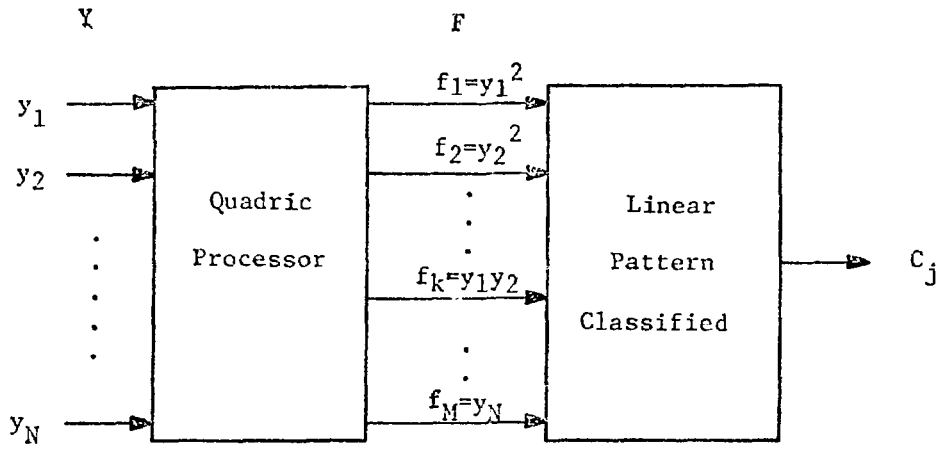


Figure 3.14 Quadratic Discriminant Functions

$$g(Y) = w_1 f_1(Y) + w_2 f_2(Y) + ... + w_M f_M(Y) + w_{M+1}$$

Such a g(Y) is frequently referred to as a ϕ function[15].

3.3) Sequential Statistical Classification Techniques (Case E)

The main incentive for using a statistical approach in pattern classification is that many processes can best be characterized in statistical terms. It is also often desirable to evaluate a pattern classifier in terms of its statistical performance. The statistical

classifier investigated in this research is similar to the deterministic classifier described in Section 3.2 in that a form of g(Y) is assumed (i.e. g(Y)=f(W,Y)). The following notation will be used when referring to the statistical nature of pattern classifiers:

$p(Y|C_i)$ = the probability density function of those vectors Y which belong to C_i

$P(C_i)$ = the probability of class C_i occurring ($P(C_1)+P(C_2)=1$)

$p(Y) = p(Y|C_1)P(C_1)+p(Y|C_2)P(C_2)$ = the probability density of Y

$p(C_i|Y) = p(Y|C_i)P(C_i)/p(Y)$ = the probability of the vector Y belonging to class C_i.

3.3.1) Linear Discriminant Functions

In Section 3.2.1 the feature vectors were assumed to be linearly separable. By use of statistical techniques, linear discriminant functions (i.e. $g(Y)=W\cdot Y^*$) can be used with non-linearly separable data in a least error sense.

Let the function $S(W,Y,C_i|C_k)$ be defined as the loss incurred[16] when a pattern or feature vector, Y, actually belonging to class C_k is placed in class C_i (note that S is a function of the weight vector W). A vector Y is said to belong to class C_k if

$$p(C_k|Y) > p(C_i|Y) \qquad \text{for } i \neq k.$$

This loss function provides a means of weighing specific classification errors more heavily than others. For example the distance from a misclassified prototype to the decision surface (see Figure 3.15) as given by

$$d(W,Y) = \frac{|g(Y)|}{|W'|} \tag{3.13}$$

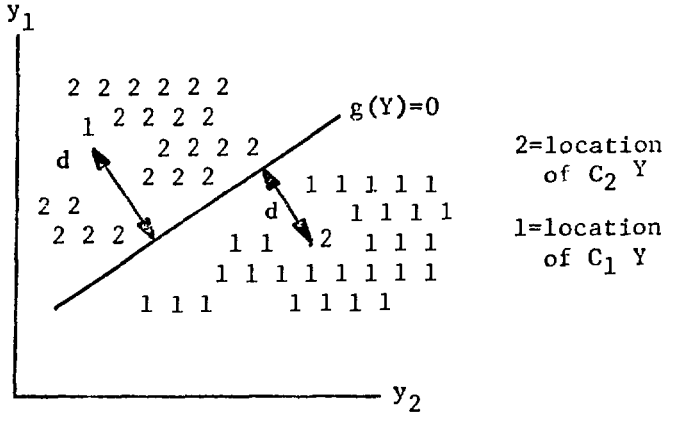


Figure 3.15 Misclassification Distance

where W' is the weight vector W with $w_{N+1}=0$, is frequently used as a loss function[28] which for two classes results in

$$S(W,Y,C_1|C_2) = S(d,C_1|C_2) = d(W,Y),$$
$$S(W,Y,C_1|C_1) = S(d,C_1|C_1) = 0,$$
$$S(W,Y,C_2|C_2) = S(d,C_2|C_2) = 0, \text{ and}$$
$$S(W,Y,C_2|C_1) = S(d,C_2|C_1) = d(W,Y). \qquad (3.14)$$

The variable d is frequently referred to as the "misclassification distance"[28] and should not be confused with the distance, D, as given by Equation 3.11. The average loss[16] $L(W,Y,C_i)$ as given by

$$L(W,Y,C_i) = S(W,Y,C_i|C_k)p(C_i|Y) \qquad \begin{matrix} i=1,2 \\ k=1,2 \\ i\neq k \end{matrix} \qquad (3.15)$$

can be interpreted as the average $S(W,Y,C_i|C_k)$ associated with vector Y and class C_i. If $L(W,Y,C_i)$ is integrated over all feature space, the result is the <u>risk</u>[16] associated with each class:

$$R(W,C_i) = \int L(W,Y,C_i)p(Y)dY \qquad \begin{matrix} i=1,2 \\ k=1,2 \\ i\neq k \end{matrix}$$

$$R(W,C_i) = \int S(W,Y,C_i|C_k)p(C_i|Y)p(Y)dY \qquad (3.16)$$

The total risk in the classification problem is the sum of the risks involved in each class:

$$R(W) = R(W,C_1) + R(W,C_2) \tag{3.17}$$

The purpose of the pattern classifier is to minimize the risk with respect to W. Assuming R(W) is differentiable and has a global minimum with respect to W, the optimum W is the solution of $\nabla R(W)=0$. However, as seen by

$$\begin{aligned} \nabla R(W) &= \nabla \int p(Y)L(W,Y,C_1)dY + \nabla \int p(Y)L(W,Y,C_2)dY \\ &= P(C_1) \int p(Y|C_1)\nabla S(W,Y,C_1|C_2)dY \\ &\quad + P(C_2) \int p(Y|C_2)\nabla S(W,Y,C_2|C_1)dY, \end{aligned} \tag{3.18}$$

this requires that $p(Y|C_i)$ be known. This problem can be alleviated if R(W) is approximated by a summation over prototypes

$$\tilde{R}(W) = \sum_{k=1}^{2} \frac{1}{M_k} \sum_{n=1}^{N_k} S(W,Y_n^k,C_i|C_k) \qquad \begin{matrix} i=1,2 \\ k=1,2 \\ k\neq i \end{matrix} \tag{3.19}$$

where M_k = no. of prototypes in class C_k

Y_n^k = nth misclassified prototype of class k

N_k = number of misclassified prototypes in C_k

rather than an integration over densities. Using this approximation for R(W), W can be incremented proportional to the negative of $\nabla\hat{R}(W)$ as given by

if $g(Y_i) > 0$ and Y_i belongs to C_1 (3.20)

or

if $g(Y_i) < 0$ and Y_i belongs to C_2

then

$$W_{i+1} = W_i - \lambda\nabla\hat{R}(W)$$

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial w_1} \\ \frac{\partial}{\partial w_2} \\ \vdots \\ \frac{\partial}{\partial w_{N+1}} \end{bmatrix}$$

where λ is a proportionality constant or learning parameter similar to the λ described in Section 3.2.1 and

$$\nabla\tilde{R}(W) = \sum_{k=1}^{2} \frac{1}{M_k} \sum_{n=1}^{N_k} \nabla S(W, Y_n^k, C_i | C_k) \qquad \begin{matrix} i=1,2 \\ k=1,2 \\ k\neq i \end{matrix} \qquad (3.21)$$

3.3.2) Quadratic Discriminant Functions

Quadratic discriminant functions using statistical techniques are treated the same as linear discriminant functions except that the feature vector is first processed by the quadric processor described in Section 3.2.2.

3.4) Feature Selection

The objective of feature selection is to retain that information necessary for classification and to eliminate that information which is not. Feature selection often results in greatly reducing the demands on the classifier. For example, a feature selector may process non-linearly separable data into linearly separable data as shown in Figure 3.16.

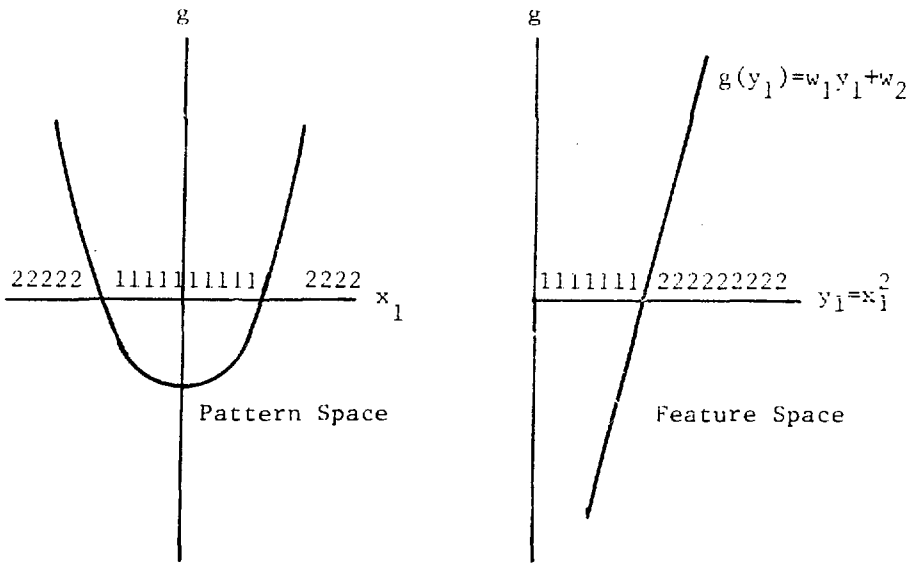


Figure 3.16 Linearization by Feature Selection

Unfortunately, the operation of feature selection is far less defined mathematically than that of pattern classification. Although a human can implement feature selection with ease, the techniques used are heuristic in nature and usually highly problem dependent. At the present time, selection decisions trivial to a human may take a great deal of effort to model and even then may take a large amount of computer time to implement. Thus this research will rely upon the heuristic techniques of the user to supply the feature selection process.

The pattern space of a particle transport problem consists at most of eight basic parameters:

- three position parameters (x,y,z)
- three angular parameters** (u,v,w)
- energy (E)
- time (t)

Certainly, if a problem independent of time is under investigation, it is much easier for the human user to remove t from feature space than it is for a computer based selection system to recognize that there is no correlation between time and classification. Another case is a problem involving spherical symmetry in the geometry in which three variables (x,y,z) can be replaced by one, r($r= x^2+y^2+z^2$). This not only reduces the dimensionality of the problem but can also linearize the feature space. Such a substitution is easily specified by the user but would take numerous operations to recognize computationally.

3.5) Multiclass Problems

The previous sections have considered pattern recognition problems involving two classes. In this research discriminant functions are learned for two classes at a time* resulting in a single discriminant function for classes i and i+1 (see Equation 3.2)

---

* The reason for this will become apparent in Chapter V.

** Only two of the angular parameters are independent.

$$g_{i,i+1}(Y) = g_{i+1}(Y) - g_i(Y) \qquad i=1,\ldots,J-1 \tag{3.22}$$

where J is the total number of classes. These classes are ordered such that

if $g_{i,i+1}(Y) < 0$,

then Y must belong to a class C_j where $j=1,\ldots,i$

if $g_{i,i+1}(Y) > 0$

then Y must belong to a class C_j where $j=i+1,\ldots,J$

Because of this characteristic, the class of a prototype can be determined as shown by the flow diagram in Figure 3.17.

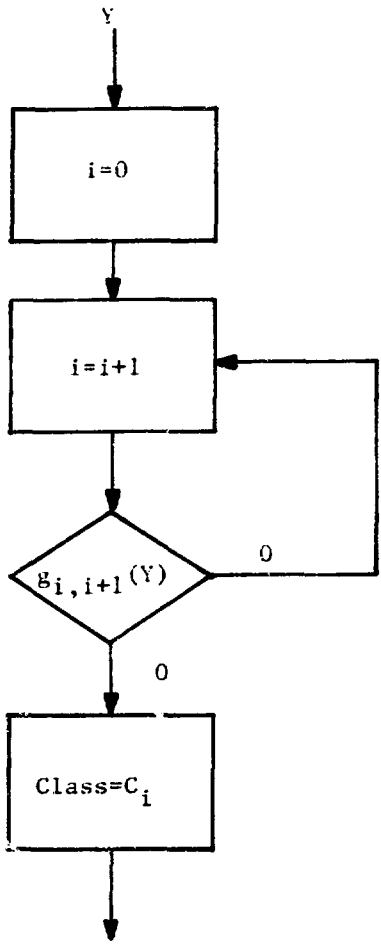


Figure 3.17 Multi-Class Problems

IV. Recognition of Splitting Surfaces

In Section 2.4 splitting in state space was described and it was suggested that pattern recognition be used for the identification of the splitting surfaces. It is the purpose of this chapter to describe how pattern recognition can be implemented and to investigate the performance of the technique.

Since the purpose of this dissertation is to demonstrate proof of principle, the technique is not applied to a general purpose Monte Carlo code. The Monte Carlo problems investigated in this research have been chosen for their simplicity and their minimal use of computer time. These sample problems include: (1) a one-dimensional one-region homogeneous slab, (2) a one-dimensional multi-region slab, and (3) a two-dimensional, multi-region slab. These problems illustrate the basic treatment of distance and direction variables in a Monte Carlo problem.

This chapter is concerned only with the learning of the splitting surface. A description of how to use the splitting surface and what surfaces are desirable as splitting surfaces is given in Chapter V.

Section 4.1 describes in general how pattern recognition is used to identify splitting surfaces. In Section 4.2 both deterministic (see Section 3.2) and statistical (see Section 3.3) classifiers are used to identify splitting surfaces for a one-region slab Monte Carlo problem and studies are made to determine: (1) the effects of slab thickness and class overlapping, (2) the improvement due to the use of

buffer zones,(3) the best choice of a loss function (see Section 3.3.1) for the statistical classifier, (4) computer time spent for pattern recognition, and (5) the sensitivity of the classifiers to the learning parameter λ. In Section 4.3 the same classifiers developed in Section 4.2 are used to identify surfaces for multi-region problems and again the sensitivity to λ is investigated. Section 4.3 also includes a study of the sensitivity of the classifiers to the selection of initial conditions (i.e., the initial guess for W). Section 4.4 increases the pattern space to two dimensions, distance and angle, thus requiring normalization of the feature vector. Studies of the learning parameter and initial conditions are then repeated for the two-dimensional problem. Section 4.5 summarizes the results of the chapter.

4.1) Basic Principles

The purpose of this section is to relate the pattern recognition system described in Chapter III to the problem of identifying Monte Carlo splitting surfaces as described in Chapter II.

In Section 3.1.2, the concept of "prototypes" or "training sets" was introduced. This concept is very important in this research and is discussed with respect to Monte Carlo calculations in Section 4.1.1.

Although feature selection is not discussed until Chapter V, Section 4.1.2 does describe how feature selection relates to the simple Monte Carlo problems of this chapter.

Two pattern classification systems are investigated in this chapter: (1) the Case D deterministic classifier using the fractional correction rule (see Section 3.2) and (2) the Case E statistical classifier (see Section 3.3). The structure of these classifiers and the operations performed for learning a splitting surface are described in Section 4.1.3.

In Chapter III, the term "learning parameter" was introduced for both deterministic and statistical classifiers. This parameter, λ, plays an important role in this research. Its importance is explained in Section 4.1.4.

In order to evaluate different classifiers and classifier parameters, one must be able to measure the performance of the classifier. The performance measures used in this dissertation are described in Section 4.1.5.

4.1.1) Prototypes from Monte Carlo Calculations

Prototypes were described in Section 3.1.2 as being pattern vectors with known classification. The prototypes allow Case B, D, and E classifiers to learn a discriminant function, g(Y), (see Section 3.1.1) which is necessary before classification can take place.

In Section 3.14 it was stated that for the general Monte Carlo problem pattern space consists of 8 variables:

- three position parameters (x,y,z)
- three direction parameters (u,v,w)

- energy (E)

- time (t)

As a particle travels through a material region, it undergoes numerous collisions. At each collision point a new set of (x,y,z,u,v,w,E,t) is calculated for the particle (see Appendix A). This new set of values consists of a point in state space (see Section 2.4) and can be represented by the state space vector X as given by

$$X = \begin{bmatrix} x \\ y \\ z \\ u \\ v \\ w \\ E \\ t \end{bmatrix}$$

This vector X is also a pattern vector; therefore,

state space vector $\equiv$ pattern vector

and

state space $\equiv$ pattern space.

Thus pattern vectors are created in a Monte Carlo problem whenever a particle undergoes a collision.

Before the pattern vectors can be used as prototypes, their classification must be known. In Section 3.5 it was stated that discriminant functions are learned for two classes at a time. Therefore, before the pattern vector, X, can be used as a prototype, it must be known to which class, $C_i (i = 1,2)$, the vector belongs. This is done by introducing the concept of "importance".

In this research the importance of the vector X in state space is defined as the average contribution to the tally by particles which pass through X divided by the average weight (see Section 2.2) that particles have at X. If the importance were known for all X, there would be little reason to solve the Monte Carlo problem since the average importance of the source particles would be equivalent to the desired tally. Therefore, only approximations to the importance as defined above will be used. The approximation to the importance at X of a single particle passing through X is given by

$$I(X) = \frac{T}{Wt(X)} , \tag{4.1}$$

where T = the contribution of the particle to the tally

Wt(X) = the weight of the particle when it existed at X.

Using Equation 4.1 for the importance, one can classify the pattern vector X as follows:

$$\text{if } I(X) < \bar{I}, \text{ X belongs to } C_1 \tag{4.2}$$

$$\text{if } I(X) > \bar{I}, \text{ X belongs to } C_2 ,$$

where $\bar{I}$ is an importance which is used to discriminate between C_1 and C_2 ($\bar{I}$ will be discussed later).

Thus prototypes, X, are created at collision points with their classification determined by Equation 4.2. The following example illustrates the creation of prototypes.

Example: Consider the problem of the one-dimensional homogeneous slab shown in Figure 4.1 with a unidirectional source at x=0 and a tally of particles as they cross the surface at x=L. A single particle is shown traversing the slab and undergoing five collisions before it is tallied. The absorption probability at each collision is .2; thus, the weight of the particle is multiplied by .8 at each collision. When the particle is tallied approximations of the importances at the various collision points can be found by

$$I(x) = \frac{Wt(L)}{Wt(x)} ,$$

where $I(x)$ = importance of a particle at x

$Wt(L)$ = tally contribution of the particle= weight of the particle at x=L, and

$Wt(x)$ = weight of the particle at x.

The values of I(x) for the five collisions shown in Figure 4.1 are given in Table 4.1. If $\bar{I}$=.75, the pattern vectors are classified as shown in Table 4.1.

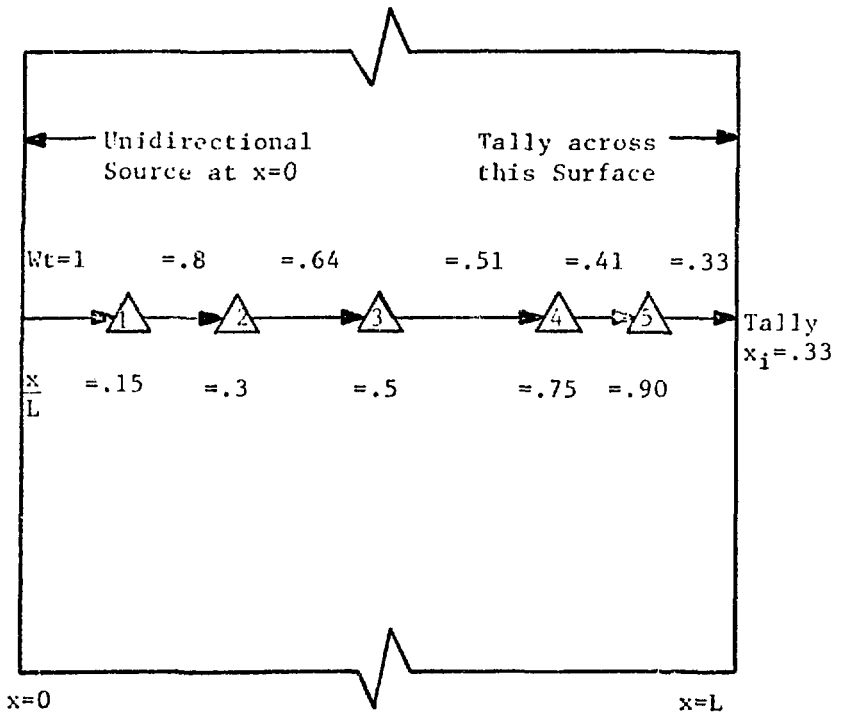


Figure 4.1 Prototypes from a One-dimensional Slab Monte Carlo Problem

4.1.2) Feature Selection

Sections 4.2 and 4.3 use one dimensional slab Monte Carlo problem and thus pattern space consists of the single variable x, where x is the distance from the source (see Figure 4.1). For this problem feature space will be the same as pattern space; thus

$$X = [x_1] = Y = [y_1] = x.$$

The two dimensional problem of Section 4.4 will be treated similarly. In this case

$$X = \begin{bmatrix} x_1 \\ \\ x_2 \end{bmatrix} = Y = \begin{bmatrix} y_1 \\ \\ y_2 \end{bmatrix} = \begin{bmatrix} x \\ \\ \phi \end{bmatrix}$$

Table 4.1 Prototypes for Problem Illustrated in Figure 4.1 and $\bar{I} = .75$

$\frac{x}{L}$	Wt(x)	I(x)	C_i
.15	.8	.41	1
.30	.64	.51	1
.50	.51	.64	1
.75	.41	.8	2
.90	.33	1.0	2

where ϕ will be defined in Section 4.4. Some experiments are also made in Section 4.4. for which $X \neq Y$.

4.1.3) The Pattern Classifier

The classifiers used in this research assume a linear form for g(Y) as described in Sections 3.2.1 and 3.3.1 and operate in two stages:

(1) prototypes are used to learn the weight vector W for the linear discriminant function $g(Y) = W \cdot Y^*$ (see Equation 3.8)

(2) the discriminant function g(Y) is used to classify the feature vector Y where Y is of unknown classification.

This chapter is concerned only with the first operation. The second operation is discussed in Chapter V.

The learning of the weight vector, W, consists of:

(1) selecting an initial value of W and

(2) incrementing W (using Equations 3.10 and 3.20) whenever a prototype belonging to class C_i (as determined by Equation 4.2) is classified into C_j $(j \neq i)$ as determined by the sign of g(W).

The sensitivity of the classifier to the initial selection of W is investigated in Section 4.3.1.

The incrementing of W is the major operation of the learning process. The classification of a feature vector according to Equation 4.2 is referred to as the "teacher". The classification according to

the pattern classifier is determined by Equation 3.3 which is repeated here as

if $g(Y) < 0$ Y belongs to C_1

if $g(Y) > 0$ Y belongs to C_2

The classification according to Equation 3.3 is referred to as the "student". Thus whenever the student disagrees with the teacher, the student is corrected by adjusting W. This process continues until the agreement between student and teacher meets some threshold value. At this point the classifier has learned the desired W.

The intersection of the discriminant function g(Y) with state space is called the decision surface (see Section 3.1.1) and is given by $g(Y) = 0$. This surface separates state space into two regions: (1) X for which $g(Y) > 0$ and (2) X for which $g(Y) < 0$. Since this is the purpose of a splitting surface, it follows that

decision surface $\equiv$ splitting surface.

These two terms will be used interchangeably throughout the remainder of this dissertation.

4.1.4) The Learning Parameter, λ

The weight adjustment algorithm using the deterministic classifier and the fractional correction rule is given by (see Equations 3.10 and 3.12)

$$W_{i+1} = W_i \pm \frac{\lambda |g(Y)|}{Y^* \cdot Y^*} Y^* .$$

For the statistical classifier the adjustment algorithm is given by (see Equation 3.20)

$$W_{i+1} = W_i - \lambda \nabla \tilde{R}(W).$$

In both cases the amount of the adjustment of W_i is determined by the learning parameter, λ.

When a pattern classifier is presented with overlapping distributions* (see Section 3.1.2), the selection of an optimum λ becomes quite complex. Before using the adjustment algorithms, the classifier must be told to which class a feature vector belongs.

*All data produced by Monte Carlo will be overlapping since the prototype classification is determined by Equation 4.2 and I(X) is only a single estimate of the true value of the importance at X. Overlapping will be discussed further in Section 4.2.

When Y can belong to either class with probability $p(C_1|Y)$ and $p(C_2|Y)$ (see Section 3.1.2), the classifier should be told

$$\text{if } p(C_1|Y) > p(C_2|Y) \qquad Y \text{ belongs to } C_1$$
$$\text{if } p(C_2|Y) > p(C_1|Y) \qquad Y \text{ belongs to } C_2$$

In this research the $p(C_i|Y)$ are unknown. As a result, when the classifier is told that Y belongs to C_2, it may be that

$$p(C_1|Y) > p(C_2|Y) \ .$$

In such a case, the classifier should not adjust the weights if $g(Y)<0$, since it is the prototype classification that is wrong, not the classifier. However, since the $p(C_i|Y)$ are not known, it is impossible to determine which classification is right.

As the confidence in a prototype's classification becomes small, i.e., as

$$p(C_1|Y) \to .5 \text{ and } p(C_2|Y) \to .5$$

a smaller value of λ should be used than when the confidence becomes great, i.e., as

$$p(C_i|Y) \rightarrow 1 \qquad i=1 \text{ or } 2$$

Thus, it would be beneficial to use a λ that is a function of

$$\Delta(Y) = 1 - |p(C_1|Y) - p(C_2|Y)|$$

which is not possible since $\Delta(Y)$ is unknown. If the average value of $\Delta(Y)$ is known for a problem as given by

$$\overline{\Delta} = \frac{\int \Delta(Y) dY}{\int d\,Y}$$

a "semi-optimal" constant λ can be chosen for each problem such that

$$\lambda \propto \overline{\Delta}$$

For Monte Carlo problems, the value of $\bar{\Delta}$ is not known. Therefore a single λ must be used for all problems which from the above discussion is definitely sub-optimal. One of the purposes of this dissertation is to determine the sensitivity of classifier performance to λ so that a suitable λ can be chosen for a large range of problems.

4.1.5) Classifier Performance

Two parameters are used in this dissertation to measure the performance of the various classifiers: misclassification rate and variability. In addition the classifiers are timed in order to estimate how much computer time is spent in learning a splitting surface.

A prototype is said to be misclassified if the student disagrees with the teacher (see Section 4.1.3). This can be summarized as follows:

$$\text{if } I(Y) < \bar{I} \text{ and } g(Y) > 0$$

or

$$\text{if } I(Y) > \bar{I} \text{ and } g(Y) < 0 \text{ ,}$$

then the prototype Y has been misclassified. The misclassification rate used in this dissertation is given by

$$R_{MC} = \frac{N_1 + N_2}{M_1 + M_2},$$

where N_i = number of prototypes belonging to C_i (according to the teacher) that are misclassified into C_j $(i \neq j)$ and

M_i = number of prototypes belonging to C_i (according to the teacher).

It is important to note that the above misclassification rate is that seen by the teacher. Because of this the misclassification rate can never be lower than the misclassification rate of the teacher. Therefore, a problem in which the class distributions, $p(C_i|Y)$, overlap by 30% will never have a misclassification rate below .3. Similarly a problem in which the $p(C_i|Y)$ have no overlap can theoretically have a misclassification rate of zero. Because of the Monte Carlo problem dependence of the misclassification rate another parameter, the variability, is used to measure performance.

The variability is a measure of the amount of fluctuation of the decision surface. The decision surface after the j'th prototype for the one dimensional problem is given by

$$S_{j+1} = \left(\frac{-w_2}{w_1}\right)_{j+1}$$

where S_1 = the initial selection of the decision surface

$(w_i)_{j+1}$ = the i'th component of the weight vector W_{j+1} that exists after the j'th prototype

W_1 = the initial selection of W

The mean value of the decision surface after J prototypes is given by

$$\bar{S} = \frac{\sum_{j=1}^{J+1} \left(\frac{-w_2}{w_1}\right)_j}{J+1}$$

The variability of the decision surface as used in this dissertation is given by

$$\text{Var} = \left[\frac{\sum_{j=1}^{J+1} \left(\frac{-w_2}{w_1}\right)_j^2}{J+1} - \bar{S}^2}{J}\right]^{1/2} \times \frac{1}{\bar{S}}$$

The variability can be thought of as the relative error of the decision surface. As the decision surface converges the variability decreases. The variability of the statistical classifier approaches zero; however, as will be seen, this is not true for the deterministic classifier.

The misclassification rate and variability can also be used by the classifier to determine when a decision surface has been learned. This can be done by first setting a threshold on the misclassification rate and variability. Once the misclassification rate threshold has been reached, the classifier continues until the variability threshold is reached. The values to use as thresholds will require experience with a general purpose Monte Carlo code; however, an indication of those values is given in this research.

Because the purpose of using pattern recognition is to save computer time, it is important to know how much computer time is used by the classifier to learn a splitting surface. This is done in this research by using the timing routine described in Reference 36. It should be noted that the FORTRAN programming used in this research is in no way optimized. Therefore, the timing data should be considered as an upper limit to the values that could be obtained using optimized FORTRAN or assembly language programming. It is also important to realize that the relative difference between timing values for different operations, quoted in this research could also change depending on the programming used. Thus when it is stated that operation x is quicker than operation y, it should be remembered that this is relative to the programming techniques used in this research.

4.2 One-Dimensional One-Region Slab

The Monte Carlo problem used in this section consists of calculating the transmission probability through a homogeneous slab of thickness L (see Figure 4.2). At each collision the particle's weight is reduced by the absorption probability and then allowed to continue in the forward direction only. The computer code for this problem is shown in Appendix C. The analytical solutions for the transmission probability to x and the importance at x are given by

$$TP = e^{-\Sigma_a x} \quad \text{and} \tag{4.2a}$$

$$I = e^{-\Sigma_a (L - x)} . \tag{4.2b}$$

Figure 4.3 shows the distribution of importances obtained from Monte Carlo runs of 200 particles for various thickness slabs and macroscopic cross sections of $\Sigma_t = .5$ and $\Sigma_s = .4$. The discrete behavior of this distribution can be attributed to the fact that only an integer number of collisions can occur resulting in importances given by

$$I(x) = (.8)^n , \tag{4.3}$$

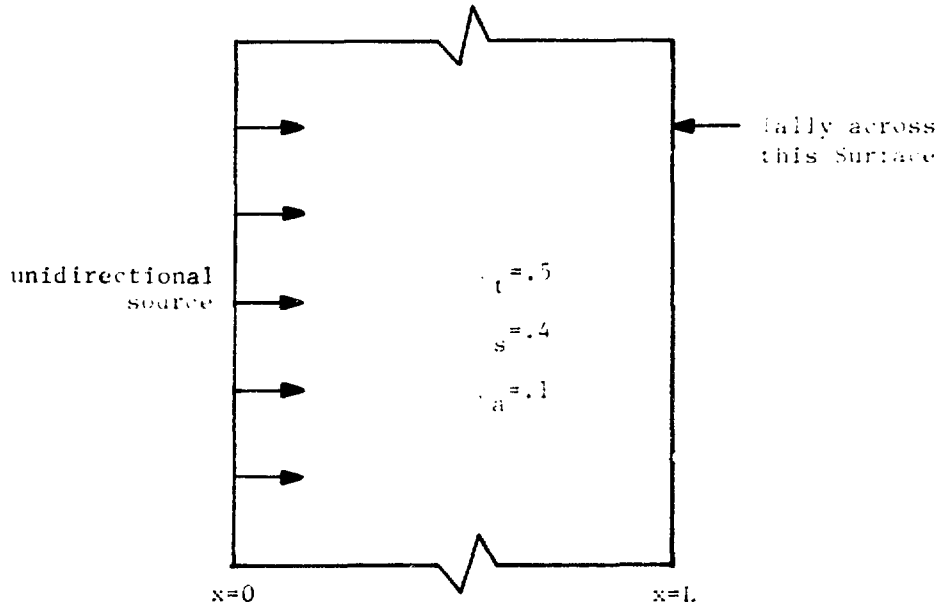


Figure 4.2 The One-Dimensional, One-Region, Homogeneous Slab

where n is the number of collisions a particle has undergone between the time it leaves x and is tallied. Because these importance distributions are unknown prior to the beginning of a Monte Carlo calculation, the mean or median must be learned during the initial stages of the run. This value can then be used as the $\bar{I}$ mentioned in Section 4.1.

Before applying pattern recognition it is helpful to observe the probability densities, $p(C_i|Y)$, of the two classes separated by $\bar{I}$. These densities are shown in Figure 4.4 for several thickness slabs using the mean importance for $\bar{I}$ and cross sections of $\Sigma_t = .5$ and

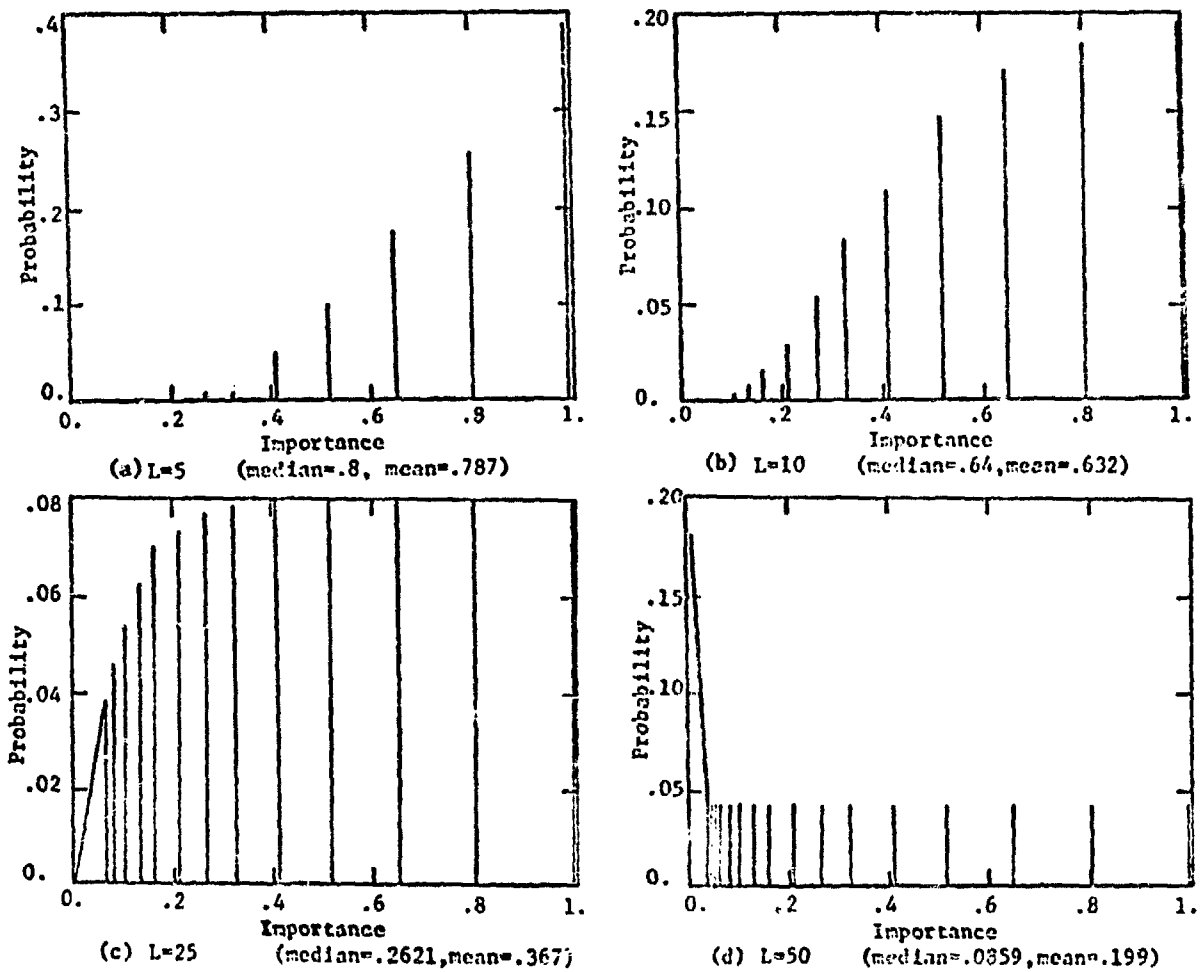


Figure 4.3 Importance Distributions for Several Slab Thicknesses

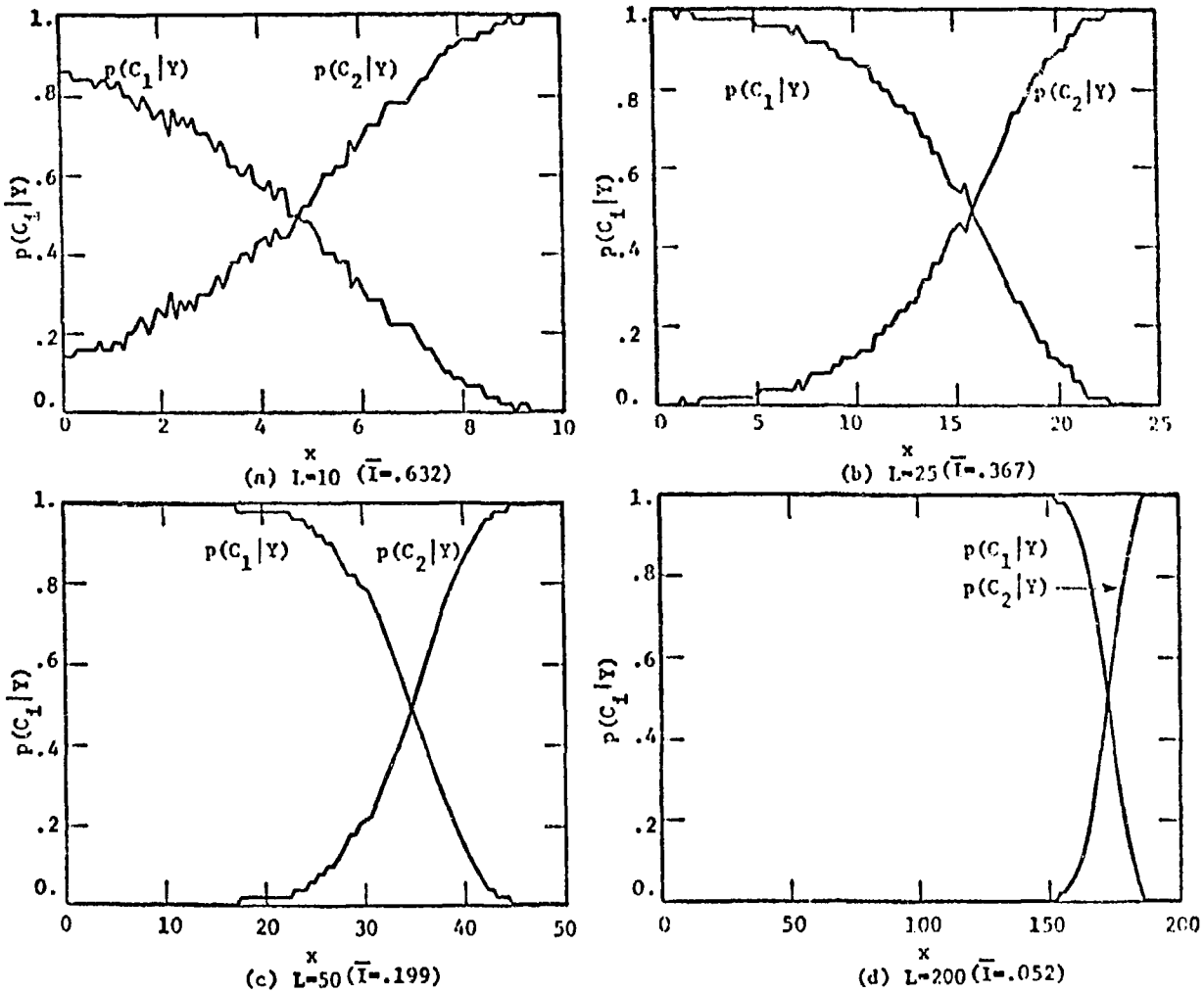


Figure 4.4 Class Distributions for Several Slab Thicknesses

$\Sigma_s = .4$. For this problem p(Y) is constant (except at x = 0) since the particles are never killed but allowed to continue through the slab until they are tallied. Therefore, since the $P(C_i)$ are also constant, the $p(Y|C_i)$ will have the same shape as the $p(C_i|Y)$ only different magnitudes relative to each other.

The overlapping of the class distributions is due to the fact that prototypes created at x do not all have the same importance I(x), but instead have a distribution of importances (with a mean importance of $\bar{I}(x)$ as is shown in Figure 4.5 for a slab with L = 10 and values of x = 0, 5, and 9). If all the prototypes created at x had the same importance, $\bar{I}(x)$, then these prototypes would all belong to the same class depending on whether $\bar{I}(x)$ is greater or less than $\bar{I}$. However, since prototypes are distributed about $\bar{I}(x)$, prototypes originating from the same x can belong to both classes depending on the location of $\bar{I}$ within the distribution. This point is illustrated by Figure 4.5 where the classes are separated by $\bar{I} = .632$ (shaded regions = C_1, unshaded = C_2). As $\bar{I}(x)$ approaches $\bar{I}$, the split of the distribution of prototypes at x into two classes becomes more pronounced (this is illustrated by Figure 4.5 where x = 5).

The remainder of this section investigates the behavior of both deterministic and statistical pattern classification procedures when applied to this problem. In all cases the initial decision surface will be chosen at L/2 ($w_1 = 1.0$, $w_2 = -L/2$).

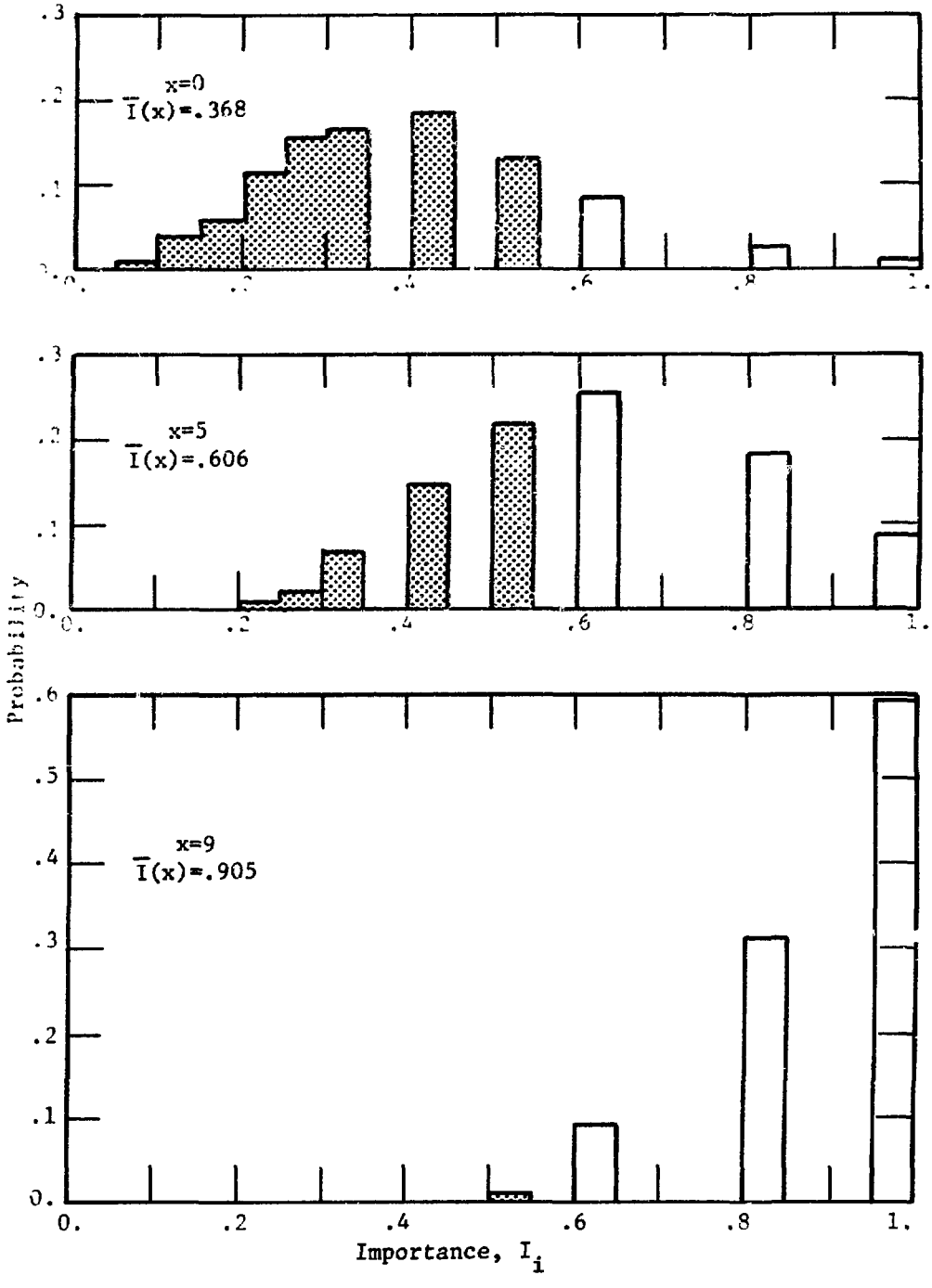


Figure 4.5 Distribution of Importances at x= 0,5, and 9 for L=10 and $\bar{I}$=.632
(600 prototypes are sampled for each x)

4.2.1) Deterministic Techniques

Figure 4.4 illustrates the fact that this problem does not produce linearly separable data since the $p(C_i|Y)$ are overlapping. Deterministic techniques will never converge to a single g(Y) since there exists no W which will correctly classify all Y's. However, if one is willing to use a g(Y) which satisfies a large percentage of the Y's, then deterministic techniques can be used to find an appropriate g(Y).

In the case of linearly inseparable data Equation 3.6 cannot be used directly but must be replaced by the following

if $g(Y_i) > 0$ and Y_i belongs to class C_1,

$$W' = W - c\left(\frac{N_2}{N_1}\right)Y^*_i \quad \text{or} \tag{4.4}$$

if $g(Y_i) < 0$ and Y_i belongs to class C_2,

$$W' = W + c\,Y^*_i ,$$

where N_i = number of prototypes in class i.

Equation 4.4 prevents the class with the most prototypes from influencing the selection of W simply because it has more prototypes and not because it has a higher percentage of misclassified prototypes.

In this study the fractional correction rule will be used since it allows for a more controlled convergence when used with overlapping distributions (i.e., the correction increment can be controlled through λ). Therefore, as given in Section 3.2.1, $c = \lambda|g|/Y^* \cdot Y^*$. The FORTRAN coding necessary to use the fractional correction rule with this problem is shown in Appendix D.

The results after 100 source particles for several slabs of varying thickness and four values of λ are summarized in Table 4.2. The behavior of the decision surface as a function of the number of source particles and a plot of weight space is shown in Figure 4.6 for a λ of .5. As is seen from the data of Table 4.2 the value of λ has only a small effect on the misclassification rate; however, it can greatly decrease the variability of the decision surface. This is useful since it is desirable to stop calculating the discriminant function once a suitable W has been found. Thus the variability can be used as an indicator of when to use the present discriminant function and stop adjustment of the weights.

The misclassification of prototypes after convergence is primarily due to the overlapping of the distributions shown in Figure

L Slab Thickness	Learning Parameter	Misclassification Rate C_1	Misclassification Rate C_2	Misclassification Rate C_1+C_2	Mean Decision Surface	Variability
10	$\lambda = .05$	.357	.181	.259	3.57	.128
	$\lambda = .2$	.397	.174	.273	3.64	.235
	$\lambda = .5$	.371	.177	.263	4.02	.364
	$\lambda = 1.0$	.348	.181	.255	4.63	.446
50	$\lambda = .05$	.116	.079	.104	32.01	.086
	$\lambda = .2$	.103	.077	.094	32.76	.105
	$\lambda = .5$	.115	.051	.094	32.92	.148
	$\lambda = 1.0$	.140	.052	.112	32.33	.194
200	$\lambda = .05$	.043	.034	.041	167.4	.048
	$\lambda = .2$	.040	.021	.037	169.1	.050
	$\lambda = .5$	.057	.026	.052	166.7	.082
	$\lambda = 1.0$	.089	.026	.080	162.7	.125

Table 4.2 Results of λ and L Variations After 100 Source Particles Using the Deterministic Classifier

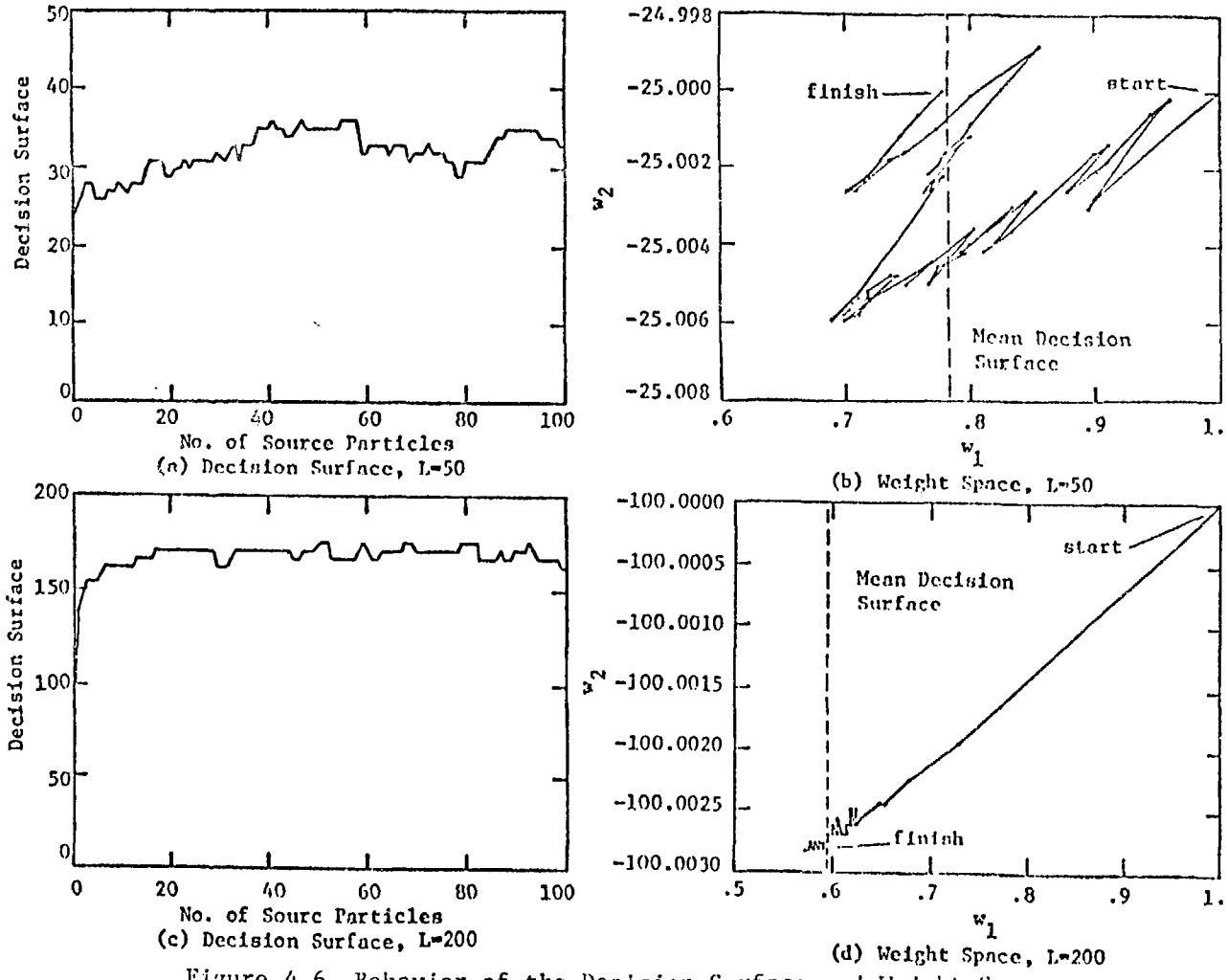


Figure 4.6 Behavior of the Decision Surface and Weight Space using the Deterministic Classifier for 100 Source Particles

4.4. This overlapping is measured in terms of the error rate as given by

$$\text{Error Rate} = E_r = \frac{100}{L}\left[\int_0^{\bar{x}} p(C_2|Y)dx + \int_{\bar{x}}^{L} p(C_1|Y)\right] \qquad (4.5)$$

where E_r = % misclassification of prototypes due to overlapping of $P(C_i|Y)$

L = slab thickness

$\bar{x}$ = the value of x for which $p(C_1|Y) = p(C_2|Y)$

Since the majority of misclassified prototypes come from an importance close to $\bar{I}$, much of the error can be eliminated by the introduction of a "buffer zone".[16] A buffer zone consists of a band of importances from I_1 to I_2 ($I_1 < \bar{I} < I_2$). Any track which has an importance I such that $I_1 < I < I_2$, is not placed in either class and W is left unchanged. Results of using various buffer zones on the $P(C_i|Y)$ distributions are shown in Figure 4.7. The reduction in the overlapping area is given in Table 4.3 along with results obtained by using various buffer zones. As can be seen from this data, the buffer zones have the effect of lowering the misclassification rate considerably. The variability is lowered in some cases, but raised in others. This is due to the fact that the majority of variability is caused by prototypes outside the buffer

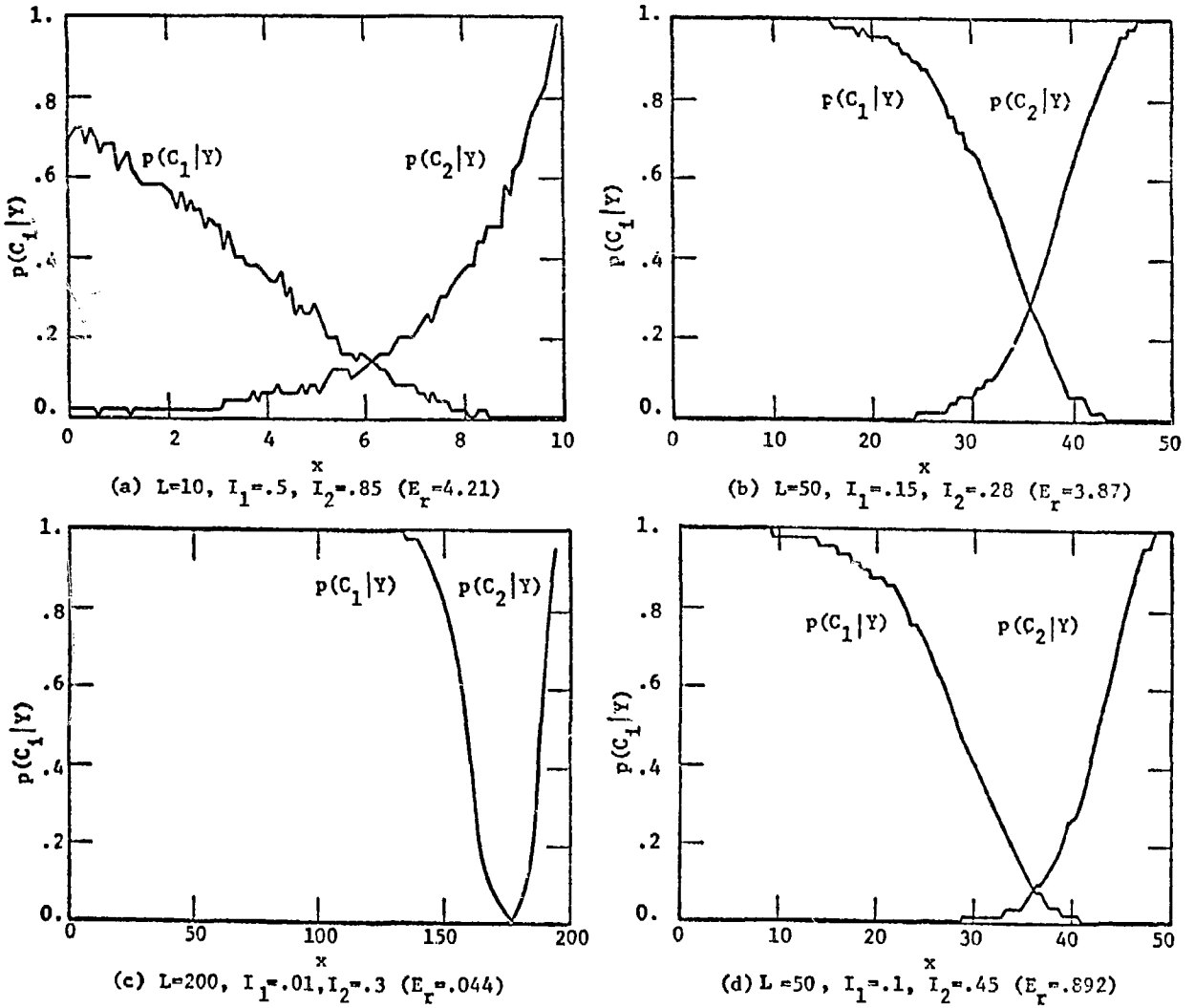


Figure 4.7 Effect of Buffer Zones on Class Distributions for Several Slab Thicknesses

Buffer Zone	E_r	Misclassification Rate	Mean Decision Surface	Variability
L = 10				
none	23.0	.259	3.572	.128
.6-.75	14.1	.240	3.929	.089
.5-.85	4.2	.153	4.532	.056
L = 50				
none	8.85	.104	32.01	.086
.18-.22	6.72	.086	32.97	.085
.15-.28	3.87	.064	33.04	.089
.10-.45	.89	.045	31.80	.101
L = 200				
none	2.96	.041	167.4	.048
.025-.12	.64	.024	167.0	.054
.010-.30	.04	.024	161.3	.059

Table 4.3 Effect of Buffer Zones on the Performance of the Deterministic Classifier after 100 Source Particles

zone. These misclassifications thus have more effect since there are fewer prototypes* in the buffer zone runs.

In conclusion, deterministic techniques work satisfactorily if λ is kept small ($\lambda = .05$ is sufficient) and buffer zones are used. One interesting characteristic is that the larger L, the better the classifier performs. This is quite favorable since it is for large L that one needs variance reduction the most.

4.2.2) Statistical Techniques

In Section 3.3 the statistical approach to pattern recognition was described in which weight adjustments were made based on the average behavior of the prototypes. In this section Equations 3.20 and 3.21 are used to adjust the weights for several different loss functions, $S(W,Y,C_i|C_k)$. Table 4.4 lists the loss functions which are investigated and their corresponding ∇S components. A description of these loss functions and the derivation of the ∇S components are given in Appendix E. From Equation 3.20 the weight vector, W, is incremented according to

*Feature vectors falling within the buffer zone are not counted as prototypes since their classification is not determined.

$S(W,Y,C_1\|C_j)$	$\frac{\partial S(W,Y,C_2\|C_1)^*}{\partial w_1}$	$\frac{\partial S(W,Y,C_2\|C_1)^*}{\partial w_2}$
$d = \frac{\|g\|}{w_1}$	$-\frac{w_2}{w_1^2}$	$\frac{1}{w_1}$
$d = \sqrt{\frac{\|g\|}{w_1}}$	$-\frac{w_2}{w_1^2}\left(\frac{1}{2}\sqrt{\left\|\frac{w_1}{g}\right\|}\right)$	$\frac{1}{w_1}\left(\frac{1}{2}\sqrt{\left\|\frac{w_1}{g}\right\|}\right)$
$d^2 = \frac{g^2}{w_1^2}$	$-\frac{w_2}{w_1^2}\left(\frac{2\|g\|}{w_1}\right)$	$\frac{1}{w_1}\left(\frac{2\|g\|}{w_1}\right)$
$D = \frac{\|g\|}{\sqrt{1+y^2}}$	$\frac{y}{\sqrt{1+y^2}}$	$\frac{1}{\sqrt{1+y^2}}$

*For all loss functions

$$\frac{\partial S(W,Y,C_2|C_1)}{\partial w_1} = \frac{-\partial S(W,Y,C_1|C_2)}{\partial w_1}$$

$$\frac{\partial S(W,Y,C_2|C_1)}{\partial w_2} = \frac{-\partial S(W,Y,C_1|C_2)}{\partial w_2}$$

Table 4.4 Loss Functions and ∇S Components

$$W_{i+1} = W_i - \nabla\tilde{R}(W) = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_{i+1} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_i - \lambda \begin{bmatrix} \frac{\partial\tilde{R}}{\partial w_1} \\ \frac{\partial\tilde{R}}{\partial w_2} \end{bmatrix} \qquad (4.6)$$

Appendix F contains a listing of the FORTRAN code used to implement Equation 4.6. The risk as defined by Fquation 3.19 is given by

$$\tilde{R}(W) = \tilde{R}(W,C_1) + \tilde{R}(W,C_2)$$

$$= \frac{\sum_{n=1}^{N_1} S(W,Y_n^1,C_2|C_1)}{M_1} + \frac{\sum_{n=1}^{N_2} S(W,Y_n^2,C_1|C_2)}{M_2} \qquad (4.7)$$

where: N_i = total number of misclassified prototypes in class C_i

M_i = total number of prototypes in class C_i

Y_n^i = the n'th misclassified feature vector of class C_i

$S(W,Y_n^j,C_i|C_j)$ = the loss incurred when feature vector Y_n^j is misclassified into class C_i. The loss S is evaluated using the value of W that exists at the time Y_n^j is misclassified.

By using Equation 4.7 and the $S(W,Y,C_i|C_j)$ given in Table 4.4, $\tilde{R}(W)$ can be evaluated for different values of W. This was done for the prototypes created by the slab Monte Carlo problem of Figure 4.2 (with L = 10 and $\bar{I}$ = mean importance = .632) using 12 different values of a fixed decision surface (decision surface location = $-w_2/w_1$ = constant = 1,3,3.5,4,4.5,5,5.5,6,7,8,9,10) for each loss function. For these calculations $w_1 = 1$ resulting in $-w_2$ being equal to the above decision surface locations. Each estimate of $\tilde{R}(W)$ was determined by using 200 source particles (1024 prototypes). The resulting 12 values of $\tilde{R}(W)$ for each loss function were then fitted[37] with a second order polynomial. The resulting curves are illustrated in Figure 4.8 for the loss functions of Table 4.4 as well as the loss function

$$S(W,Y,C_i|C_j) = \text{constant}$$

Each curve has been normalized to its minimum risk, $\tilde{R}_{min}(W)$, resulting in a relative loss function. The location of the decision surfaces for minimum risk, x_{min}, as determined by the fitted polynomials are:

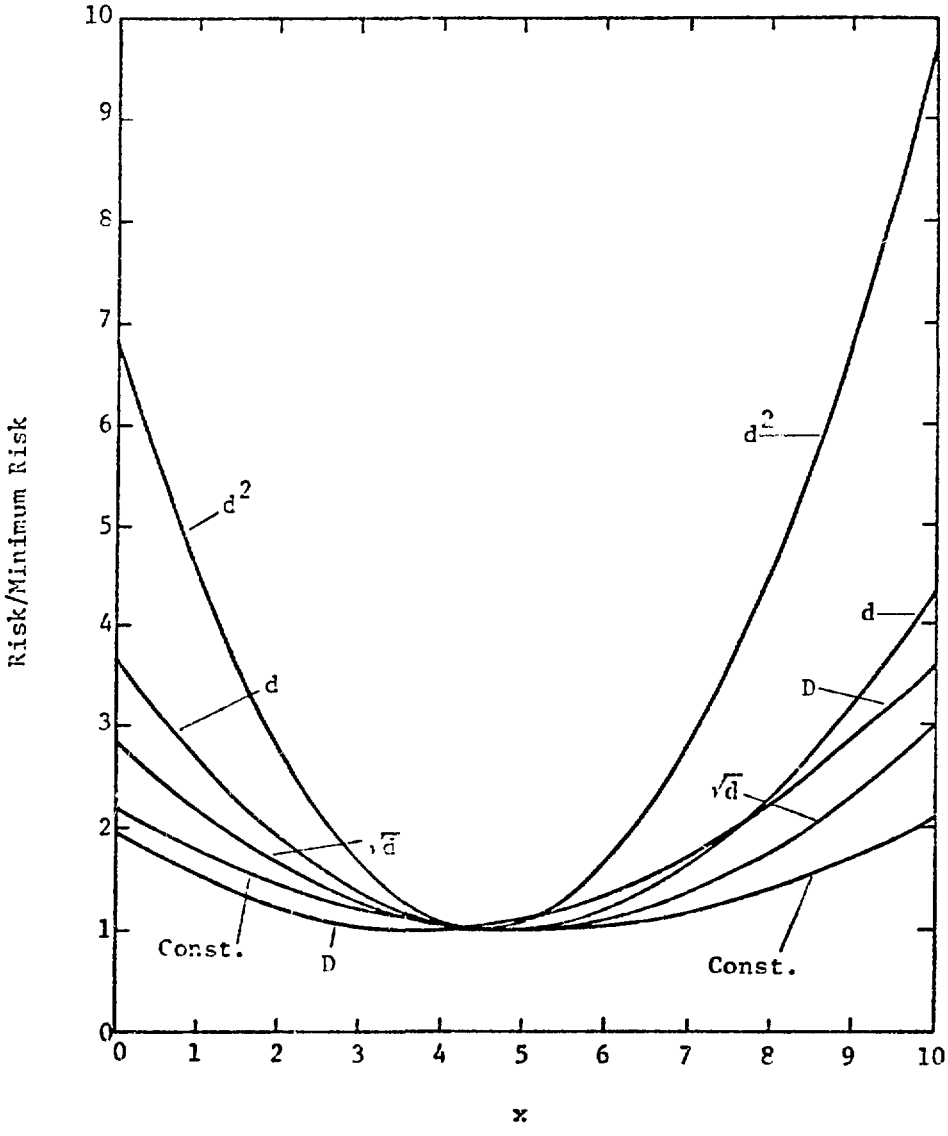


Figure 4.8 Relative Risk as a Function of a Decision Surface Location for Various Loss Functions

Loss Function	Decision Surface Location for Minimum Risk (x_{min})	Error*
Constant	5.11	.187
$\sqrt{d}$	4.50	.074
d	4.73	.022
d^2	4.89	.104
D	3.78	.442

*This estimated error is for the 1σ level and includes only the error introduced by the least squares curve fitting procedure[37].

Equation 4.7 is equivalent to the following:

$$\text{Risk} = R_1 + R_2 \qquad \text{for loss = Constant}$$

$$\text{Risk} = R_1\overline{\sqrt{d_1}} + R_2\overline{\sqrt{d_2}} \qquad \text{for loss} = \sqrt{d}$$

$$\text{Risk} = R_1\bar{d}_1 + R_2\bar{d}_2 \qquad \text{for loss = d}$$

$$\text{Risk} = R_1\left(\overline{d_1^2}\right) + R_2\left(\overline{d_2^2}\right) \qquad \text{for loss} = d^2$$

$$\text{Risk} = R_1\bar{D}_1 + R_2\bar{D}_2 \qquad \text{for loss = D}$$

where R_i = misclassification rate for $C_i = N_i/M_i$

N_i = no. of misclassified prototypes for C_i

M_i = total no. of prototypes for C_i

$\overline{d}_i$ = average d of misclassified prototypes of C_i

$\overline{\sqrt{d_i}}$ = average $\sqrt{d}$ of misclassified prototypes of C_i

$\left(\overline{d_i^2}\right)$ = average d^2 of misclassified prototypes of C_i

D_i = average D of misclassified prototypes of C_i

$i = 1,2$

The decision surface location for which

$$p(C_1|Y) = p(C_2|Y)$$

minimizes the total misclassification rate as given by

$$R_T = \frac{N_1 + N_2}{M_1 + M_2}$$

but does not minimize the risk as given by

$$\text{Risk} = R_1 + R_2 = \frac{N_1}{M_1} + \frac{N_2}{M_2}$$

unless the median is used for $\bar{I}$ (i.e., $M_1 = M_2$). Therefore, the minimum risk location for loss = constant is not the same as the $p(C_1|Y) = p(C_2|Y)$ location of Figure 4.4a. The other loss functions further affect the optimum location by introducing the average value of a function of d or D. Of these, the risk using loss = D has the smallest x_{min}. This is because the y_1 (=x) in the denominator of the expression for D (See Table 4.4) causes misclassified prototypes with small x to be weighed more heavily than those with large x. This causes the optimum decision surface to move so as to decrease g(Y) for smaller x (i.e., it moves to a smaller x_{min}). As L becomes large, the $p(C_i|Y)$ become more symmetric and (when the mean importance is used for $\bar{I}$) the $p(C_1|Y) = p(C_2|Y)$ location increases (see Figure 4.4). These two effects cause the x_{min} of different loss functions to approach the same value. It was found that for the case where L = 200, the differences in x_{min} are indistinguishable as far as the classifier is concerned.

For the general loss function given by

$$\text{Loss} = d^k \tag{4.8}$$

as $k \to \infty$ the maximum of the misclassification distances are minimized and as $k \to 0$ the percentage of misclassified patterns is minimized. This

is illustrated in Figure 4.8 by the high ratio of $\tilde{R}(W)/\tilde{R}_{min}(W)$ for loss = d^2 and the lower ratio for loss = $\sqrt{d}$.

The misclassification rate and variability of the decision surface are shown in Figure 4.9 and 4.10 respectively for the different loss functions over a range of λ. These plots are for the problem described earlier in Section 4.2,using a slab thickness of 200 and a source of 1000 particles. For all loss functions, the final value of the decision surface was between x = 166 and x = 167. These tests illustrate two important phenomena: (1) there is not a great amount of difference between the performance of the different loss functions after 1000 particles (i.e., when using optimum values of λ for each) and (2) the performance of the pattern classifier is dependent on the value of λ. For each loss function there is a range of λ (approximately three decades wide) over which the performance is relatively constant. A λ below this range leads to decreased performance because of the increase in convergence time it requires. A λ above this range leads to poor performance since it overcompensates for the correction.

The misclassification rate as a function of source particles started is shown in Figure 4.11. The values of λ used in these runs were chosen from Figures 4.9 and 4.10 so as to optimize performance and are

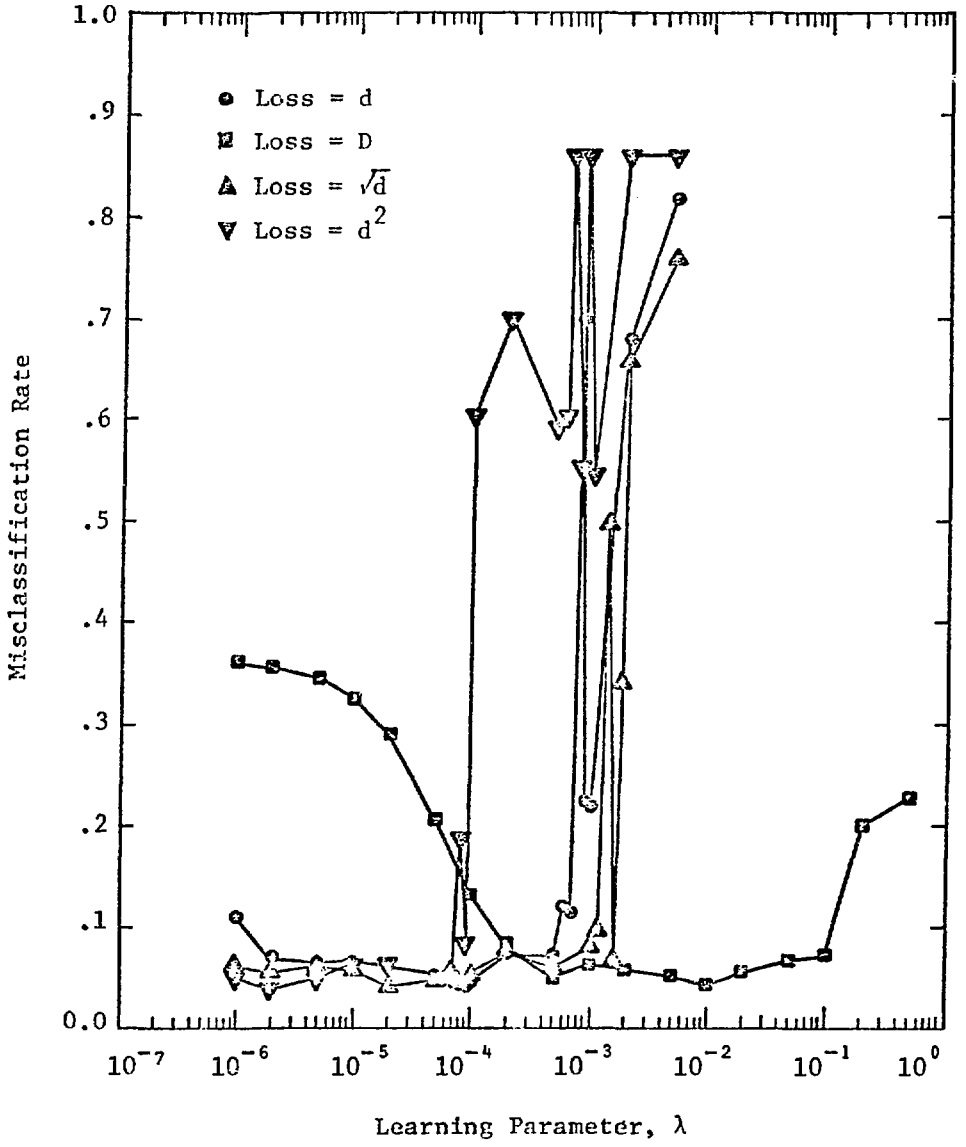


Figure 4.9 Misclassification Rate Vs. λ for Various Loss Functions

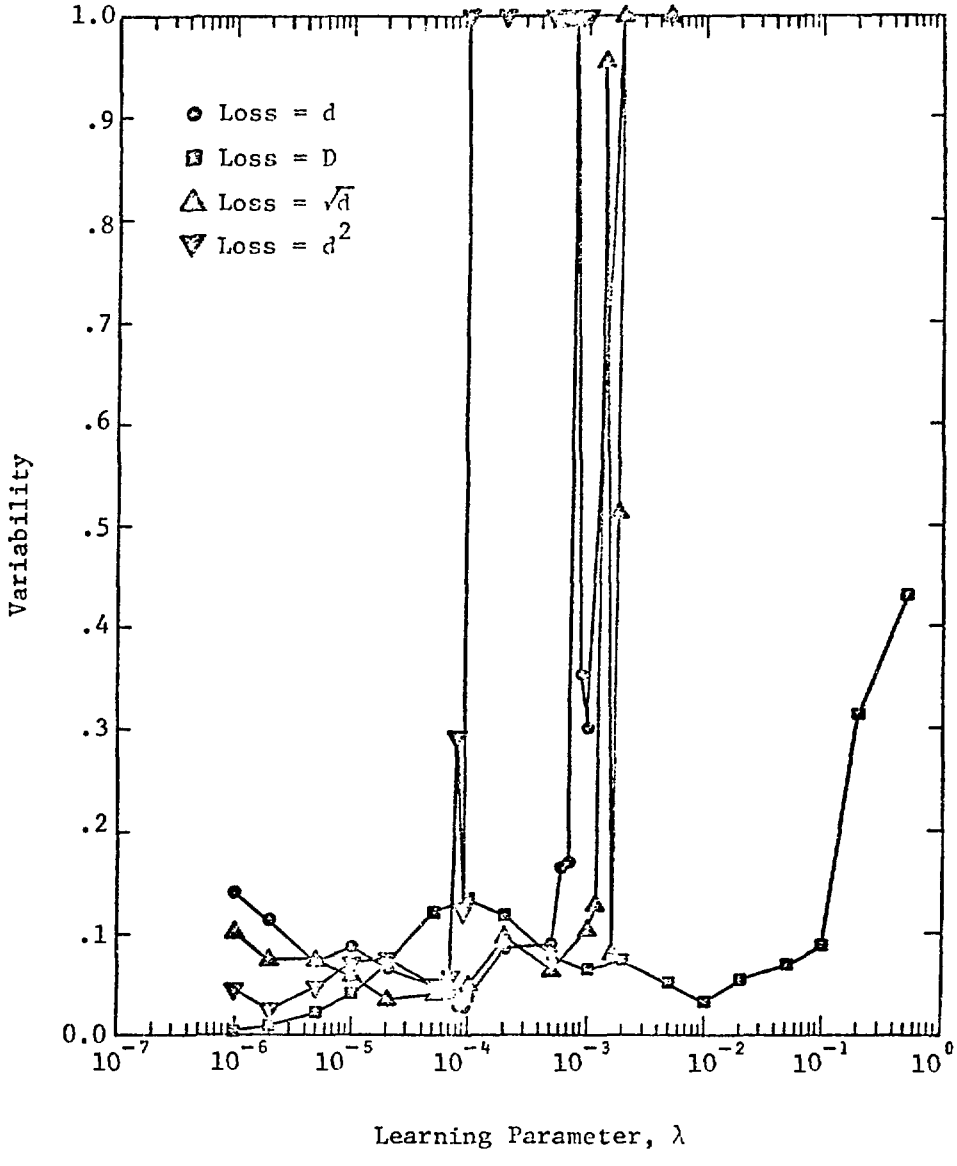


Figure 4.10 Variability Vs. λ for Various Loss Functions

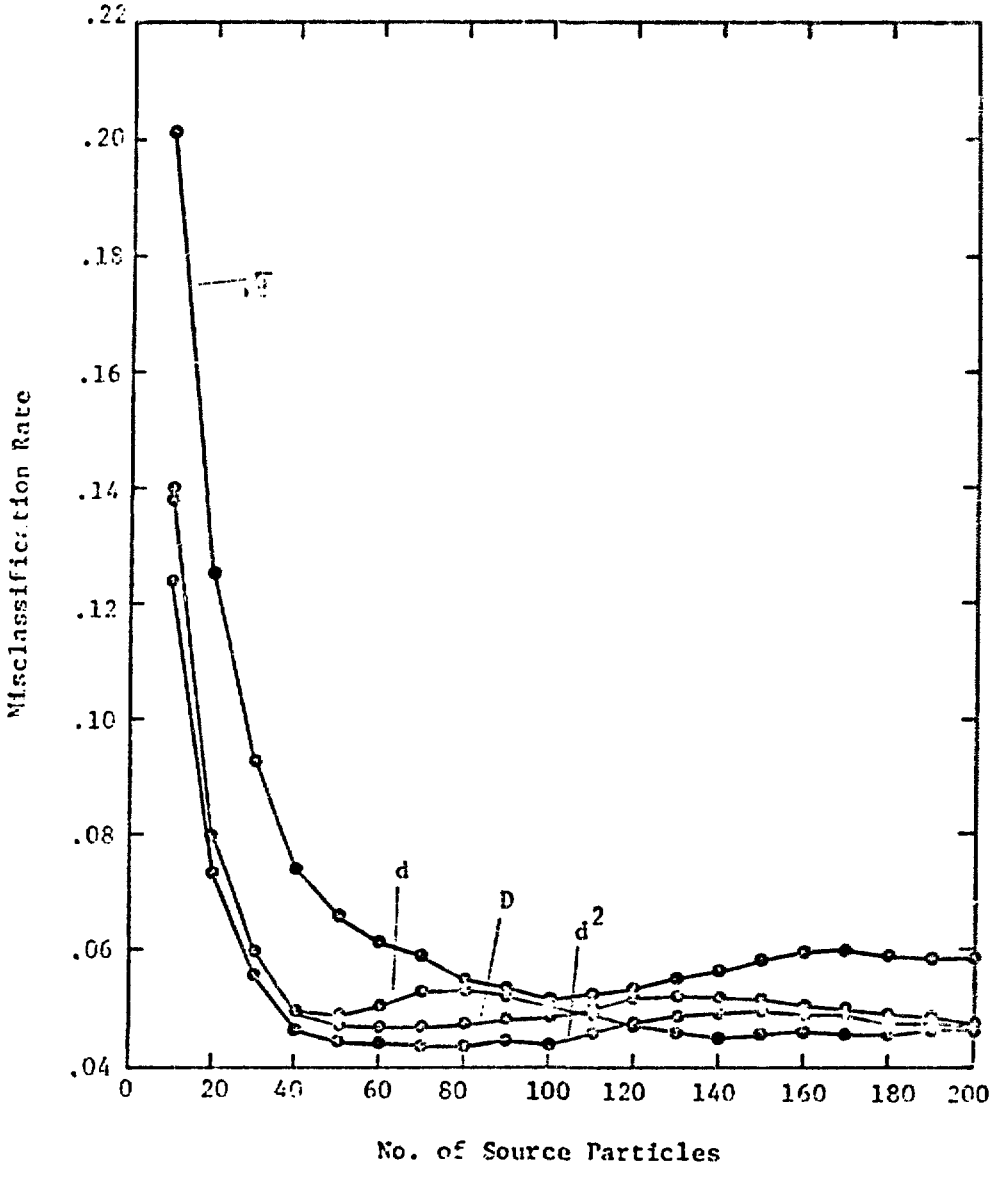
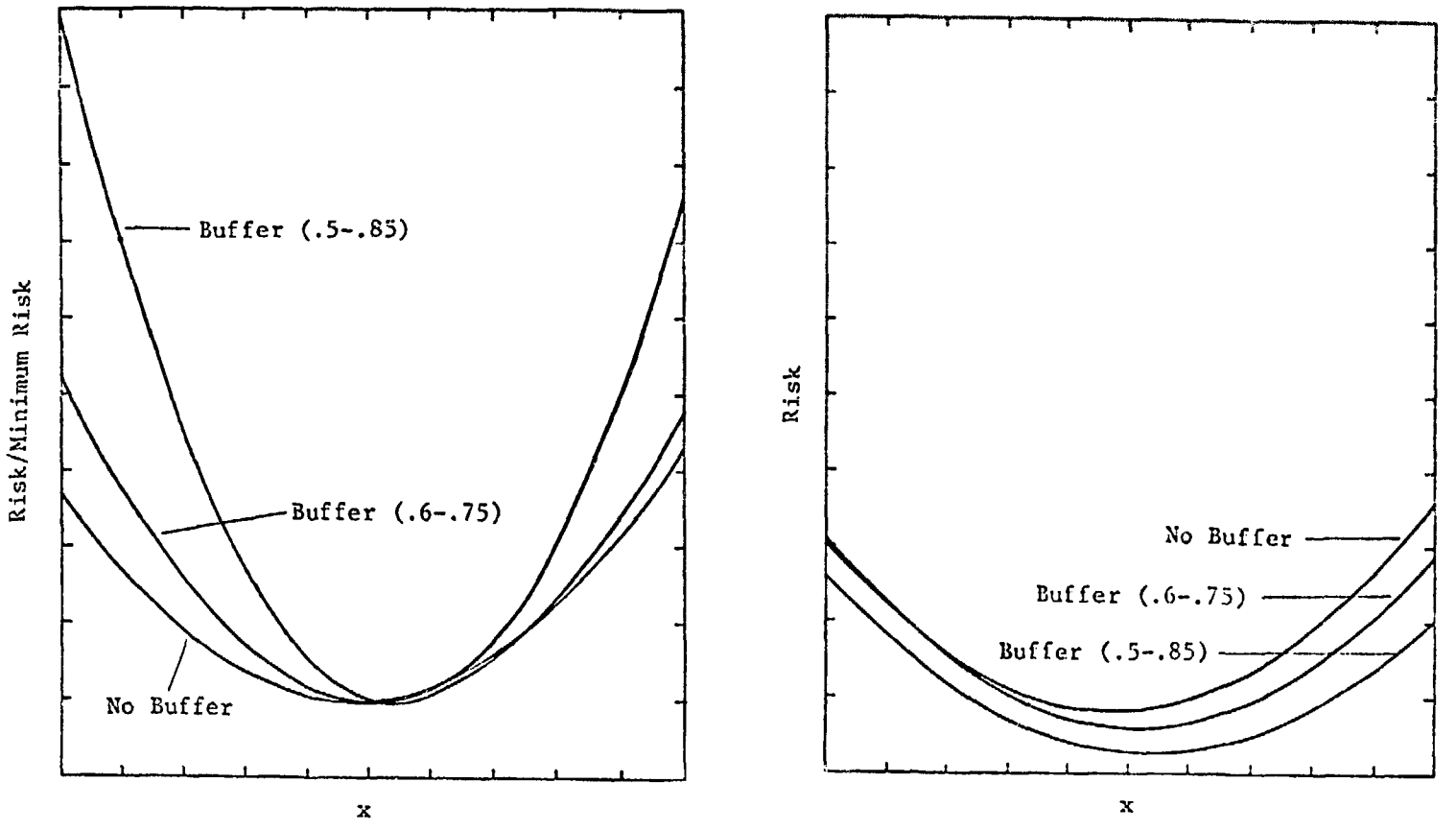


Figure 4.11 Misclassification Rate Vs. Number of Source Particles for Various Loss Functions

Loss Function	λ
$\sqrt{d}$	.00002
d	.00008
d^2	.000002
p	.01

With the exception of loss = $\sqrt{d}$, all loss functions lead to convergence in approximately the same number of particles with loss = d^2 being slightly quicker. Because of the small difference in performance of the above loss functions, loss = d appears to be the most attractive because of its computational simplicity. In all cases the statistical approach results in oscillations about the optimum decision surface. Although the technique is guaranteed to converge,[31] runs made with as many as 20,000 source particles still show the presence of this oscillation although it does decrease in amplitude.

To alleviate this oscillation, buffer zones are introduced. The effect of using buffer zones on the risk is illustrated in Figures 4.12a and 4.12b. Figure 4.12a indicates that buffer zones have the same effect as increasing k in Equation 4.8; however, Figure 4.12b shows that although the relative risk is increased, the absolute risk is actually decreased. Figure 4.13 shows the effect of the buffer zones on the misclassification rate and the variability of the decision surface. Although the variability is less for the buffer zone runs



(a) Risk Relative to Minimum Risk

(b) Absolute Risk

Figure 4.12 Effect of Buffer Zones on the Risk

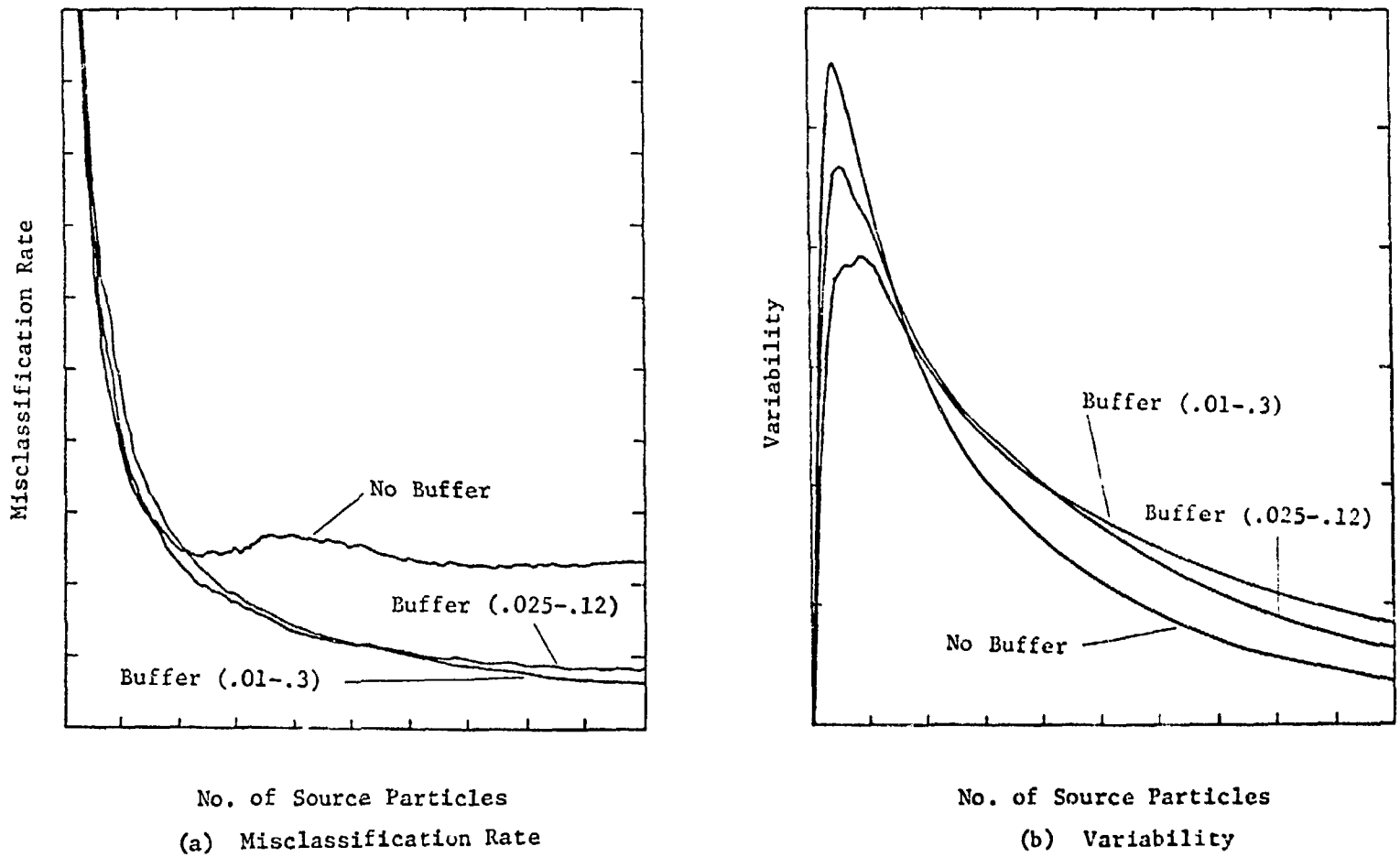


Figure 4.13 Effect of Buffer Zones on Misclassification Rate and Variability

during the initial approach to the decision surface, after this approach the variability of the non-buffer zone run is less. This behavior is consistent with the deterministic buffer runs of Section 4.2.1 and indicates that although the Y_i that lie in the buffer zone are not classified, they should be counted as prototypes when calculating the variability in order to obtain a true measure of the convergence. The smoothing of the convergence due to buffer zones is illustrated by Figure 4.14 for the problem used earlier in this section.

In summary, the statistical approach produced satisfactory results which do not depend greatly upon the selection of the loss function. A loss function proportional to the misclassification distance appears to be the most attractive since it requires fewer computations. Although the statistical approach does converge with time, it appears that like the deterministic approach, a decision surface will have to be selected prior to a final convergence. The use of buffer zones reduces the error after the initial approach and the variability (as calculated) during the initial approach. The increase in variability after the decision surface has begun to converge, when buffer zones are used, can be reduced by treating buffer zone feature vectors as prototypes when calculating the variability. Buffer zones are also attractive since by reducing the number of weight adjustments, they reduce the amount of computer time used.

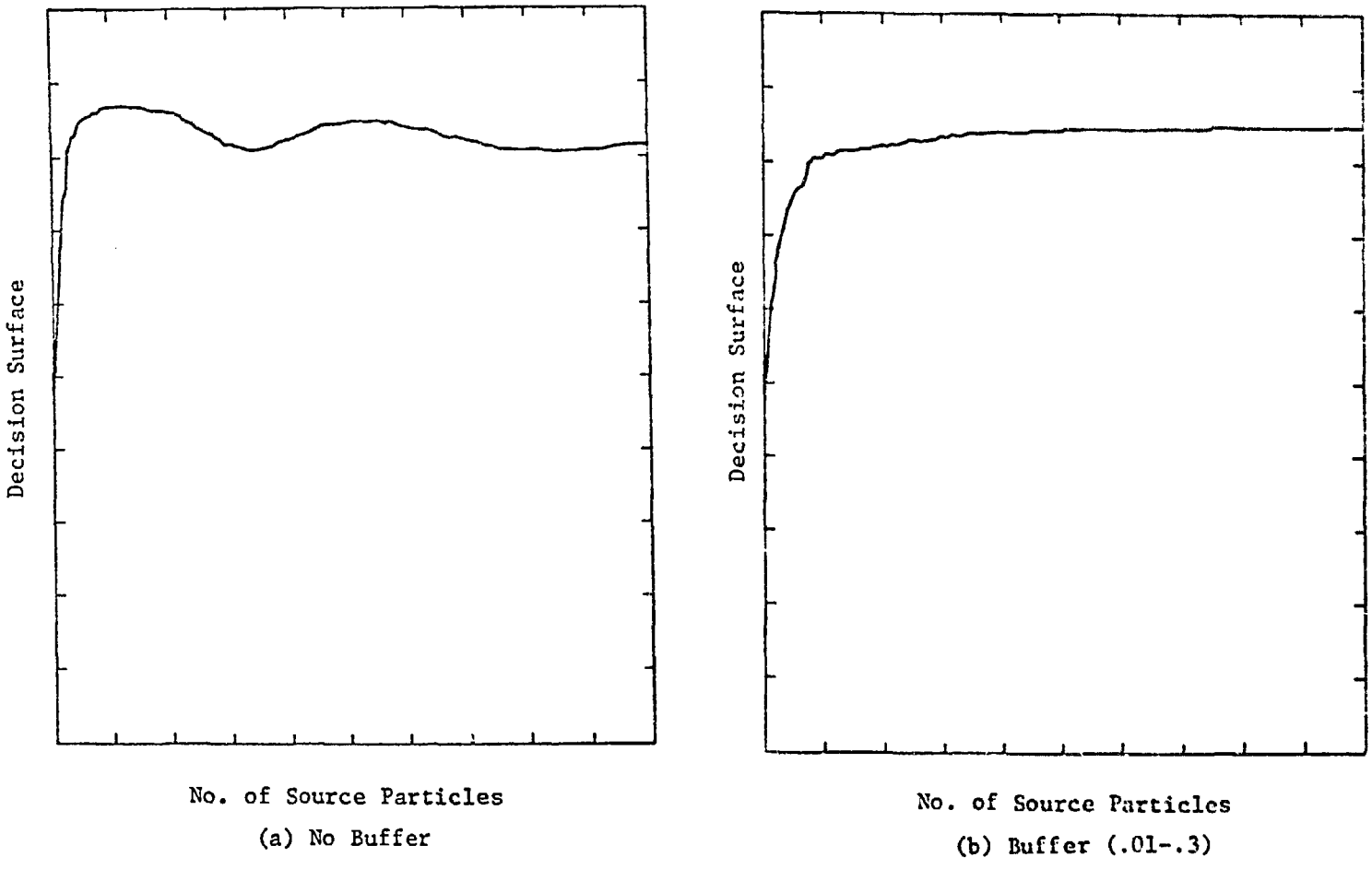


Figure 4.14 Smoothing of the Convergence Due to Buffer Zones

4.2.3) Comparison of Deterministic and Statistical Techniques

In the previous two sections statistical and deterministic techniques have been implemented on different thickness slabs using a range of learning parameters (λ) and different buffer zones. Figure 4.15 compares deterministic and statistical classifiers using $L = 200$, $\Sigma_t = .5$, $\Sigma_s = .4$, no buffer zones, and an $\bar{I} = .05$. Both runs use an optimum λ for the specific technique ($\lambda = .05$ for deterministic and $\lambda = .00008$ for statistical). Although the deterministic classifier leads to slightly fewer misclassifications in the early stages of the run, as the decision surface begins to converge (after about 40 particles) the two techniques are very competitive.

The variability of the decision surface appears to be considerably lower for the deterministic case. The reason for this can be seen from Figure 4.16 in which the average decision surface and the decision surface are plotted as a function of particles started. In the deterministic case a single prototype can cause the decision surface to be altered in a given direction. However, for the statistical classifier, the decision surface is altered in the direction indicated by the average properties of the prototypes. This behavior leads to a buildup of misclassification in the other direction. Although the variability of the decision surface does appear greater for the statistical classifier during the early stages, the final variability of the deterministic classifier will never go to zero; whereas for the statistical, it is guaranteed.

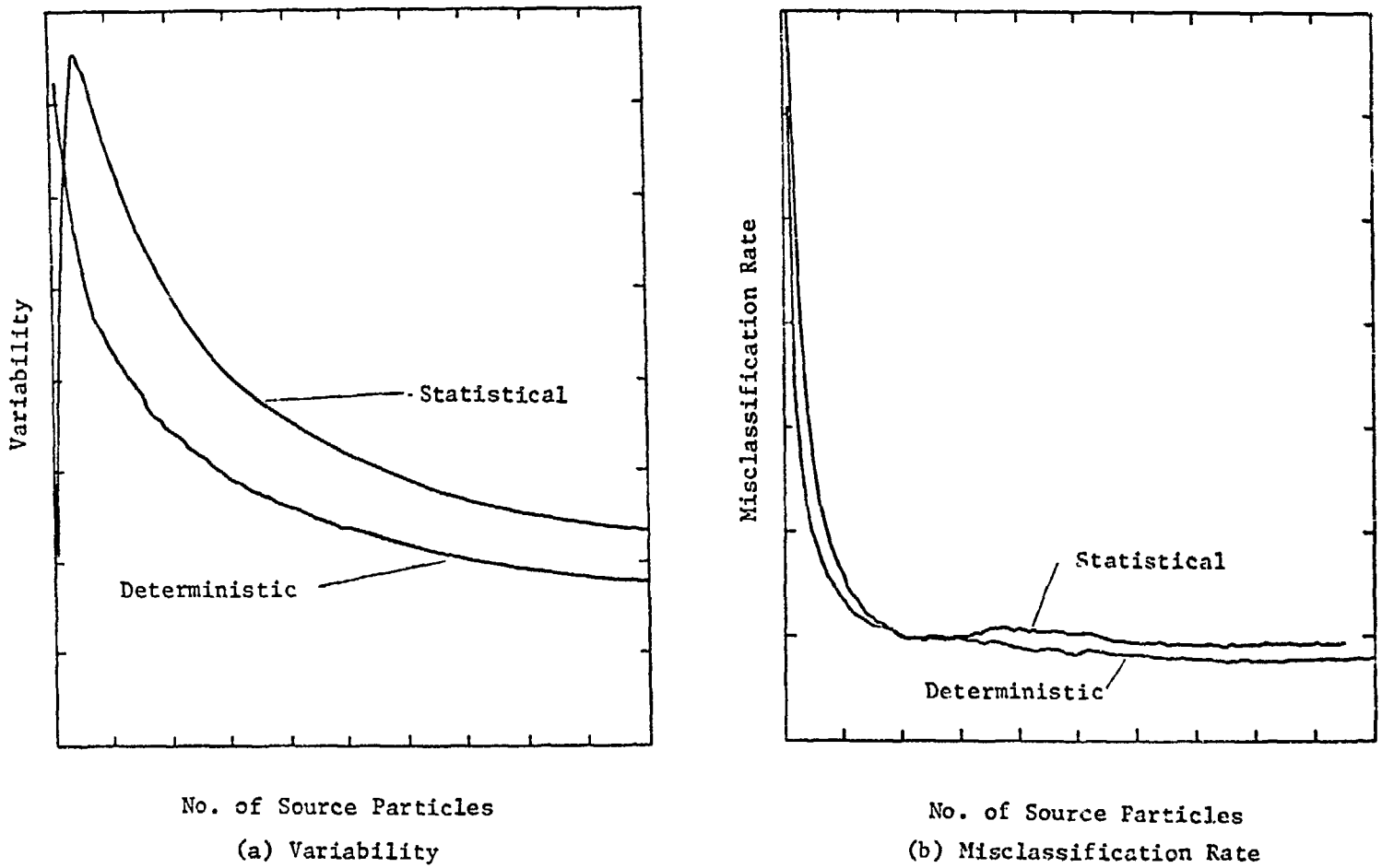


Figure 4.15 Comparison of the Performance of Deterministic and Statistical Classifiers

Figure 4.16 Behavior of the Decision Surface

The computer time spent for the two techniques can be compared by using Equation 4.9

$$T = N(t_A + t_L f_L + t_C f_L f_C), \tag{4.9}$$

where T = total computer time spent for pattern recognition,

N = total number of prototypes produced,

t_A = time required to determine the classification of a prototype,

t_L = time required to determine if a prototype is misclassified,

t_C = time required to adjust weights,

f_C = fraction of classifiable prototypes which are misclassified, and

f_L = fraction of prototypes which are classifiable (lie outside the buffer zone).

Using suitable timing schemes results in the values shown in Table 4.5. The following results are obtained for T after 40 particles are started (resulting in N = 4013) with no buffer zone (i.e., $f_L = 1.0$).

Statistical ($f_C = .050$)

$$T = 1.65 \times 10^{-2} + .40 \times 10^{-2} + .21 \times 10^{-2} = 2.26 \times 10^{-2}$$

	Time in Seconds	
	Statistical	Deterministic
t_C	10.5×10^{-6}	3.5×10^{-6}
t_L	1.0×10^{-6}	1.0×10^{-6}
t_A	4.1×10^{-6}	4.1×10^{-6}

Table 4.5 Timing Parameters for the One Dimensional Problem

Deterministic ($f_C = .051$)

$$T = 1.65 \times 10^{-2} + .40 \times 10^{-2} + .07 \times 10^{-2} = 2.12 \times 10^{-2}$$

From these results it is seen that the two techniques are comparable since the majority of time spent is not for weight adjustment but for class identification. However, since the variability is less for the deterministic classifier, the statistical classifier requires more prototypes (greater N) to get the same results. Thus the deterministic classifier appears to be the most advantageous with respect to time.

The above times can be reduced by the use of buffer zones as shown below for a buffer zone of .01 to .30 ($f_L = .85$).

Statistical ($f_C = .051$)

$$T = 1.65 \times 10^{-2} + .34 \times 10^{-2} + .17 \times 10^{-2} = 2.16 \times 10^{-2}$$

Deterministic ($f_C = .048$)

$$T = 1.65 \times 10^{-2} + .34 \times 10^{-2} + .06 \times 10^{-2} = 2.05 \times 10^{-2}$$

The optimum λ's found in previous sections are a function of slab thickness, thus requiring normalization of the correction algorithms. This is done by replacing the augmented feature vector Y* by Y' as given by

$$\text{unnormalized } Y^* = \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \text{normalized } Y' = \begin{bmatrix} \frac{x}{L} \\ \frac{1}{L} \end{bmatrix} \tag{4.10}$$

This results in

$$w_1 = w_1 \pm c'x' \text{ and} \tag{4.11}$$

$$w_2 = w_2 \pm \frac{c'}{L},$$

$$\text{where } c' = \frac{\lambda |g'|}{x'^2 + \frac{1}{L^2}} = Lc$$

$$g' = w_1 x' + w_2/L$$

$$x' = x/L$$

$$L = \text{slab thickness}$$

for the correction algorithms. Similarly, for the statistical classifier the following should be used:

$$W_{i+1} = W_i - \lambda' \nabla \tilde{R}'(W_i), \tag{4.12}$$

where $\lambda' = \lambda/L$ and

$$\begin{bmatrix} \frac{\partial R}{\partial w_1} \\ \\ \frac{\partial R}{\partial w_2} \end{bmatrix}' = \begin{bmatrix} \frac{-w_2 L}{w_1^2} \\ \\ \frac{-1}{w_1 L} \end{bmatrix}$$

In summary, for the one-dimensional homogeneous slab the deterministic approach appears to have an advantage over the statistical approach because of its reduced variability at early stages. Furthermore, if the variability is used as a threshold for using a decision surface shortly after the approach to such a surface, the deterministic classifier also presents a savings in computer time. However, the variability of the deterministic classifier is bounded by a minimum (depending on the amount of overlapping of the class distributions); whereas the statistical classifier has a variability which is guaranteed to approach zero.

4.3) Multi-Region Slab and Initial Conditions

In the previous section a one region slab was used to demonstrate the type of behavior which might be expected when pattern recognition is used to identify splitting surfaces. Although convenient for demonstration purposes, such problems are little challenge to the skilled Monte Carlo user as far as identifying splitting planes is concerned. If one considers the problem of a multi-region slab consisting of several layers of different materials, the problem becomes much more difficult for the "human intuition" approach. This section considers this slightly more complex problem using the identical pattern recognition techniques described in the previous section.

The problem is illustrated in Figure 4.17 for a slab consisting of four materials. The FORTRAN coding for the Monte Carlo program used to solve this problem is shown in Appendix G. Figure 4.18 shows the importance distributions for several different combinations of materials (see Table 4.6). Each region is 50 cm thick and has a total macroscopic cross section of $\Sigma_t = .5/cm$. The variety of distributions results from rearranging the sequence of the materials and in one case even leads to a bi-modal distribution. The class distributions for these same problems is shown in Figure 4.19 where the median importance has been used for $\bar{I}$.

Because Σ_t is the same for all four cases, the collision points of each Monte Carlo run are identical (since each Monte Carlo run uses the same set of random numbers). Therefore the set of feature vectors

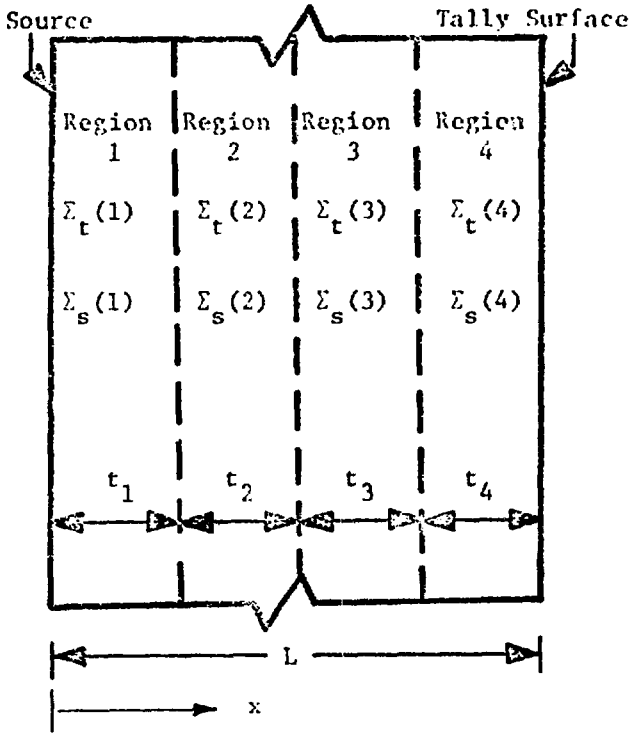


Figure 4.17 The Multi-Region Slab Problem

Y_i, used for prototypes is identical for each case. The variation in results of the four cases can therefore be attributed solely to the effect of different class and importance distributions as caused by the different Σ_s's.

Because particles are allowed to continue through the slab, there is an equal probability throughout the slab of having a collision (and thus a prototype created) between x and x+dx. As a result for half the prototypes

$$y_1 = x < 100$$

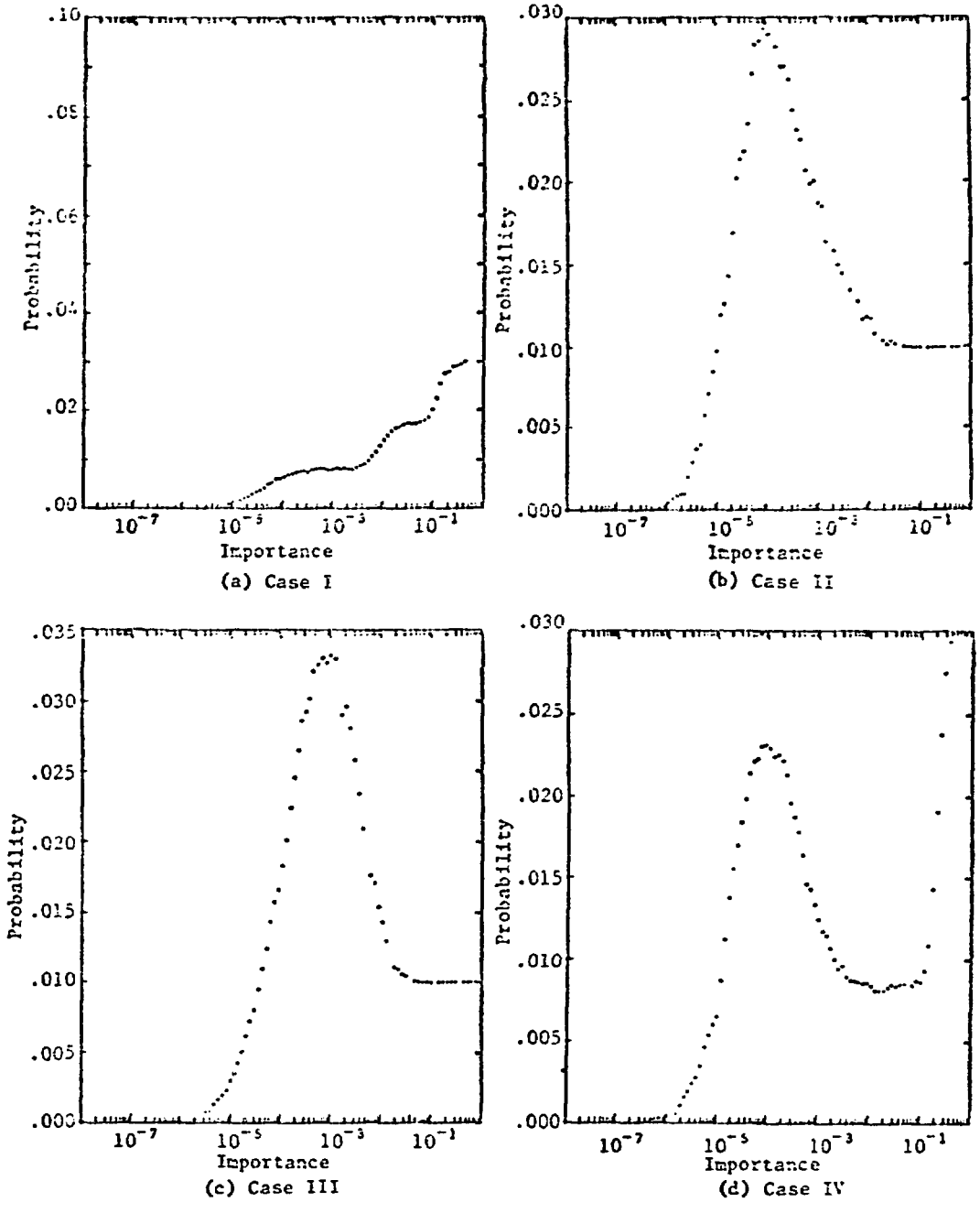


Figure 4.18 Importance Distributions for the Multi-Region Slab Problem

	Case I	Case II	Case III	Case IV
$\Sigma_s(1)$	.40	.49	.45	.49
$\Sigma_s(2)$	.45	.47	.49	.45
$\Sigma_s(3)$	.47	.45	.47	.40
$\Sigma_s(4)$	.49	.40	.40	.47
Tally	6.7×10^{-5}	7.6×10^{-5}	7.5×10^{-5}	7.0×10^{-5}
Median I	.112	.00035	.00105	.0011
Mean I	.288	.0502	.0508	.140
Er*	3.2	12.6	19.1	5.6

*Using Median I for $\bar{I}$

Table 4.6 Multi-Region Sample Problems

and for the other half

$$y_1 = x > 100.$$

Therefore the optimum position to locate the splitting surface (using the median $\bar{I}$) is the same for all four cases and is located at $x = 100$.

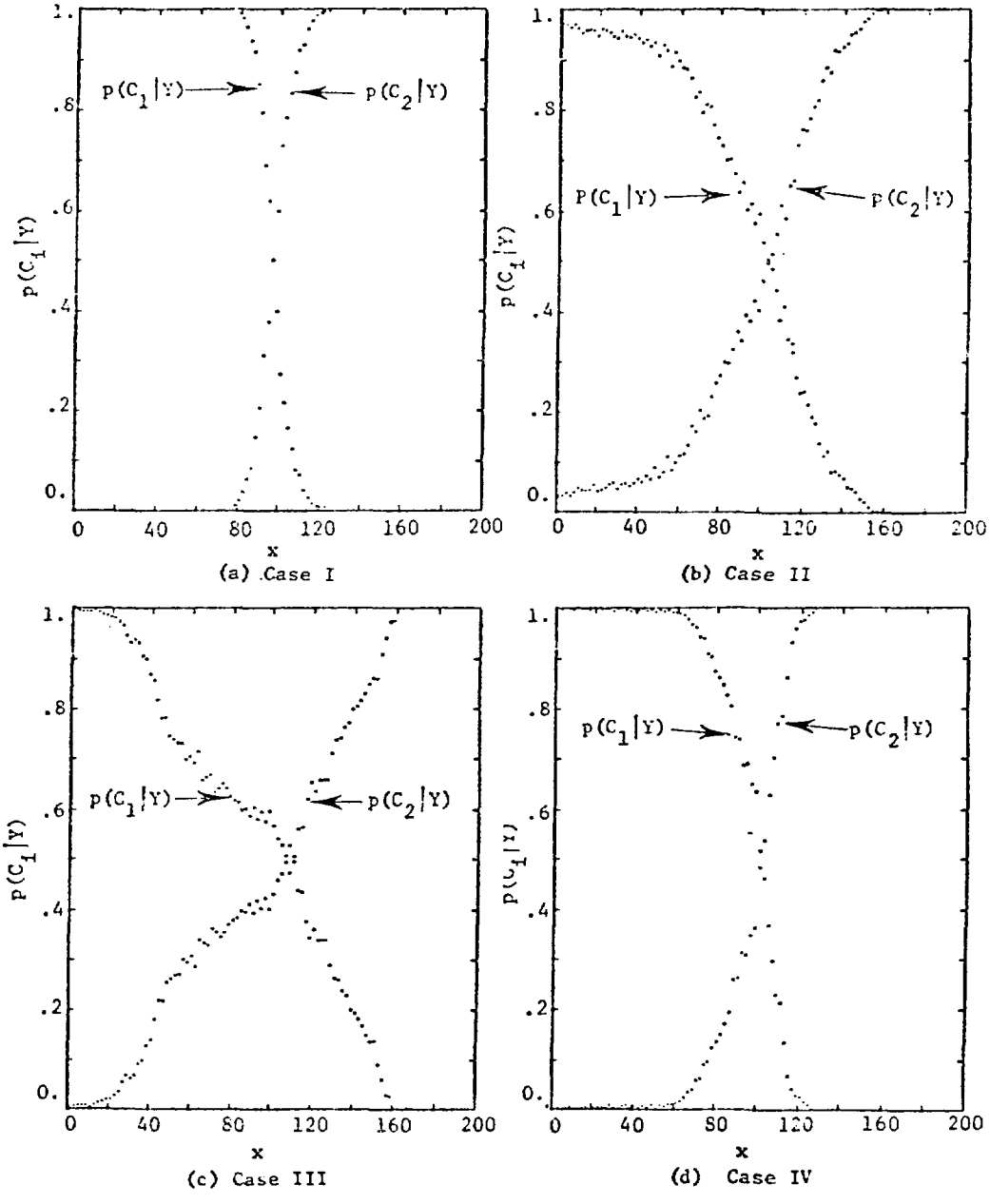


Figure 4.19 Class Distributions for the Multi-Region Slab Problem

An additional set of computational experiments is performed in this section to determine the sensitivity of the pattern classifiers to the selection of an initial decision surface, S_1, where $S_1 = -w_2/w_1$ for the initial selection of w_1 and w_2.

4.3.1) Initial Conditions

In Section 4.2, the midpoint of the slab was used as the initial location of the decision surface. In this section, the effect of the initial location, S_1, on the performance of several pattern classifiers is analyzed using Case I. Several computation experiments are made using different classification algorithms and different learning parameters, λ. The initial location can be varied in two ways: (1) variation of the initial value of w_1 and (2) variation of the initial decision surface location (i.e., $-w_2/w_1$).

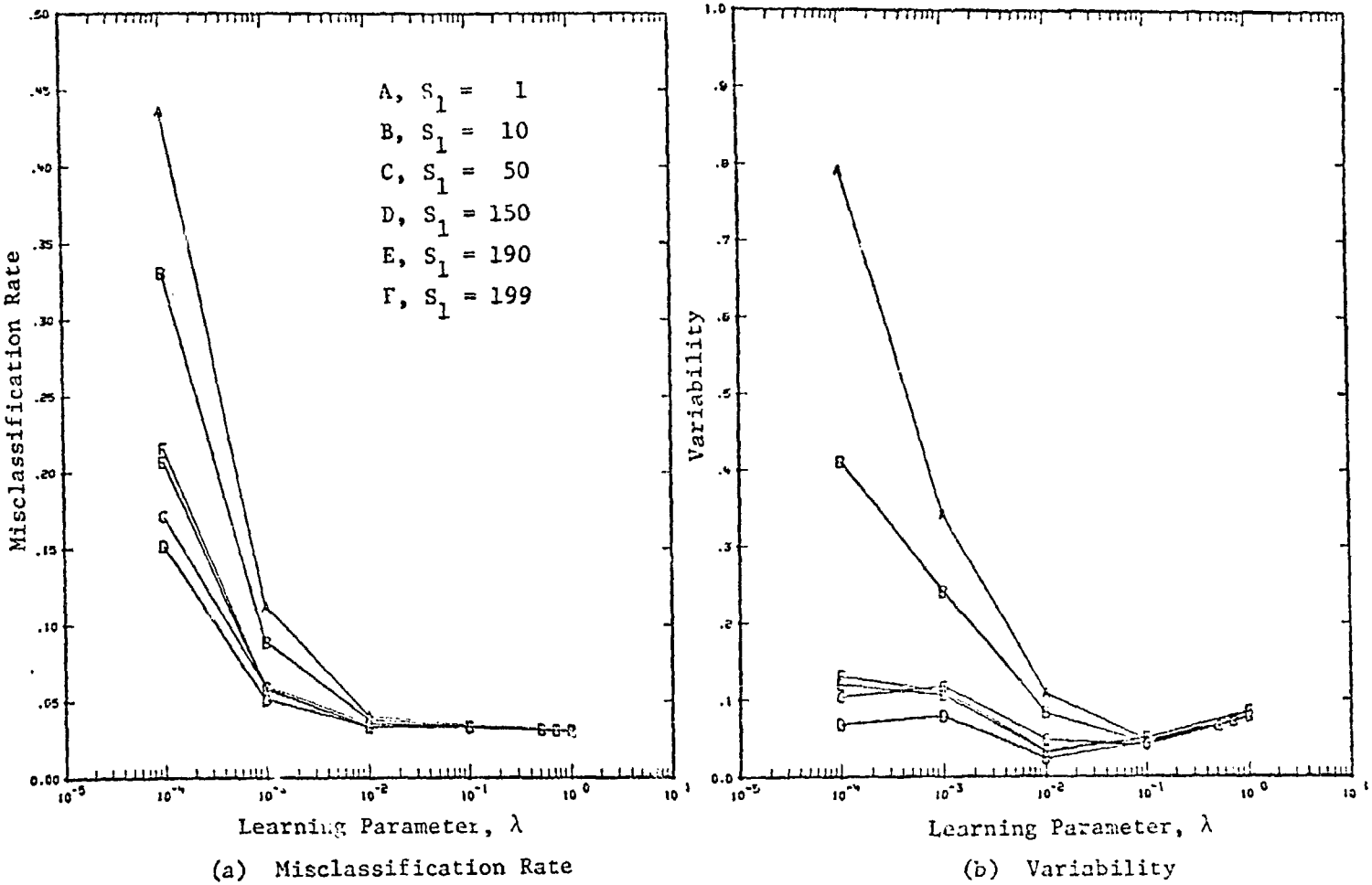
4.3.1a) Deterministic Classifier. The location of the decision surface after the j'th prototype belonging to C_k has been misclassified is given by

$$S_{j+1} = \left(\frac{-w_2}{w_1}\right)_{j+1} = \left(\frac{-w_2 + \frac{(-1)^{k+1}\lambda|g(Y)|}{(x^2+1)}}{w_1 + \frac{(-1)^{k}\lambda|g(Y)|x}{(x^2+1)}}\right)_j$$

$$= \frac{S_j(x^2+1) + \lambda(x-S_j)}{(x^2+1) - \lambda x(x-S_j)} \qquad (4.13)$$

where x is the feature, y_1, of the j'th misclassified prototype. Since S_{j+1} does not depend on w_1 (independently of S_j), the value of w_1 for any given S_j has no effect upon the behavior of the classifier. Therefore the initial selection of w_1 (for any given S_1) causes no change in classifier performance. The effect of varying the initial decision surface, S_1, after 1000 source particles is demonstrated in Figure 4.20 for $10^{-4} \leq \lambda \leq 1$ and six different S_1. This figure illustrates that after 1000 source particles the performance is relatively independent of initial conditions and λ for $\lambda > 1.2$.

The performance as a function of the number of source particles is shown in Figure 4.21 for runs using $\lambda = .05$. In these runs it was found that when S_1 is selected below the final decision surface location, (i.e. S_1=1,10,50), the decision surface converges monotonically to the final location. This behavior is reflected in Figure 4.21a by the monotonically decreasing misclassification rate when S_1=1, 10, and 50. However, when S_1 is started above the final decision surface



(a) Misclassification Rate

(b) Variability

Figure 4.20 Performance Vs. λ for Various Initial Conditions, Deterministic Classifier

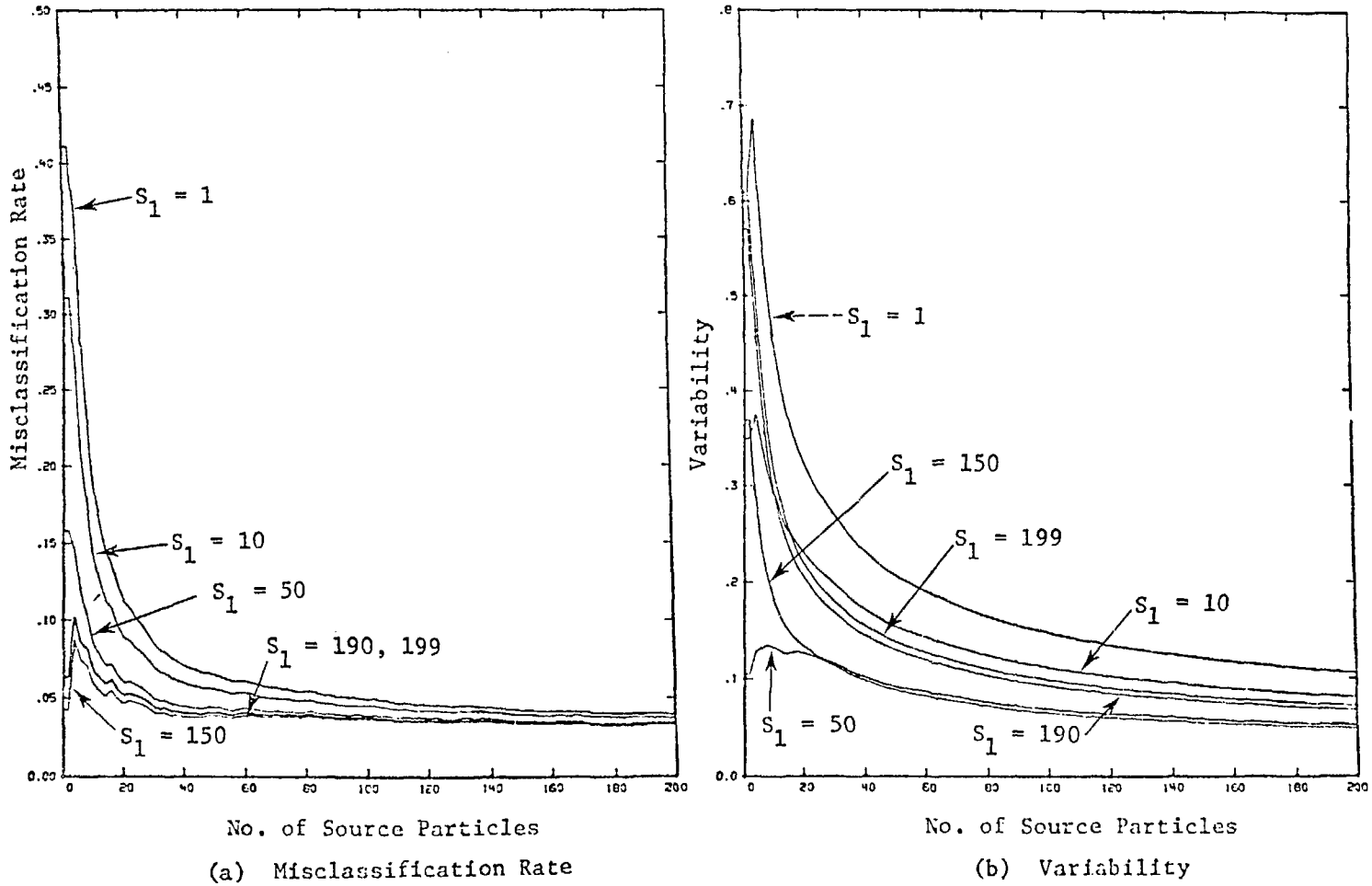


Figure 4.21 Performance Vs. Number of Source Particles for Various Initial Conditions, Deterministic Classifier

location (i.e., $S_1 = 150$, 190, 199), it was found that the decision surface very quickly (in less than 10 source particles) moved past the optimum location and then converged to its final location from below. This behavior causes the misclassification rate to increase when the decision surface overshoots the optimum location; however, as the surface begins its final convergence, the misclassification rate decreases. This behavior is demonstrated by the $S_1 = 150$, 190, and 199 curves of Figure 4.21. Thus although two S_1 may be located equidistant from the optimum surface location (for example: $S_1 = 10$ and $S_1 = 190$) the S_1 located above the surface leads to better classifier performance. This behavior is explained by investigating Equation 4.13 and noting that the change in S_j, ΔS_j, is affected by the value of the feature x as well as the misclassification distance, $\Delta = |x - S_j|$. Contours of $\Delta S_j = |S_{j+1} - S_j|$ using Equation 4.13 are shown in Figure 4.22* for $\Delta = 10$, 20 and $\lambda = .01$, .05. This figure iilustrates that for misclassifications in class C_1, $\Delta S_j \to 0$ as $S_j \to 0$ which explains the slow convergence when $S_1 << 100$. The increasing slope of ΔS_j as $S_j \to 100$ from the right as opposed to the decreasing slope as $S_j \to 100$ from the left explains why the decision surface overshoots and then approaches from below when $S_1 = 150$, 190, and 199.

---

*Because of overlapping classes the curves shown in Figure 4.22 actually extend across $S_j = 100$. However, the class 1 misclassification curve has been plotted only for $S_j < 100$ and class 2 curve only for $S_j > 100$.

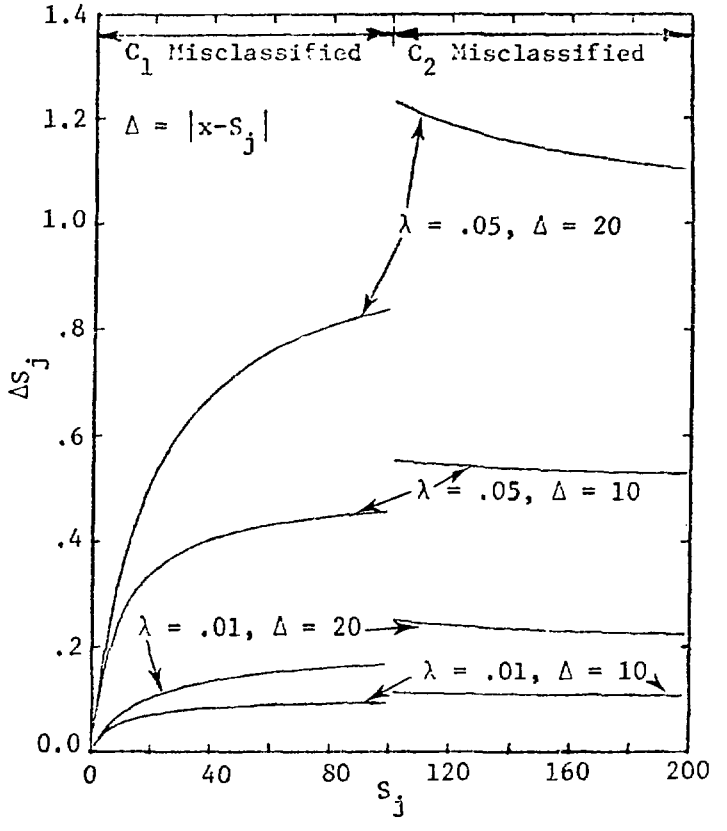


Figure 4.22 Behavior of S_{j+1} for the Deterministic Classifier

These results indicate that for the deterministic classifier there is a very distinct advantage to locating S_1 above the optimum location.

4.3.1b) Statistical Classifier with Loss = d. The decision surface location after the j'th prototype belonging to C_k has been misclassified is derived in Appendix K and is given by

$$S_{j+1} = \left\{ \frac{S_j w_1 + \lambda \left[T_2 - \frac{A_{k,2}}{(M_k+1)} + \frac{(-1)^{k+1}}{w_1 (M_k+1)} \right]}{w_1 - \lambda \left[T_1 - \frac{A_{k,1}}{(M_k+1)} + \frac{(-1)^{k+1} S_j}{w_1 (M_k+1)} \right]} \right\}_j \qquad (4.14)$$

where M_k = number of prototypes in the k'th class (not counting the last misclassified prototype)

$$A_{k,1} = (-1)^k \sum_{n=1}^{N_k} \left(\frac{w_2}{w_1^2} \right)_n$$

$$A_{k,2} = (-1)^{k+1} \sum_{n=1}^{N_k} \left(\frac{1}{w_1} \right)_n$$

$$T_1 = A_{1,1} + A_{2,1}$$

$$T_2 = A_{1,2} + A_{2,2}$$

N_k = number of misclassified prototypes in the k'th class (not counting the last misclassified prototype)

Unlike the deterministic classifier, S_{j+1} depends on the selection of an initial w_1. However, by proper selection of λ(i.e., by keeping λ/w_1^2 = constant), this effect can be cancelled. The effect of varying λ/w_1^2 is shown in Figure 4.23a for the case where $T_1 = T_2 = A_{i,j} = M_k = 0$. Although, in general $M_k >> 0$, this figure does illustrate the relative effects of S_j and λ/w_1^2 on the convergence of the classifier.

Once the decision surface has converged to a final location, T_1 and T_2 approach zero. However, due to the overlapping of classes, $A_{k,i}$ approaches a value proportional to the amount of overlapping. Fig-4.23b illustrates that this overlapping has very little influence on ΔS_j. This is to be expected since this term of Equation 4.14 varies as $1/M_k^2$,whereas the other terms vary as $1/M_k$ ($A_{k,i}$ itself varies as $1/M_k$).

When the decision surface is still approaching a final location, T_1 and T_2 will not be zero. If the surface is approaching the optimum location from below (i.e., $S_1 < 100$), T_1 and T_2 are positive. The behavior of ΔS_j for this case is illustrated in Figure 4.23c for $T_1 = 0$, 1, 10, $T_2 = .01T_1$, and $M_k = 100$, 1000. Similarly the case for $S_1 > 100$ is illustrated in Figure 4.23d. These figures illustrate the relative effects of two factors: (1) the statistical effects of past prototypes reflected in T_1 and T_2 and (2) the effect of the last misclassified prototype (the $1/w_1$ and S_j/w_1 terms of Equation 4.14).

As the misclassification rate of past prototypes becomes large compared with the effects of the last misclassified prototype,

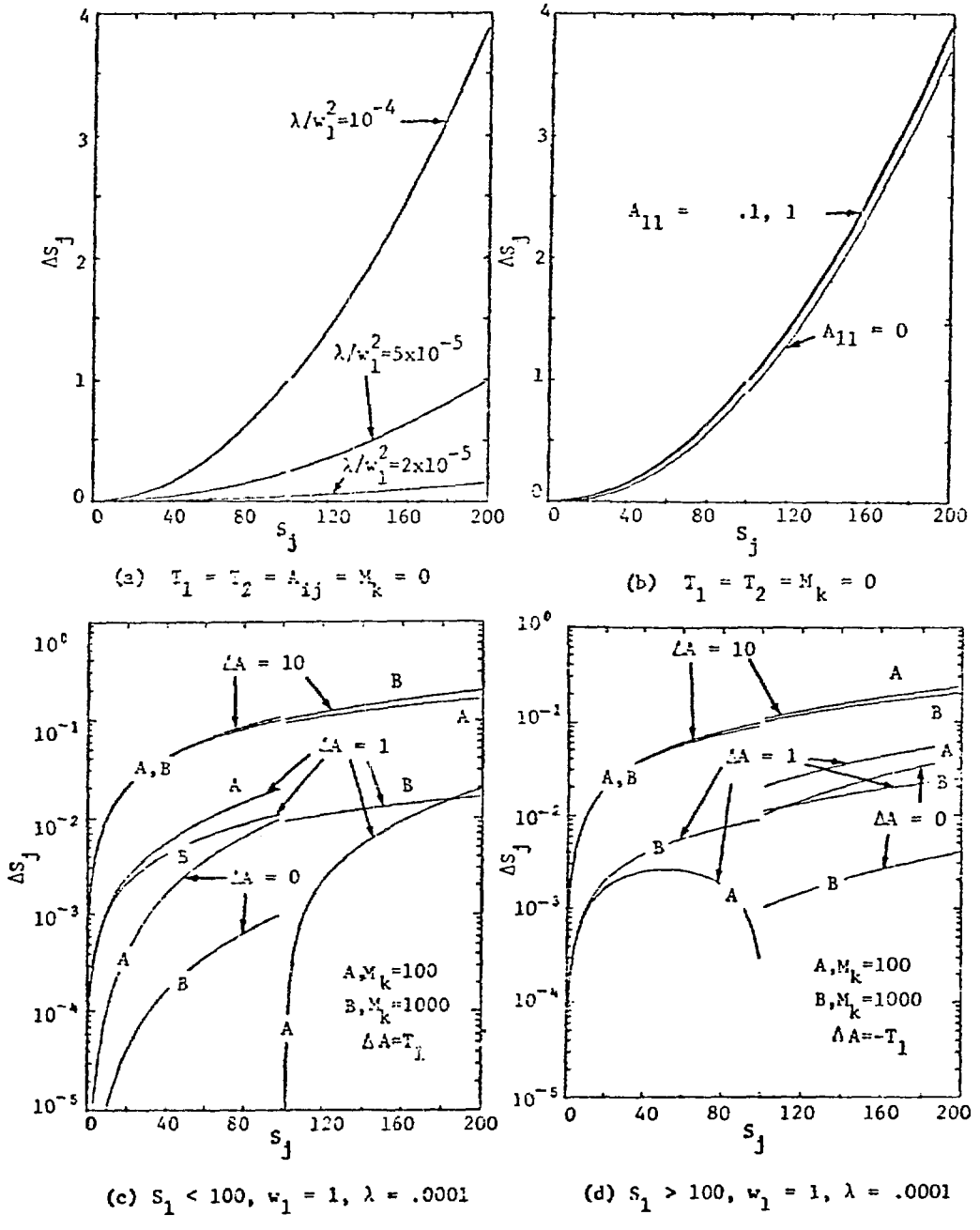


Figure 4.23 Behavior of S_{j+1} for the Statistical Classifier with Loss = d

$$|T_2| >> \left| \frac{A_{k,2}}{(M_k+1)} + \frac{(-1)^{k+1}}{w_1(M_k+1)} \right|$$

and

$$|T_1| >> \left| \frac{A_{k,1}}{(M_k+1)} + \frac{(-1)^{k+1}S_j}{w_1(M_k+1)} \right|$$

which using Equation 4.14 results in

$$\Delta S_j = S_{j+1} - S_j \approx \frac{\lambda(T_2 + T_1 S_j)}{w_1 - \lambda T_1} \tag{4.15}$$

If it is further assumed that $w_1 >> \lambda T_1$, and $T_2 << T_1 S_j$ Equation 4.15 reduces to

$$\Delta S_j \approx \frac{\lambda T_1 S_j}{w_1} \tag{4.16}$$

Thus ΔS_1 varies linearly with S_j as is demonstrated in Figures 4.23c and 4.23d by the curves for which $|T_1| = 10$.

When past misclassification becomes small (i.e., after convergence)

$$|T_1| << \left| -\frac{A_{k,1}}{(M_k+1)} + \frac{(-1)^{k+1}S_j}{w_1(M_k+1)} \right|$$

and

$$|T_2| << \left| -\frac{A_{k,2}}{(M_k+1)} + \frac{(-1)^{k+1}}{w_1(M_k+1)} \right|$$

And since in general

$$\left| \frac{A_{k,1}}{(M_k+1)} \right| << \left| \frac{S_j}{w_1(M_k+1)} \right|$$

$$\left| \frac{A_{k,2}}{(M_k+1)} \right| << \left| \frac{1}{w_1(M_k+1)} \right|$$

Equation 4.14 results in

$$\Delta S_j \approx \frac{(-1)^{k+1}\lambda(1+S_j^2)}{w_1^2(M_k+1) - (-1)^k\lambda S_j} \qquad (4.17)$$

If it is further assumed that $S_j^2 >> 1$ and $w_1^2(M_k+1) >> \lambda S_j$, Equation 4.17 reduces to

$$\Delta S_j \approx \frac{\lambda(-1)^{k+1} S_j^2}{w_1^2(M_k+1)} . \qquad (4.18)$$

Equation 4.18 indicates that as $T_i \to 0$ (i=1,2), ΔS_j varies quadratically with S_j. However, unlike Equation 4.16, the ΔS_j given by Equation 4.18 goes to zero as M_k increases. This behavior is illustrated by the curves in Figures 4.23c and 4.23d for which $T_1=0$. For these curves, the effect of M_k is much more pronounced than the curves for which $|T_1| = 10$. Although ΔS_j as given by Equation 4.18 varies quadratically with S_j as opposed to linearly for Equation 4.16, the coefficient of S_j^2 is much smaller than the coefficient of S_j, and the linear term dominates. Thus in Figures 4.23c and 4.23d, the ΔS_j for $|T_1| = 10$ is much greater than for $|T_1| = 0$.

When neither the past misclassification term nor the current misclassified prototype term is dominant, ΔS_j varies proportional to the difference between the two effects. This is illustrated by the $|T_1| = 1$ curves in Figures 4.23c and 4.23d.

The performance of the classifier after 1000 source particles for six different S_1 is shown in Figure 4.24 for a range of λ from 10^{-6}

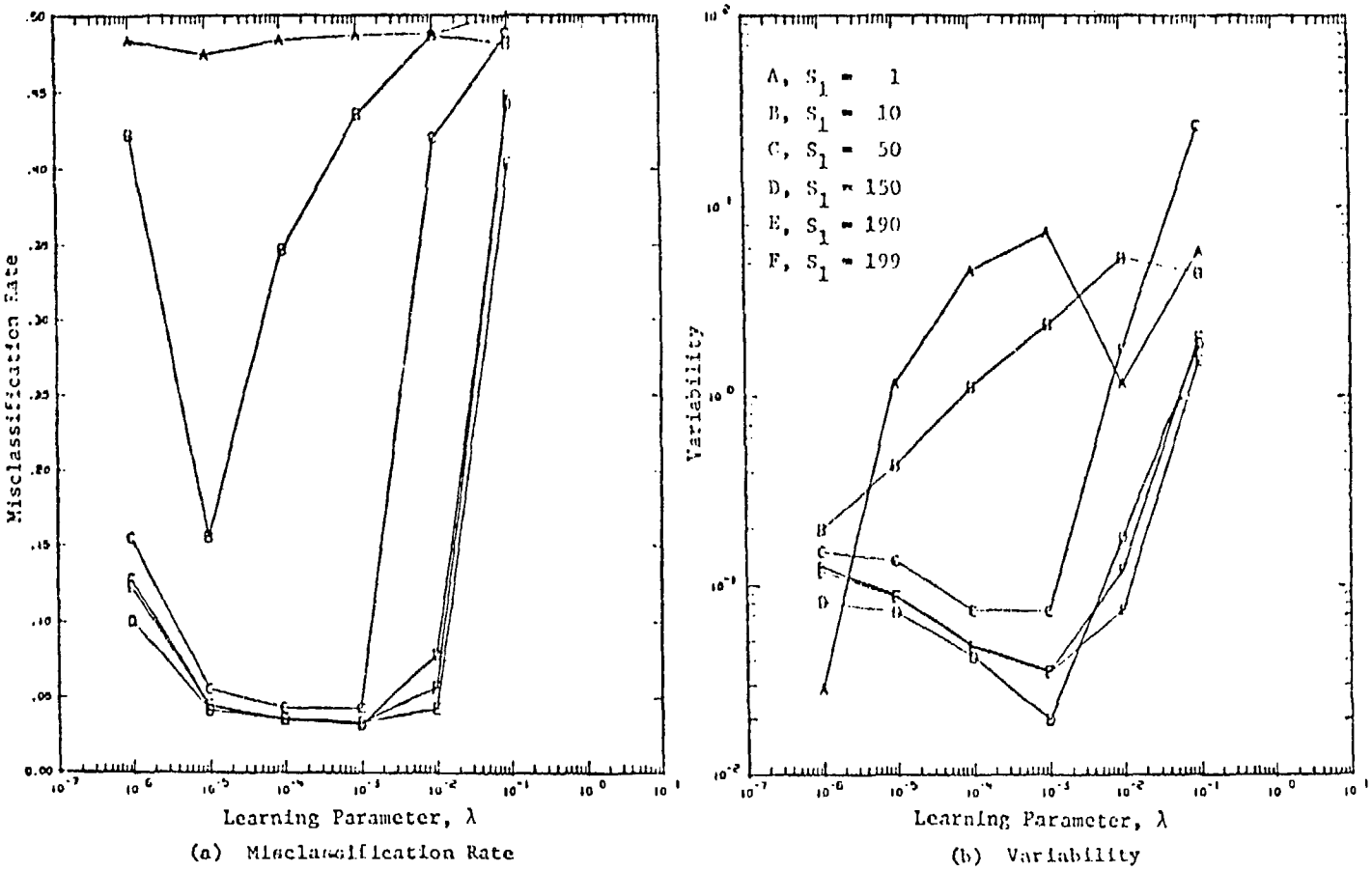


Figure 4.24 Performance Vs. λ for Various Initial Conditions, Statistical Classifier with Loss = d

to 10^{-1}. Unlike the deterministic classifier there is no value of λ for which the performance is acceptable for all S_1.

The performance as a function of the number of source particles is shown in Figure 4.25 using $\lambda = .0001$ (for $S_1 = 50, 150, 190, 199$) and $\lambda = .00001$ (for $S_1 = 1, 10$). The behavior of the curves for $S_1 < 100$ can be explained as follows. When S_1 is very small, ΔS_j is also very small as explained in the previous analysis of Equation 4.14. For many iterations there is little change in S_j and thus the misclassification rate remains approximately constant (see $S_1 = 1, 10$ curves of Figure 4.25a). Although ΔS_j is small, the variability increases since S_j is also very small (see $S_1 = 1, 10$ curves of Figure 4.25b). Eventually w_1 is decreased to such an extent that $1/w_1$ and $1/w_1^2$ begin to increase rapidly, resulting in large ΔS_j which eventually leads to a large overshoot of the optimum surface location. The magnitude of the overshoot and the following oscillations about the optimum decision surface location depends upon the value of S_1. The variability is a good indicator of when this overshoot occurs. Figure 4.25b illustrates that for: (a) $S_1 = 50$, the overshoot occurs between 20 and 40 source particles (b) $S_1 = 10$, although S_j is increasing rapidly the overshoot does not occur until after 200 source particles (c) $S_1 = 1$, S_j has not even begun its approach to the overshoot.

The primary reason for this difficulty is that Equation 4.14 is not dependent upon the location, x or $|S_j - x|$, of the prototype but is strongly dependent on the current location of the decision surface, S_j. This problem can be alleviated by using a λ that varies with the

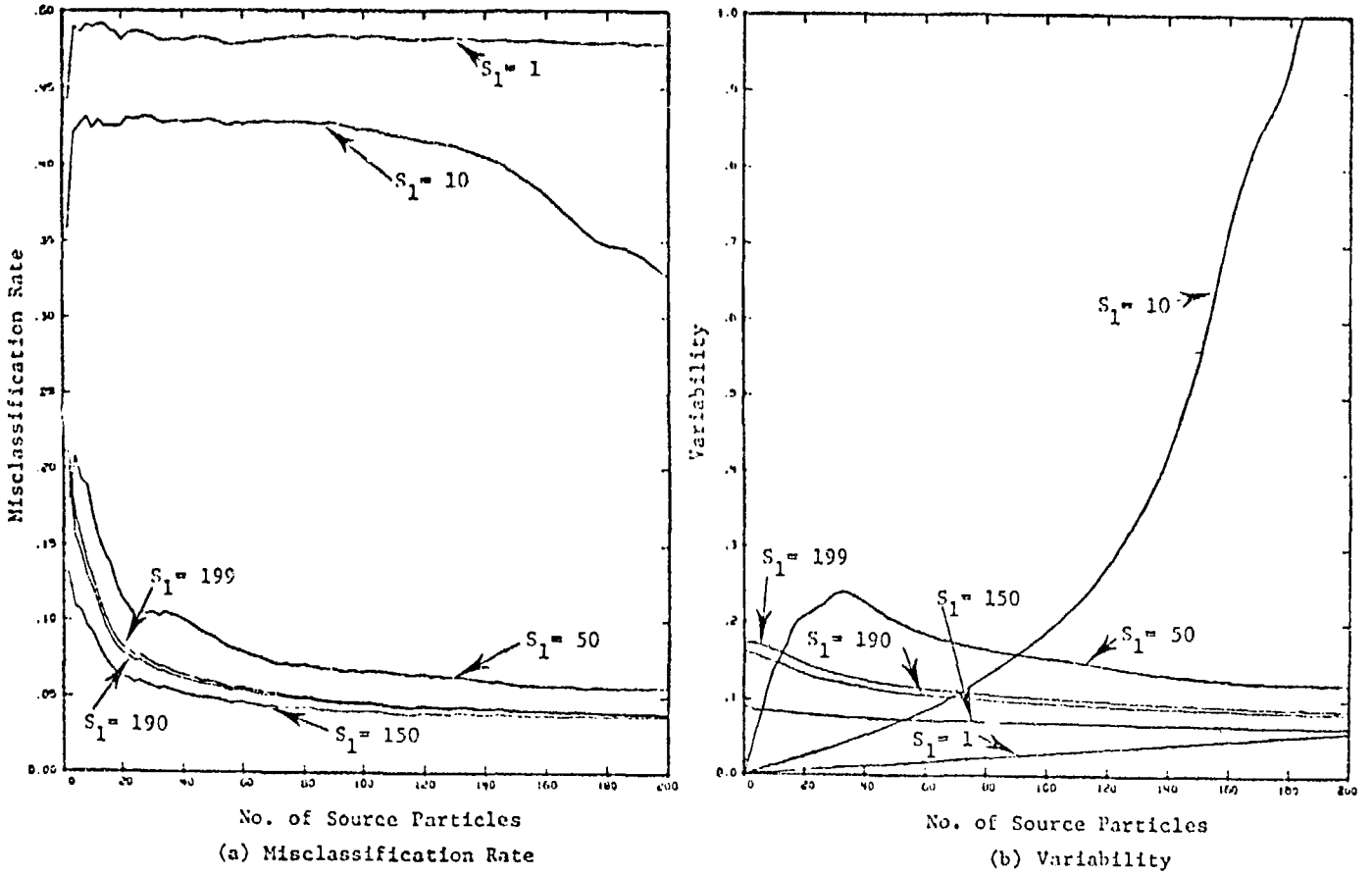


Figure 4.25 Performance Vs. Number of Source Particles for Various Initial Conditions, Statistical Classifier with Loss = D

location of S_j; however, this would lead to other complexities and would involve the use of more computer time. Because of this difficulty another statistical classifier is investigated which uses a different loss function.

4.3.1c) Statistical Classifier with Loss = D. The derivation of S_{j+1} with loss = D (see Table 4.4) is similar to that of S_{j+1} using loss = d (see Appendix K) and results in

$$S_{j+1} = \left\{ \frac{S_j w_1 + \lambda \left[T_2 - \frac{A_{k,2}}{(M_k+1)} + \frac{(-1)^{k+1}}{(M_k+1)\sqrt{1+x^2}} \right]}{w_1 - \lambda \left[T_1 - \frac{A_{k,1}}{(M_k+1)} + \frac{x(-1)^{k+1}}{(M_k+1)\sqrt{1+x^2}} \right]} \right\}_j , \qquad (4.19)$$

$$\text{where } A_{k,2} = \frac{(-1)^{k+1} \sum_{n=1}^{N_k} \left(\frac{1}{\sqrt{1+x^2}} \right)_n}{M_k} ,$$

$$A_{k,1} = \frac{(-1)^{k+1} \sum_{n=1}^{N_k} \left(\frac{x}{\sqrt{1+x^2}} \right)_n}{M_k} , \text{ and}$$

$$x = y_1 = \text{distance from the source,}$$

and all other parameters are identical to those found in Equation 4.14. Unlike the statistical classifier with loss = d, S_{j+1} for this classifier depends on the location of the prototype, x, as well as the current position of the decision surface, S_j. The dependence of S_{j+1} on an initial selection of w_1 can be eliminated by keeping λ/w_1 constant.

When past misclassification effects are large

$$|T_2| >> \left| \frac{-A_{k,2}}{(M_k+1)} + \frac{(-1)^k x}{(M_k+1)\sqrt{1+x^2}} \right| \quad \text{and}$$

$$|T_1| >> \left| \frac{-A_{k,1}}{(M_k+1)} + \frac{(-1)^{k+1}}{(M_k+1)\sqrt{1+x^2}} \right| \quad ,$$

and Equation 4.19 can be reduced the same as for the loss = d classifier resulting in Equation 4.16. Thus for the initial approach to the decision surface, this classifier behaves the same as the loss = d classifier.

When the decision surface is converging to a final surface

$$|T_2| << \left| \frac{-A_{k,2}}{(M_k+1)} + \frac{(-1)^k x}{(M_k+1)\sqrt{1+x^2}} \right| \quad \text{and}$$

$$\left| T_1 \right| << \left| \frac{-A_{k,1}}{(M_k+1)} + \frac{(-1)^{k+1}}{(M_k+1)\sqrt{1+x^2}} \right| .$$

Through the assumptions that

$$\left| \frac{-A_{k,1}}{(M_k+1)} \right| << \left| \frac{(-1)^{k+1}}{(M_k+1)\sqrt{1+x^2}} \right| \quad \text{and}$$

$$\left| \frac{-A_{k,2}}{(M_k+1)} \right| << \left| \frac{(-1)^{k}x}{(M_k+1)\sqrt{1+x^2}} \right| ,$$

Equation 4.19 can be reduced to

$$\Delta S_j \approx \frac{\lambda(-1)^{k+1}(1+S_j x)}{w_1(M_k+1)\sqrt{1+x^2} - \lambda(-1)^{k+1}x} \tag{4.20}$$

and under the further assumptions that

$$S_j x >> 1, \sqrt{1+x^2} \approx x, \text{ and } \lambda << \left| w_1(M_k+1) \right|$$

it can be reduced to

$$\Delta S_j \approx \frac{\lambda(-1)^{k+1} S_j}{w_1(M_k+1)} \tag{4.21}$$

A comparison of Equation 4.21 with Equation 4.18 shows that using loss = D linearizes the dependence of ΔS_j on S_j for the case when the decision surface is undergoing the final convergence. However, Equation 4.21 indicates that the dependence of ΔS_j on S_j on x (as shown by Equation 4.19) cancels out (based upon the assumptions made to obtain Equation 4.21). A plot of Equation 4.19 for $A_{k,i} = M_k = 0$ is shown in Figure 4.26 for three values of w_1 and $\lambda = .05$. The discontinuity of these curves is caused by the $-\lambda$ term in the denominator of Equation 4.20 and becomes less pronounced as $w_1(M_k+1) >> \lambda$.

Plots of Equation 4.19 for various values of $A_{k,i}$, M_k, etc. are not presented here but are very similar to the plots of Figure 4.23c and 4.23d. This is because, like the loss = d classifier, the coefficient of S_j in Equation 4.21 is small compared with the coefficient of S_j in Equation 4.16.

The performance of the loss = D classifier is shown in Figure 4.27 for several S_1 and a range of λ, $10^{-6} \leq \lambda \leq 10$. Although the misclassification rate is similar to the loss = d case, the variability for small S_1 has increased considerably.

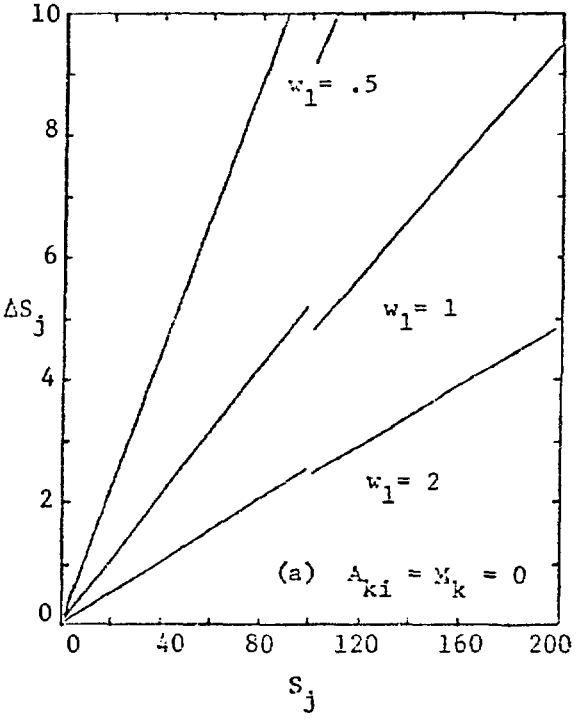


Figure 4.26 Behavior of S_{j+1} for the Statistical Classifier with Loss = D

Figure 4.28 shows the performance for the first 200 source particles using $\lambda = .05$. The extreme oscillations for the $S_1 = 1$ curve can be explained as follows. During initial operation, w_1 is initially decreased. Like the loss = d classifier as w_1 decreases, Δw_1, increases. This positive feedback can lead to an instability as $w_1 \to 0$ and w_1 can actually become negative. In this case S_j changes sign which leads to a very large variability as seen by Figure 4.28b.

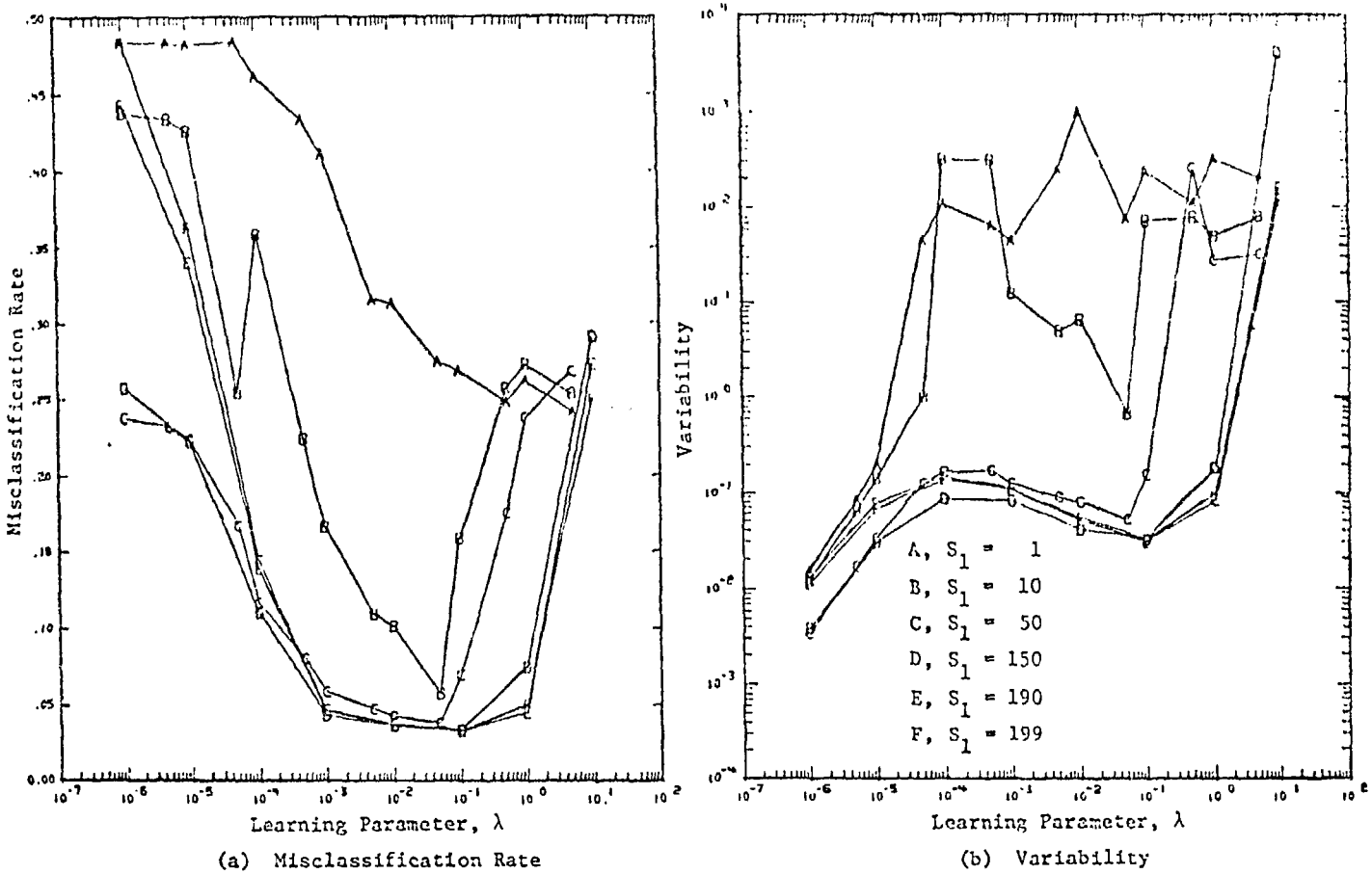


Figure 4.27 Performance Vs. λ for Various Initial Conditions, Statistical Classifier with Loss = D

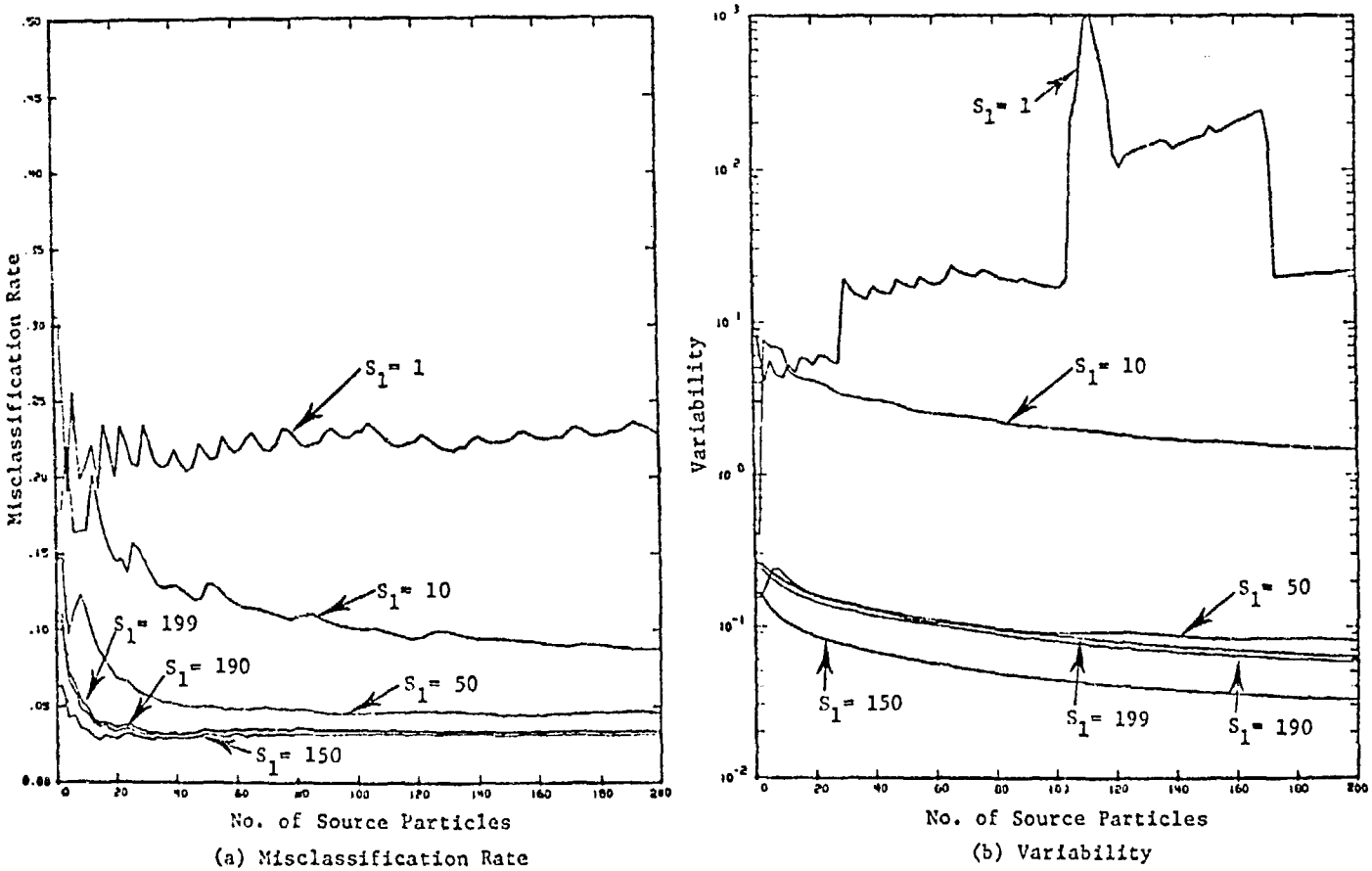


Figure 4.28 Performance Vs. Number of Source Particles for Various Initial Conditions, Statistical Classifier with Loss = D

In summary, although by using loss = D, ΔS_j, has been made to depend on x, the dependence is very weak. The classifier is therefore no improvement over the loss = d classifier for small S_1.

4.3.1d) Summary. It has been found in this section that the performance of the various classifiers does depend strongly on the choice of an initial S_1. The deterministic classifier is far superior to the statistical ones with regard to initial condition selection below the final decision surface. However, when S_1 is chosen above the final surface, the deterministic and statistical techniques are competitive.

It has also been found that the statistical techniques do depend on w_1. However, by using an appropriate (for loss = d, λ/w_1^2 = constant; for loss = D, λ/w_1 = constant), the effect of selecting an initial w_1 can be eliminated.

It should be noted that although the statistical classifier appears to perform poorly for small S_1, this effect can be eliminated by using the distance from the detector $(L - x)$ instead of the distance from the source (x) in the feature vector prototypes. This would necessitate changing the initial weight to correspond to the new feature space. This will be done for the two-dimensional problem of Section 4.4. This new feature selection would allow the initialization of splitting surfaces at the origin ($S_j = 0$); however, it would prohibit originating at $S_j = L$.

4.3.2) Multi-Region Slab Variations

In this section, three different classifiers (deterministic, statistical with loss = d, and statistical with loss = D) will be used to identify splitting planes or decision surfaces for the four Monte Carlo problems described in Table 4.6. The median importances of the distributions shown in Figure 4.18 will be used to separate the classes resulting in the class distributions shown in Figure 4.19. In the previous section it was found that a high value for the initial decision surface is better than a low one. Therefore an S_1 of 200 will be used in all cases.

Because of the different types of distributions that can occur (see Figures 4.18 and 4.19) the optimal λ for different problems varies (see Figures 4.29 and 4.30). If two classes are entirly non-overlapping, a λ which is large enough to compensate completely for a misclassified prototope is desirable since no prototype will ever appear to belong to more than one class. However, as classes begin to overlap, the amount of correction and thus λ must be decreased because falsely labeled prototypes could lead to a large variability of the decision surface. This phenomonon is apparent from Figures 4.29 and 4.30 which show the results after 100 particles. In the deterministic case the misclassification rate decreases for increasing λ, but the variability increases. In Cases II and III the overlapping of classes results in a high variability which is understandable since the deterministic classifier is not designed for overlapping classes. The statistical classifier leads to a large improvement in variability for Cases II and III but only a

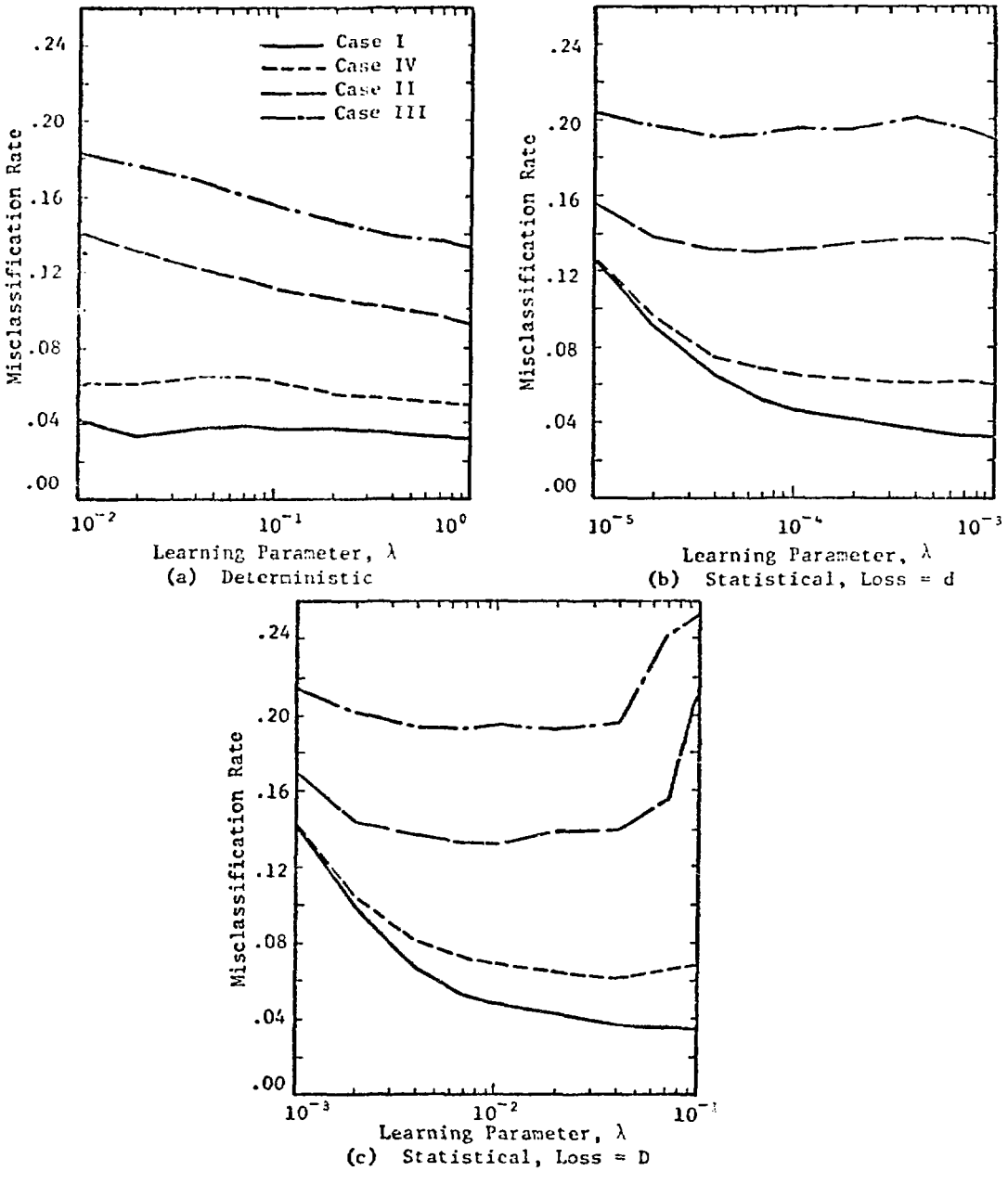


Figure 4.29 Misclassification Rate Vs. λ for Four Different Multi-Region Problems

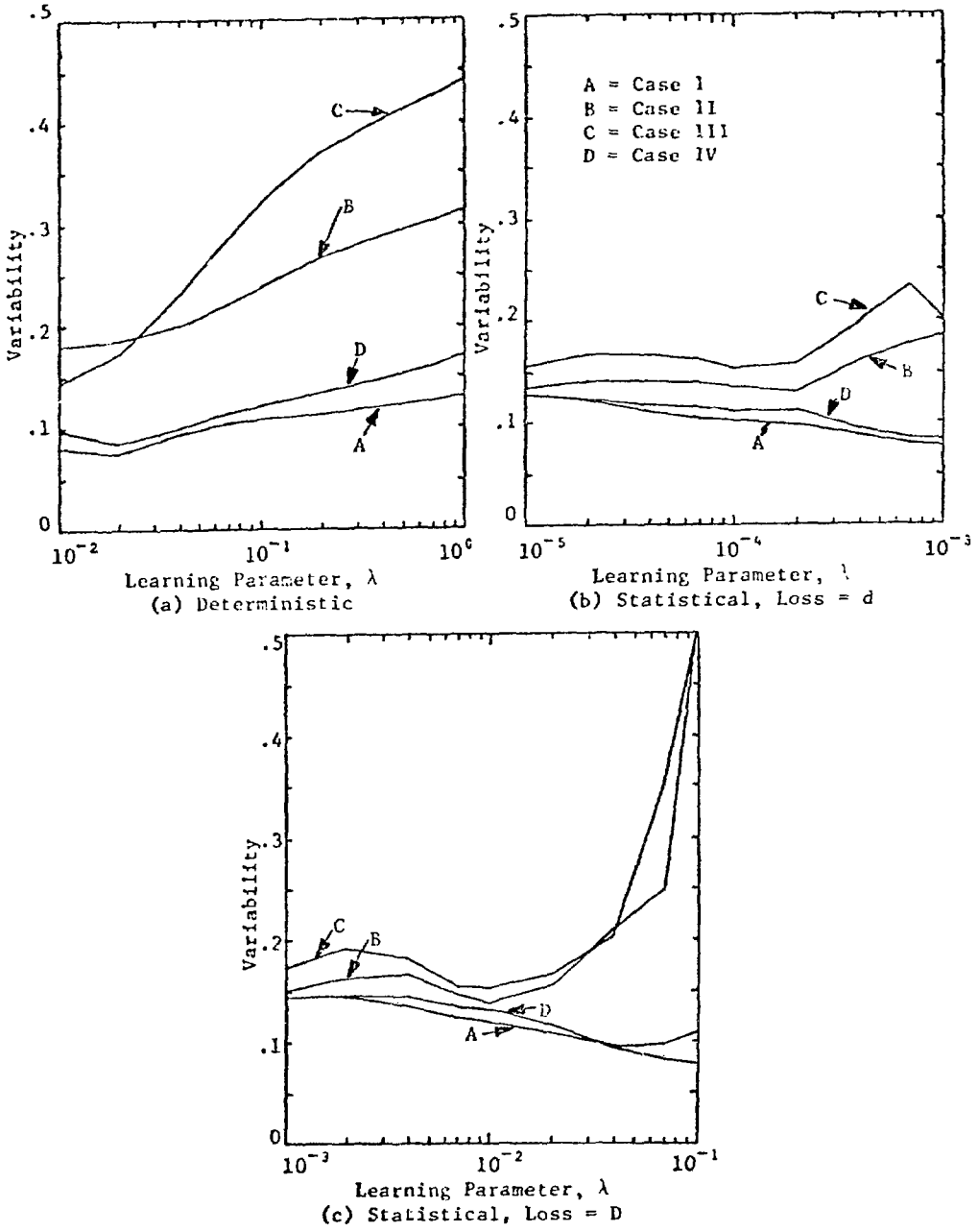


Figure 4.30 Variability Vs. λ for Four Different Multi-Region Problems

small improvement for Cases I and IV. Of the two statistical techniques the loss function using d results in slightly better performance although the difference is small.

Because a pattern classifier must be able to operate on a large range of problems without alteration, a single λ must be chosen for each classifier. For the purpose of this investigation, λ's of 10^{-1}, 10^{-4}, and 10^{-2} will be used for the deterministic, statistical (loss = d), and statistical (loss = D) classifiers respectively. Although this selection has not been optimized, these values are at least a reasonable compromise as is seen from Figures 4.29 and 4.30 for decreasing the error while keeping the increase in variability down.

For these values of λ, the performance for the first 100 particles is shown in Figures 4.31 and 4.32. It is interesting to note that in all cases the deterministic classifier leads to a smaller misclassification rate. Because the prototypes in these problems are presented in the order that they occur (i.e., for a single source particle $x_1 < x_2 < x_3 \ldots < x_N$, where N is the number of collisions) and because the deterministic classifier responds only to a single prototype at a time, it has an advantage over the statistical technique. For example, consider the following prototypes and a current $S_i = 100$:

$$i = \ldots 9,\ 10,\ 11,\ 12,\ 13,\ 14,\ 15\ldots$$

$$x_i = \ldots 75,\ 83,\ 91,\ 97,\ 109,\ 114,\ 119\ldots$$

$$\text{Class} = \ldots C_1,\ C_2,\ C_2,\ C_2,\ C_2,\ C_2,\ C_2\ldots$$

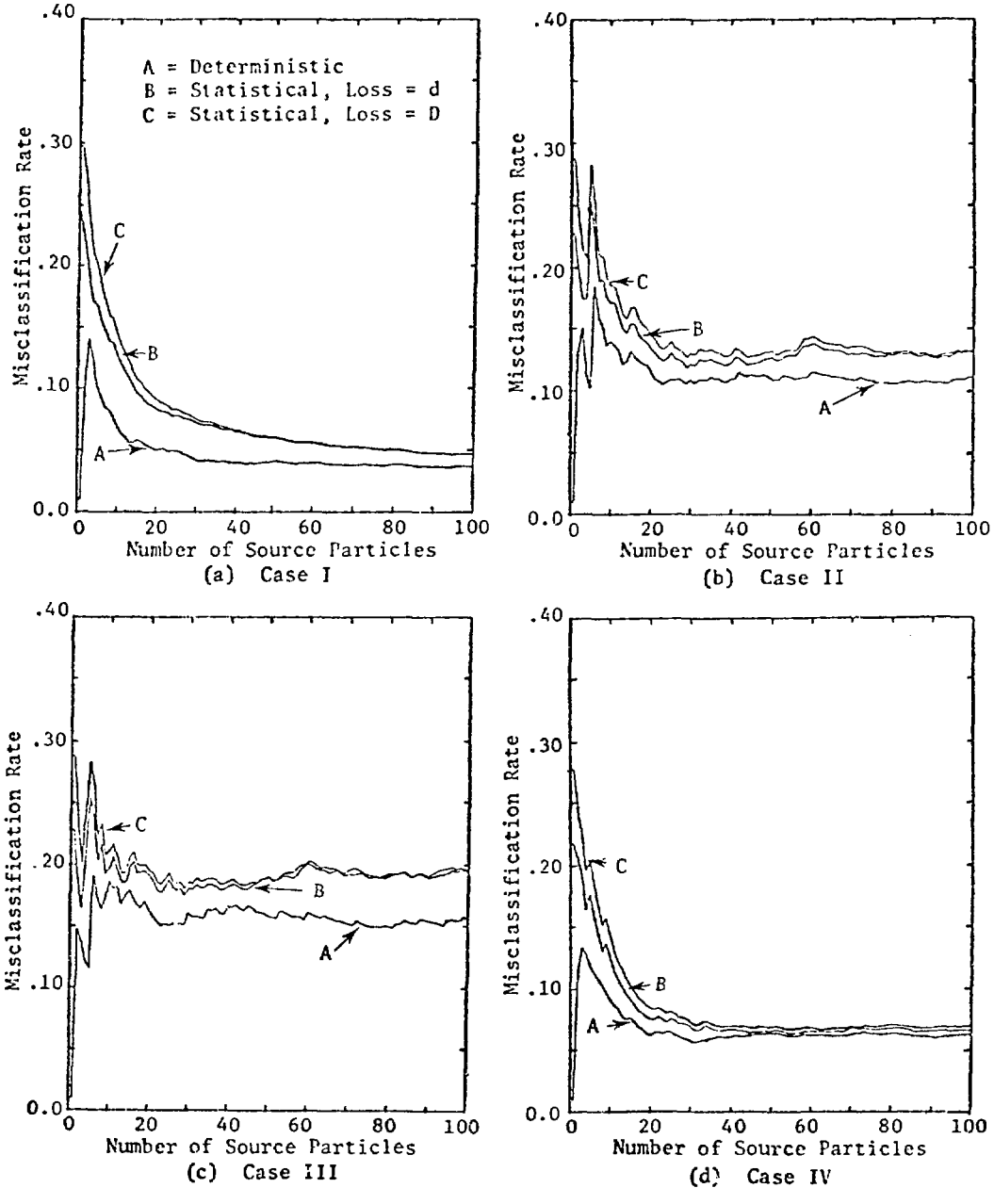


Figure 4.31 Misclassification Rate Vs. Number of Source Particles for Four Different Multi-Region Problems

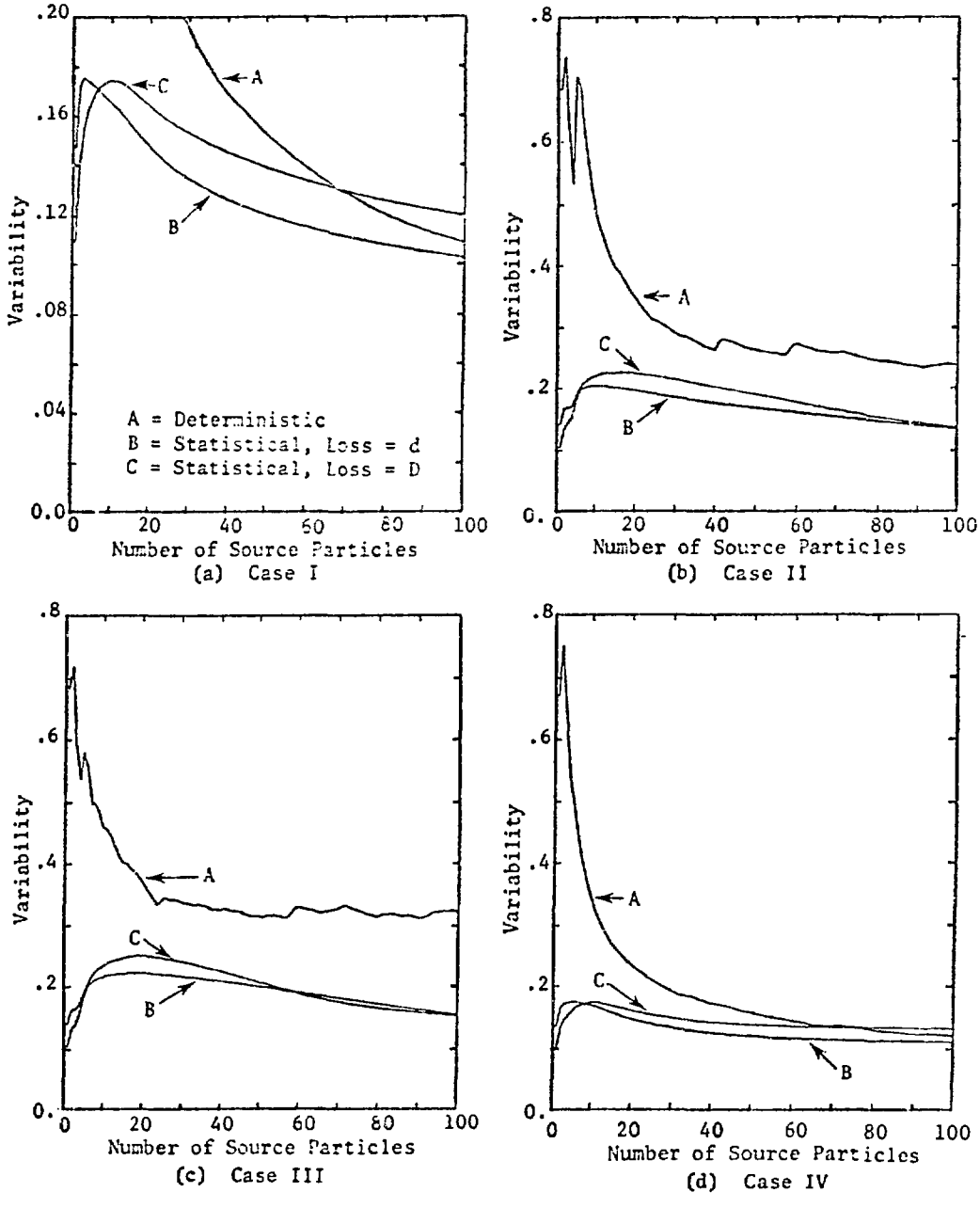


Figure 4.32 Variability Vs. Number of Source Particles for Four Different Multi-Region Problems

If $\lambda = 1$, and x_{10} is presented to the deterministic classifier, the new $S_{j+1} = 83$ which in turn makes x_{11} and x_{12} correctly classified. For $\lambda < 1$, the deterministic classifier still has this effect to some degree. The statistical classifier may or may not decrease S_j in this case depending on the value and sign of T_1 and T_2 (see Equations 4.14 and 4.15). Although this appears to be an advantage of the deterministic classifier, it may actually be a disadvantage since the objective is to minimize the number of misclassified prototypes according to their average importances and not the importance encountered for each prototype. Thus the misclassification rate shown in Figure 4.29 may be a "false misclassification" rate since this is the misclassification rate as seen by observing prototypes one at a type instead of looking at the average importance of each prototype. This phenomenon is often referred to as "learning from a teacher who makes mistakes" since the classifier must use prototypes which are misclassified part of the time.

In summary, the deterministic classifier performs better for non-overlapping or slightly overlapping classes. However, as the overlapping increases statistical techniques become increasingly more attractive because of their lower variability.

4.4) Distance and Angle, A Two-Dimensional Problem

In previous sections of this chapter the pattern space has consisted of one dimension--distance. In this section the Monte Carlo problem consists of the multi-region problem shown in Figure 4.17 except that after each collision, a scattering angle is calculated

(see Figure 4.33). For the purposes of this research, scattering is assumed isotropic in the laboratory system. The FORTRAN coding used for the calculations in this section is shown in Appendix H.

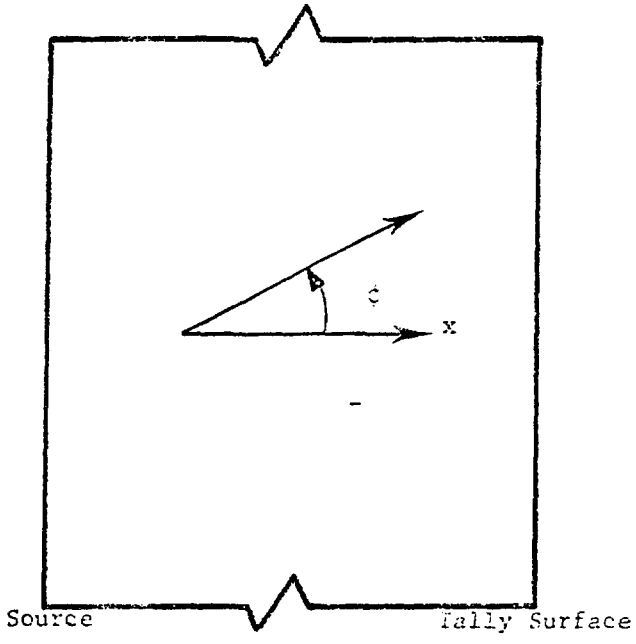


Figure 4.33 The Two Dimensional Monte Carlo Problem

Because the number of regions in the slab does not affect the structure of the pattern classifier, a single region will be used with $\Sigma_t = .5$ and $\Sigma_s = .4$. A slab thickness of five mean-free-paths ($L = 10$) is chosen for this example in order to keep computer time at a minimum while still maintaining the characteristics of a two-dimensional problem.

A significant difference between this problem and the previous one-dimensional problems for which $L = 200$ is the number of prototypes (equal to the number of collisions). The one-dimensional problem creates 100 prototypes per source particle, whereas the two-dimensional problem of this section produced only 11. Consequently the two-dimensional problem investigated in this section requires more source particles to learn a splitting surface. An important characteristic of the two-dimensional problem is that particles are allowed to escape from the system without contributing to the tally. As a result the distribution of track importances consists of a distribution of importances created by non-zero tallies plus a number of track importances equal to zero. This is a common feature of the majority of Monte Carlo problems. Because of this, one wishes to locate splitting surfaces so as to get particles from regions of zero importance into regions of non-zero importance. Therefore, the $\bar{I}$ described in the introduction to this chapter is equal to zero.

In the sample problem of this section approximately 59% of all tracks have zero importance. The distribution of the remaining 41% is shown in Figure 4.34a. The probability distribution of prototypes, $p(Y)$, as a function of x is shown in Figure 4.34b for three different contours through x-ϕ space and in Figure 4.34c as a function of ϕ. The decreasing probability of prototypes with increasing x is a result of fewer particles penetrating the slab and thus fewer prototypes occur deep within the slab. The flat distribution with respect to $\cos\phi$ results from isotropic scattering.

Given the feature vector $Y=[x\ \phi]$, the probability that Y belongs to class C_i is given by $p(C_i|Y)$, (see Section 3.3). Plots of $p(C_i|Y)$ for the same contours used in Figure 4.34b and c are shown in Figure 4.35. The symmetry of the class distributions is due to isotropic scattering and the single material region. The intersections of the distributions define the decision surface. A particle with coordinates on this decision surface has an equal probability of being tallied or escaping. Unlike the one-dimensional problem the probability densities of each class, $p(Y|C_i)$ are not proportional to $p(C_i|Y)$ since $p(Y)$ is not constant. The misclassification rate due to the overlapping of $p(C_i|Y)$ is 29% (i.e., Er = 29 where Er is defined by Equation 4.5).

In Section 4.2.3 normalization was achieved by multiplying the feature vector Y by 1/L and altering the adjustment algorithms accordingly. This is not possible for two-dimensional pattern space since both variables ϕ and x must be normed with respect to each other. This is

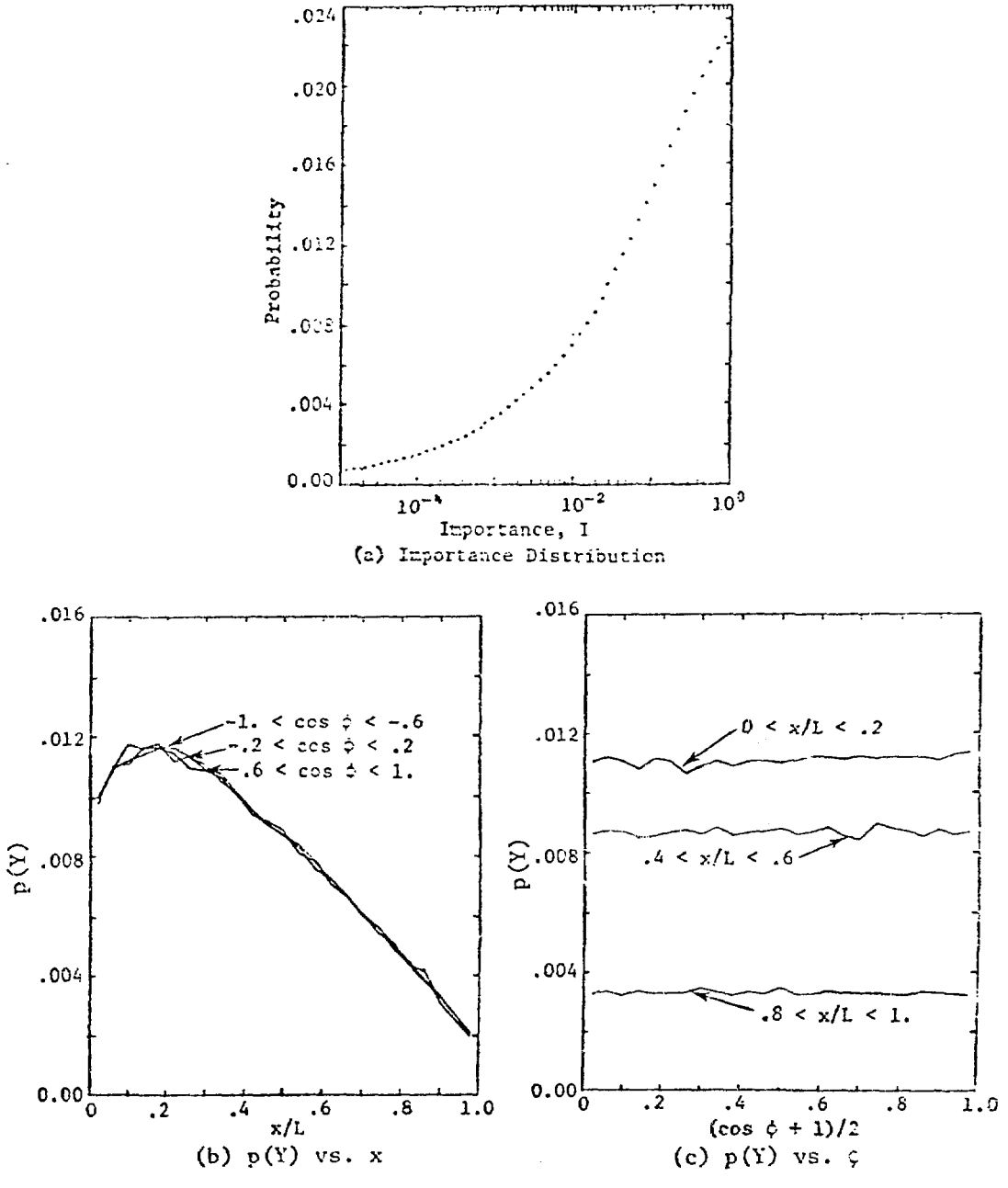


Figure 4.34 Importance and Prototype Distributions for the Two Dimensional Monte Carlo Problem

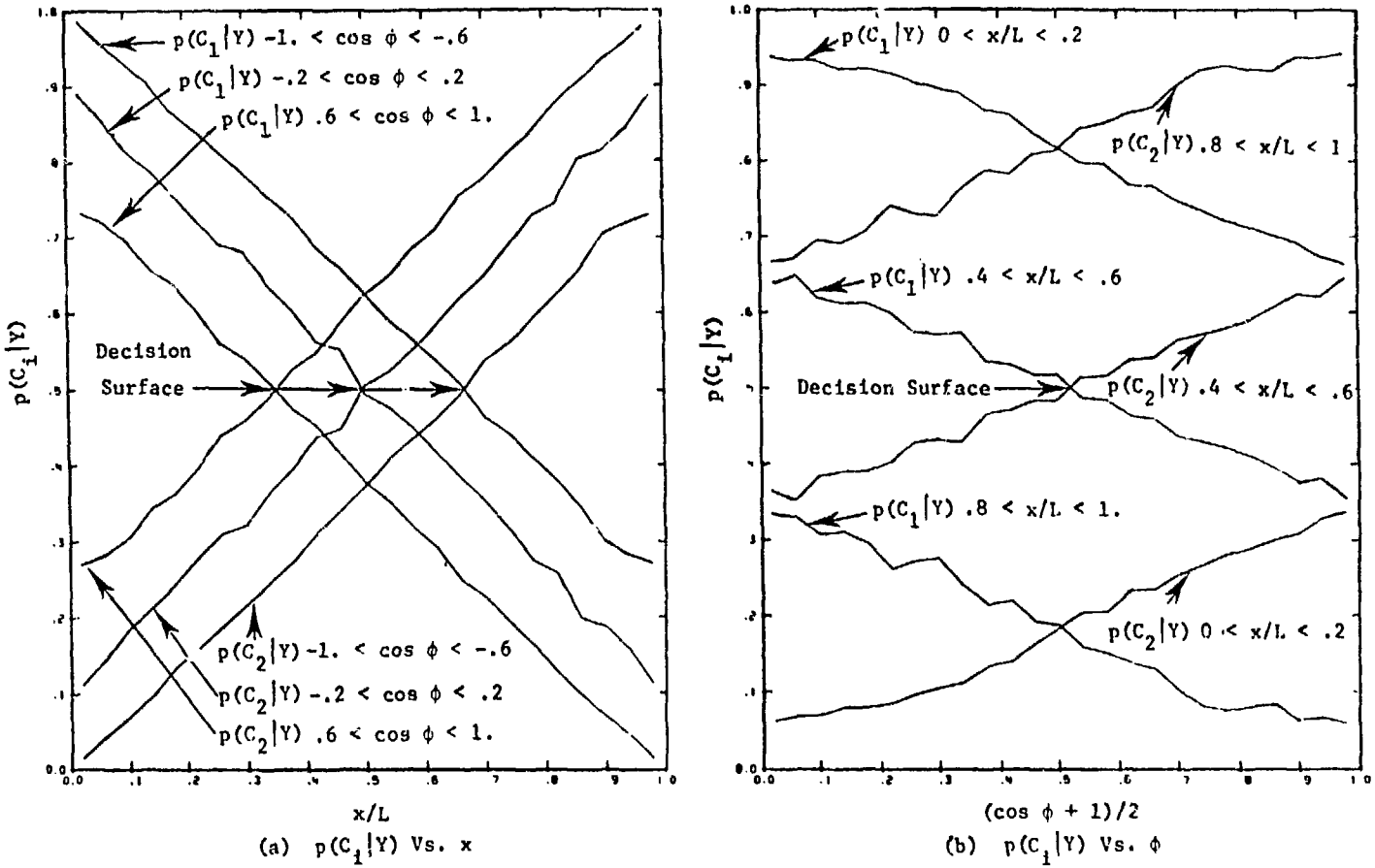


Figure 4.35 Class Distributions for the Two Dimensional Monte Carlo Problem

done by normalizing both angle and distance to one. Thus, pattern space is transformed to feature space as shown by Equation 4.22.

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x \\ \phi \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x/L \\ (\cos\phi + 1)/2 \end{bmatrix} \tag{4.22}$$

In the one-dimensional slab problem the performance of the classifier is measured in terms of the misclassification rate and the variability of the decision surface. The misclassification rate will be used unchanged; however, the variability of the decision surface (a line in two dimensions) is given by

$$\text{Variability} = \frac{\sigma_x \sigma_\phi}{\sqrt{\bar{\phi}^2 + \bar{x}^2}} \tag{4.23}$$

where $\bar{\phi}$ = the mean value of the decision surface with the y_2 $[(\cos\phi + 1)/2]$ axis

$\bar{x}$ = the mean value of the decision surface with the $y_1 (x/L)$ axis

σ_x = standard deviation of the mean $\bar{x}$

σ_ϕ = standard deviation of the mean $\bar{\phi}$

In addition to investigating learning parameter influence, this section studies the effects of initial conditions as was done in Section 4.3.1 for the one-dimensional case. For one-dimensional feature space the initial condition consists of a single variable $S_1 = -w_2/w_1$. The choice of an initial w_1 was found to have no effect provided the proper λ were selected. Thus by keeping λ = constant, an initial $w_1 = 1$ could be used without loss of generality. Figure 4.36 illustrates the decision surface for normalized two-dimensional feature space. From this figure it is seen that two parameters are required to specify the decision surface, $-w_3/w_2$ and $-w_3/w_1$. A third parameter (in this case let it be w_1) determines the slope of the discriminant function and like the one-dimensional case can be set equal to 1 provided that λ

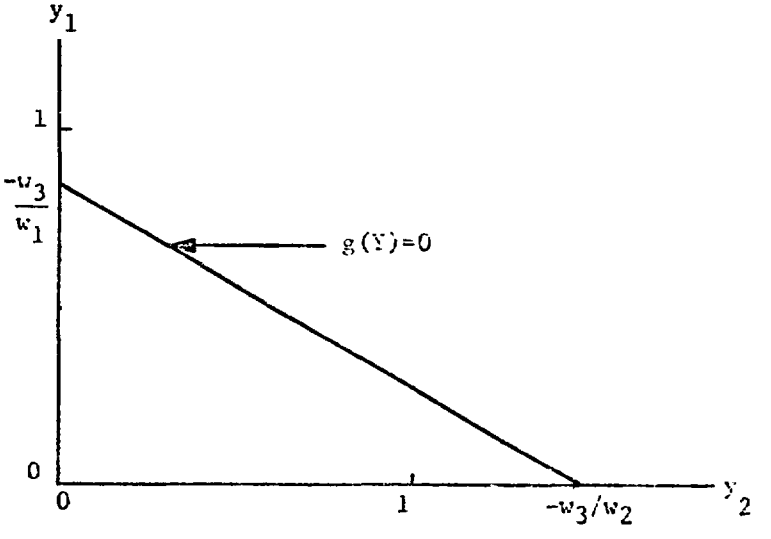
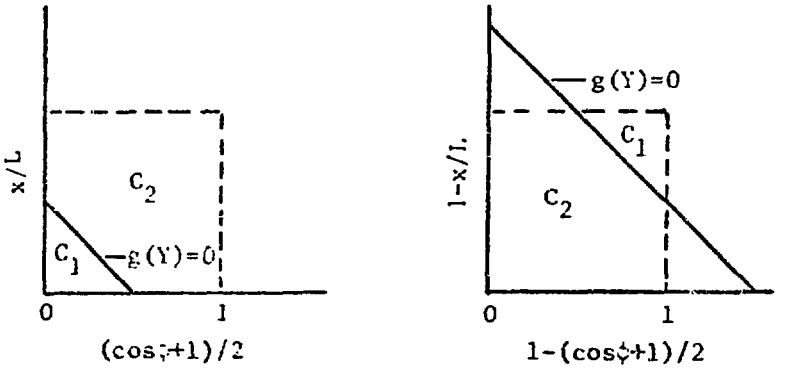


Figure 4.36 Decision Surface for the Two Dimensional Problem

remains constant. A simplifying assumption made only for the purposes of this research is to use $w_2 = w_1 = 1.0$ which results in a decision surface (line) oriented at 45° to the y_1 and y_2 axes. Using this assumption, the initial conditions consist of specifying w_3, where $-w_3$ is the y_1 and y_2 intercept.

It was demonstrated in Section 4.3.1 that an initial decision surface near the origin leads to decreased performance and it was recommended that this problem be alleviated by changing the feature x to L - x. This situation also occurs in the two-dimensional problem. Therefore, several runs will be made using $y_1 = 1 - x/L$ and $y_2 = 1 - (\cos\phi + 1)/2$. For these runs $w_1 = w_2 = -1.0$ and w_3 will be chosen to correspond to an equivalent w_3 in x/L, $(\cos\phi + 1)/2$ space. To illustrate this feature conversion two equivalent decision surfaces are shown in Figure 4.37. In the following sections a negative value of w_3 implies $w_1 = w_2 = 1$ (as in Figure 4.37a), a positive w_3 implies $w_1 = w_2 = -1$ (as in Figure 4.37b).



(a) $w_1 = w_2 = 1$, $w_3 = -.5$ (b) $w_1 = w_2 = -1$, $w_3 = 1.5$

Figure 4.37 Feature Vector Conversion

4.4.1 Deterministic Classifier

The algorithm for weight adjustment is the same as given by Equation 4.4 except that the dimension of the vectors W and Y_i^* is increased from two to three. The FORTRAN coding of this classifier is shown in Appendix H. The effect of the learning parameter, λ is illustrated in Figures 4.38a and 4.38c for several di ferent initial conditions after 5000 source particles. The behavior of the classifier as a function of λ is similar to that illustrated in Figures 4.29 and 4.30 for the problems in which the class distributions have a large amount of overlap (Cases II and III). The misclassification rate drops with increasing λ while the variability increases. The fact that the misclassification rate drops below that due to overlapping classes (29%) indicates the effect of "false misclassification" described in Section 4.3.2.

The misclassification rate and variability for runs started near the origin ($w_3 = -.25, -.05$) are higher than the other runs (see Figures 4.39a and 4.40a). When feature space is converted ($w_3 = 1.75, 1.95$) the same prototypes lead to much better performance.

4.4.2 Statistical Classifier

The correction vector for the two-dimensional problem with a loss function equal to d is given by

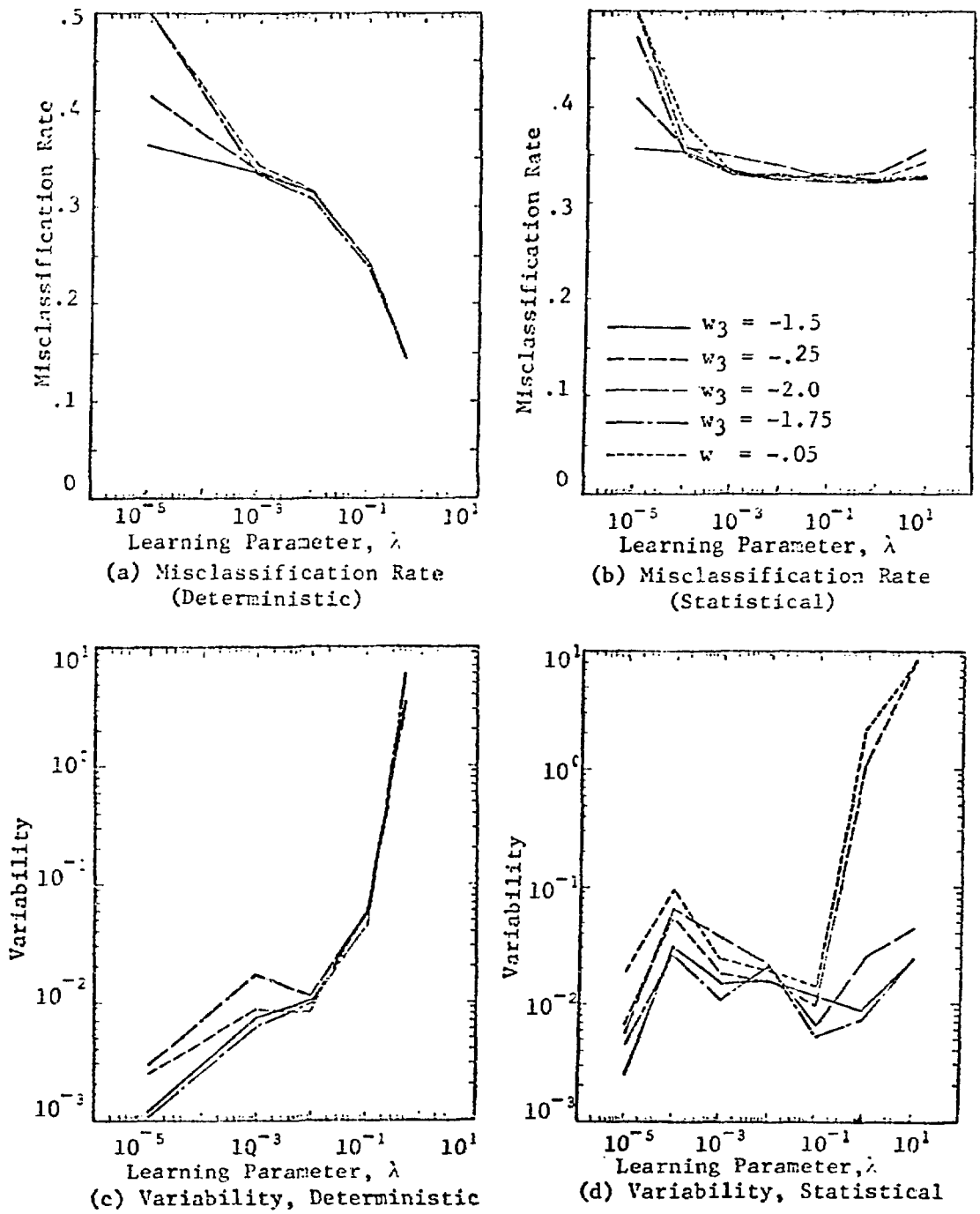


Figure 4.38 Performance Vs. λ for the Two-Dimensional Problem

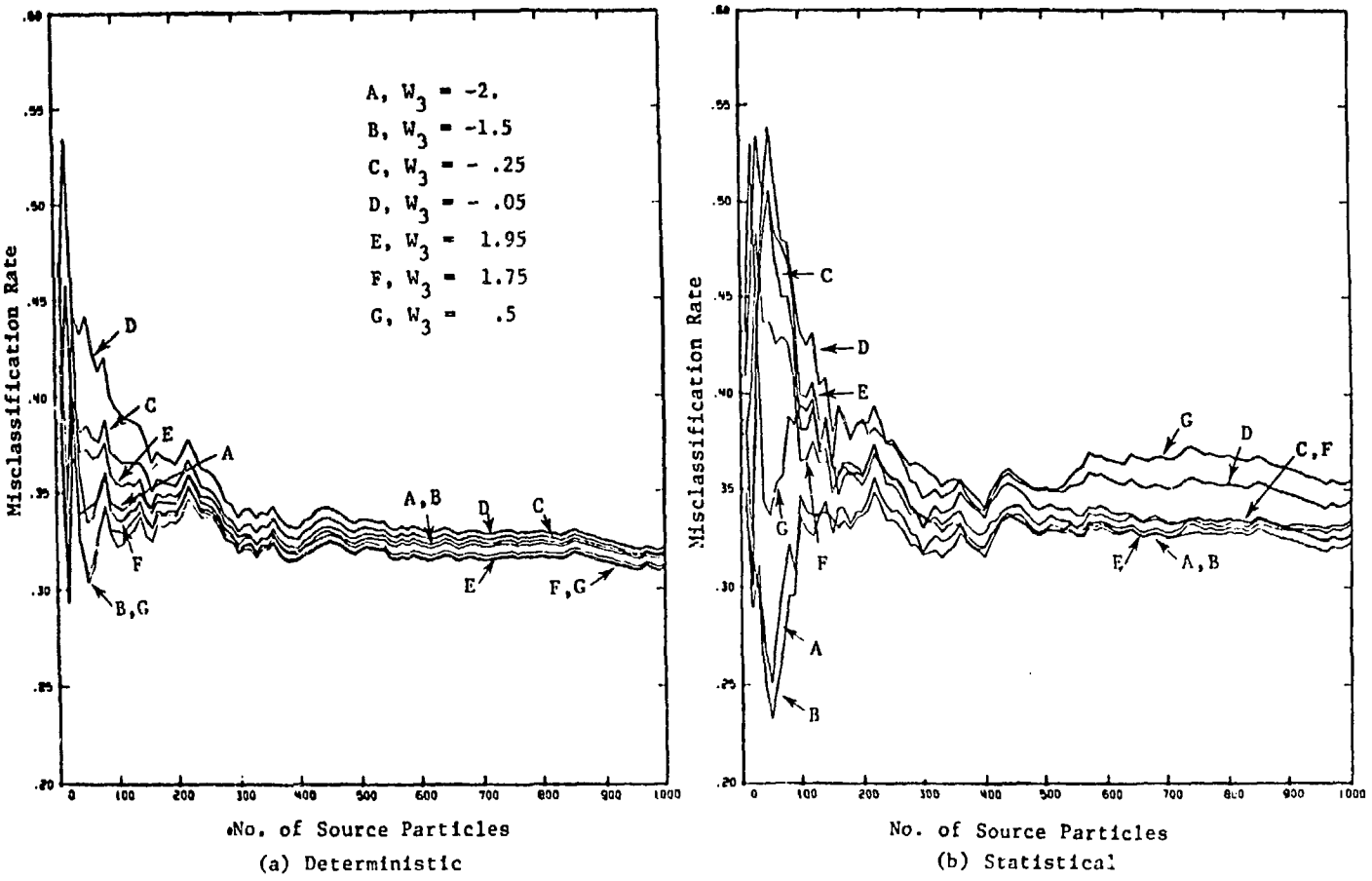


Figure 4.39 Misclassification Rate Vs. Number of Source Particles for the Two Dimensional Problem

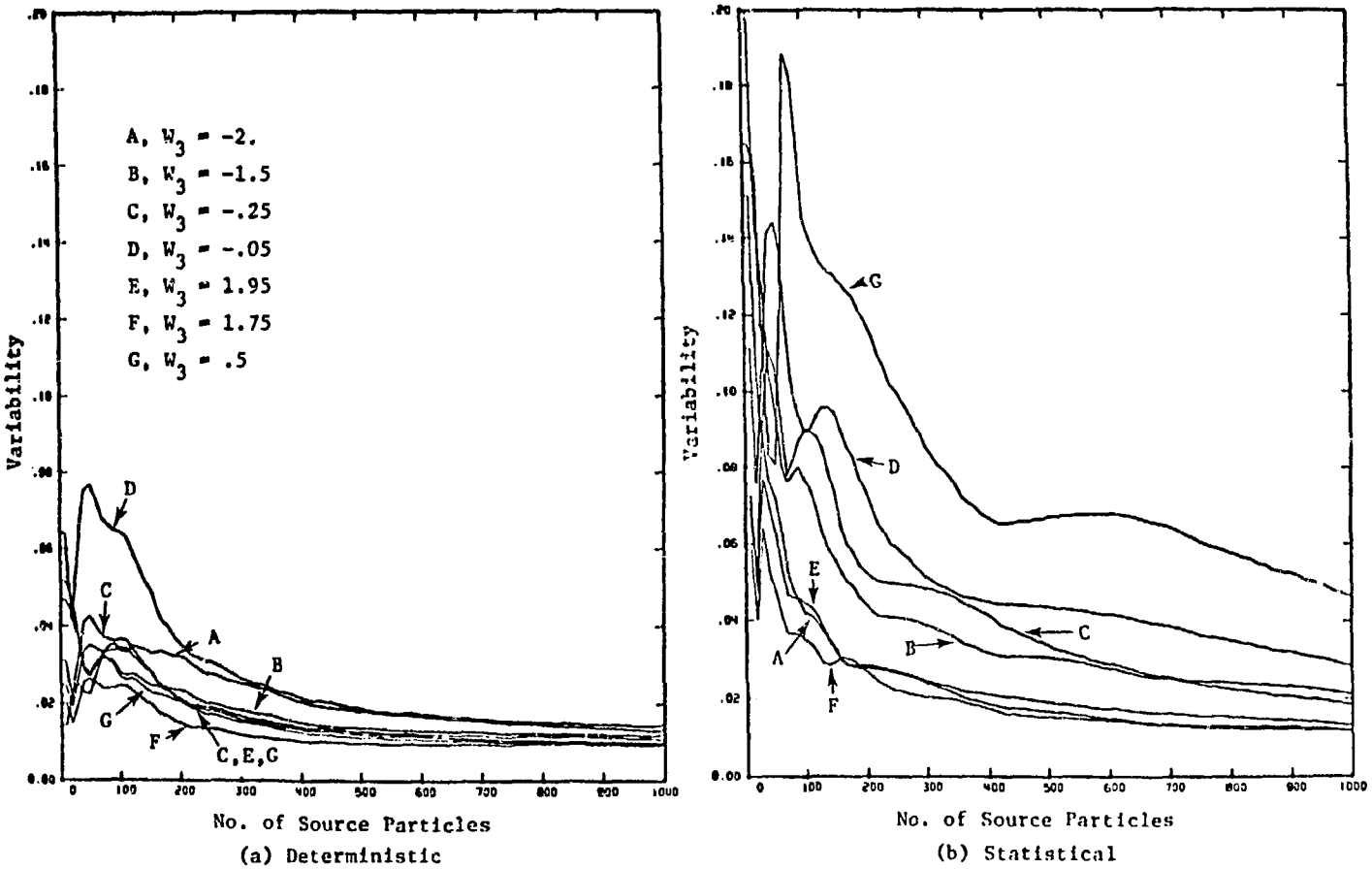


Figure 4.40 Variability Vs. Number of Source Particles for the Two Dimensional Problem

$$\nabla\tilde{R} = \begin{bmatrix} \frac{\partial\tilde{R}}{\partial w_1} \\ \\ \frac{\partial\tilde{R}}{\partial w_2} \\ \\ \frac{\partial\tilde{R}}{\partial w_3} \end{bmatrix} = \begin{bmatrix} \frac{y_1\left(w_1^2+w_2^2\right)-g(Y)}{\left(w_1^2+w_2^2\right)^{3/2}} \\ \\ \frac{y_2\left(w_1^2+w_2^2\right)-g(Y)}{\left(w_1^2+w_2^2\right)^{3/2}} \\ \\ \frac{1}{\left(w_1^2+w_2^2\right)^{1/2}} \end{bmatrix} \tag{4.24}$$

The FORTRAN coding for the statistical classifier is given in Appendix I. The misclassification rate after 5000 particles is relatively flat for λ's between 10^{-3} and 1 (see Figure 4.38b); however, the variability has a minimum at $\lambda = 10^{-1}$. Unlike the deterministic classifier, normalization affects the optimum λ. The misclassification rate (see Figure 4.39b) is more spread than the deterministic classifier due to the sensitivity of the statistical classifier to an initial decision surface near the origin. The variability is also more spread (see Figure 4.40b) since initial surfaces nearer the origin ($w_3 = -.25, -.05, .5$) performed considerably worse than their counterparts away from the origin ($w_3 = 1.75, 1.95, -1.5$). Thus the statistical classifier behaves the same as previously illustrated in one dimension.

4.4.3) Summary

Aside from the normalization procedure the two-dimensional problem behaves similar to the one dimensional case. Once again the classi-

fiers perform better when not operating near the origin and, as has been shown, inversion of the feature vector can be used to alleviate this problem.

The time spent by the two techniques can be compared by using Equation 4.9 and the timing parameters given in Table 4.7. The time required for normalization of features is not included in this data since it is a feature selection process and will be treated in Chapter V. The time to determine classification, t_A, is unchanged since it does not involve the dimensionality of Y. The increase in t_L is due to the additional terms necessary to calculated g(Y). The weight adjustment times increase by 100% for the statistical case and by 66% for the deterministic case. The example times given below are for 300 source particles (3,590 prototypes), an initial $w_3 = 1.95$ (see Figures 4.39 and 4.40, runs E) and no buffer zones.

	Time in Seconds	
	Statistical	Deterministic
t_C	21.5×10^{-6}	5.8×10^{-6}
t_L	1.7×10^{-6}	1.7×10^{-6}
t_A	4.1×10^{-6}	4.1×10^{-6}

Table 4.7 Timing Parameters for the Two Dimensional Problem

Deterministic ($f_c = .327$)

$$T = 1.47 \times 10^{-2} + .61 \times 10^{-2} + .68 \times 10^{-2} = 2.76 \times 10^{-2}$$

Statistical ($f_c = .330$)

$$T = 1.47 \times 10^{-2} + .61 \times 10^{-2} + 2.55 \times 10^{-2} = 4.63 \times 10^{-2}$$

Unlike the one dimensional case of Section 4.2.3 the difference between the two techniques is significant. This difference is due primarily to the increase in the amount of class overlap (from 4% to the 1-D case to 29% for 2-D) which causes the effect of t_c to be more pronounced in the two-dimensional case.

4.5 Summary

In this chapter statistical and deterministic classifiers have been used to learn splitting surfaces for several different Monte Carlo problems. The conclusions drawn from these numerical experiments are listed below for each topic.

Slab Thickness and Class Overlapping. It was found in Section 4.2 that the smaller the distance variable (in this case slab thickness) is in terms of mean free paths, the greater is the amount of class overlapping. Because class overlapping decreases classifier performance, geometries of many mean free paths are easier problems as far as pattern recognition is concerned.

Buffer Zones. Buffer zones decrease the misclassification rate of the classifier by removing prototypes which are very close to class boundaries (according to the teacher). However, if prototypes falling within the buffer zone are not used as prototypes, the variability can be increased since the variability is proportional to the inverse of the number of prototypes. Therefore, prototypes within the buffer zone should be considered as correctly classified prototypes. A further benefit of buffer zones is that they decrease the amount of computer time necessary for pattern recognition.

Loss Functions. Although there was no great difference in performance for the loss functions used, loss = d appears to be the most attractive since it is the simplest computationally.

Learning Parameter, λ. It is desirable to use a single λ for all problems since this greatly simplifies the classification process. As has been seen in previous sections a range of λ does exist for both statistical and deterministic classifiers over which the performance is relatively constant. If a λ below this range is used, the convergence of the classifier is slowed down requiring additional time for the misclassification rate to decrease sufficiently. A λ above this range causes the variability to increase although it may actually decrease the misclassification rate. This range is uaffected by normalization for the deterministic classifier but is strongly affected for the statistical classifier. The following appear to be reasonable values of λ that are suitable for a wide range of problems.

Deterministic $.01 \leq \lambda \leq .1$

Statistical (normalized to 1.0) $.01 \leq \lambda \leq 1$

Problems with a large amount of overlap require λ's in the lower part of this range (small overlap, the upper part). Monte Carlo problems affect the choice of λ only because of differences in the amount of class overlap. Values of .05 and .1 can be used for the deterministic and statistical classifiers respectively without greatly penalizing performance generality.

Initial Conditions. Although both techniques are sensitive to the selection of an initial decision surface, this selection is far more critical for the statistical classifier. The problem can be alleviated by altering the feature vector and thus changing from the region near the origin to a region near the point $(y_1, y_2, \ldots y_n) = (1,1,\ldots,1)$. As will be seen in the next chapter decision surfaces will be started at the origin; therefore, this alteration of feature space is necessary.

Computer Time. Although programming efficiency has been overlooked in the construction of the classifiers, some conclusions can be drawn from the timing data. These values can then be used as an upper limit for the time spent in pattern classification. The time to determine prototype classification, t_A, is a major contributor to time spent on pattern classification, amounting to about 4.1×10^{-6} seconds per prototype. Using Tables 4.5 and 4.7 and extrapolating to feature space of N dimensions results in

$$t_L \approx 1.0 \times 10^{-6} + (N-1)\ .7 \times 10^{-6} \tag{4.25}$$

Performing the same extrapolation for t_C results in

$$\text{Deterministic:} \quad t_C \approx 3.5 \times 10^{-6} + (N-1)\ \ 2.3 \times 10^{-6} \tag{4.26}$$

$$\text{Statistical:} \quad t_C \approx 10.5 \times 10^{-6} + (N-1)\ \ 11. \times 10^{-6} \tag{4.27}$$

Both the one-dimensional and two-dimensional problems require about the same number of prototypes (3 to 4 thousand) until convergence. However, due to difference in the overlapping of class distributions, the one-dimensional case converges to a misclassification rate of ≈5% whereas the two-dimensional case converges to a rate of ≈30%.

Normalization. The normalization of feature space for multi-dimensional feature vectors has been accomplished by normalizing the feature space to a 1×1 coordinate system.

Deterministic vs. Statistical. Although the statistical classifier has the advantage that it is guaranteed to converge in the presence of overlapping class distributions, this advantage is not great since the decision surface will be used prior to final convergence. The deterministic classifier appears to be the most attractive because: (1) its performance (variability and misclassification rate) is usually better than the statistical classifier (2) it uses less

computer time (3) its learning parameter, λ, is not affected by normalization and (4) it is much less affected by the initial selection of a decision surface.

Chapter V. Discussion of General Application

Many practical considerations remain before the techniques of Chapter IV can be applied to a multi-purpose particle transport code. The purpose of this chapter is to describe and analyze these considerations. The first problem is that given the ability to recognize surfaces, what surfaces are desirable, and in what order should those surfaces be learned. This problem is analyzed in Section 5.1 for a single tally, the point detector tally, and multiple-tallies.

Section 5.2 describes the problems encountered in feature selection and suggests a scheme for implementing the feature selection process in a general purpose code. Section 5.3 investigates the amount of time required for pattern recognition and its related operations and its effectiveness as a variance reduction techniques. The limitations of the pattern recognition system in reducing variance are discussed in Section 5.4. Finally, Section 5.5 summarizes the chapter.

5.1) Implementation

The first requirement of implementation is to make prototypes available to the classifier. In this research prototypes are created after each collision;* however, not until after a particle is lost to

*Prototypes could also be created when entering a new geometric region; however, energy and angle variables will be unchanged from the previous prototype.

the system or is tallied can the importance be determined (this is not true for the point detector, see Section 5.1.3). Figure 5.1 illustrates when the information for the prototypes is extracted from the Monte Carlo problem and Figure 5.2 shows when pattern recognition is performed. Unlike the Monte Carlo problems used in Chapter IV, for many problems a single source particle can lead to multiple contributions to the tally. Such a case is illustrated in Figure 5.3 where the track length in region 2 is tallied. Due to scattering in region 1, it is possible for a particle to pass through region 2 (and thus be tallied) any number of times before leaving the system. Because of these multi-contributions the pattern classification loop of Figure 5.2 can be entered many times during the trace of a single source particle.

The pattern classification block can be considered to be a FORTRAN subroutine. Sections 5.1.1, 5.1.2, and 5.1.3 discuss the structure of this subroutine for a single tally, multiple tallies, and a point detector tally, respectively, using the information gained in Chapter IV.

5.1.1) Single Tally

The majority of Monte Carlo problems which require importance sampling to reduce the variance are characterized by a seldom occurring tally. Thus the majority of prototype importances will be zero. The problem with zero importance prototypes is that they contain much less information than a non-zero tally (i.e., all prototypes with zero importances look the same to the classifier whereas non-zero importances can

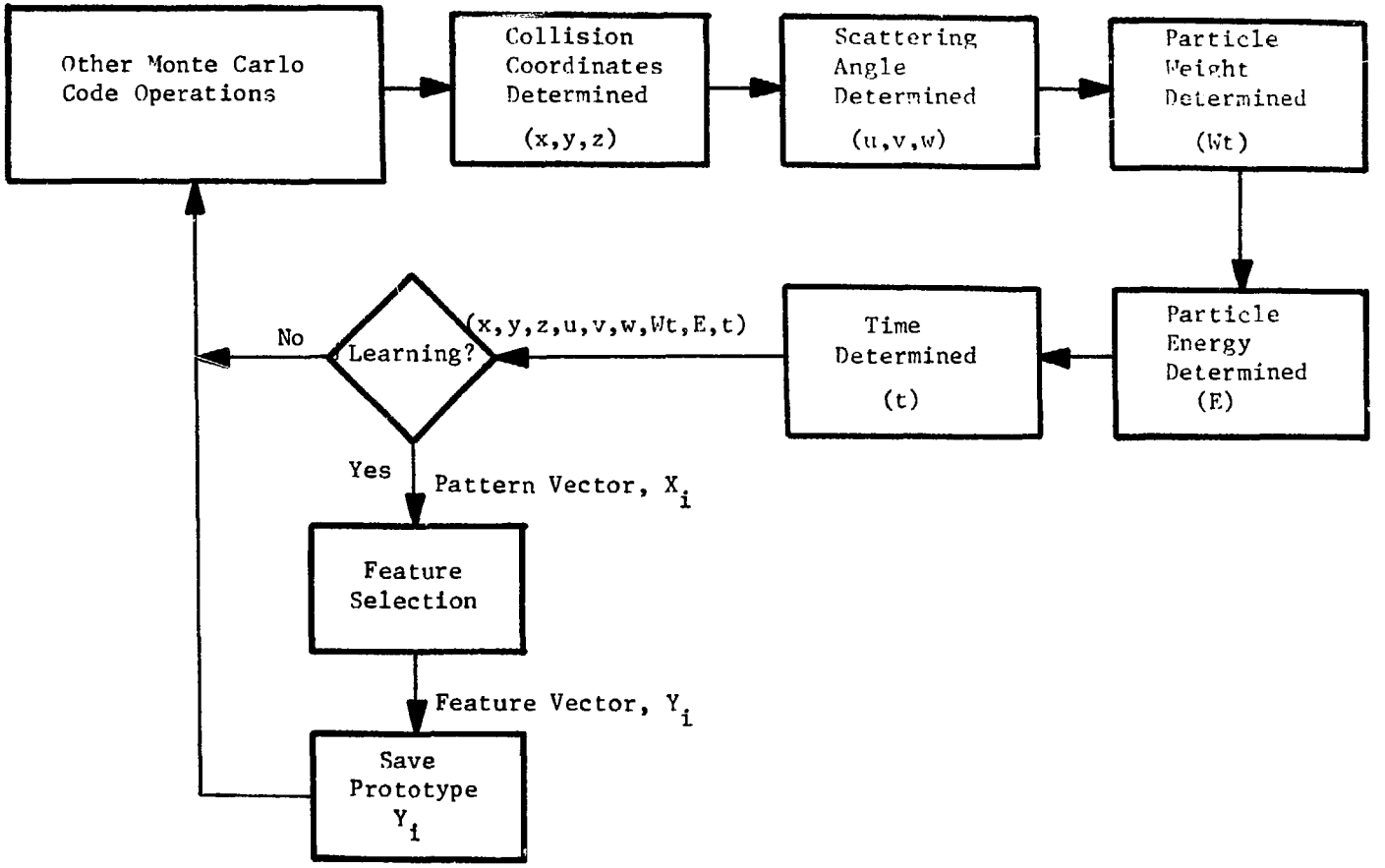


Figure 5.1 Saving Information for Prototypes

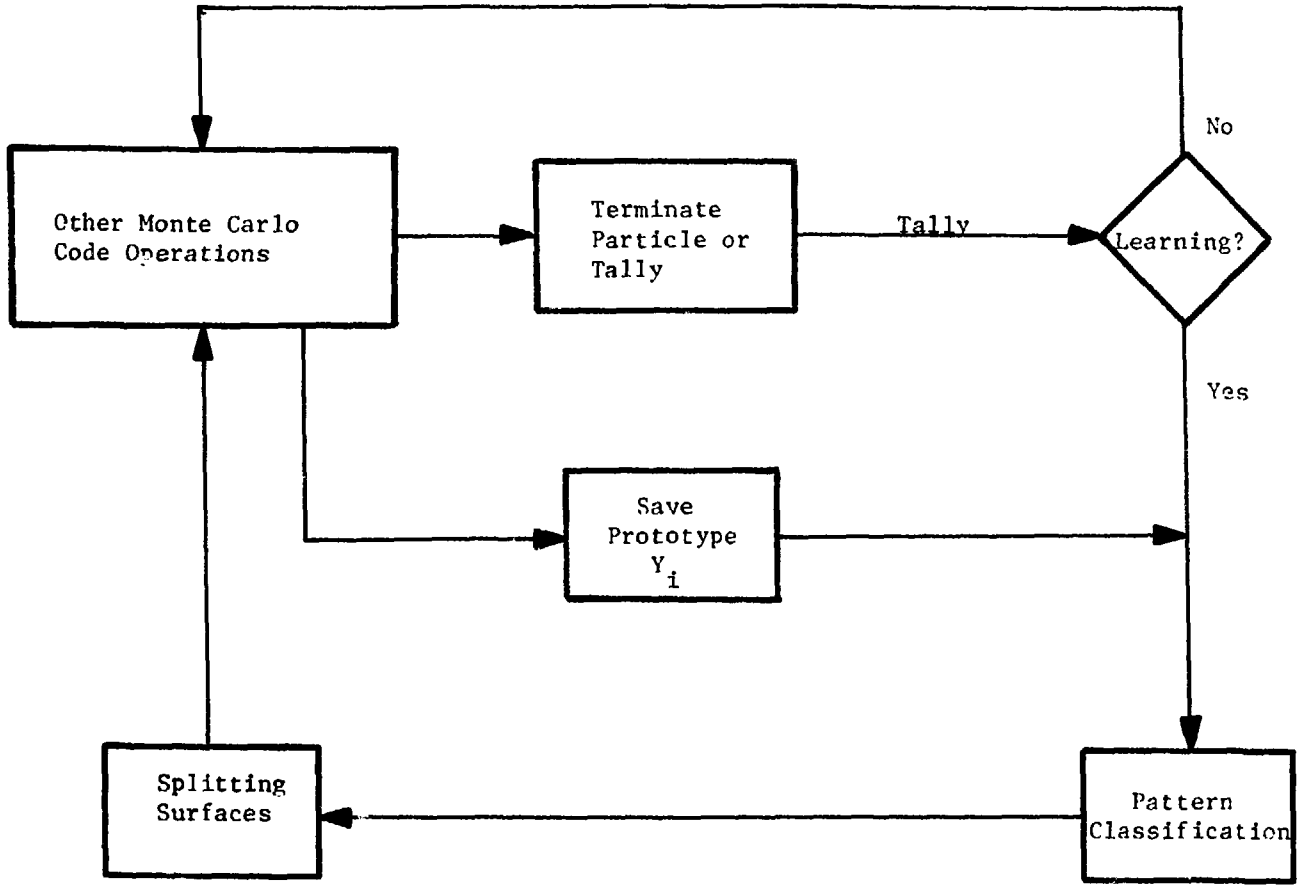


Figure 5.2 Communication between the Pattern Classifier and the Monte Carlo Program

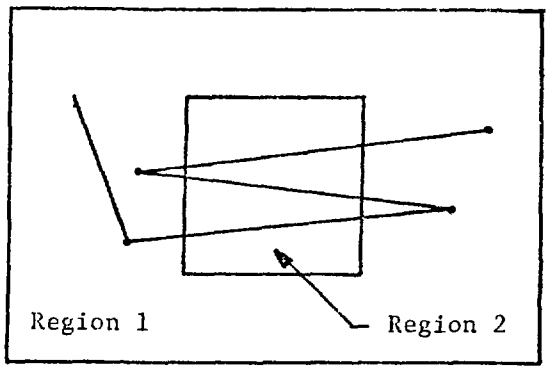


Figure 5.3 Multiple Contributions to a Tally

be classified among themselves). Therefore the first splitting surface to be learned is the surface separating zero from non-zero importance regions.

The initial location of this first splitting surface should be well within the class 1 (zero importance) region. Default* values may be used to locate this surface; however, if the user can supply more information, it should be used in order to speed up convergence. Suitable default values for the components of W are

$$w_i = -1.0 \qquad i = 1, \ldots, N \qquad (5.1)$$

$$w_{N+1} = N$$

---

* A default value is one which is supplied automatically by the computer program; however, the program also allows for the user to provide a different value. If the user provides a value, it is used; if he does not, the default value is used.

where it has been assumed that the feature selector has normalized all feature components to an interval from zero to one (see Section 5.2) such that importance decreases as $y_i (i = 1, \ldots, N)$ increases.

Once this initial location has been set, prototypes can then be used as the Monte Carlo calculation progresses. This splitting surface can be utilized while W is being learned. By doing this more prototypes are introduced on the class 2 side which will accelerate the convergence. When the classifier attains a suitable misclassification rate and variability (as set by the user or default), weight adjustment on the first splitting surface can be stopped. During this first phase, the number of prototypes in each class, M_i (i=1,2) is calculated. This variable can be used in the following expression

$$\left\{ \begin{array}{c} \text{if } F_2 = \dfrac{M_2}{M_1+M_2} > R \\ \\ \text{then no more splitting surfaces are learned} \end{array} \right\} \qquad (5.2)$$

to determine if more splitting is necessary where R is determined by the user. The first splitting surface (between zero and non-zero importances) divides state space into two regions - in one region particles have a probability >50% of contributing to the tally, in the other region the particles have a probability >50% of not contributing to the tally. As F_2 increases less splitting is necessary.

If $F_2 < R$ then an additional surface within class 1 is necessary to get more prototypes into the important regions of the problem. However, since all prototypes in class 1 have zero importance, a sub-goal or sub-tally must be used for classification. One such choice is to observe when a particle enters class 2 according to the student. This is done by checking the sign of g(Y) after each collision (this is already done since g(Y) is being used to split particles). If it is found that a particle is entering C_2 (i.e., $g(Y) > 0$), the particle is tallied (only for the pattern classifier). The prototypes are then sent to the pattern classifier. After the classification process, the particle continues the random walk. However, prototypes are not created until the particle re-enters C_1 (i.e. the "No" learning branch of Figure 5.1 is used). If the particle re-enters C_1, prototypes are again saved until the particle either enters C_2 or is lost to the system. Like the C_1-C_2 splitting surface, a single source particle can contribute several "sets" of prototypes (one set for each tally). Unlike the C_1-C_2 splitting surface, prototypes are not created after each collision, but only after collisions occurring within C_1. With this procedure, class 1 can be subdivided into two sub-classes, class 3 and class 4 where class 4 particles have a 50% or greater chance of becoming a class 2 particle and class 3 particles have less than 50%. The choice of an initial w for the second splitting surface can be made by using Equation 5.3

$$ {}_{I}w_i = \frac{2({}_{I}w_{N+1})}{\frac{{}_{2}w_{N+1}}{{}_{2}w_i} + \frac{{}_{1}w_{N+1}}{{}_{1}w_i}} \qquad I = 1,\ldots,N \tag{5.3} $$

$$ {}_{I}w_{N+1} = \frac{1}{2}\left({}_{1}w_{N+1} + {}_{2}w_{N+1}\right) $$

where ${}_{I}w_i$ = initial weight of new discriminant function

${}_{1}w_i$ = initial weight of previous discriminant function

${}_{2}w_i$ = final weight of previous discriminant function

which results in a splitting surface located mid-way between the first and last positions of the class 1-2 decision surface. When the second splitting surface has been identified a check is again made on F_2. If $F_2 < R$, three things can be done: (1) class 3 can be subdivided (2) class 4 can be subdivided (3) class 3 and 4 can be subdivided at the same time. If $F_2 << R$, then the third choice is needed since a considerable amount of splitting is necessary to make F_2 larger. Otherwise the choice between 1 and 2 is decided as shown by relation 5.4 with $I = 3$, $J = 4$.

If $M_I > M_J$, subdivide class I.

If $M_J > M_I$, subdivide class J. (5.4)

This process continues until either a maximum number of surfaces have been identified or $F_2 > R$. A flow diagram of the above process is shown in Figure 5.4. Table 5.1 lists the eight regions formed by the first 4 splitting surfaces.

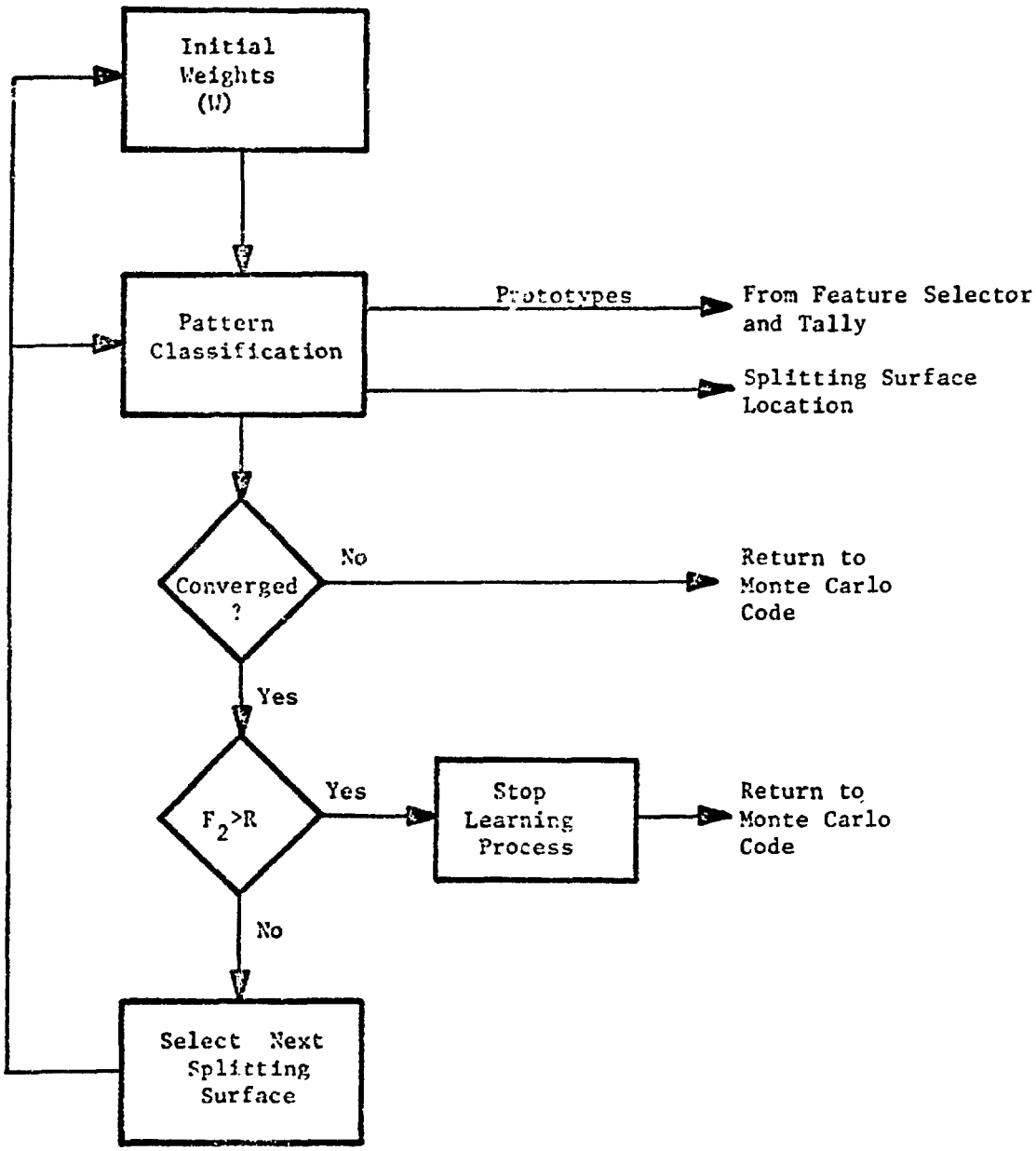


Figure 5.4 Iterative Procedure for Generating Splitting Surfaces

Table 5.1

Class	Probability of Tally
1	$\leq 50\%$
2	$\geq 50\%$
3	$\leq 25\%$
4	$\geq 25\%$, $\leq 50\%$
5	$\leq 12.5\%$
6	$\geq 12.5\%$, $\leq 25\%$
7	$\geq 25\%$, $\leq 37.5\%$
8	$\geq 37.5\%$, $\leq 50\%$

5.1.2) Multiple Tallies

In the previous section only the splitting surface between classes 1 and 2 depended upon the importances, I, of the prototypes (this is not true if class 2 is subdivided). The problem of multiple tallies consists of defining what is meant by I when more than one tally is considered. This section considers only the C_1-C_2 splitting surface. Subdivision of class C_2 is discussed in Section 4.1.2 and subdivision of class C_1 is identical to that described in Section 5.1.1.

The importance I_i ($i=1,\ldots,n_i$ n=no. of tallies) is the importance relative to a particular tally. Therefore each prototype actually has n importances for n tallies. One could assign a g(Y) for each tally; however, this would consume n times as much computer time. An easier solution is to assign a probability, p_i, to each tally such that

$$I = \sum_{i=1}^{n} \delta_i I_i \tag{5.5}$$

where $\delta_i = 1$ with probability p_i

$\delta_i = 0$ with probability $(1-p_i)$

I_i = importance with respect to i'th taliy

I = overall importance of prototype.

This results in a tally with a high p_i having more particles directed towards it. Although the user can aasign the p_i, it is possible to have this done automatically by using the variance of each tally during the Monte Carlo calculation. The p_i of tallies with high variances should be increased, with low variances decreased. This splits particles so as to produce a uniform variance for all tallies. Although this process affects only the C_1-C_2 surface directly, since all other subclasses of C_1 are related to this surface, they are also affected indirectly.

5.1.3) The Point Detector Tally

The point detector tally described here assumes the same structure as is used in the MCN code at Los Alamos Scientific Laboratory[19], usually used for the tallying of flux at a point. This tally

differs from other tallies in that after each collision, the contribution to the tally, $x_{i,j}$, given by

$$x_{i,j} = wt_{i,j}\, p_j \tag{5.6}$$

where $x_{i,j}$ = contribution to the tally of the i'th particle after the j'th collision ($x_i = \sum_{j=1}^{J} x_{i,j}$ where J = no. of collisions)

$wt_{i,j}$ = weight of the i'th particle before the j'th collision

p_j = (probability of scatter) × (the probability of scattering toward and being detected by the point detector at the j'th collision)

is calculated. Unlike other tallies each prototype has a non-zero importance which eliminates the C_1-C_2 surface as described in previous sections. Instead an appropriate $\bar{I}$ must be selected which falls within the distribution of I_j's. (I_j = importance of the j'th prototype, not to be confused with I_i of Equation 5.5 which is the importance of a prototype with respect to the i'th tally). The median is a good choice for $\bar{I}$ since it is a measure of the number of prototypes. The mean is a less useful measure since it can be strongly influenced by the value of I_j (i.e. a few very large I_j affects the mean $\bar{I}$ more than a much larger number of prototypes with small I_j). For some cases (especially when the distribution of I_j is over many orders of magnitude) the mean falls at the tail

of the distribution which would not be very useful for splitting. Another possible value for $\bar{I}$ is the logarithmic mean (see Equation 5.7)

$$\bar{I} = \text{logarithmic mean} = \frac{\sum_{j=1}^{N} \log_{10} I_j}{N} \qquad (5.7)$$

where N = number of prototypes

I_j = importance of the j'th prototype

which compensates for distributions that cover many orders of magnitude. An important consideration in the choice of an $\bar{I}$ is the computer time required. Thus, although the median is the best choice from a statistical point of view (it creates classes which originally have the same number of prototypes in each class) it is also expensive calculationally.

The initial stages of the Monte Carlo calculation are used to calculate $\bar{I}$. During this initial stage weight adjustment is allowed with the initial W chosen the same as described in Section 5.1.1. The splitting surface is not used until a final $\bar{I}$ has been determined. This prevents splitting from influencing the selection of $\bar{I}$. When the first splitting surface has converged, F_2 is checked (see Equation 5.2). If $F_2 < R$ then the procedure continues as illustrated by Figure 5.4. As described in Section 5.1 this process continues until either $F_2 > R$ or a maximum number of surfaces has been created.

The importance of a prototype for multiple point detectors is determined by

$$I = \sum_{i=1}^{n} m_i I_i \qquad (5.8)$$

m_i = weight of the i'th detector

I_i = importance relative to the i'th detector (same as Equation 5.5)

n = number of detectors

The weights, m_i, serve two purposes. First of all they are used to normalize tallies. This is required when tallies with different units are used (for example fissions at a point, captures at a point, flux at a point) and must be normalized to $0 < I_i < 1$. The second purpose of m_i is to direct particles towards certain tallies. This is done by increasing m_i for tallies to which more particles are needed in the same manner as was suggested for the p_i (see Section 5.1.2). Also like p_i, optimum m_i can be learned during the calculations or can be supplied by the user.

5.2) Feature Selection

As stated in Section 3.4, feature selection is best served for this research if it remains a human function. This can easily be done if the feature selector consists of a user supplied subroutine in much the same was as for source subroutines[19]. The function of the feature selection subroutine is shown in Figure 5.5. The user must supply the following information:

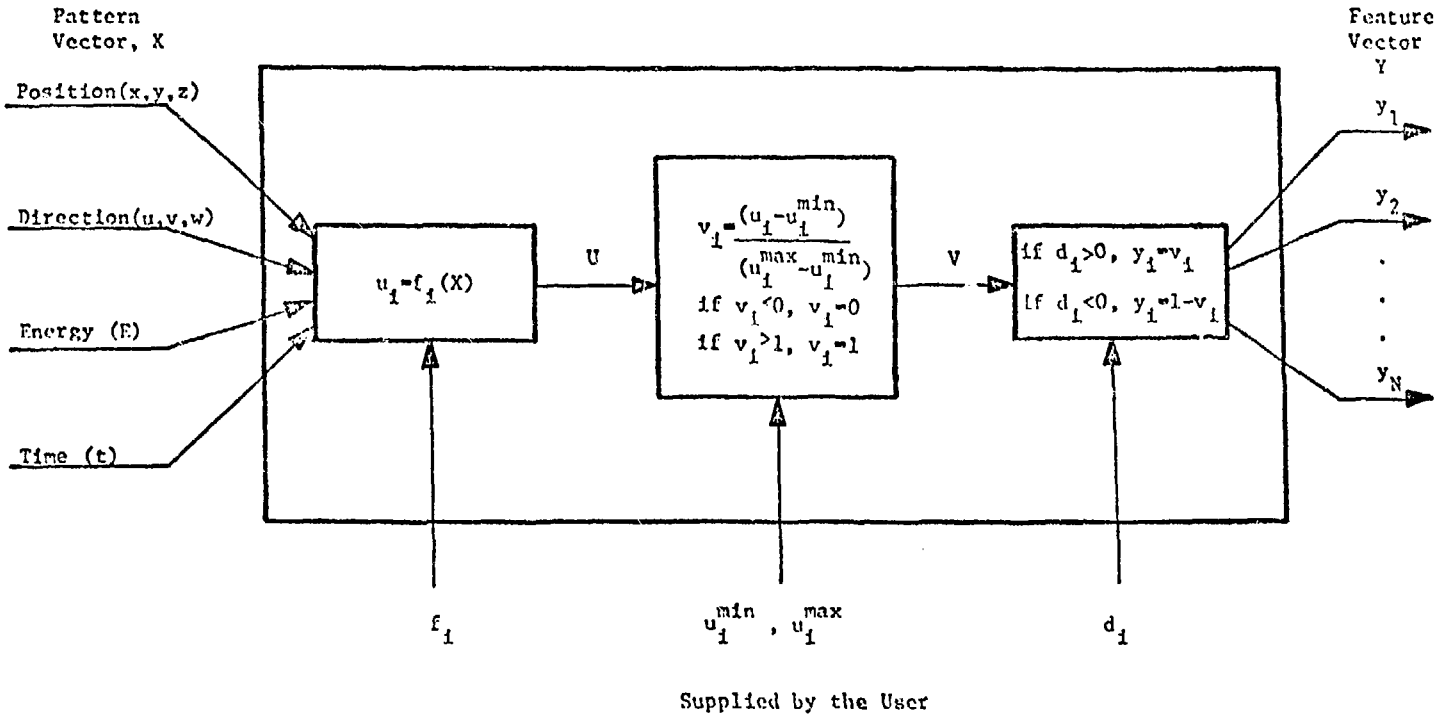


Figure 5.5 Structure of the Feature Selector

f_i = the functional form of the i'th components of feature space (e.g., $f_i = r = x^2+y^2+z^2$)

u_i^{min} = the minimum value encountered for the i'th component of feature space

u_i^{max} = the maximum value encountered for the i'th component of feature space

d_i = +1, if the importance increases as v_i (see Figure 5.5) increases

= -1, if the importance decreases as v_i increases

The selection of the f_i can do more to improve the behavior of the pattern recognition system than any other user input. As users gain experience they will undoubtedly gain skill in the selection of f_i. The following heuristics* or rules of thumb should prove of value in this process.

(1) Omit any variables which are not used in the problem.

(2) Make use of any symmetries in the problem.

(3) If a variable spans several orders of magnitude, use the logarithm of the variable.

(4) Limit the span of a pattern vector component to that range

---

* Heuristics are non-analytically derived rules usually gained through insight or experience and are used when analytical techniques are inadequate or non-existent.[38]

over which splitting surfaces are likely to occur. This is done by specifying the range (u_i^{min}, u_i^{max}) to be smaller than the actual range of variable u_i.

(5) If the importance varies very weakly with a variable, omit the variable.

(6) Determine what variables are important from the "tally point of view".

(7) Select features for which the importances either monotonically increases or decreases.

Heuristics 1, 2, and 5 are aimed primarily at reducing the dimensionality of feature space. Heuristics 3 and 4 are used to prevent surfaces from being too close to each other in feature space. The results of these heuristics is to expand the space in the vicinity of splitting surfaces. The purpose of heuristic 6 is to select optimum feature components for splitting. Heuristic 7 is aimed at preventing class distributions which cannot be treated with linear discriminant functions. Suggestions for each pattern vector component are given below using the above heuristics as guidelines.

Time: A great number of transport problems are concerned only with steady state situations. For these problems time should be omitted from all f_i (heuristic 1). Problems which do involve time often involve several decades. In this case heuristics 3 and 4 should be used. Time will usually not be combined with any other variables to form a new y_i but will be used alone.

Energy: Most transport problems of interest are usually strongly dependent on energy due to the energy dependence of nuclear cross sections. An example of this is the fission cross section which varies from a few barns in the Mev region to several hundred barns in the ev region. Therefore, the probability of a fission occurring varies by two orders of magnitude due to energy alone. Heuristics 3 and 4 are almost always necessary unless dealing with only a very small energy interval. There are problems in which, over the energy range of interest, the cross sections of the materials involved (for example, carbon) are relatively flat. In such cases heuristic 5 should be executed. Like time, the energy variable will seldom be combined with other variables.

Spatial Coordinates x,y,z: Heuristic 2 is intended primarily for the spatial coordinates. For example, if the geometry of a problem is symmetric about an axis, cylindrical coordinates should be used for the features. Once the geometry has been transformed to cylindrical coordinates, it may be found that the importance does not vary with y or θ (see Figure 5.6). Then by using $f_i = r = x^2 + z^2$, three components (x,y,z) have been reduced to one. An analogous situation exists for spherically symmetric problems. Figure 5.7 illustrates a case where heuristic 7 is necessary. By transforming x as given by

$$f_i = |(x-x_1)| \tag{5.9}$$

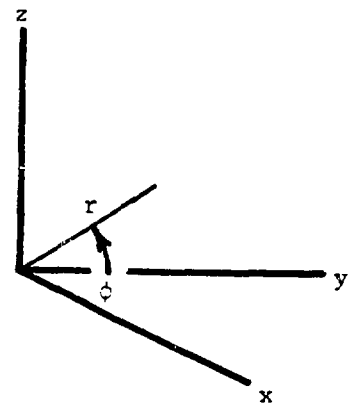


Figure 5.6 Cylindrical Symmetry

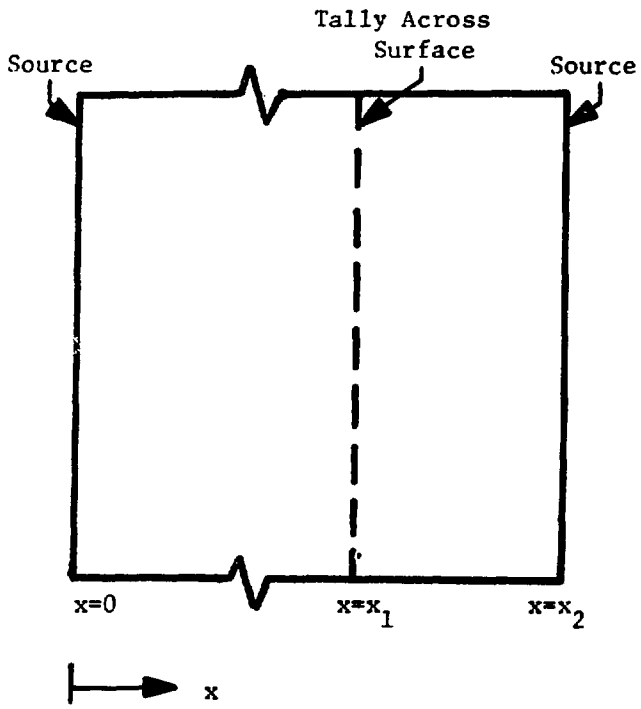


Figure 5.7 Feature Selection to Avoid a Quadratic g(Y)

a linear discriminate function can be used. In general the distance to a tally is the best choice of a spatial coordinate (heuristic 6) as given by

$$f_i = d = \sqrt{(x-x_d)^2 + (y-y_d)^2 + (z-z_d)^2} \qquad (5.10)$$

where x_d, y_d, z_d are the coordinates of the tally

Such a choice also reduces the number of features. Spatial features will usually not require the use of heuristics 3 and 4.

Direction Coordinates (u,v,w): The direction cosines are similar to the spatial coordinates in that they are usually better combined. For example, the cosine of the angle between the present line of flight (vector=v_1) and the direction to the tally (vector=v_2) given by

$$f_i = \cos \phi = \frac{V_1 \cdot V_2}{|V_1 V_2|} \qquad (5.11)$$

is similar to d (Equation 5.10). Using the above feature also allows linear discriminant functions (heuristic 7). As in the case of the spatial coordinates, cylindrical and spherical symmetry can often reduce the number of angular features necessary.

The problem illustrated in Figure 5.8 will be used to demonstrate the proper selection of features. In this problem material 2 is highly absorbing and material 1 is only slightly absorbing but has a moderately

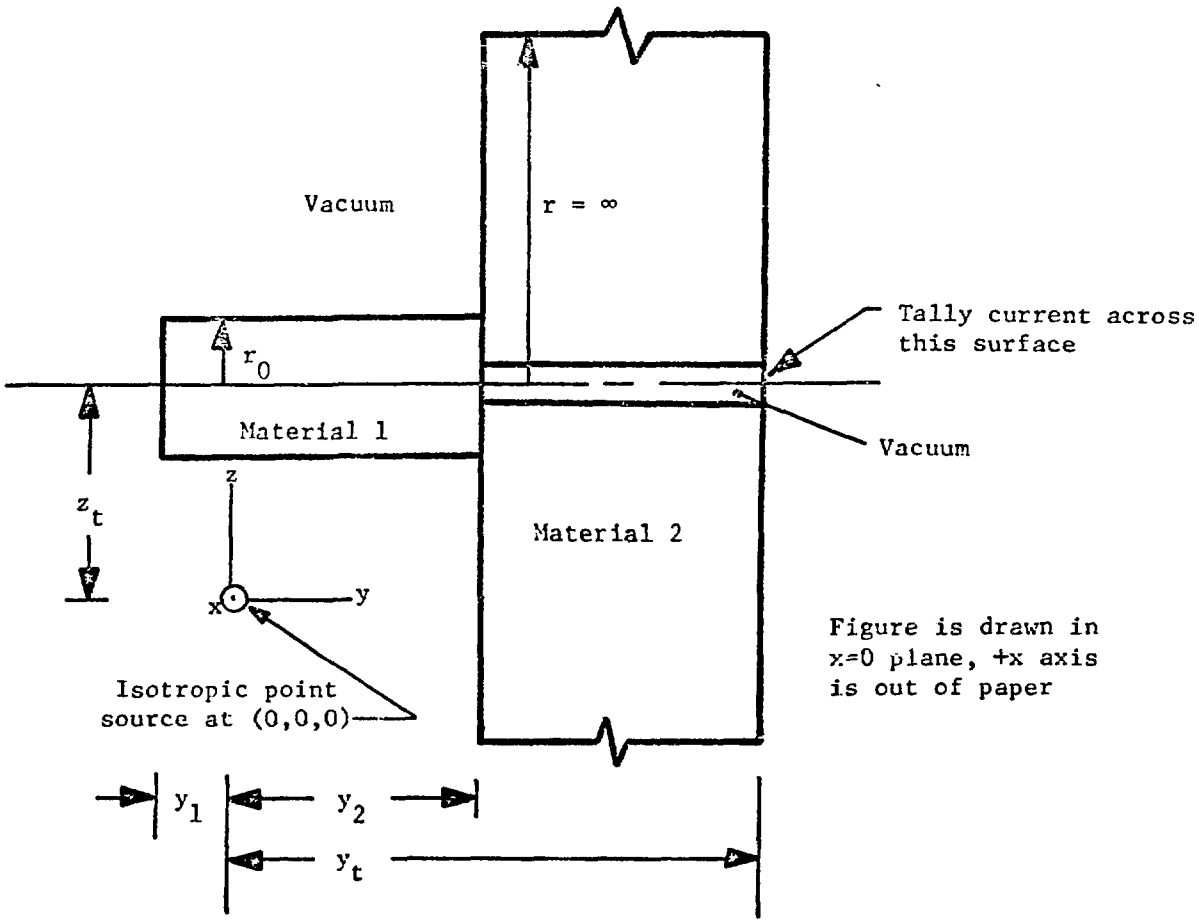


Figure 5.8 Sample Problem for Feature Selection

high scattering cross section. An isotropic source of neutrons is located at the origin. It is assumed that all cross sections are flat with energy and that time is of no interest. The tally consists of counting the number of neutrons which cross the surface located at

$y = y_t$ The material regions shown are cylindrically symmetric about the line drawn through the point $(0,0,z_t)$ and parallel to the y axis.

The following f_i would be suitable for the feature vector of this problem:

$$f_1 = y$$

$$f_2 = r = \sqrt{(z_t - z)^2 + x^2}$$

$$f_3 = v \cdot v_d |v\ v_d|$$

$$u_1^{min} = -y_1 \qquad u_1^{max} = y_t$$

$$u_2^{min} = 0 \qquad u_2^{max} = r_0$$

$$u_3^{min} = -1.0 \qquad u_3^{max} = 1.0$$

where v = direction cosine with y axis

v_d = direction cosine to point $(0,y_t,z_t)$

The previous seven heuristics relate to the above selection as follows:

Heuristic 1. Since cross sections are flat, the problem is independent of energy and as was stated time is of no interest. Therefore time and energy do not appear as variables in any of the f_i.

Heuristic 2. The form of f_2 was chosen due to the symmetry of the problem. This reduced the number of variables and is also a more effective variable for splitting.

Heuristic 3. Not used.

Heuristic 4. This heuristic was used to set u_2^{max}. For $r>r_o$ particles all belong to the same class thus limiting splitting to $r<r_o$. The other u_i^{max} and u_i^{min} are set to the limits of the problem.

Heuristic 5. Not used.

Heuristic 6. This heuristic influenced the choice of all three f_i. If one is at the tally looking at contributing neutrons, almost all the neutrons will be coming down the channel from material 1 (few come from material 2 since it is highly absorbing). Because of this, the probability of a neutron arriving at the tally depends primarily on the f_i chosen.

Heuristic 7. As f_1 increases from $-y_1$ to y_t, the probability that it is tallied also increases monotonically. This is also true for f_2 and f_3.

Although the selection of features does require user information, this information is qualitative in nature as opposed to the quantitative information required by normal geometry splitting.

5.3) Timing Considerations and Effectiveness

The purpose of this section is to determine the factors necessary for estimating the effectiveness of state space splitting using pattern recognition. The only definitive way to determine the effectiveness as given by Equation 2.16 is to apply the technique to a full scale

Monte Carlo Code and use it both with and without the variance reduction for a large number of problems. The approach taken in this section is to compare the various operations involved in state space splitting as opposed to conventional splitting (Section 5.3.1). The additional operations involved in using pattern recognition to identify the splitting surfaces are then described (Section 5.3.2). Although this analysis does not result in an absolute evaluation for all problems, some assumptions can be made concerning the effectiveness as compared to presently used splitting technique.

5.3.1) State Space Splitting vs. Conventional Splitting

The basis of this comparison is the current version of the MCN neutron Monte Carlo code at Los Alamos Scientific Laboratory[19] which contains both energy splitting and geometry splitting. For the energy splitting the user specifies a number of energy splitting surfaces, E_i, i=i,...,N (N=number of splitting surfaces) and the ratio for splitting between two energy regions (separated by E_i) R_i, i=1,...,N. The energy of the particle is checked after each collision. If the energy drops below an E_i, the particle is split into R_i particles with weights equal to the particles original weight divided by R_i.

Each geometric cell or region (see Appendix J) is assigned an importance I_i. Whenever a particle crosses a surface S_j which bounds two or more cells, a check is made to determine which cell the particle is entering. This check is made by: (1) determining which cell or cells are on the other side of S_j (2) if more than one cell is on the

other side of S_j the senses (see Appendix J) are calculated with respect to the bounding surfaces (with the exception of S_j) of each cell (3) the cell for which all the senses agree is used as the next cell (if there is only one cell, it is used). The importance of the new cell, I_n, is then compared with the importance of the previous cell, I_o, and the operations indicated in Figure 5.9 are performed. In the case of splitting, if the ratio of importances I_n/I_o is a non-integer value given by

$$R = I_n/I_o - J \tag{5.12}$$

where J = largest integer that will go into I_n/I_o,

then J+1 particles are created with probability R and J particles are created with probability (1-R). When particles are split one of the particles is continued and the others are banked* (i.e., the state space description of each is saved) until the first particle leaves the system, at which time the banked particles are continued. Of the above process, the only time used for splitting and Russian roulette is the time spent in the operations shown in Figure 5.9 plus the time spent in banking particles. The average time per particle spent performing these operations will be designated Δt_1. Unfortunately, this is not the only

---

* The banking of a particle consists of recording its state space description (x,y,z,u,v,w,E,t) and its weight, wt, on a pushdown list. This list is so constructed and maintained that the next particle retrieved from the list is the last particle placed on the list.

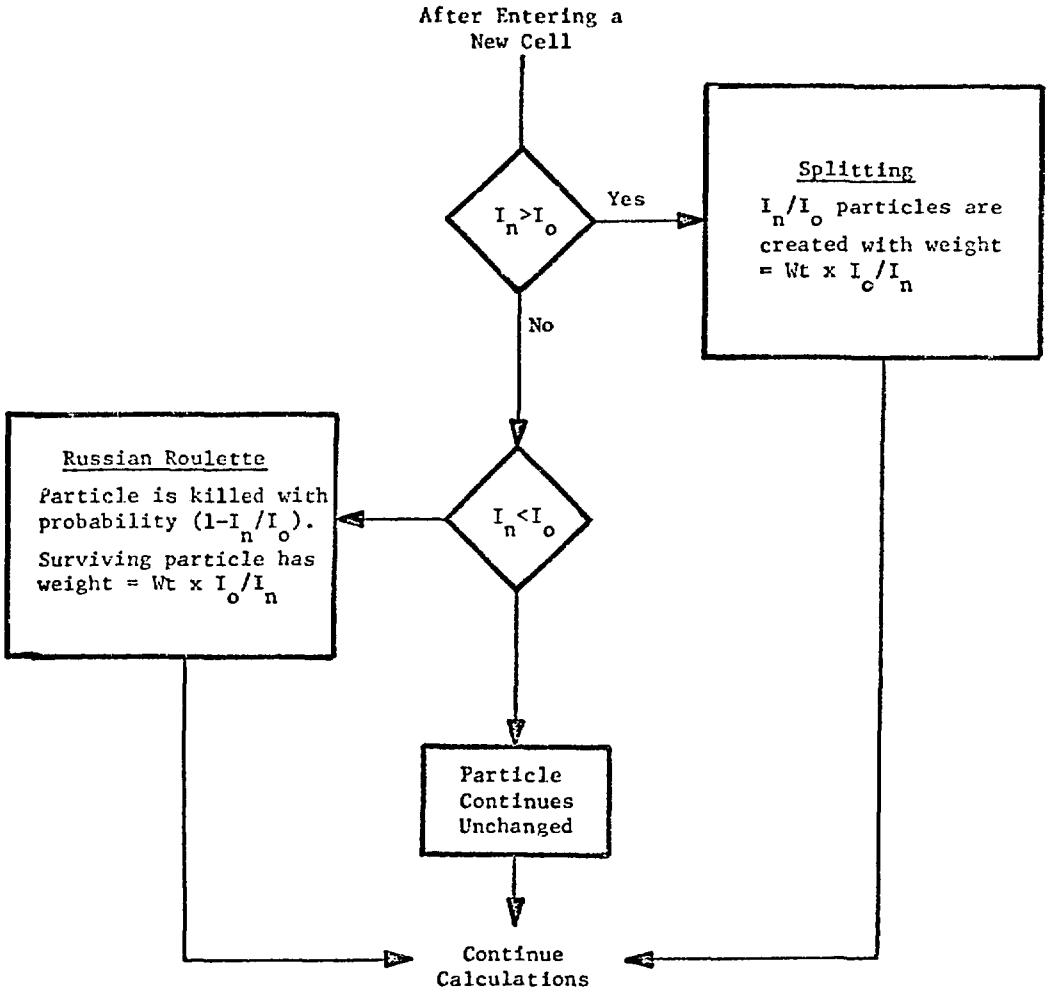


Figure 5.9 Conventional Geometry Splitting

increase in time due to the splitting process. Because splitting is often desired at surfaces which are not material boundaries, the user must subdivide the already existing cells into smaller cells in order to split at the surfaces he desires. The introduction of these new "splitting cells and surfaces" means that more intersections will have to be calculated and more senses will have to be checked. It also requires more calculations of the distance until a collision. These three operations can consume a considerable amount of time (the majority of computer time spent is often spent in these geometry related calculations). In fact, for some problems, the additional time required by these operations can completely offset any gains due to variance reduction. The average time per particle spent due to increased geometry calculations caused by "splitting cells" will be denoted as Δt_2.

If the additional time per particle spent due to energy splitting is designated Δt_3, then Equation 5.13

$$\Delta t_g = \Delta t_1 + \Delta t_2 + \Delta t_3 \qquad (5.13)$$

where Δt_1 = time to perform processes of Figure 5.9 plus banking of particles

Δt_2 = time required to perform geometry calculations due to additional "splitting cells"

Δt_3 = time required for energy splitting

gives the total additional time spent due to the use of splitting and Russian roulette.

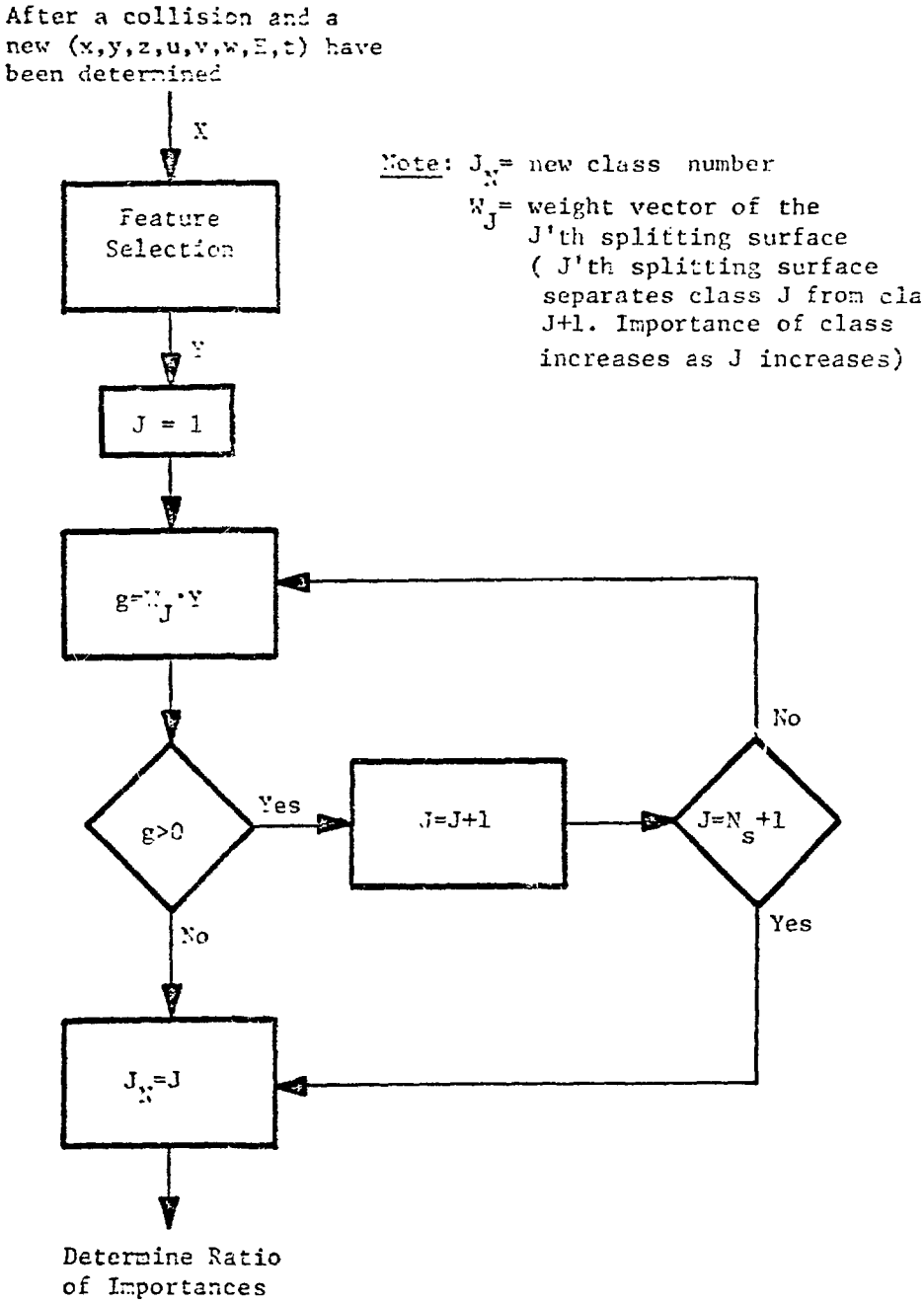


Figure 5.10 Determining the New Class for State Space Splitting

If state space splitting is used with known splitting surfaces (i.e., W is known) the operations shown in Figure 5.10 are performed after each collision. Once the new class, J_n, has been found it is used with the old class, J_o, to determine the ratio of importances given by

$$I_n/I_o = 2^{\Delta J}, \qquad \Delta J=J_n-J_o \tag{5.14}$$

After this ratio has been calculated, the procedure illustrated in Figure 5.9 is used. Therefore, using state space splitting, the total additional time per particle, Δt_s, is given by

$$\Delta t_s = \Delta t_4 + \Delta t_1 \tag{5.15}$$

where Δt_4 = average time spent per particle to perform operations of Figure 5.10

Δt_1 = see Equation 5.12.

Subtracting Equation 5.15 from 5.13 results in

$$\Delta t_g - \Delta t_s = (\Delta t_2 + \Delta t_3) - \Delta t_4 \tag{5.16}$$

All three of these parameters (Δt_2, Δt_3, and Δt_4) are dependent upon the amount of splitting used. The parameter Δt_2 is also very dependent on the number of "splitting cells" and the complexity of the surfaces involved (higher order surfaces require more time to calculate intersections). The parameter Δt_4 is dependent on the complexity and number

of the $f_i(Y)$. A great deal of experimentation with many different Monte Carlo problems would have to be run to determine whether Δt_g or Δt_s is larger. It is most probable that each technique has a class of problems for which it operates quicker.

If both techniques are used to represent the same surfaces, the relative effectiveness can be calculated by comparing the above times only. However, this is not the case since the conventional technique splits only in energy and position space (independently) and the other approach splits in state space. Furthermore, the splitting surfaces of the conventional approach are the results of heuristic guesses by the user; whereas, in the state space approach, the surfaces are the result of the pattern recognizer. Figure 5.11 demonstrates the advantage of splitting in state space for the two dimensional problem considered in Section 4.4. The solid line indicates the splitting surface found by pattern recognition. The dashed lines are a few of the splitting surfaces possible if geometry splitting is used alone. As can be seen there is a great amount of deviation regardless of which geometry splitting surface is chosen. The problem is caused by the fact that one cannot split along constant contours of I using geometry and energy splitting only. As a result, many particles are split which should not be. As a result, N_e (see Equation 2.15) will in general be much smaller for state space splitting than for the conventional splitting approach.

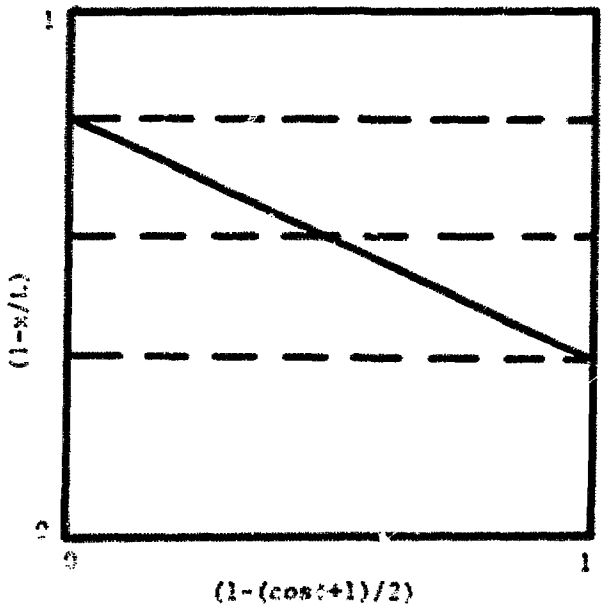


Figure 5.11 Splitting Surfaces for the Problem of Section 4.4

5.3.2) Time Spent for Pattern Recognition

Section 5.3.1 discussed the time required to split particles in state space. This section discusses the additional time required to learn the splitting surfaces and includes:

(1) T_f = the additional time required to save prototypes (see Figure 5.1)

(2) T_p = the time required to classify patterns and adjust W (see Sections 5.2.3, 4.4.3, and 4.5)

(3) T_m = time required to "manage" the splitting surface selection (see Figure 5.4).

The above operations are different from those described in Section 5.3.1 since they are only performed while the splitting surfaces are being learned. Because of this, the time spent for pattern recognition is distributed only over those particles which exist while the pattern recognizer is being used.

The discussion in this section assumes that a point detector tally is not used and that the splitting surfaces are learned one at a time (i.e., one surface has passed the convergence test before another is begun) as described in Section 5.1.1. The following notation is used throughout this section:

$S_{k,j}$ = the splitting surface between classes k and j where the k and j are defined as given by Table 5.1. The i'th splitting surface refers to surface $S_{2i-1,2i}$.

N_i = the average number of collisions per source particle in a particular Monte Carlo calculation before the i'th surface is learned but after the (i-1)'th surface is learned. ($N_1 = N_i$ before any surfaces are learned)

h_i = average fraction of N_i which occur in C_i before the i'th surface is learned but after the (i-1)'th surface is learned ($h_1 = h_i$ before any surface is learned = 1).

Q_i = the average number of times a source particle contributes to the tally (or class sub-tally) while the i'th surface is being learned.

B_i = number of source particles required to learn the i'th splitting surface.

$P_i = N_i h_i B_j$ = number of prototypes required to learn the i'th splitting surface.

M_s = total number of splitting surfaces learned. M_s is either supplied by the user or is determined by the F_I test (see Figure 5.4).

The splitting surfaces are learned in the following order:

$$S_{1,2},\ S_{3,4},\ S_{5,6},\ S_{7,8},\ \ldots\ldots\ldots S_{2M_s-1,2M_s}$$

T_f (saving prototypes)

The total time spent transforming vectors from pattern to feature space is given by

$$T = t_f \sum_{i=1}^{M_s} N_i h_i B_i = t_f \sum_{i=1}^{M_s} P_i \qquad (5.17)$$

where t_f = average time spent transforming one prototype (see Figure 5.5)

The time parameter t_f depends on the number and computational complexity of the f_i (see Figure 5.5) involved and like the parameters N_i, h_i, and B_i, is highly problem dependent. The u_i (see Figure 5.5) are often already calculated within the Monte Carlo Code and therefore need only be stored.

After the first learning surface has been identified, the feature vector is calculated after every collision in order to utilize state space splitting (see Section 5.3.1). Since this operation need not be performed twice, the <u>additional</u> time used to transform pattern vectors is given by the equation

$$T' = t_f N_1 B_1 \tag{5.18}$$

If the additional time spent in storing the Y_i is designated by T_s, then T_f is given by the following equation

$$T_f = T' + T_s = t_f N_1 B_1 + T_s \tag{5.19}$$

In general $T_s << t_f N_1 B_1$ resulting in the equation

$$T_f \approx T' = t_f N_1 B_1 \tag{5.20}$$

T_p (pattern classifier)

The time spent by the pattern classifier has been discussed in Sections 4.2.3, 4.4.3, and 4.5 for the pattern classifiers used in this research and are summarized in Tables 4.5 and 4.7. It should be noted that these times indicate a maximum of the values to be expected. Very simple changes in the FORTRAN coding can lead to large reductions in time. Some examples of these changes are:

(1) For non-point detector tallies where the classes are determined by "tally" or "no tally", the importance I_i need not be calculated. In such a case all prototypes in a "set" (see Section 5.1.1) belong to the same class. The only test necessary to determine classification is to test the tally for zero, non-zero. For the problems run in Chapter IV, this results in changing Equation 4.9 to Equation 5.21 thus reducing T by $(N-B)t_A$.

$$T = Bt_A + N(t_L f_L + t_C f_C f_L) \tag{5.21}$$

where B = number of source particles

(2) The dot product of two vectors (W·Y* and Y*·Y*) can be performed much quicker by using system functions (for example DOTPRO[39] at LASL). The larger the dimensions of W, the more time can be saved. In this research each vector element was multiplied in FORTRAN.

(3) Vector addition ($W_{i+1} = W_i + cY^*$) can be performed much faster using system functions (such as ADDVEC[40] at LASL) In this research each element was added separately using a FORTRAN statement for each element.

Since the purpose of this research is not to develop an optimum FORTRAN program, these operations have not been used since the more straightforward multiplication and additions are easier for the reader to follow (see Appendices D, F, H, I).

The total time spent by the pattern classifier to learn M_s splitting surfaces is given by Equation 5.22.

$$T_p = T_1 + T_2 + \ldots\ldots\ldots T_{M_s} \tag{5.22}$$

$$T_p = \sum_{i=1}^{M_s} (t_L\, {}_i f_L + t_C\, {}_i f_C\, {}_i f_L) P_i + t_A \sum_{i=1}^{M_s} Q_i B_i$$

where: t_A, t_L, t_C are given by Equation 4.9

${}_i f_L$ and ${}_i f_C$ are the same as f_L and f_C of Equation 4.9 except that they refer to prototypes used in learning the i'th splitting surface

T_i = time spent learning surface i

T_m (splitting surface selection)

The time T_m includes the following operations (see Figure 5.4):

(1) Initial Weight Selection

(2) Convergence testing including the calculations involved in determining the misclassification rate and variability.

(3) Splitting Surface Selection including $F_2 > R$ test.

Operations (1) and (3) are only performed M_s times throughout the entire Monte Carlo calculation and therefore their times will be ignored compared to operation (2) which is orders of magnitude more time consuming. The convergence is tested after every C_T source particles. The choice of C_T depends upon the number of source particles required for convergence, B_i. However since $B_i = P_i/(N_i h_i)$ and since P_i does not vary greatly from problem to problem ($P_i \sim 4{,}000$ from Chapter IV), B_i can be approximated from the above relationship (h_i and N_i are approximated within the Monte Carlo calculation). As experience is gained with this technique, a suitable C_T can probably be user specified.

Calculation of the misclassification rate involves storing the number of prototypes and the number of misclassified prototypes. The former is already performed by the classifier for each class and the latter consists of incrementing a counting register every time a prototype is misclassified. A ratio of the two is then performed after every C_T source particles.

The calculation of the variability is more complex. For every feature vector component, y_i, there is an intercept of the decision surface with the y_i axis which is given by

$A_{i,j}$ = intercept with y_i axis after j'th prototype =

$$- \frac{w_{N+1,j}}{w_{i,j}} \qquad i = 1,\ldots,N \tag{5.23}$$

where N = number of feature vector components

$w_{i,j}$ = i'th weight component of W_j, where W_j is the weight vector after the j'th prototype

The variability of each intercept is given by

$$\sigma_i = \frac{\left[\sum_{j=1}^{J}(A_{i,j} - \bar{A}_i)^2\right]^{\frac{1}{2}}}{\sqrt{J}\ \bar{A}_i} = \frac{A_i^2 - \bar{A}_i^2}{\bar{A}_i} \tag{5.24}$$

where J = number of prototypes

$$\bar{A}_i = \sum_{j=1}^{J} \frac{A_{i,j}}{J}$$

$$\overline{A_i^2} = \sum_{j=1}^{J} \frac{A_{i,j}^2}{J}$$

Thus for each y_i both the first and second moments of $A_{i,j}$ must be accumulated. If the J'th prototype is misclassified the accumulated moments are given by

$$\sum_{j=1}^{J} A_{i,j} = \sum_{j=1}^{J-n} A_{i,j} + (n-1)(A_{i,J-1}) + A_{i,J} \qquad i=1,\ldots,N \tag{5.25a}$$

$$\sum_{j=1}^{J} A_{i,j}^2 = \sum_{j=1}^{J-n} A_{i,j}^2 + (n-1)(A_{i,J-1}^2) + A_{i,j}^2 \qquad i=1,\ldots,N \tag{5.25b}$$

where n = number of prototypes since the last misclassified prototype

If the J'th prototype is not misclassified then the accumulated moments are given by

$$\sum_{j=1}^{J} A_{i,j} = \sum_{j=1}^{J-n} A_{i,j} + n(A_{i,j}) \qquad i=1,\ldots,N \tag{5.25c}$$

$$\sum_{j=1}^{J} A_{i,j}^2 = \sum_{j=1}^{J-n} A_{i,j}^2 + n(A_{i,j}) \qquad i=1,\ldots,N \tag{5.25d}$$

After every C_T source particles the variability of each intercept is found using Equation 5.24 and the accumulated values of Equation 5.25. Each σ_i can be tested separately or combined to give an overall variability as given by

$$\sigma = \frac{\prod_{i=1,N} \sigma_i}{\sqrt{\sum_{i=1}^{N} \bar{A}_i^2}} \quad . \tag{5.26}$$

The time required to do these operations is summarized by the relation

$$T_m \approx t_I \sum_{i=1}^{M_S} {}_i f_C P_i + \frac{t_v}{C_T} \sum_{i=1}^{M_S} B_i \tag{5.27}$$

where t_I = time required per operation to store the first and second moments of the y_i intercept (Equation 5.25)

t_v = time required per operation to calculate the misclassification rate and variability and test for convergence.

Using a FORTRAN code to similate these operations results in the following approximate values for t_I and t_v.

$$t_I \approx 4.4 \times 10^{-7} N \text{ seconds} \tag{5.28a}$$

$$t_v \approx 6.7 \times 10^{-7} N \text{ seconds} \tag{5.28b}$$

where N = number of features

Summary

Combining Equations 5.20, 5.22, and 5.27 results in

$$T_{PR} = T_f + T_p + T_m \tag{5.29}$$

$$\approx t_f P_1 + \sum_{i=1}^{M_S} P_i \left[t_L i f_L + i f_c (t_c i f_L + t_I) + \frac{(t_A Q_i + \overline{C_T}) t_v}{N_i h_i} \right]$$

The above parameters fall into the following groups:

(1) t_A, t_L, t_C, t_I, t_V are constants which depend only on the programming efficiency used in the pattern classifier

(2) N_i, Q_i, h_i, ${}_if_c$, M_s are parameters which depend primarily on the characteristics of the Monte Carlo problem involved

(3) t_f, ${}_if_L$, C_T are parameters whose values depend on user specification.

The timing parameters from Section 4.5 and Equation 5.28 are summarized in Table 5.2 for the deterministic classifier and N features. If it is assumed that the 4000 prototypes found necessary for convergence in Chapter IV are characteristic of all splitting surfaces (actually it should be less for each new surface that is learned since the initial guess for W improves with the number of splitting surfaces) and if no buffer zones are used (${}_if_L = 1.0$), Equation 5.29 reduces to

$$T_{PR} \approx 4\times10^{3}t_f + 4\times10^{-3}\sum_{i=1}^{M_s} 0.3+1.2\,{}_if_C + N(0.7+2.7\,{}_if_C) + \frac{\left(4.10_i + \frac{0.67N}{C_T}\right)}{N_i h_i} \quad (5.30)$$

If it is further assumed that:

(1) ${}_if_c = 0.3$ for all i (i.e. a large amount of overlap)

(2) $C_T = 10$ (a very conservative choice considering the value of P_i)

(3) $N = 5$ (the maximum using unreduced pattern space is 8)

Table 5.2

t_A	4.1×10^{-6}
t_L	$1.0\times10^{-6} + (N-1).7\times10^{-6}$
t_C	$3.5\times10^{-6} + (N-1)2.3\times10^{-6}$
t_I	$4.4\times10^{-7}N$
t_V	$6.7\times10^{-7}N$

Equation 5.30 reduces to

$$T_{PR} \approx 4\times10^{3}t_f + 3.28\times10^{-2}M_s + 4\times10^{-3}\sum_{i=1}^{M_s}\frac{4.1Q_i + 0.335}{N_i h_i} \qquad (5.31)$$

The value of $N_i h_i$ will in general be greater than $.5P_i$ for initial surfaces. However as surfaces progress this value will decrease since splitting will cause more particles in the more important classes. Assuming the very conservative value $N_i h_i = .1P_i$ reduces Equation 5.31 to

$$T_{PR} \approx 4\times10^{3}t_f + 3.28\times10^{-2}M_s + 10^{-5}\sum_{i=1}^{M_s}(4.1Q_i + 0.335) \qquad (5.32)$$

The value of Q_i will seldom be over 1 (for any problem requiring variance reduction) and therefore the third term can be ignored compared with the second resulting in

$$T_{PR} \approx 4\times10^{3}t_f + 3.28\times10^{-2}M_s \qquad (5.33)$$

Although Equation 5.33 is only a very rough approximation it is still useful as an indicator of the amount of time spent for learning surfaces. Furthermore, the assumptions have been made in such a manner so as to overestimate T_{PR}. Choosing any reasonable value for t_f (10^{-4} to 10^{-6}) and M_s (<10) and allowing for approximation errors of a factor of ten still results in a T_{PR} on the order of 10 seconds. For Monte Carlo runs of many minutes this time is quite small compared with the time saved by variance reduction.

5.3.3) Effectiveness

Geometry splitting and energy splitting techniques have proven to be quite effective techniques for a great variety of Monte Carlo problems.[6] Since state space splitting is closely related to these techniques the effectiveness of learning state space splitting surfaces by pattern recognition is described relative to conventional techniques. This results in the relation

$$E_R = \frac{N_e \, \Delta t_e}{N^*_e \, \Delta t^*_e} \tag{5.34}$$

where E_R = effectiveness of state space splitting using surfaces learned by pattern recognition

N_e = number of source particles required to achieve the desired relative error using state space splitting

N_e^* = number of source particles required to achieve the desired relative error using conventional splitting

Δt_e = computer time required per particle when state space splitting is used

Δt_e^* = computer time required per particle when conventional splitting is used.

If t_B is the computer time required per particle to perform the Monte Carlo calculations without any form of splitting, then

$$\Delta t_e^* = \Delta t_g + \Delta t_B = \Delta t_1 + \Delta t_2 + \Delta t_3 + \Delta t_B \tag{5.35}$$

$$\Delta t_e = \Delta t_s + \Delta t_B + \frac{T_{PR}}{N_e} = \Delta t_4 + \Delta t_1 + \Delta t_B + \frac{T_{PR}}{N_e} \tag{5.36}$$

where Δt_g, Δt_1, Δt_2, Δt_3 are defined by Equation 5.13.

Δt_s, Δt_4 are defined by Equation 5.15.

T_{PR} is given by Equation 5.29.

Substituting Equations 5.35 and 5.36 into Equation 5.34 results in

$$E_R = \frac{N_e(\Delta t_4 + \Delta t_1 + \Delta t_B) + T_{PR}}{N_e^*(\Delta t_3 + \Delta t_2 + \Delta t_1 + \Delta t_B)} \tag{5.37}$$

The amount of variance reduction is indicated by the ratio

$$R = N_e/N_e^* \tag{5.38}$$

which measures how much more efficient for reducing the variance state space splitting is than conventional splitting. The values of N_e and N_e^* can be further reduced to

$$N_e^* = N_A - \Delta N_C \tag{5.39}$$

$$N_e = N_A - \Delta N_S + \Delta N_L \tag{5.40}$$

where N_A = number of source particles required to achieve the desired relative error if neither conventional nor state space splitting is used

ΔN_C = reduction of N_e^* due to variance reduction when using conventional splitting

ΔN_S = reduction of N_e due to variance reduction when using state space splitting, assuming the splitting surfaces are known <u>prior</u> to the Monte Carlo calculation.

ΔN_L = increase in N_e due to the learning of splitting surface.

If the state space splitting surfaces are known <u>prior</u> to the Monte Carlo calculation ($\Delta N_L = 0$), then in general $R < 1$. This is because state space splitting includes the advantages of conventional splitting (i.e., ΔN_C) as well as the additional advantages of splitting in time and direction space (i.e., $\Delta N_S > \Delta N_C$). Through proper feature selection optimal combinations of state space parameters can be used which further

increases ΔN_S.

When splitting surfaces must be learned, the value of $(\Delta N_S - \Delta N_L)$ varies throughout the calculation starting at zero and increasing to ΔN_S for which splitting surfaces are known a priori. In this case R could possibly be >1 for some Monte Carlo problems; however, by using decision surfaces for splitting before they have converged, ΔN_L can be greatly reduced. Monte Carlo problems exist for which conventional splitting is inadequate; whereas state space splitting appears to be quite attractive (such problems are now usually split into two separate problems where the tally from one provides the source to the other). For this type of problem R<<1. If several similar Monte Carlo problems are being calculated, splitting surfaces can be learned in the first problem and used in the others, thus making $\Delta N_L = 0$ for subsequent similar problems. The value of ΔN_L depends on the amount of learning performed per source particle which in turn depends on the number of prototypes created per source particle, N_t (see Section 5.3.2). Since prototypes are created at collisions, problems in which the particles have a high number of collisions before leaving the system result in converged splitting surfaces after fewer source particles. This point is illustrated in Chapter IV by comparing the number of source particles until convergence (relative to each problem) for the problems described in Sections 4.2.3 and 4.4.3:

	Collisions/Source Particle	No. Source Particles until Convergence
(4.2.3)	100	40
(4.4.3)	12	300

Problems which do not have a large number of collisions are often problems which do not require much variance reduction if any. If a problem containing few collisions requires variance reduction, the learning technique can be modified to produce prototypes at surface intersections as well as collisions. However, in general problems containing many collisions are the most suitable for learning state space splitting surfaces. For many problems Δt_B will be much greater than the other time parameters Δt_1, Δt_2, Δt_3, Δt_4 and T_{PR}/N_e thus reducing Equation 5.37 to

$$E_R \approx \frac{N_e}{N_e^*} = R \tag{5.41}$$

For problems in which this is not true, the time parameters of Equation 5.37 must be taken into consideration. Since these parameters and R depend upon many problem characteristics and user supplied input, it is ludicrous to propose that E_R will always be less than 1.0. However, from the results and analysis of this chapter and Chapter IV, it does appear that for many problems, this will be the situation. Only a great number of computational experiments using a general purpose Monte Carlo code will provide the data necessary to define for which problem types $E_R < 1$.

5.4) Limitations of this Research

The research described in this thesis represents an initial investigation into the area of pattern recognition applied to state space splitting surface identification. The following sub-sections include the more important limitations of this research and the learning of state space surfaces.

5.4.1) Classifier Selection

Both the deterministic and statistical classifiers investigated in Chapter IV are of the same construction in that they assume a linear form of g(Y). It is very unlikely that any optimal splitting surface in state space having overlapping distributions will be linear. It has been assumed that in most cases a linear surface is a sufficient approximation to adequately split particles. Furthermore, feature space can be altered to improve this approximation. Experience with the technique will be the ultimate test in determining whether or not linear discriminant functions are sufficient.

There are many different types of classifiers for Case D and E data (see Section 3.1.2), for example, the expansion given by Equation 3.4. Some other technique may prove more attractive for classifying the types of distributions resulting from Monte Carlo calculations. In Chapter IV it was found that the statistical classifier used is hindered by its accumulated overshoot and that the deterministic classifier has

the characteristic that for overlapping $p(C_i|Y)$, the variability never goes to zero. An attractive alternative might be a hybrid classifier that uses deterministic classification during the initial approach and then uses statistical classification until the final convergence criterion is met.

5.4.2) User Input

One of the incentives for using pattern recognition to identify splitting surfaces is to relieve the user of the burden of providing a priori information to the Monte Carlo calculation. However, as is described in Section 5.2 the user must provide the values of f_i, u_i^{min}, u_i^{max}, and d_i for the feature selector (see Figure 5.4). Once the f_i's have been selected, the values of u_i^{min}, u_i^{max}, and d_i are determined and can be easily supplied. The main problem then is selecting a good choice for the f_i This choice will have to be based upon intuition and experience using heuristics similar to those described in Section 5.2. Although certainly a non-trivial operation, the above selection is a qualitative decision for which humans are much more suited than computers. Similarly, the learning of a splitting surface using the f_i is a quantitative decision for which the computer is much better suited. In conventional splitting the user is required to perform both these tasks.

There is other input which could be made optional as user input and includes:

(1) Initial selection of W

(2) Convergence criteria for the variability and misclassification rate

(3) A choice for R (see Equation 5.2)

(4) The limit on the number of splitting surfaces, M_s (see Section 5.3.2)

(5) For multiple tallies, the values of p_i (see Equation 5.5)

(6) For multiple point detector tallies, the values of m_i (see Equation 5.8)

It would be a little presumptuous to expect the general Monte Carlo user to input all this additional information. The default values can be chosen as described in Sections 5.1 and 5.2; however, after experience has been gained with the technique better values will undoubtedly be found.

5.4.3) Range of Application

As with any variance reduction technique this one has the problem that the user is not certain when to use it. The result is that problems not requiring splitting or for which splitting will not help may be slowed down. This technique does have the advantage that by using the F_2 test (see Equation 5.2), the code will stop learning surfaces if they are not needed. However, this test is made only after the first splitting surface has been learned.

5.5) Summary

A systematic approach has been described in Section 5.1 for collecting prototypes, determining classification, and managing the selection of splitting surfaces. The feature selection process can be performed by a user-supplied subroutine containing the information shown in Figure 5.5. The selection of the features (f_i) is the most important user input and must be based on the intuition or experience of the user. Recommended heuristics have been given to aid in this selection. Although an absolute statement cannot be made about the timing and effectiveness, by using constants derived in this research and assuming reasonable values for problem parameters, it does appear that the technique should be quite attractive compared with conventional splitting techniques.

VI. Conclusions and Recommendations

In Section 1.4 four steps toward demonstrating "proof of principle" were stated. These steps are repeated below along with the conclusions drawn from this research.

(1) Develop a pattern recognition system that can be used to learn splitting surfaces in Monte Carlo transport calculations.

A scheme has been designed whereby pattern recognition can be used to learn splitting surfaces in state space. Training sets or prototypes are required before the splitting surface can be learned. These prototypes consist of points in state space (collision points were used in this research) with their corresponding importances. The importances are determined by measuring the probability that a particle leaving a given point contributes to the Monte Carlo tally under consideration. Both statistical and deterministic classifiers learn splitting surfaces satisfactorily. From the examples of this research it appears that the number of source particles required to learn a surface is small enough to make the technique useful for practical applications.

(2) Investigate the performance of statistical and deterministic classifiers when used to recognize splitting surfaces. Analyze the sensitivity of the parameters involved.

The deterministic classifier using the fractional correction rule is superior to the statistical one with respect to per-

formance, computer time, sensitivity to the learning parameter, and sensitivity to initial conditions. The learning parameter, λ, affects the rate of convergence of the splitting surface. Although the optimum choice of a value for λ depends upon the characteristics of the Monte Carlo problem under investigation, it was found that a single λ (λ=.05) can be used with the deterministic classifier without seriously penalizing convergence. The convergence is also affected by the initial guess of the splitting surface location. Initial splitting surfaces should be located in the least important region of state space. Each component, y_i, of feature space should be normalized to the interval from zero to one and should be selected so that as y_i increases, the importance decreases. As a result the least important point in feature space is given by $Y = [y_1, y_2, \ldots, y_N] = [1, 1, \ldots, 1]$.

(3) Propose a system for applying pattern recognition to a general purpose Monte Carlo code.

A system for implementing pattern recognition techniques in a general purpose Monte Carlo code is given in Chapter V. The scheme requires the user to supply the information needed by the feature selector. Other parameters can be supplied by the user or by default values. Selection of these values must be gained from experience with the technique.

(4) Analyze the effectiveness that can be expected by using pattern recognition as a variance reduction technique.

The effectiveness of the technique is problem dependent and user input dependent. The parameters involved have been identified and values have been determined for those parameters which are independent of the Monte Carlo problem or the user input. Based upon these values and reasonable choices for the other parameters, it appears that the technique should be more useful than conventional splitting. The problem scope and degree of effectiveness can only be determined by applying the technique to many Monte Carlo problems using a multi-purpose code.

Based upon these conclusions, it is recommended that a multi-purpose Monte Carlo code such as MCN at the Los Alamos Scientific Laboratory be modified to split in state space instead of the presently used splitting. Having done this the system described in Chapter V can be included to provide the splitting surfaces. The deterministic classifier described in this research can be used directly except that an allowance must be made for variable dimensioned feature vectors. The modified code should be used on a range of Monte Carlo problems to determine many of the unknown problem related parameters described in this research and to determine the success of the technique for reducing computer time. The code should then be used routinely to determine user related problems and to gain experience with the technique. Only after such an investigation can the success of the technique be determined.

Acknowledgements

I wish to thank Dr. Billy V. Koen for his continual guidance as my supervising professor during both my M.S. and Ph.D. studies. Through his intellectual curiosity and professional honesty, Dr. Koen has been a strong influence on my engineering education as well as my philosophical views. I wish to also thank my other committee members at the University of Texas, Dr. E. Linn Draper, Dr. Baxter F. Womack, Dr. Robert N. Little, and Dr. Joe O. Ledbetter for their help with this dissertation.

I am very grateful to the many people at the Los Alamos Scientific Laboratory (LASL) who made this research possible. In particular I wish to thank Dr. Edmond D. Cashwell for making it possible for me to perform this research at LASL and for the many hours he has spent reading and discussing this dissertation with me. I consider it an honor to have a man of Dr. Cashwell's capabilities on my committee.

This research was initially supported by the Atomic Energy Commission by the award of a Traineeship in Nuclear Science and Engineering administered by the University of Texas at Austin. Preliminary computer calculations were performed at the University of Texas Computation Center. The calculations and results reported in this dissertation were performed at the Los Alamos Scientific Laboratory Central Computer Facility.

Appendix A. Monte Carlo Sampling for Neutron Transport

The purpose of this Appendix is to give the reader a "physical feel" for the processes involved in using the Monte Carlo method for neutron transport. It is not the purpose of this Appendix to derive and prove the mathematical basis of Monte Carlo. The structure of this description of Monte Carlo is modeled after reference 7 with many of the ideas taken from reference 34. These references should be consulted if more details are desired. Section A.1 presents the basic principles of Monte Carlo as used for neutron transport. The remaining sections treat individual topics of neutron transport and give examples of how the physics of the problem is programmed into a statistical model.

A.1 Basic Principles

The basic problem of neutron transport consists of calculating the number of neutrons that are emitted from a source, undergo any number of collisions with atomic nuclei, and finally arrive at some region of state space (defined below) of interest to the person trying to solve the transport problem. The term state space refers to the complete description of the physical properties of the neutrons as given by:

$$(x,y,z,u,v,w,E,t)$$

where x,y,z are the coordinates of the neutron in a Cartesian Coordinate System (see Figure A.1)

u,v,w are the direction cosines of the neutron with respect to the x,y, and z axis (see Figure A.1)

E is the kinetic energy of the neutron

t is the time elapsed since the process under study has begun

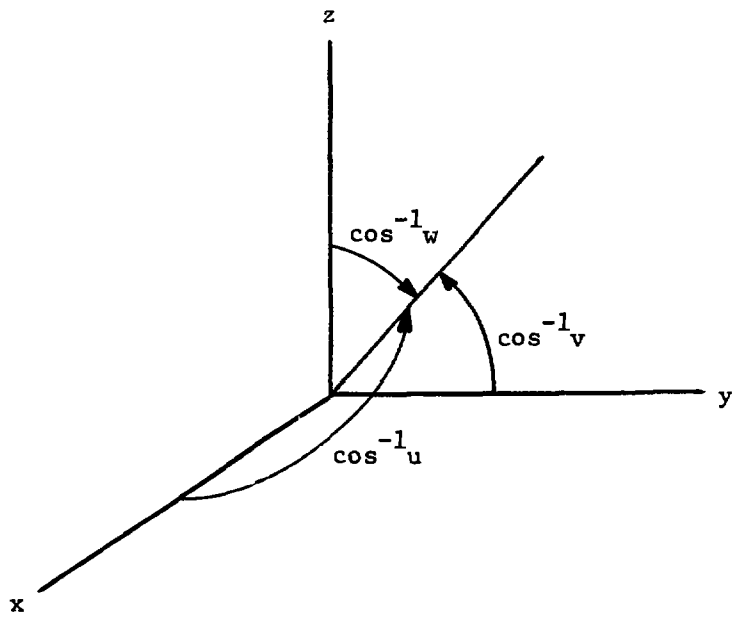


Figure A.1 Geometry Coordinates

Most methods used to solve the transport equation treat all the neutrons as a whole (i.e., they solve for the flux of neutrons) and then

proceed to find the value of the flux at all points in state space. The Monte Carlo method works quite differently in that neutrons are treated one at a time independently of the others. After calculating the behavior of many individual neutrons, the average or mean behavior of the neutron "sample" is determined. This mean value is used as an approximation of the value that would be obtained if an infinite number of neutrons (i.e., the total population) had been treated. Corresponding to this mean estimate is a probabilistic error based upon the statistical behavior of the neutrons.

Neutron transport is ideally suited for the Monte Carlo method since most of the physical processes involved are probabilistic in nature. For example:

(1) a neutron undergoes a collision per unit path length with probability Σ_t.

(2) the probability of a particular nuclear reaction, i, occurring during a collision with a nucleus is σ_i/σ_t.

(3) the probability of scattering at an angle θ (in the center of mass system) is determined by $\sigma(\theta)$.

(4) the probability per unit time that a nucleus decays is given by the decay constant λ.

(5) the probability of a prompt fission neutron being emitted with energy between E and E+dE is given by $\chi(E)dE$.

Figure A.2 illustrates the major operations performed by a Monte Carlo program when used to follow neutrons through state space. These operations are introduced below and described in more detail in the following sections.

(1) Source (Section A.2). The source of neutrons is usually described as a user supplied subroutine in a Monte Carlo program. It is the purpose of this subroutine to describe the initial state space parameters (x,y,z,u,v,w,E,t).

(2) Collision or Escape Decision (Section A.3). The geometry of any Monte Carlo problem is divided into regions or cells which may or may not correspond to material regions. This decision process consists of "sampling" the probability of having a collision as determined by Σ_t of the material in the given region. It is for this decision that the majority of geometry calculations are made since the distance from the present (x,y,z) to the next intersection with the cell boundary must be known.

(3) Entering a New Region (Section A.4). This process consists of determining which region the particle is entering after leaving the present region.

(4) Collision Type (Section A.5). The process for determining which type of reaction occurs is determined by the relationship of the respective cross sections.

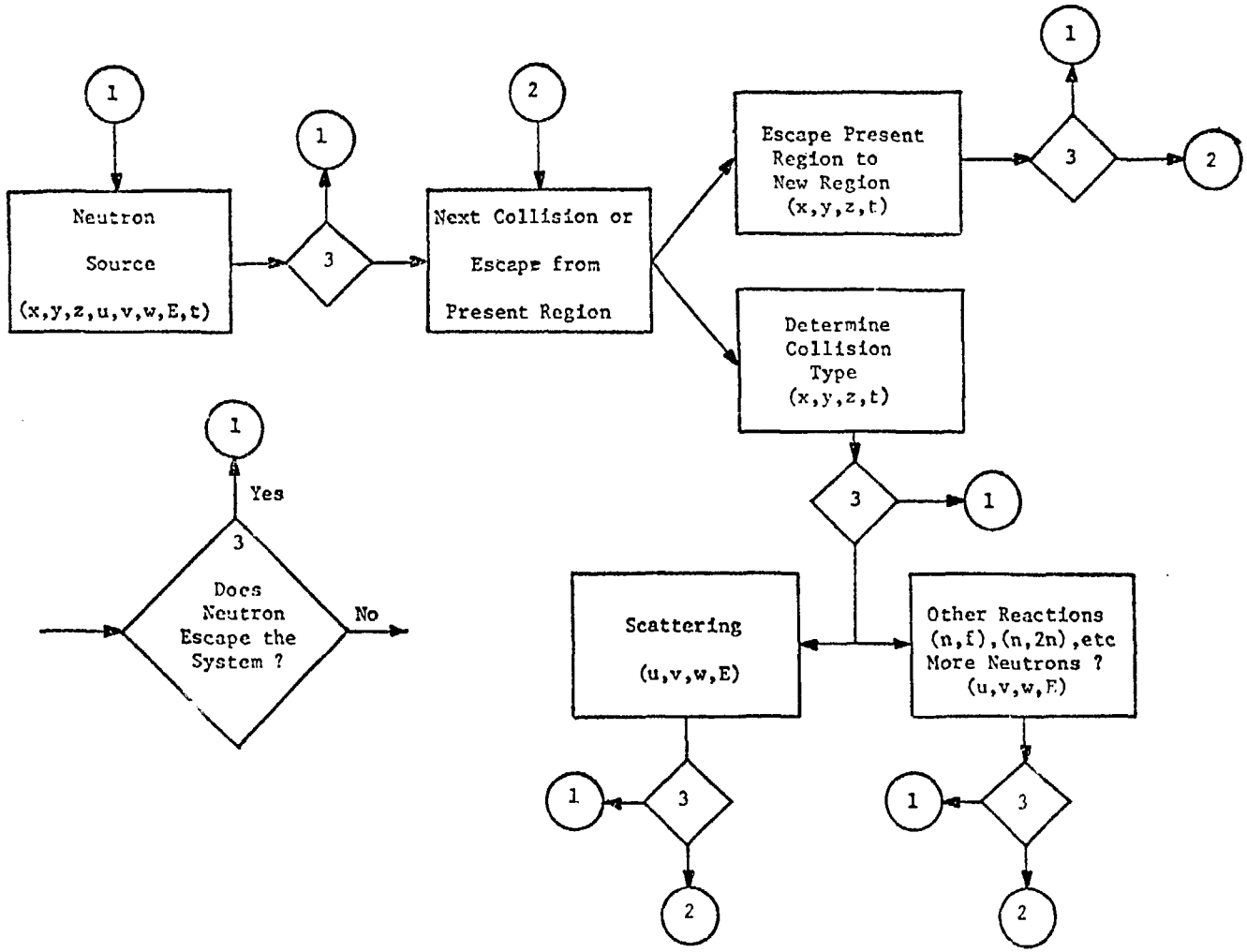


Figure A.2 Major Monte Carlo Operations for Neutron Transport

(5) Scattering (Section A.6). Both elastic and inelastic scattering result in a change in energy which is related to the scattering angle (usually given in the center of mass system). The scattering decision consists of sampling for this scattering angle. The new neutron energy and direction cosines can then be determined from this angle.

(6) Other Reactions (Section A.7). Many other reactions can occur (i.e., fission, [n,2n], etc.) and often lead to the creation of additional neutrons. In such a case an energy E and direction (u,v,w) must be determined for each neutron emitted. In addition, for fission reactions the number of neutrons emitted can be sampled or an average value used.

(7) Tallies (Section A.8). The tally of a Monte Carlo transport problem consists of the "answer" that one is seeking as determined by the mean behavior of the neutron histories followed. Examples of such tallies are: current crossing a particular surface, flux at a point, number of captures in a region, etc. Because of the generality involved, the information necessary to determine the tally can be extracted from any part of the Monte Carlo calculation. Therefore, the tally operation is not shown in Figure A.2.

The following sections frequently refer to the choice of a random number, r, for sampling probability distributions. The mathematical description of "random" can be found in many of the references[7,14,29]. For the purpose of this description it is assumed that the random number, r, has an equal probability of falling between the value of 0 and 1 and that all subsequent random numbers are uncorrelated[14,29].

The coordinate system used in the following description is as shown in Figure A.1 where the angles are in the laboratory system unless otherwise noted. Energy is also relative to the laboratory system.

A.2 The Neutron Source

The sampling of a general neutron source involves the following operations:

(1) sampling of the space coordinates (x,y,z)

(2) sampling of the direction cosines (u,v,w)

(3) sampling of the energy spectrum

(4) determination of the time

The above operations are performed independently and will be described in this section by using several examples. In these examples all sampling will be performed uniformly over the quantities (surface, volume, etc.) of interest.

A.2.1 Space Coordinates

Two examples of sampling the space coordinates (x,y,z) are described in this section: (1) a flat two dimensional surface and (2) a right circular cylinder. Although these examples comprise only a small fraction of the possible sources, the techniques used to sample are similar for all sources and thus provide the reader with a "feel" for the processes involved.

In many Monte Carlo problems neutrons are emitted from a flat two dimensional surface. Examples of such sources are cross sections of a neutron beam, the side of a rectangular reactor, a spill of radioactive liquid on a flat surface, etc. Three examples are shown in Figure A.3 - a rectangle centered at the origin, a circle centered at the origin, and a general function f(y,z).

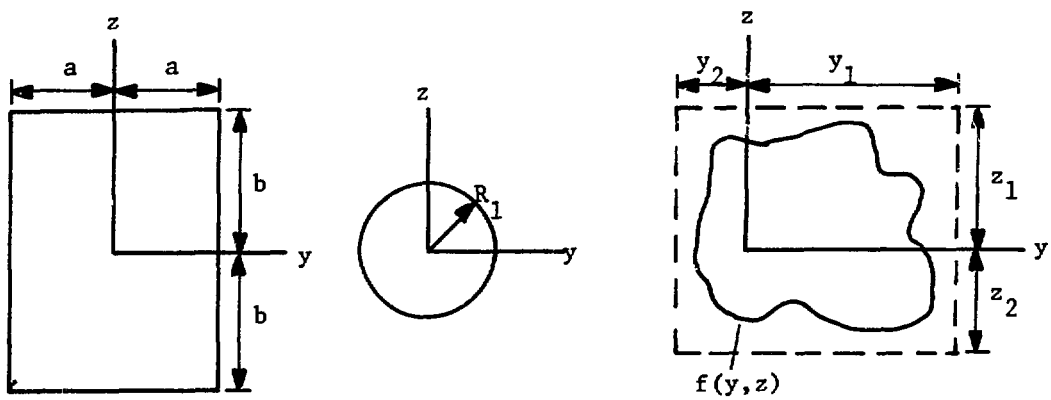


Figure A.3 Sampling Two-Dimensional Sources

The rectangular source is the easiest to sample and consists of:

(1) choosing a point randomly along the y axis between $y = -a$ and $y = +a$ and

(2) choosing a point randomly along the z axis between $z = -b$ and $z = +b$.

This is done by performing the operations shown in Figure A-4 which results in uniform sampling over the area of the rectangle.

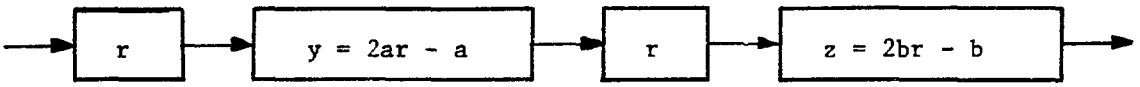


Figure A.4 Sampling a Rectangular Two Dimensional Source

The sampling of the circle is facilitated by using polar coordinates as shown by Figure A.5. This is done since one wishes to sample evenly over the area of the circle and the area of a circle is more easily defined by polar coordinates.

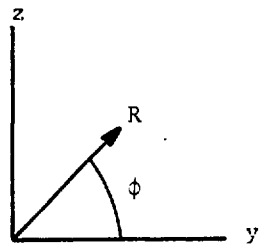


Figure A.5 Coordinates Used for Sampling a Circle

The sampling of the circle consists of:

(1) choosing a R randomly between R = 0 and R = R_1 (actually R^2 is chosen randomly between $R^2 = 0$ and $R^2 = R_1^2$)

(2) choosing a ϕ randomly between $\phi = 0$ and $\phi = 2\pi$

(3) converting polar coordinates (R,ϕ) to rectangular coordinates (y,z)

These operations are performed as shown by Figure A.6.

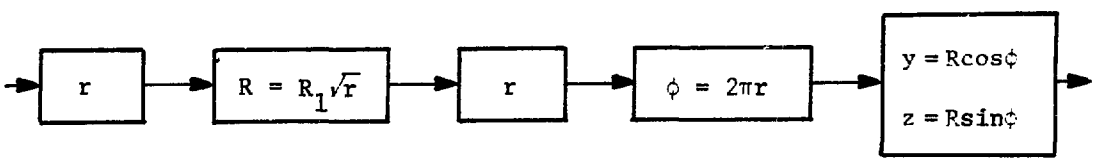


Figure A.6 Sampling a Circular Two Dimensional Source

The sampling of the general function f(y,z) is usually best solved by the method of "rejection". This consists of defining a rectangular area completely enclosing f(y,z) and:

(1) sample the rectangle the same as for the earlier case resulting in a (y',z')

(2) test to see if this (y',z') lies within the area defined by f(y,z). This is done by checking the sign of f(y',z'). [assuming that (y',z') which are inside f(y,z) = 0 result in f(y',z') < 0]

(3) if (y',z') does not lie within the area defined by f(y,z) then repeat the above steps, otherwise use the value of (y',z').

Figure A.7 illustrates this process.

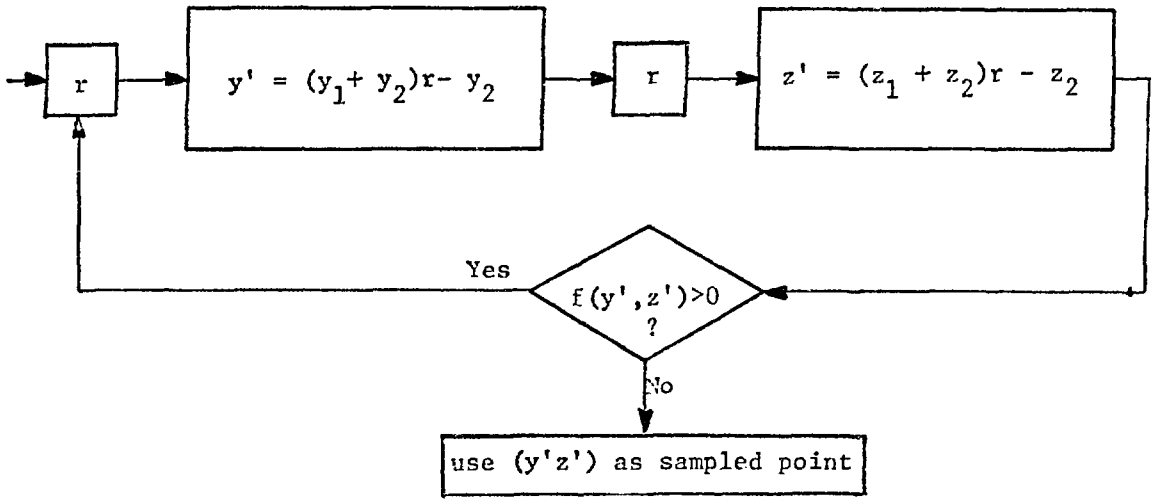


Figure A.7 Sampling by the Rejection Technique

A source within a right circular cylinder (see Figure A.8) is frequently encountered in reactor related problems dealing with cylindrical fuel rods and pellets. This problem consists of sampling a volume as opposed to an area in the earlier examples.

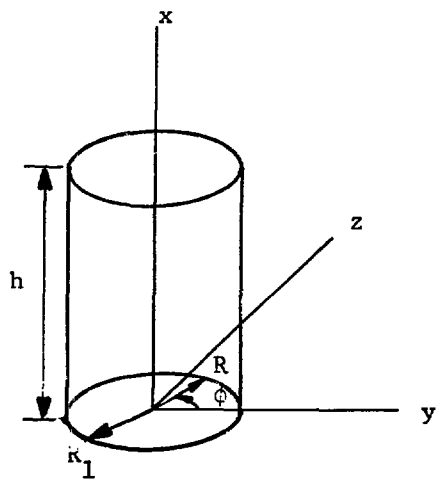


Figure A.8 Right Circular Cylinder

Once again cylindrical coordinates are preferred as shown by Figure A.5. A ϕ and R are selected as given by Figure A.6 followed by the selection of x (see Figure A.9).

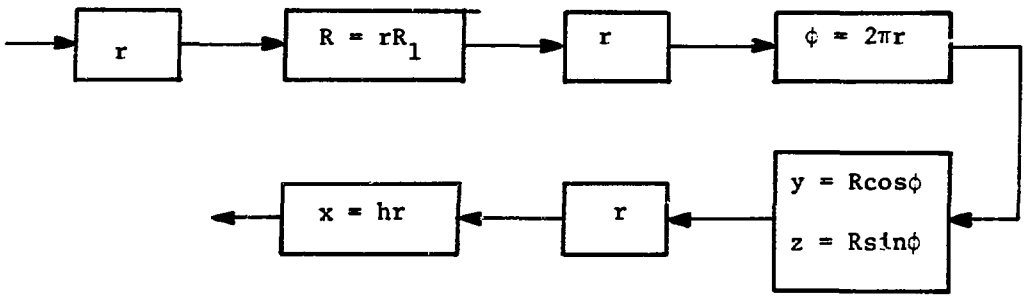


Figure A.9 Sampling the Right Circular Cylinder

A.2.2 Direction Cosines

This section considers two different sources: (1) a directed beam and (2) an isotropic source. The directed beam as shown by Figure. A.10 has the trivial solution

$$u = w = 0$$

$$v = 1 .$$

Since each neutron has the same direction there is no distribution to sample.

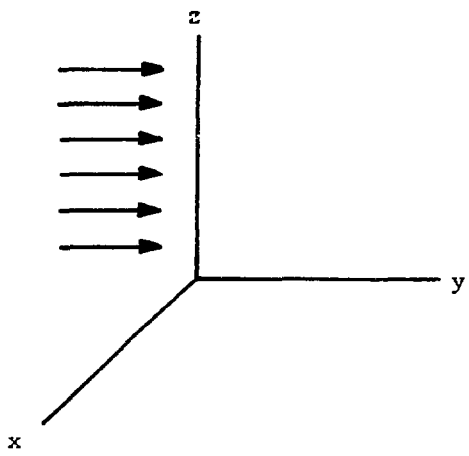


Figure A.10 The Directed Beam Source

The problem of selecting a (u,v,w) for an isotropic source is equivalent to that of choosing a point (u,v,w) uniformly distributed on the unit sphere $u^2 + v^2 + w^2 = 1$ (see Figure A.11).

The unit area on this surface is given by $\sin\theta\, d\theta\, d\phi$. This distribution can be sampled by the following operations (see Figure A.12):

(1) select a w randomly between -1 and +1

(2) determine ρ (see Figure A.11)

(3) select a ϕ randomly between 0 and 2π

(4) convert from coordinates (ϕ,ρ) to (u,v)

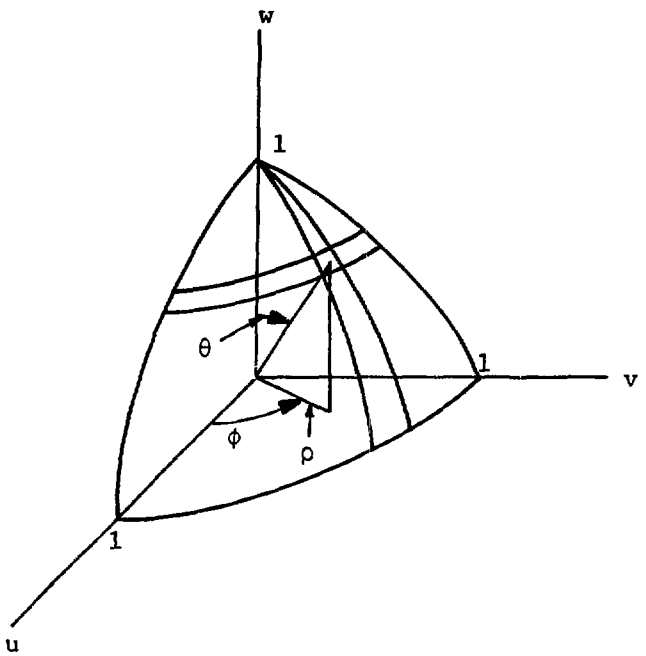


Figure A.11 The Isotropic Source

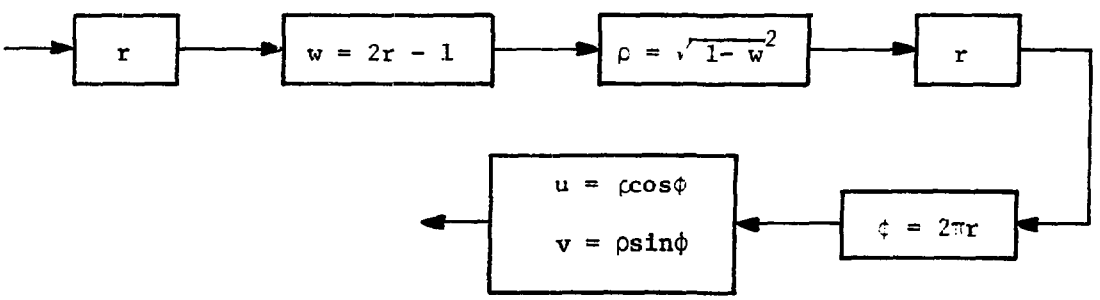


Figure A.12 Sampling an Isotropic Source

A.2.3 Energy Distribution

The energy of source neutrons is frequently a single value for all particles and thus has no distribution. The fission spectrum is used for some sources and is discussed in section A.7. Many times experimental data results in an energy distribution which consists of the number of neutrons lying within specified energy intervals

$$N(E_i,E_{i+1}) = \text{number of neutrons having energy between } E_i \text{ and } E_{i+1} \quad (A.1)$$

where $i = 0, \ldots M-1$

and E_0 = minimum energy of neutrons

E_M = maximum energy of neutrons

In such a case the data must first be normalized by dividing by the total number of neutrons

$$p(E_i,E_{i+1}) = \frac{N(E_i,E_{i+1})}{\sum_{i=0}^{M-1} N(E_i,E_{i+1})} \quad (A.2)$$

which results in the probability of a source neutron having an energy between E_i and E_{i+1}. The probability distribution is given by summing the $p(E_i,E_{i+1})$ as given by

$$P(E_{i+1}) = \sum_{j=0}^{i} p(E_j,E_{j+1}) \quad (A.3)$$

where $P(E_M) = 1.0$. A random number is then selected between 0 and 1.0 and if

$$P(E_i) < r \leq P(E_{i+1}), \qquad i = 0, \ldots M-1 \qquad (A.4)$$

$$\text{where } P(E_0) = 0.0,$$

then the energy of the neutron lies between E_i and E_{i+1}. The energy can be further specified by linearly interpolating between $P(E_i)$ and $P(E_{i+1})$ (see Figure A.13) resulting in

$$E = E_1 + \frac{(r-P_1)\ (E_2-E_1)}{(P_2-P_1)} \qquad (A.5)$$

where r is the same random number used in equation A.4.

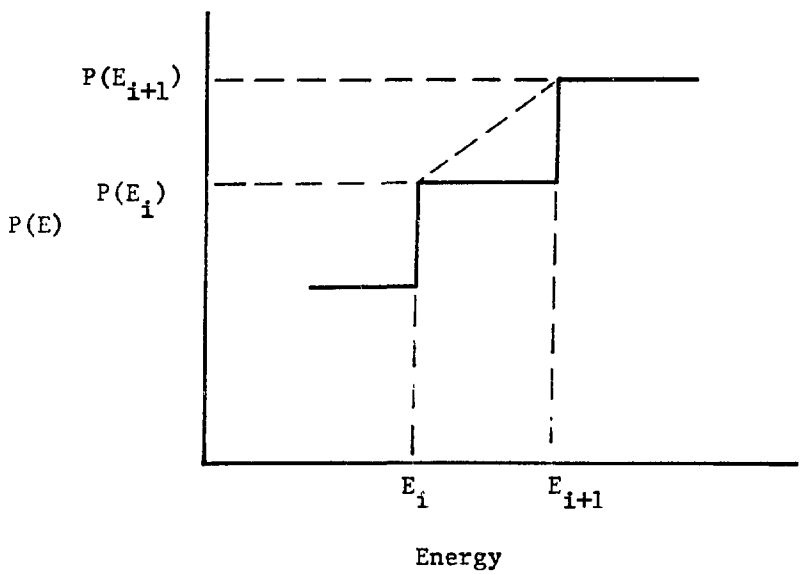


Figure A.13 Linear Interpolation to Determine the Energy

A.2.4 Time

Source neutrons are generally considered to be emitted at time $t = 0$; however, there are instances in which the source is time dependent. An example of this is the emission of delayed neutrons which are emitted as

$$dn(t) = \lambda n_0 e^{-\lambda t} dt \tag{A.6}$$

where $dn(t)$ = the number of neutrons emitted between time t and $t + dt$

n_0 = the number of precursor nuclei at time $t = 0$

λ = the decay constant of the precursor

The probability that a neutron is emitted between t and $t + dt$ is given by

$$p(t)dt = \lambda e^{-\lambda t} dt \tag{A.7}$$

The probability distribution function is given by

$$\int_0^t p(t')dt' = 1 - e^{-\lambda t} \tag{A.8}$$

Sampling this distribution results in

$$t = \frac{\ln(1-r)}{-\lambda} \quad \frac{\ln(r)}{-\lambda} \tag{A.9}$$

for the time of emission.

A.3 Collision or Escape from the Present Region

After a collision or after being emitted from the source a neutron is positioned at (x,y,z) and headed in the direction indicated by (u,v,w). The geometry of the problem is divided into cells or regions (see Appendix J) which are bounded by surfaces. The equations of the N_i surfaces bounding the i'th cell are given by

$$\begin{aligned} f_{i,1}(x,y,z) &= 0.0 \\ f_{i,2}(x,y,z) &= 0.0 \\ &\vdots \\ f_{i,N_i}(x,y,z) &= 0.0 \end{aligned} \tag{A.10}$$

where N_i = total number of surfaces bounding the i'th cell.

If the neutron lies in the i'th cell then the first operation that must be performed is to find the intersections of the neutron's present line of flight [as given by (u,v,w) and (x,y,z)] with the N_i surfaces as

given by Equation A.10. Only positive real intersections are of interest and of these the one that yields the shortest distance from the neutron's previous position is used. This distance d_{max} is the maximum distance the neutron can travel before it leaves the present region.

The probability that a neutron undergoes a collision between c and c + dc is given by

$$p(c)dc = {}_i\Sigma_t \; e^{-{}_i\Sigma_t c} dc \qquad (A.11)$$

where ${}_i\Sigma_t$ is the total macroscopic cross section of the material in the i'th cell

c is the distance as measured from the current position (x,y,z) of the neutron to the point of collision

Using equation A.11 the probability distribution function is given by

$$P(c) = \int_0^c p(c')dc' = 1 - e^{-{}_i\Sigma_t c} \qquad (A.12)$$

Sampling this distribution for the distance to the point of collision results in

$$c = \frac{\ln(r)}{-{}_i\Sigma_t} \qquad (A.13)$$

The decision as to whether the neutron either collides or escapes can then be made as follows:

$$\text{if } c \leq d_{max}$$

then the neutron undergoes a collision at

$$(x',y',z') \tag{A.14}$$

where $x' = x + uc$

$y' = y + vc$

$z' = z + wc$

$$\text{if } c > d_{max}$$

then the neutron escapes the present cell at the point

$$(x',y',z') \tag{A.15}$$

where $x' = x + ud_{max}$

$y' = y + vd_{max}$

$z' = z + wd_{max}$

A.4 Entering a New Region

When a neutron is crossing an intersection, the time must be updated as given by

$$t' = t + d_{max}/Vel \tag{A.16}$$

where t' = new time

d_{max} = distance traveled since t was calculated

Vel = velocity of the neutron = $\sqrt{\frac{2E}{m_n}}$

m_n = mass of a neutron.

The decision as to which region is being entered is made by comparing the sense (see Appendix J) of the neutron's present position (x',y',z') with the senses of points in neighboring cells (the sense with respect to the surface which the neutron is on is not checked). Since the senses of a cell are unique, only one cell will agree with this sense check. This cell and its material are used as the next cell.

A.5 Collision Type

After it has been determined that a collision has occurred (see Appendix A.3) the time of the particle must be updated as given by equation A.1 except that c (see equation A.13) is used instead of d_{max}. The following decisions must be made:

(1) which nuclide is the neutron interacting with (for materials composed of more than one nuclide)

(2) what type of reaction occurs.

If the total cross section of nuclide j of the i'th cell is given by ${}_{i,j}\Sigma_t$, then the probability that the neutron interacts with nuclide j is given by

$$p_j = \frac{{}_{i,j}\Sigma_t}{\Sigma}, \qquad j = 1,\ldots,M_i \tag{A.17}$$

where ${}_i\Sigma_t$ is defined by equation A.11 and M_i is the number of nuclides making up the material in cell i. Adding the probabilities given by p_j results in the probability distribution function

$$P_j = \sum_{i=1}^{j} p_i. \tag{A.18}$$

The P_j is sampled by selecting a random number r and

if $P_k < r < P_{k+1}$ $\qquad k = 0,\ldots,M_i-1$

where $P_0 = 0.0$

then the neutron interacts with the (k+1) nuclide.

The probability of reaction l occurring is given by

$$p_l = \frac{\sigma_{j,l}}{\sigma_{j,t}} \qquad l = 1,L_j \tag{A.19}$$

where L_j = the number of possible reactions for the j'th isotope

$\sigma_{j,l}$ = the microscopic cross section of the l'th reaction of the j'th isotope

$\sigma_{j,t}$ = the microscopic total cross section of the j'th isotope

The resulting probability distribution is given by

$$P_l = \sum_{l=1}^{L_j} P_l \qquad (A.20)$$

and is sampled by the random number r where

if $P_k < r < P_{k+1}$ $\qquad k = 0,\ldots,M_i-1$

where $P_0 = 0.0$

then the neutron undergoes the (k+1)'th reaction

A.6 Scattering

In this section both elastic and inelastic scattering events are described. A scattering event results in a new set of (u,v,w) and a new E. These new coordinates are dependent only upon the scattering angle of the neutron in the laboratory system, (see Figure A.14) and the original (u,v,w). Because scattering is often isotropic in the center of mass (COM) coordinates, the scattering angle in the COM system, θ, is usually used instead of ϕ.

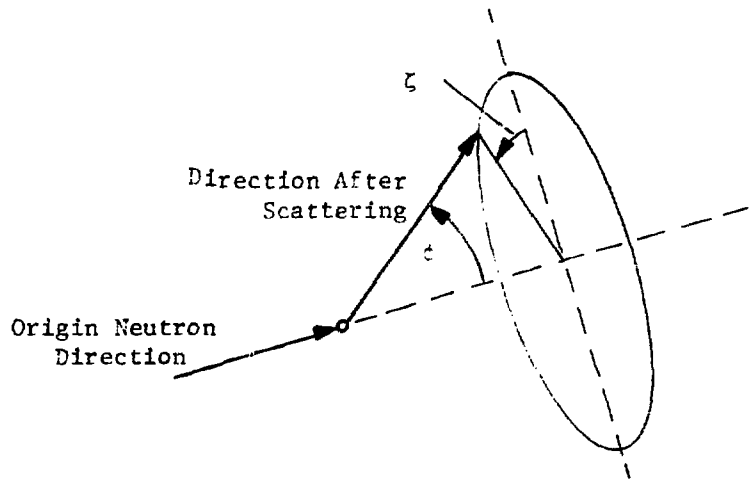


Figure A.14 Scattering in the Lab System

A.6.1 Elastic Scattering

For isotropic scattering in the COM system the scattering angle θ can be sampled by

$$\cos\theta = 2r - 1 \tag{A.22}$$

which when converted to the laboratory system results in

$$\cos\phi = \frac{1 + A\cos\theta}{\sqrt{A^2 + 2A\cos\theta + 1}} \tag{A.23}$$

where A is the mass of the target nucleus in units of the mass of a neutron

The angle ζ (see Figure A.14) is sampled by

$$\zeta = 2\pi r \tag{A.24}$$

The above ζ and cos ϕ can then be converted to a new (u',v',w') by using suitable transformations and the (u,v,w) before the collision.

The emerging neutron energy in the lab system is given by

$$E' = \frac{E}{2}\left[(1 - \alpha)\cos\theta + 1 + \alpha\right] \tag{A.25}$$

where $\alpha = \left(\frac{A-1}{A+1}\right)^2$

E = incoming neutron energy in the lab system

In the event that scattering is not isotropic in the COM system (i.e. $\sigma_s(\theta) \neq \sigma_s/4\pi$) tables of $\sigma_s(\theta)$ vs. θ are usually supplied and can be used in the same manner as the source energy tables (see section A.2.3). Once a value of cos θ has been sampled the process is the same as for isotropic scattering.

The above descriptions have assumed that the bombarded nucleus is stationary in the lab system. If this is not true, a thermal scattering treatment must be used as is described in reference 9.

A.6.2 Inelastic Scattering

The sampling of cos θ and ζ is the same for inelastic scattering as it is for elastic scattering; however, since the nucleus is excited, equations A.23 and A.25 are no longer valid. The energy of the emerging neutron in the COM system after an (n,n') reaction is given by

$$E'_{COM} = \left(\frac{A}{A+1}\right)^2 E + Q \left(\frac{A}{A+1}\right). \tag{A.26}$$

where E is the energy of the incident neutron in the lab system and Q is the Q value of the reaction (Q = rest energy of the nucleus before collision - rest energy after). The value of Q is determined by the particular (n,n') reactions; however, for heavy nuclei the levels of (n,n') reactions can become very dense. As a result Q may in some cases have to be sampled. Once E'_{COM} has been determined, the emergent energy in the lab system can be found by the relation

$$E'_{Lab} = E'_{COM} + \frac{\left[E + 2\cos\theta\ (A+1)\ \sqrt{E\ E'_{COM}}\right]}{(A+1)^2} \tag{A.27}$$

and the scattering angle in the lab system is given by

$$\cos\phi = \sqrt{\frac{E'_{COM}}{E'}}\ \cos\theta + \sqrt{\frac{E}{E'}}\ \left(\frac{1}{A+1}\right) \tag{A.28}$$

After determining ϕ and ζ the new (u,v,w) is found in the same way as for the elastic scattering case.

A.7 Other Reactions

Other reactions that can occur besides scattering events are capture, fission, (n,2n), etc. The capture event is treated differently than any other reaction and involves the concept of the neutron "weight". The weight of a neutron can be thought of as representing a fraction of a neutron. Although only an integer number of neutrons can be transported, a fraction of a neutron can be represented by a neutron with a weight less than one (weights are further discussed in section 2.2). Whenever a neutron undergoes a collision with a nucleus the weight of the neutron is multiplied by

$$\frac{\sigma_t - \sigma_c}{\sigma_t} \tag{A.29}$$

where σ_c is the capture cross section. The reason for this treatment of capture will be given in section 2.2.

Of the other possible reactions only fission will be described in this section. Three features must be sampled for the neutrons emitted from fission:

(1) the number of neutrons emitted

(2) the energy of the neutrons emitted

(3) the directions of neutrons emitted.

The neutrons are emitted isotropically in the lab system resulting in the same sampling scheme shown in Figure A.12 for the new (u,v,w).

The average number of neutrons emitted per fission for an incoming energy E is given by [18]

$$\bar{\nu}(E) = \nu_0 + aE . \tag{A.30}$$

where ν_0 and a are experimentally determined constants which depend on the fissioning nucleus. It is possible to sample the distributions for each reaction based on $\sigma_f(n,n)$, $\sigma_f(n,2n)$, $\sigma_f(n,3n)$, etc. and follow the corresponding number of neutrons. However, it is unnecessary since a single neutron can be created with a weight equal to $\nu(E)$.

The energy distribution of neutrons emitted from fission can be approximated by [18]

$$\chi(E')dE' = .453e^{-1.036E'}\sinh\sqrt{2.29E'} , \tag{A.31}$$

where $\chi(E')dE'$ is the probability that a fission neutron is emitted with energy between E and E'+dE' (lab system)

The value of E is found by sampling the probability distribution function

$$r = \int_0^E \chi(E')dE' \ . \qquad (A.32)$$

The method for performing this sampling is non-trivial and is given in references 41 and 42.

A.8 Tallies

A Monte Carlo tally can be anything the user wishes to specify. Some commonly used tallies are:

(1) the current of neutrons crossing a specified surface

(2) the number of neutron collisions in a cell

(3) the number of fissions in a cell

(4) the flux of neutrons at a point.

In most cases, the neutron can contribute to the tally more than once during its lifetime. If each contribution to the tally of the n'th neutron is given by $x_{i,n}$ then the total contribution of the n'th neutron is given by

$$x_n = \sum_{i=1}^{C_n} x_{i,n} \ , \qquad (A.33)$$

where C_n is the number of times the n'th neutron contributes to the tally. It is the value of x_n that is used to calculated the mean and variance of the tally.

The tally x_n is evaluated when the neutron has escaped the system (see Figure A.2). The system is defined as that region of state space which is of interest to the tally or tallies under consideration. For example, consider a single tally that consists of counting the number of neutrons with energies between E_1 and E_2 ($E_2 > E_1$) crossing a surface. For this example, whenever a neutron's energy falls below E_1, the neutron can no longer contribute to the tally and therefore has left the system. Similar situations exist for the other state space variables.

The individual contributions $x_{i,n}$ must be saved whenever they occur. For the example tallies presented earlier, this is done

(1) When a neutron crosses a surface bounding two cells, this surface is checked. If the surface corresponds to a tally surface, $x_{i,n}$ is calculated.

(2) When a new cell is entered, a check is made to see whether or not the cell is a tally cell. If it is, then $x_{i,n}$ is calculated each time a collision occurs in the cell.

(3) Same as (2) except for fissions.

(4) After being emitted from the source and after each collision, the probability of being scattered toward the unit area of the point detector is calculated. This probability is multiplied by the probability of the neutron arriving at the point detector uncollided and weight of the neutron prior to the collision resulting in $x_{i,n}$. (The point detector is described further in Section 5.1.3).

Appendix B. Statistical Errors in Monte Carlo

Section B.1 derives the expression for the variance of the sample mean. The central limit theorem is discussed in section B.2.

B.1 Derivation of the Sample Mean

Let x(s) be any integrable function of s and p(s) be the probability density of s. The mean or expected value of x is given by

$$\langle x \rangle = \int_{-\infty}^{\infty} x(s)p(s)ds \tag{B.1}$$

The variance of x is given by

$$\begin{aligned}
\sigma^2(x) &= \int_{-\infty}^{\infty} (x(s)-\langle x \rangle)^2 p(s)ds \\
&= \int_{-\infty}^{\infty} x^2(s)p(s)ds \; -2\langle x \rangle \int_{-\infty}^{\infty} x(s)p(s)ds + \langle x \rangle^2 \int_{-\infty}^{\infty} p(s)ds \\
&= \langle x^2 \rangle - 2\langle x \rangle^2 + \langle x \rangle^2 \\
&= \langle x^2 \rangle - \langle x \rangle^2
\end{aligned} \tag{B.2}$$

When N values of the random variable s, $s_1, s_2, \ldots\ldots, s_N$, are choosen according to the probability density p(s) the resulting values of $x(s_i)$ are given by $x(s_1), x(s_2), \ldots\ldots, x(s_N)$. The sample mean is given by

$$\bar{x} = \frac{\sum_{i=1}^{N} x(s_i)}{N} \tag{B.3}$$

The expected value of the sample mean is given by

$$\langle \bar{x} \rangle = \left\langle \frac{\sum_{i=1}^{N} x(s_i)}{N} \right\rangle$$

$$= \frac{1}{N} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \ldots\ldots \int_{-\infty}^{\infty} \left[\sum_{i=1}^{N} x(s_i) \right] p(s_1) \ldots p(s_N) ds_1 \ldots ds_N \tag{B.4}$$

where each s_i has been treated as an independent variable.[43] Using

$$\int_{-\infty}^{\infty} p(s_i) ds_i = 1 \tag{B.5}$$

Equation B.4 reduces to

$$\langle \bar{x} \rangle = \frac{1}{N} \int_{-\infty}^{\infty} \sum_{i=1}^{N} x(s_i) p(s_i) ds_i$$

$$= \frac{1}{N} \sum_{i=1}^{N} \int_{-\infty}^{\infty} x(s_i) p(s_i) ds_i$$

$$\langle \bar{x} \rangle = \frac{1}{N} \sum_{i=1}^{N} \langle x(s_i) \rangle = \langle x \rangle \tag{B.6}$$

Using Equation B.2, the variance of the sample mean about $\langle x \rangle$ is given by

$$\sigma^2(\bar{x}) = \langle \bar{x}^2 \rangle - \langle x \rangle^2 \tag{B.7}$$

where the first term is given by

$$\langle \bar{x}^2 \rangle = \frac{1}{N^2} \left\langle \left[\sum_{i=1}^{N} x(s_i) \right]^2 \right\rangle$$

$$= \frac{1}{N^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left[\sum_{i=1}^{N} x(s_i) \right]^2 p(s_1) \ldots p(s_N) ds_1 \ldots ds_N$$

$$= \frac{1}{N^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left[\sum_{i=1}^{N} x^2(s_i) + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} x(s_i) x(s_j) \right] p(s_1) \ldots p(s_N) ds_1 \ldots ds_N \tag{B.8}$$

Using Equation B.5, Equation B.8 reduces to

$$\langle \bar{x}^2 \rangle = \frac{1}{N^2}\left[\int_{-\infty}^{\infty} \sum_{i=1}^{N} x^2(s_i)p(s_i)ds_i + \int_{-\infty}^{\infty} \sum_{i=1}^{N} x(s_i)p(s_i)ds_i \int_{-\infty}^{\infty} \sum_{\substack{j=1 \\ j\neq i}}^{N} x(s_j)p(s_j)ds_j \right]$$

$$= \frac{1}{N^2}\left[\sum_{i=1}^{N} \langle x^2(s_i)\rangle + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \langle x(s_i)\rangle^2 \right]$$

$$= \frac{\langle x^2 \rangle}{N} + \frac{(N-1)\langle x \rangle^2}{N} \qquad \text{(B.9)}$$

Substituting Equation B.9 into Equation B.7 results in

$$\sigma^2(\bar{x}) = \frac{\langle x^2 \rangle}{N} + \frac{(N-1)}{N}\langle x \rangle^2 - \langle x \rangle^2 = \frac{\langle x^2 \rangle - \langle x \rangle^2}{N} = \frac{\sigma^2(x)}{N} \qquad \text{(B.10)}$$

which is the desired result.

B.2) The Central Limit Theorem

The central limit theorem[17] states:

> If x is any random variable which has a mean and a variance, then for N sufficiently large, $\bar{x}$, the mean of a random sample of size N, has approximately a normal distribution.

A normal distribution is given by

$$f(s) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{1}{2}\left[(s-\mu)/\sigma \right]^2 \right\} \qquad (B.11)$$

where σ is the is the standard deviation and μ is the mean of the distribution. For Monte Carlo calculations

$$\sigma = \sigma(\bar{x})$$

$$\mu = \langle x \rangle$$

$$s = \bar{x}$$

The probability that $\bar{x}$ falls within the interval

$$\langle x \rangle + \alpha\sigma(\bar{x}) < \bar{x} < \langle x \rangle + \beta\sigma(\bar{x})$$

where $\alpha<\beta$ is given by

$$\frac{1}{(x)\sqrt{2\pi}} \int_{\langle x \rangle + \alpha\sigma(\bar{x})}^{\langle x \rangle + \beta\sigma(\bar{x})} \exp\left\{ -\frac{1}{2}\left[(\bar{x} - \langle x \rangle)/\sigma(\bar{x}) \right]^2 \right\} d\bar{x} \qquad (B.12)$$

Substituting

$$t = \frac{\bar{x} - \langle x \rangle}{\sigma(\bar{x})} \tag{B.13}$$

and

$$dt = \frac{d\bar{x}}{\sigma(\bar{x})} \tag{B.14}$$

into Equation B.12 gives

$$\text{prob}\left[\alpha\sigma(\bar{x}) < \bar{x} - \langle x \rangle < \beta\sigma(\bar{x})\right] = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\beta} \exp(-t^2/2)\, dt \tag{B.15}$$

which is the desired result.

Appendix C: FORTRAN Coding for the One Dimensional Homogeneous Slab Monte Carlo Problem

The following computer listing has had all input - output and other extraneous coding removed. The significant FORTRAN variables are defined below.

N = number of source particles

X = distance particle has travelled since leaving the source

WTN = weight of particle

L = thickness of slab

SIGT = total macroscopic cross section, Σ_t

FRAC = non-absorption probability, $(\Sigma_t-\Sigma_a)/\Sigma_t$

RAT = estimate of the fraction of particles that leave the source and cross the surface at X=L, (the Monte Carlo Tally)

RE = relative error of the estimate due to Monte Carlo statistics (at the 1σ level)

NMAX = maximum number of particles to be started

```
C       SOURCE
 40     CONTINUE
        R=RANF(0)
        N=N+1
        WTN=1.0
        X=0.0

C       TRACK LENGTH
 50     CONTINUE
        R=RANF(0)
        D=ALOG(1./R)/SIGT
        DMAX=(L-X)
        IF (D.GT.DMAX) GO TO 60
        X=X+D

C******REDUCE WEIGHT BY ABSORPTION PROBABILITY*****
        WTN=FRAC*WTN
           GO TO 50

C       TALLY
 60     CONTINUE
        ANS=ANS+WTN
        ANS2=ANS2+WTN**2


        RAT=ANS/N
        X2BAR=ANS2/N

C       ERROR CALCULATIONS
        SIG2=(X2BAR-RAT**2)/N
        IF (SIG2.EQ.-0.0) SIG2=0.0
        RE=SQRT(SIG2)/RAT
        IF(N.EQ.NMAX) GO TO 181
           GO TO 40
 181    CONTINUE
```

Appendix D: FORTRAN Coding for the Fractional Correction Rule Deterministic Classifier

The following computer listing has had all input - output and other extraneous coding removed. The term "TEACHER" in the comment statements refers to the classification that a track appears to belong to after observing the importances created by the last source particle. The term "STUDENT" refers to the classification given by the current discriminant function. The Monte Carlo parameters are the same as those described in Appendix C. Significant pattern recognition parameters are given below.

INC = the number of tracks created by the last source particle (also used to number the prototypes)

WTREM(I) = the weight of the particle after the I'th collision (the source is the first collision)

NXREM(I) = the distance from the source to the I'th collision (source is the first collision). These values are used as the prototypes for the pattern classifier.

G = the discriminant function, g(Y)

WT(I) = the I'th component of the weight vector, $W=w_i$

SEP2 = upper limit of the importance buffer zone used to separate classes, I_2

SEP1 = lower limit of the importance buffer zone used to separate classes, I_1

SUBTOT(I) = number of prototypes belonging to the I'th class (as seen by the "TEACHER")

LAMDA = learning parameter, λ

WTN = weight of the particle when it is either lost or tallied

SCORE = Importance of the prototype, I_i

```
C       SOURCE
 40     CONTINUE
        R=RANF(0)
        N=N+1
        WTN=1.0
        X=0.0
        INC=1
        WTREM(INC)=WTN
        XREM(INC)=X

C       TRACK LENGTH
 50     CONTINUE
        R=RANF(0)
        D=ALOG(1./R)/SIGT
        DMAX=(L-X)
        IF (D.GT.DMAX) GO TO 60
        X=X+D

C*****REDUCE WEIGHT BY ABSORPTION PROBABILITY*****
        WTN=FRAC*WTN
        INC=INC+1
        XREM(INC)=X
        WTREM(INC)=WTN
           GO TO 50

C       TALLY
 60     CONTINUE
        ANS=ANS+WTN
        ANS2=ANS2+WTN**2

C*************** ADJUST WEIGHTS - PATTERN CLASSIFIER ***************
C*                                                                  *
C*                                                                  *
        DO 141 I=2,INC

C       CALCULATE VALUE OF PRESENT DISCRIMINANT FUNCTION
        G=WT(I)*XREM(I)+WT(2)

C       SEE WHICH CLASS THE TEACHER SAYS PARTICLE IS IN
        SCORE=WTN/WTREM(I)
        IF(SCORE.GT.SEP2) 100,90
 90     IF(SCORE.LT.SEP1) 110,140

C       TEACHER SAYS CLASS 2
 100    CONTINUE
        SUBTOT(2)=SUBTOT(2)-1.
        RATOT=SUBTOT(1)/SUBTOT(2)
           IF (G) 130,130,140
```

```
C     TEACHER SAYS CLASS 1
 110  CONTINUE
      SUBTOT(1)=SUBTOT(1)+1.
      IF(SUBTOT(2).EQ.0.0) GO TO 111
      RATOT=SUBTOT(1)/SUBTOT(2)
      GO TO 113
 111  RATOT=1.0
 113  IF(G) 140,120,120

C     TEACHER SAYS STUDENT HAS
C     MISCLASSIFIED CLASS 1 INTO CLASS 2
 120  CONTINUE
      C=LAMDA*ABS(G)/(NNREM(I)**2+1.)
      WT(1)=WT(1)-C*NNREM(I)
      WT(2)=WT(2)-C
      GO TO 131

C     TEACHER SAYS STUDENT HAS
C     MISCLASSIFIED CLASS 2 INTO CLASS 1
 130  CONTINUE
      C=LAMDA*RATOT*ABS(G)/(NNREM(I)**2+1.)
      WT(2)=WT(2)-C
      WT(1)=WT(1)-C*NNREM(I)

 131  CONTINUE
 141  CONTINUE

C*                                                                    *
C*                                                                    *
C*********************** END OF PATTERN CLASSIFIER ********************

      RAT=ANS/N
      M2BAR=ANS2/N

C     ERROR CALCULATIONS
      SIG2=(M2BAR-RAT**2)/N
      IF (SIG2.EQ.-0.0) SIG2=0.0
      RE=SQRT(SIG2)/RAT
      IF(N.EQ.NMAX) GO TO 181
         GO TO 40
 181  CONTINUE
```

Appendix E : Derivation of $\nabla\tilde{R}$ for Various Loss Functions

An approximation to the risk was given by Equation 3.15 and is repeated here as

$$\tilde{R}(W) = \sum_{k=1}^{2} \frac{1}{M_k} \sum_{n=1}^{N_k} S(W, Y_n^k, C_i | C_k) \qquad i=1,2 \quad i \neq k \qquad (E.1)$$

where:

M_k = number of prototypes in class C_k

Y_n^k = n'th misclassified prototype of class C_k

$S(W, Y_n^k, C_i | C_k)$ = loss incurred when prototype Y_n^k actually belonging to class C_k is placed in class C_i

The gradient of R(W) for one dimensional feature space is given by

$$\nabla\tilde{R}(W) = \begin{bmatrix} \frac{\partial R}{\partial w_1} \\ \frac{\partial R}{\partial w_2} \end{bmatrix} = \sum_{k=1}^{2} \frac{1}{M_k} \sum_{n=1}^{N_k} \nabla S(W, Y_n^k, C_i | C_k) \qquad i=1,2 \quad i \neq k$$

$$= \frac{1}{M_1} \sum_{n=1}^{N_1} \nabla S(W, Y_n^1, C_2 | C_1) + \frac{1}{M_2} \sum_{n=2}^{N_2} \nabla S(W, Y_n^2, C_1 | C_2)$$

$$\nabla\tilde{R}(W) = \frac{1}{M_1}\sum_{n=1}^{N_1}\begin{bmatrix}\frac{\partial S(W,Y_n^1,C_2|C_1)}{\partial w_1}\\ \\ \frac{\partial S(W,Y_n^1,C_2|C_1)}{\partial w_2}\end{bmatrix} + \frac{1}{M_2}\sum_{n=1}^{N_2}\begin{bmatrix}\frac{\partial S(W,Y_n^2,C_1|C_2)}{\partial w_1}\\ \\ \frac{\partial S(W,Y_n^1,C_1|C_2)}{\partial w_2}\end{bmatrix} \quad \text{(E.2)}$$

Equation E.2 will now be evaluated for several different loss functions.

Loss Function S = d

The distance d from a prototype , Y, to the decision surface g(Y) = 0 in feature space is given by

$$d = \frac{|W \cdot Y^*|}{|W'|} \quad \text{(E.3)}$$

where W' is the weight vector W with $w_{N+1} = 0$ and Y* is the augumented feature vector as given by Equation 3.5. For S = d, the gradient of S for prototypes belonging to class C_1 but misclassified into C_2 is given by

$$\nabla S(W,Y_n^1,C_2|C_1) = \nabla S(d,C_2|C_1) = \frac{\partial S}{\partial d}\begin{bmatrix}\frac{\partial d}{\partial w_1}\\ \\ \frac{\partial d}{\partial w_2}\end{bmatrix} \quad \text{(E.4)}$$

Substituting Equation E.3 into Equation E.4 and using the relation

$$\frac{\partial S}{\partial d} = 1 \tag{E.5}$$

results in

$$\nabla S(W, Y_n^1, C_2 | C_1) = \begin{bmatrix} \frac{\partial d}{\partial w_1} \\ \frac{\partial d}{\partial w_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial \left(\frac{w_1 y + w_2}{w_1} \right)}{\partial w_1} \\ \frac{\partial \left(\frac{w_1 y + w_2}{w_1} \right)}{\partial w_2} \end{bmatrix} = \begin{bmatrix} -\frac{w_2}{w_1^2} \\ \frac{1}{w_1} \end{bmatrix} \tag{E.6}$$

where y is the single component of the feature vector Y_n^1 (Equation E.1) and w_1 and w_2 are the components of the weight vector W which exist at the time prototype Y_n^1 is misclassified. Similarly for prototypes belonging to class C_2 but misclassified into class C_1, the gradient of S is given by

$$\nabla S(W, Y_n^2, C_1 | C_2) = \begin{bmatrix} \frac{w_2}{w_1^2} \\ -\frac{1}{w_1} \end{bmatrix} \tag{E.7}$$

Loss Function $S = \sqrt{d}$

For this loss function

$$\frac{\partial S}{\partial d} = \frac{\partial(\sqrt{d})}{\partial d} = \frac{1}{2\sqrt{d}} \tag{E.8}$$

For Equation E.3 and $g = W \cdot Y^*$, Equation E.8 reduces to

$$\frac{\partial S}{\partial d} = \frac{1}{2}\left(\left|\frac{w_1}{g}\right|\right)^{1/2} \tag{E.9}$$

For Equations E.9 and E.6, Equation E.4 reduces to

$$\nabla S(W, Y_n^1, C_2 | C_1) = \frac{1}{2}\left(\left|\frac{w_1}{g}\right|\right)^{1/2} \begin{bmatrix} -\frac{w_2}{w_1^2} \\ \frac{1}{w_1} \end{bmatrix} \tag{E.10}$$

Simlarly, for prototypes belonging to class C_2 but misclassified into class C_1 , the gradient of S is given by

$$\nabla S(W, Y_n^2, C_1 | C_2) = \frac{1}{2}\left(\left|\frac{w_1}{g}\right|\right)^{1/2}\begin{bmatrix} \frac{w_2}{w_1^2} \\ -\frac{1}{w_1} \end{bmatrix} \qquad \text{(E.11)}$$

Loss Function $S = d^2$

This loss function results in

$$\frac{\partial S}{\partial d} = \frac{\partial (d^2)}{\partial d} = 2d = 2\left|\frac{g}{w_1}\right| \qquad \text{(E.12)}$$

which when substituted into Equation E.4 along with Equation E.6 results in

$$\nabla S(W, Y_n^1, C_2 | C_1) = 2\left|\frac{g}{w_1}\right|\begin{bmatrix} -\frac{w_2}{w_1^2} \\ \frac{1}{w_1} \end{bmatrix} \qquad \text{(E.13)}$$

and similarly for class C_2 prototypes

$$\nabla S(W, Y_n^2 C_1 | C_2) = 2\left|\frac{g}{w_1}\right| \begin{bmatrix} \frac{w_2}{w_1^2} \\ -\frac{1}{w_1} \end{bmatrix} \qquad \text{(E.14)}$$

Loss Function S = D

The distance in weight space from the weight vector W to the pattern hyperplane defined by Y is given by

$$D = \frac{|W \cdot Y^*|}{|Y^* \cdot Y^*|} \qquad \text{(E.15)}$$

This is the same distance that is used by the deterministic fractional correction rule as described in Section 3.2.1 and implemented in Section 4.2.1. Using Equations E.3 and E.15 the relationship between D and d is given by

$$D = d\frac{|W' \cdot W'|}{|Y^* \cdot Y^*|} \qquad \text{(E.16)}$$

which for one dimensional feature space reduces to

$$D = d \frac{|w_1|}{\sqrt{y_1^2 + 1}} \tag{E.17}$$

For $y_1 >> 1$ (which is usually the case for the unnormalized runs made in this research)

$$D \simeq d \left|\frac{w_1}{y_1}\right| \tag{E.18}$$

Equation E.18 indicates that for $y_1 >> 1$, when using loss=D, prototypes with small y_1 will have more effect on the classifier than those with large y_1.

For one dimensional feature space and S = D

$$\begin{bmatrix} \frac{\partial D}{\partial w_1} \\ \frac{\partial D}{\partial w_2} \end{bmatrix} = \frac{1}{\sqrt{y_1^2 + 1}} \begin{bmatrix} \frac{\partial (w_1 y_1 + w_2)}{\partial w_1} \\ \frac{\partial (w_1 y_1 + w_2)}{\partial w_2} \end{bmatrix} = \frac{1}{\sqrt{y_1^2 + 1}} \begin{bmatrix} y_1 \\ 1 \end{bmatrix} \tag{E.19}$$

and since

$$\frac{\partial S}{\partial D} = 1 \tag{E.20}$$

the resulting expression for the gradient of S is given by

$$\nabla S(W, Y_n^1, C_2 | C_1) = \frac{1}{\sqrt{y_1^2 + 1}} \begin{bmatrix} y_1 \\ \\ 1 \end{bmatrix} \tag{E.21}$$

and similarly

$$\nabla S(W, Y_n^2, C_1 | C_2) = \frac{1}{\sqrt{y_1^2 + 1}} \begin{bmatrix} -y_1 \\ \\ -1 \end{bmatrix} \tag{E.22}$$

Appendix F: FORTRAN Coding for the Statistical Pattern Classifier using Loss = d

The following computer listing has had all input - output and other extraneous coding removed. The parameters defined in Appendices C and D are the same here with the below additional parameter.

$$GTRISK(I) = \frac{\partial R}{\partial w_I} \qquad \text{(see Equation 4.6)}$$

$$I=1,2$$

```
C************** ADJUST WEIGHTS - PATTERN CLASSIFIER ***************
C*                                                                 *
C*                                                                 *
 80    CONTINUE
       DO 141 I=2,INC

C      CALCULATE VALUE OF PRESENT DISCRIMINANT FUNCTION
       G=WT(1)*NXREM(I)+WT(2)

C      SEE WHICH CLASS THE TEACHER SAYS PARTICLE IS IN
       SCORE=WTN/WTREM(I)
       IF(SCORE.GT.SEP2) 100,90
 90    IF(SCORE.LT.SEP1) 110,140

C      TEACHER SAYS CLASS 2
 100   CONTINUE
       SUBTOT(2)=SUBTOT(2)+1.
          IF (G) 130,130,140

C      TEACHER SAYS CLASS 1
 110   CONTINUE
       SUBTOT(1)=SUBTOT(1)+1.
 113   IF(G) 140,120,120

C      TEACHER SAYS STUDENT HAS
C      MISCLASSIFIED CLASS 1 INTO CLASS 2
 120   CONTINUE
       Q(1,2)=Q(1,2)+1./WT(1)
       Q(1,1)=Q(1,1)-WT(2)/WT(1)**2
       GO TO 131

C      TEACHER SAYS STUDENT HAS
C      MISCLASSIFIED CLASS 2 INTO CLASS 1
 130   CONTINUE
       Q(2,1)=Q(2,1)+WT(2)/WT(1)**2
       Q(2,2)=Q(2,2)-1./WT(1)

 131   CONTINUE
```

```
C       STATISTICAL CORRECTION       METHOD 1
        IF(SUBTOT(1).LE.0.0) GO TO 133
        CRISK(1,2)=Q(1,2)/SUBTOT(1)
        CRISK(1,1)=Q(1,1)/SUBTOT(1)
  133   IF(SUBTOT(2).LE.0.0) GO TO 134
        CRISK(2,2)=Q(2,2)/SUBTOT(2)
        CRISK(2,1)=Q(2,1)/SUBTOT(2)
  134   CONTINUE
        CTRISK(1)=CRISK(1,1)+CRISK(2,1)
        CTRISK(2)=CRISK(1,2)+CRISK(2,2)
        WT(1)=WT(1)-LAMDA*CTRISK(1)
        WT(2)=WT(2)-LAMDA*CTRISK(2)
  132   CONTINUE

  141   CONTINUE
C*                                                                              *
C*                                                                              *
C********************* END OF PATTERN CLASSIFIER **************************
```

Appendix G: FORTRAN Coding for the One Dimensional Multi-Region Slab

The following computer listing has had all input - output and other extraneous coding removed. The parameter IA denotes the region that the particle is in. The parameters FRAC(I) and SIGT(I) have the same meaning as in Appendix C only with respect to the I'th region. Additional parameters are listed below.

NIA = number of regions, also the number of the region containing the tallying surface

TL(I) = the boundary of the I'th region TL(I)>TL(I-1)

```
C       SOURCE
 40     CONTINUE
        R=RANF(0)
        N=R+1
        WTH=1.0
        X=0.0
        IA=1

C       TRACK LENGTH
 50     CONTINUE
        R=RANF(0)
        D=ALOG(1./R)/SIGT(IA)
        DMAX=TL(IA)-X
        IF (D.GT.DMAX) GO TO 51
        X=X+D

C*****REDUCE WEIGHT BY ABSORPTION PROBABILITY*****
        WTH=FRAC(IA)*WTH
        XREM(INC)=X
           GO TO 50

C       ENTER A DIFFERENT REGION
 51     IF(IA.EQ.NIA) GO TO 60
        IA=IA+1
        X=X+DMAX
        GO TO 50

C       TALLY
 60     CONTINUE
        ANS=ANS+WTH
        ANS2=ANS2+WTH**2
        RAT=ANS/N
        X2BAR=ANS2/N

C       ERROR CALCULATIONS
        SIG2=(X2BAR-RAT**2)/N
        IF (SIG2.EQ.-0.0) SIG2=0.0
        RE=SQRT(SIG2)/RAT
        IF(N.EQ.NMAX) GO TO 181
           GO TO 40
 181    CONTINUE
```

Appendix H: FORTRAN Coding for the Two Dimensional Multi-Region Slab with a Deterministic Pattern Classifier

The following computer listing has had all input - output and other extraneous coding removed. The parameters required in addition to those described in Appendices G, D, and C are listed below.

PHI = angle of track with the x-axis, ϕ (see Figure 4.30)

U = cos ϕ

NPREM(I) = the value of u after the I'th collision of the last source particle (source is the first collision)

```
C       SOURCE
 40     CONTINUE
        PHI=0.0
        U=COS(PHI)
        N=N+1
        WTN=1.0
        X=0.0
        INC=1
        WTREM(INC)=WTN
        NSREM(INC)=X
        NPREM(INC)=U
        IA=1

C       TRACK LENGTH
 50     CONTINUE
        R=RANF(0)
        D=ALOG(1./R)/SIGT(IA)
        IF(U.LT.0.0) 52,53

C       BACKWARD DIRECTION
 52     IF(IA.EQ.1) 54,55
 54     DMAX=-X/U
        IF(D.GT.DMAX) 57,56

C       NO TALLY
 57     WTN=0.0
        GO TO 60

 55     DMAX= -(X-TL(IA-1))/U
        IF(D.GT.DMAX) 58,56
C       ENTER A DIFFERENT REGION
 58     IA=IA-1
        X=TL(IA)
        GO TO 50
```

```
C       FOWARD DIRECTION
 53     DMAX=(TL(IA)-X)/U
        IF(D.GT.DMAX) 51,56

 56     X=X+D*U

C*****REDUCE WEIGHT BY ABSORPTION PROBABILITY*****
        WTN=FRAC(IA)*WTN
        INC=INC+1
C       SCATTERING ANGLE
        R=RANF(G)
        U=2.*R-1.0
        PHI=ACOS(U)
        NPREM(INC)=U
        NXREM(INC)=X
        WTREM(INC)=WTN
           GO TO 50

C       ENTER A DIFFERENT REGION
 51     IF(IA.EQ.NIA) GO TO 60
        X=TL(IA)
        IA=IA+1
        GO TO 50

C       TALLY
 60     CONTINUE
        ANS=ANS+WTN
        ANS2=ANS2+WTN**2

C*************** ADJUST WEIGHTS - PATTERN CLASSIFIER ***************
C*                                                                  *
C*                                                                  *
 80     CONTINUE
        DO 141 I=2,INC

C       CALCULATE VALUE OF PRESENT DISCRIMINANT FUNCTION
        NPREM(I)=(NPREM(I)+1.0)/2.
        NXREM(I)=NXREM(I)/L
        G=WT(1)*NXREM(I)+NPREM(I)*WT(2)+WT(3)
C       SEE WHICH CLASS THE TEACHER SAYS PARTICLE IS IN
        SCORE=0.0
        IF(WTN.NE.0.0) SCORE=WTN/WTREM(I)
        IC=3
        IF(SCORE.LE.SEP1) IC=1
        IF(SCORE.GT.SEP2) IC=2
        IF(IC-2) 110,100,140

C       TEACHER SAYS CLASS 2
 100    CONTINUE
        SUBTOT(2)=SUBTOT(2)+1.
        RATOT=SUBTOT(1)/SUBTOT(2)
           IF (G) 130,130,140

C       TEACHER SAYS CLASS 1
 110    CONTINUE
        SUBTOT(1)=SUBTOT(1)+1.
        IF(SUBTOT(2).EQ.0.0) GO TO 111
        RATOT=SUBTOT(1)/SUBTOT(2)
        GO TO 113
 111    RATOT=1.0
 113    IF(G) 140,120,120
```

```
C       TEACHER SAYS STUDENT HAS
C       MISCLASSIFIED CLASS 1 INTO CLASS 2
  120   CONTINUE
        C=LAMDA*ABS(G)/(NXREM(I)**2+NPREM(I)**2+1.)
        WT(1)=WT(1)-C*NXREM(I)
        WT(2)=WT(2)-C*NPREM(I)
        WT(3)=WT(3)-C
        GO TO 131

C       TEACHER SAYS STUDENT HAS
C       MISCLASSIFIED CLASS 2 INTO CLASS 1
  130   CONTINUE
        C=LAMDA*ABS(G)*QUATOT/(NXREM(I)**2+NPREM(I)**2+1.)
        WT(1)=WT(1)+C*NXREM(I)
        WT(2)=WT(2)+C*NPREM(I)
        WT(3)=WT(3)+C

  131   CONTINUE

  141   CONTINUE

C*                                                                    *
C*                                                                    *
C********************* END OF PATTERN CLASSIFIER **********************

        RAT=ANS/N
        X2BAR=ANS2/N

C       ERROR CALCULATIONS
        SIG2=(X2BAR-RAT**2)/N
        IF (SIG2.EQ.-0.0) SIG2=0.0
        IF(RAT.EQ.0.0) GO TO 142
        RE=SQRT(SIG2)/RAT
  142   IF(N.EQ.NMAX) GO TO 181
           GO TO 40
  181   CONTINUE
```

Appendix I: FORTRAN Coding for the Statistical Classifier with Two-Dimensional Pattern Space

The following computer listing has had all input - output and other extraneous coding removed. Parameters are defined as given in Appendices H, F, D, and C.

```
C************** ADJUST WEIGHTS - PATTERN CLASSIFIER **************
C*                                                               *
C*                                                               *
 80    CONTINUE
       DO 141 I=2,IEC

C      CALCULATE VALUE OF PRESENT DISCRIMINANT FUNCTION
       HPREM(I)=(HPREM(I)+1.0)/2.
       HNREM(I)=HNREM(I)/L
       G=WT(1)*HNREM(I)+HPREM(I)*WT(2)+WT(3)
C      SEE WHICH CLASS THE TEACHER SAYS PARTICLE IS IN
       SCORE=0.0
       IF(WTN.NE.0.0) SCORE=WTN/WTREM(I)
       IC=0
       IF(SCORE.LE.SEP1) IC=1
       IF(SCORE.GT.SEP2) IC=2
       IF(IC-2) 110,100,140

C      TEACHER SAYS CLASS 2
 100   CONTINUE
       SUBTOT(2)=SUBTOT(2)+1.
          IF (G) 130,130,140

C      TEACHER SAYS CLASS 1
 110   CONTINUE
       SUBTOT(1)=SUBTOT(1)+1.
       IF(G) 140,120,120

C      TEACHER SAYS STUDENT HAS
C      MISCLASSIFIED CLASS 1 INTO CLASS 2
 120   CONTINUE
       AW=WT(1)**2+WT(2)**2
       SRI=1./SQRT(AW)
       AWG=G/AW
       Q(1,1)=Q(1,1)+SRI*(HNREM(I)-AWG*WT(1))
       Q(1,2)=Q(1,2)+SRI*(HPREM(I)-AWG*WT(2))
       Q(1,3)=Q(1,3)+SRI
       GO TO 141

C      TEACHER SAYS STUDENT HAS
C      MISCLASSIFIED CLASS 2 INTO CLASS 1
 130   CONTINUE
       AW=WT(1)**2+WT(2)**2
       SRI=1./SQRT(AW)
       AWG=G/AW
       Q(2,1)=Q(2,1)-SRI*(HNREM(I)-AWG*WT(1))
       Q(2,2)=Q(2,2)-SRI*(HPREM(I)-AWG*WT(2))
       Q(2,3)=Q(2,3)-SRI
```

```
  131  CONTINUE

C      STATISTICAL CORRECTION      METHOD 1
       IF(SUBTOT(1).LE.0.0) GO TO 133
       GRISK(1,2)=Q(1,2)/SUBTOT(1)
       GRISK(1,1)=Q(1,1)/SUBTOT(1)
       GRISK(1,3)=Q(1,3)/SUBTOT(1)
  133  IF(SUBTOT(2).LE.0.0) GO TO 134
       GRISK(2,1)=Q(2,1)/SUBTOT(2)
       GRISK(2,2)=Q(2,2)/SUBTOT(2)
       GRISK(2,3)=Q(2,3)/SUBTOT(2)
  134  CONTINUE
       GTRISK(1)=GRISK(1,1)+GRISK(2,1)
       GTRISK(2)=GRISK(1,2)+GRISK(2,2)
       GTRISK(3)=GRISK(1,3)+GRISK(2,3)
       WT(1)=WT(1)-LAMDA*GTRISK(1)
       WT(2)=WT(2)-LAMDA*GTRISK(2)
       WT(3)=WT(3)-LAMDA*GTRISK(3)

  140  CONTINUE

  141  CONTINUE

C*                                                                              *
C*                                                                              *
C************************ END OF PATTERN CLASSIFIER ***************************
```

Appendix J: Geometry Description Used in the Neutron Monte Carlo Code MCN

This appendix contains a description of the geometry used in the MCN code at Los Alamos Scientific Laboratory. The code is designed to handle any number of first, second, and some fourth degree surfaces (limited only by computer storage) which divide the geometry of the problem into geometric cells (a cell is defined below).

If $f(x,y,z)=0$ is the equation of any surface in the problem, then for any arbitrary point in space (x_0,y_0,z_0), the sign of the quantity $f(x_0,y_0,z_0)$ is defined as the sense of the point (x_0,y_0,z_0) with respect to the surface $f(x,y,z)=0$.

A geometric cell is defined such that:

(1) all points within a cell must have the same sense with respect to the bounding surfaces of that cell

(2) the senses of points within a cell must uniquely determine that cell from all other cells.

The cell corresponding to any point in space can be determined by comparing the senses of the point (with respect to the surfaces bounding each cell) to the senses that define each cell. The point belongs to the cell for which the senses agree. Cells do not necessarily correspond to material regions and are frequently specified (along with additional surfaces) for the purpose of splitting and Russian roulette. Each cell is assigned an importance which is used for splitting as described in Section 5.3.1.

APPENDIX K. DERIVATION OF S_{j+1} FOR THE STATISTICAL CLASSIFIER AND LOSS = d

Let S_j be the decision surface location as determined by

$$S_j = \frac{-w_{2,j}}{w_{1,j}} \tag{K.1}$$

where $w_{i,j}$ $(i = 1,2)$ is the i'th component of the j'th weight vector W_j. The first weight vector, W_1, is provided as an initial guess. Thereafter each W_{j+1} is determined by (see Equation 3.16)

$$W_{j+1} = W_j - \lambda \nabla \tilde{R}_{j+1}(W) \tag{K.2}$$

where $\tilde{R}_j(W)$ is the j'th approximation to the risk (see Equation 3.19). The weight adjustment as given by Equation K.2 is performed only after a prototype has been misclassified. Therefore

j = (total number of misclassified prototypes) - 1

Using Equation 3.17 it follows that

$$\nabla \tilde{R}_j(W) = \frac{1}{M_{1,j}} \sum_{n=1}^{N_{1,j}} \begin{bmatrix} \dfrac{\partial s(W_n, Y_n^1, C_2/C_1)}{\partial w_1} \\ \dfrac{\partial s(W_n, Y_n^1, C_2/C_1)}{\partial w_2} \end{bmatrix} + \frac{1}{M_{2,j}} \sum_{n=1}^{N_{2,j}} \begin{bmatrix} \dfrac{\partial s(W_n, Y_n^2, C_1/C_2)}{\partial w_1} \\ \dfrac{\partial s(W_n, Y_n^1, C_1/C_2)}{\partial w_2} \end{bmatrix} \tag{K.3}$$

where $M_{i,j}$ and $N_{i,j}$ $(i = 1,2)$ are the number of prototypes and number of misclassified prototypes respectively of class C_i after the (j-1)'th misclassification occurred ($N_{1,j} + N_{2,j} = j - 1$). Substituting the expressions for the partial derivatives of the loss function (see Table 4.4) into Equation K.3 results in

$$\nabla\tilde{R}_j(W) = \begin{bmatrix} \frac{1}{M_{1,j}} \sum_{n=1}^{N_{1,j}} \left(\frac{-w_{2,n}}{w_{1,n}^2} \right) + \frac{1}{M_{2,j}} \sum_{n=1}^{N_{2,j}} \left(\frac{w_{2,n}}{w_{1,n}^2} \right) \\ \frac{1}{M_{1,j}} \sum_{n=1}^{N_{1,j}} \left(\frac{1}{w_{1,n}} \right) + \frac{1}{M_{2,j}} \sum_{n=1}^{N_{2,j}} \left(\frac{-1}{w_{1,n}} \right) \end{bmatrix} \qquad \text{(K.4)}$$

where $w_{i,n}$ $(i = 1,2)$ is the i'th component of the n'th weight vector W_n that existed when the n'th prototype of the corresponding class was misclassified.

Defining the variables

$$A_{1,1,j} = \frac{1}{M_{1,j}} \sum_{n=1}^{N_{1,j}} \left(\frac{-w_{2,n}}{w_{1,n}^2} \right) \qquad \text{(K.5a)}$$

$$A_{2,1,j} = \frac{1}{M_{2,j}} \sum_{n=1}^{N_{2,j}} \left(\frac{w_{2,n}}{w_{1,n}^2} \right) \qquad \text{(K.5b)}$$

$$A_{1,2,j} = \frac{1}{M_{1,j}} \sum_{n=1}^{N_{1,j}} \left(\frac{1}{w_{1,n}} \right) \tag{K.5c}$$

$$A_{2,2,j} = \frac{1}{M_{2,j}} \sum_{n=1}^{N_{2,j}} \left(\frac{-1}{w_{1,n}} \right) \tag{K.5d}$$

Equation K.4 reduces to

$$\nabla \tilde{R}_j(W) = \begin{bmatrix} A_{1,1,j} + A_{2,1,j} \\ \\ A_{1,2,j} + A_{2,2,j} \end{bmatrix} = \begin{bmatrix} T_{1,j} \\ \\ T_{2,j} \end{bmatrix} \tag{K.6}$$

where $T_{1,j} = A_{1,1,j} + A_{2,1,j}$ (K.7a)

$$T_{2,j} = A_{1,2,j} + A_{2,2,j} \tag{K.7b}$$

Consider the case when the j'th misclassification consists of a prototype that has been misclassified from class C_1 into C_2. For this case

$$A_{2,2,j+1} = A_{2,2,j} \tag{K.8a}$$

$$A_{2,1,j+1} = A_{2,1,j} \tag{K.8b}$$

$$A_{1,1,j+1} = \frac{1}{M_{1,j}+1} \sum_{n=1}^{N_{1,j}+1} \left(\frac{-w_{2,n}}{w_{1,n}^2} \right) \tag{K.8c}$$

$$A_{1,2,j+1} = \frac{1}{M_{1,j}+1} \sum_{n=1}^{N_{1,j}+1} \left(\frac{1}{w_{1,n}} \right) \tag{K.8d}$$

Equation K.8c can be expanded resulting in

$$A_{1,1,j+1} = \frac{1}{(M_{1,j}+1)} \left[\sum_{n=1}^{N_{1,j}} \left(\frac{-w_{2,n}}{w_{1,n}^2} \right) + \frac{-w_{2,p}}{w_{1,p}^2} \right]$$

$$= \frac{1}{(M_{1,j}+1)} \left[M_{1,j} A_{1,1,j} + \frac{-w_{2,p}}{w_{1,p}^2} \right] \tag{K.9}$$

where $p = N_{1,j}+1$. Similarly Equation K.8d becomes

$$A_{1,2,j+1} = \frac{1}{(M_{1,j}+1)} \left[M_{1,j} A_{1,2,j} + \frac{1}{w_{1,p}} \right] \tag{K.10}$$

Again assuming that a prototype belonging to C_1 has been misclassified into C_2, Equation K.7a becomes

$$T_{1,j+1} = A_{1,1,j+1} + A_{2,1,j+1} \tag{K.11}$$

Substituting Equations K.9 and K.8b into K.11 results in

$$T_{1,j+1} = \frac{1}{(M_{1,j}+1)} \left[M_{1,j} A_{1,1,j} + \frac{-w_{2,p}}{w_{1,p}^2} \right] + A_{2,1,j} \qquad \text{(K.12)}$$

However since

$$W_p = W_j, \qquad \text{(K.13)}$$

Equation K.12 becomes

$$T_{1,j+1} = \frac{1}{(M_{1,j}+1)} \left[M_{1,j} A_{1,1,j} + \frac{-w_{2,j}}{w_{1,j}^2} \right] + A_{2,1,j} \qquad \text{(K.14)}$$

Similarly $T_{2,j+1}$ is given by

$$T_{2,j+1} = \frac{1}{(M_{1,j}+1)} \left[M_{1,j} A_{1,2,j} + \frac{1}{w_{1,j}} \right] + A_{2,2,j} \qquad \text{(K.15)}$$

Substituting Equation K.15 and K.14 into Equation K.6 results in

$$\overline{VR}_{j+1}(W) = \begin{bmatrix} \frac{1}{(M_{1,j}+1)} \left(M_{1,j} A_{1,1,j} + \frac{-w_{2,j}}{w_{1,j}^2} \right) + A_{2,1,j} \\ \frac{1}{(M_{1,j}+1)} \left(M_{1,j} A_{1,2,j} + \frac{1}{w_{1,j}} \right) + A_{2,2,j} \end{bmatrix} \qquad \text{(K.16)}$$

Using Equation K.16 and Equation K.2 gives

$$w_{1,j+1} = \left\{ w_1 - \lambda \left[\frac{1}{(M_1+1)} \left(M_1 \quad A_{1,1} + \frac{-w_2}{w_1^2} \right) + A_{2,1} \right] \right\}_j \qquad \text{(K.17a)}$$

$$w_{2,j+1} = \left\{ w_2 - \lambda \left[\frac{1}{(M_1+1)} \left(M_1 \quad A_{1,2} + \frac{1}{w_1} \right) + A_{2,2} \right] \right\}_j \qquad \text{(K.17b)}$$

where the j subscripts have been dropped on the right hand side of the equation. Using Equations K.17a and K.17b in Equation K.1 results in

$$S_{j+1} = \frac{-w_{2,j+1}}{w_{1,j+1}} = \frac{-\{w_2(M_1+1) - \lambda[A_{1,2}M_1 + \frac{1}{w_1} + A_{2,2}(M_1+1)]\}_j}{\{w_1(M_1+1) - \lambda[A_{11}M_1 - \frac{w_2}{w_1^2} + A_{2,1}(M_1+1)]\}_j}$$

$$= \frac{-\left\{ w_2 - \lambda \left[T_2 - \frac{A_{1,2}}{(M_1+1)} + \frac{1}{w_1(M_1+1)} \right] \right\}_j}{\left\{ w_1 - \lambda \left[T_1 - \frac{A_{1,1}}{(M_1+1)} - \frac{w_2}{w_1^2(M_1+1)} \right] \right\}_j}$$

$$= \left\{ \frac{S_j w_1 + \lambda \left[T_2 - \frac{A_{1,2}}{(M_1+1)} + \frac{1}{w_1(M_1+1)} \right]}{w_1 - \lambda \left[T_1 - \frac{A_{1,1}}{(M_1+1)} + \frac{S_j}{w_1(M_1+1)} \right]} \right\}_j \qquad \text{(K.18)}$$

Equation K.18 is true only when a prototype belonging to C_1 has been misclassified. If a class C_2 prototype is misclassified, a similar treatment results in

$$S_{j+1} = \left\{ \frac{S_j w_1 + \lambda \left[T_2 - \frac{A_{2,2}}{(M_2+1)} - \frac{1}{w_1 (M_1+1)} \right]}{w_1 - \lambda \left[T_1 - \frac{A_{2,1}}{(M_2+1)} - \frac{S_j}{w_1 (M_2+1)} \right]} \right\}_j \tag{K.19}$$

In summary, if a prototype belonging to C_k is misclassified, the next value of the decision surface is given by

$$S_{j+1} = \left\{ \frac{S_j w_1 + \lambda \left[T_2 - \frac{A_{k,2}}{(M_k+1)} + \frac{(-1)^{k+1}}{w_1 (M_k+1)} \right]}{w_1 - \lambda \left[T_1 - \frac{A_{k,1}}{(M_{k+1})} + \frac{(-1)^{k+1} S_j}{w_1 (M_k+1)} \right]} \right\}_j \tag{K.20}$$

which is the desired result.

References

1. J. M. Hammersley and D. C. Handscomb, Monte Carlo Methods, John Wiley & Sons, Inc. (1964).

2. M. Clark and K. F. Hansen, Numerical Methods of Reactor Analysis, Academic Press (1964).

3. A. S. Householder, Ed., "Monte Carlo Method," Proceedings of a symposium held in June and July 1949, National Bureau of Standards Applied Mathematics Series, Vol. 12, issued June 11, 1951.

4. B. I. Spinrad et al., "An Alignment Chart for Monte Carlo Solution of the Transport Problem," in Ref. 3.

5. G. W. Brown, "History of RAND's Random Digits - Summary," in Ref. 3.

6. E. D. Cashwell, Los Alamos Scientific Laboratory, personel communication, March 1975.

7. E. D. Cashwell and C. J. Everett, A Practical Manual on the Monte Carlo Method for Random Walk Problems, Permagon Press, New York (1959).

8. E. D. Cashwell et al., "Monte Carlo Code Development in Los Alamos," Presentation at the Monte Carlo Conference at Argonne National Laboratory, Argonne, Illinois, July 1-3, 1974. (Also available as LASL report LA-5903-MS).

9. J. Spanier, "A New Multi-Stage Procedure for Systematic Variance Reduction in Monte Carlo," Proceedings of Conference on New Developments in Reactor Mathematics and Applications, CONF-710302, pg. 760-770 (1971).

10. D. B. MacMillan, "Optimization of Importance-Sampling Parameters in Monte Carlo," Nuclear Science and Engineering: 48, 219-231 (1972).

11. J. Spanier and E. M. Gelbard, Monte Carlo Principles and Neutron Transport Problems, Addison-Wesley, Massachusetts (1969).

12. R. Baldini-Celio et al., "A Multistage Self-Improving Monte Carlo Method," Nuclear Instruments and Methods 72 (1969), 317-320.

13. M. H. Weik, Standard Dictionary of Computers and Information Processing, Hayden Book Company (1970).

14. L. L. Carter and E. D. Cashwell, A Review of Particle Transport Simulation with the Monte Carlo Method to appear in the AEC Critical Review Series, USAEC Technical Information Center, Oak Ridge, Tennessee.

15. N. J. Nilsson, Learning Machines, McGraw-Hill, New York (1965).

16. H. C. Andrews, Introduction to Mathematical Techniques in Pattern Recognition, Wiley-Interscience, New York (1972).

17. W. Mendenhall and R. L. Scheaffer, Mathematical Statistics with Applications, Duxbury Press, Massachusetts, 1973.

18. J. R. Lamarsh, Introduction to Nuclear Reactor Theory, Addison-Wesley Pub. Co. (1966).

19. E. D. Cashwell, et al., "MCN: A Neutron Monte Carlo Code," Los Alamos Scientific Laboratory report LA-4751 (1972).

20. L. L. Carter, "MCNA: A Computer Program to Solve the Adjoint Neutron Transport Equation by Coupled Sampling with the Monte Carlo Method," Los Alamos Scientific Laboratory report LA-4488 (1971).

21. C. E. Burgart, "A General Method of Importance Sampling the Angle of Scattering in Monte Carlo Calculations," Nuclear Science and Engineering: 42, 306-323 (1970).

22. K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York (1972).

23. K. S. Fu, Sequential Methods in Pattern Recognition and Machine Learning, Academic Press, New York (1968).

24. C. Chen, Statistical Pattern Recognition, Hayden Book Company, New Jersey.

25. T. Y. Young and T. W. Calvert, Classification, Estimation and Pattern Recognition, American Elsevier Pub. Co., New York (1974).

26. R. L. Kashyap, "Algorithms for Pattern Classification," in Adaptive, Learning and Pattern Recognition Systems, J. M. Mendel and K. S. Fu, Eds., Academic Press, New York, Chapter 3 (1970).

27. Y. C. Ho and A. K. Agrawala, "On Pattern Classification Algorithms - Introduction and Survey," Division of Engineering and Applied Physics Technical Report No. 557, Harvard University, March 1968.

28. C. Blaydon and Y. C. Ho, "On the Abstraction Problem in Pattern Classification," Proc. National Electron. Conf. (1966).

29. Y. A. Shreider, Ed., The Monte Carlo Method, Pernagon Press, New York (1966).

30. F. H. Clark, "The Exponential Transform as an Importance Sampling Device, A Review," Oak Ridge National Laboratory (ORNL-RSIC-14) (1966).

31. S. Amari, "A Theory of Adaptive Pattern Classifiers," IEEE Transactions on Electronic Computers, Vol. EC-16, No. 3, pg. 299 (June 1967).

32. M. H. Kalos, et al., "Automatic Computation of Importance Sampling Functions for Monte Carlo Transport Codes-Phase III," DNA 2890F (1972).

33. E. D. Cashwell, et al., "Monte Carlo Photon Codes: MCG and MCP," Los Alamos Scientific Laboratory report LA-5157-MS (1973).

34. W. L. Thompson, "Gamma-Ray and Electron Transport by Monte Carlo," Doctoral Dissertation, University of Virginia, Charlottesville, Virginia (August 1974).

35. S. M. Selby, Ed., Standard Mathematical Tables, Seventeenth Edition, The Chemical Rubber Co., pg. 547 (1969).

36. A. Solem, "SECOND-Elapsed CP Time for this Job," available from Los Alamos Scientific Laboratory program library, report Q110A (March 1970).

37. B. Swartz, "Least Squares Polynomial Fitting FORTRAN IV Subroutine, with Coefficients of Orthogonal and Legendre Polynomials Optional," available from Los Alamos Scientific Laboratory program library, report E208A (May 1967).

38. J. L. Macdonald, "Heuristic Learning Control for Nuclear Reactors," M.S. Thesis, University of Texas at Austin (August 1972).

39. B. L. Buzbee, "DOTPRO - Inner Product of Two Vectors," available from Los Alamos Scientific Laboratory program library, report F124A (October 1972).

40. B. L. Buzbee, "ADDVEC - Vector Addition," available from Los Alamos Scientific Laboratory program library, report F139A (August 1973).

41. C. J. Everett and E. D. Cashwell, "A Monte Carlo Sampler," Los Alamos Scientific Laboratory report LA-5061-MS (October 1972).

42. C. J. Everett and E. D. Cashwell, "A Second Monte Carlo Sampler," Los Alamos Scientific Laboratory report LA-5723-MS (September 1974).

43. C. J. Everett, Los Alamos Scientific Laboratory, personal communications, March 1975.

☆ U.S. GOVERNMENT PRINTING OFFICE: 1975—677-343/17