June 21, 1967
CGTM No. 16
Dan Ross

BOX SYNTAX -- A 2-DIMENSIONAL METALANGUAGE

INTRODUCTION

The process of translating a source language (L_S) into some other, known, target language (L_T) consists of recognizing valid constructs in L_S, and interpreting the meaning of those constructs in L_T. The description of the class of valid constructs of L_S is called the syntax of L_S. The description of the meaning of those constructs is called the semantics of L_S. The languages in which the syntax and semantics of L_S are stated are called the syntactic and semantic metalanguages.

Box Syntax is a syntactic metalanguage (abbreviated ML_{BS}) which combines some features of other metalanguages with a new 2-dimensional format. Its principal advantage is that the 2-dimensional format is easier to scan than previously existing linear formats. Comprehension of the syntax of a language L_S should be improved by describing it in Box Syntax.

DESCRIPTION OF BOX SYNTAX

(1) Constructs in ML_{BS} are classified as either terminal or nonterminal constructs.

(2) Terminal constructs in ML_{BS} are strings of text in L_S. The strings are underlined to indicate their extent. Terminal constructs are the constants of ML_{BS}. (See Fig. 1-c.)

(3) Nonterminal constructs in ML_{BS} are the variables of ML_{BS}. Each nonterminal construct is denoted by a rectangular box. The box may contain additional information which further characterizes the construct.

(4) There are several ways to indicate the class of values which may be assigned to a given nonterminal construct. One way is to define that construct in terms of other terminal and nonterminal constructs. The defined construct is denoted by a name, consisting of a string of letters and digits and blank spaces, surrounded by the box described in (3) above. (See Fig. 1-a.)

(5) The construct may be defined as the concatenation of several terminal and nonterminal constructs. (See Fig. 1-b.)

(6) A given construct may enclose other constructs within it. The given construct effectively parenthesizes the enclosed constructs. (See Fig. 1-d.)

(7) A given construct may represent an alternative choice among several other constructs. The rectangular box denoting the given construct is segmented vertically (by horizontal lines), and each of the alternative constructs is enclosed in one of the segments. (See Fig. 1-e.)

(8) The syntax of L_S may be such that a construct B in L_S may be repeatedly concatenated with itself. This is denoted in ML_{BS} by an asterisk following a single replication of the construct $\boxed{B}$, which corresponds to B. (See Figs. 1-f and 1-g.) The asterisk in Fig. 1-f indicates one or more replications of $\boxed{B}$. The asterisk surrounded by a zero in Fig. 1-g indicates zero or more replications of $\boxed{B}$.

(9) Box Syntax also allows representation of repeated concatenation of a construct B in L_S, where the replications are separated by some other delimiting construct C. However, the use of box notation requires that the ML_{BS} construct $\boxed{B}$ corresponding to B uses at least one box, on which to attach the broken-line box described in (10) below. Therefore, an additional parenthesizing box, as in (6) above, may be needed in order to draw the ML_{BS} construct properly. (See Figs. 1-h, 1-i, and 1-j.)

(10) The repeated construct of (8) above is denoted by a box drawn with solid lines, to which is adjoined a box drawn with broken lines. The broken-line box contains the ML_{BS} construct corresponding to the delimiter C in L_S. The combination of solid- and broken-line boxes is followed by either * or ⊛ , depending on whether one or more replications, or zero or more replications, are permitted in L_S.

(11) The syntax of L_S is the same regardless of whether the broken-line box of (10) above appears to the left or to the right of the solid-line box. However, the choice of left-to-right association or right-to-left association of the replications can be suggestive of the semantics of L_S. (See Figs. 1-h and 1-i.)

(12) The empty construct in ML_{BS} is denoted as in Fig. 2-a. "Empty" is not the same as the string of text in L_S consisting of a blank space character.

(13) The semantics involved in translating L_S, or in other processing associated with the syntax of L_S, optionally may be included in a box along with the corresponding syntactic construct. If so, the box is segmented vertically by a (horizontal) light solid line. The syntax, as described in (1) through (12) above, appears above the light line; the semantic constructs in an appropriate semantic metalanguage appear below the light line. (See Fig. 2-b.) By convention, the semantic constructs are executed after the corresponding syntactic construct has been recognized; that is, after a string of text in L_S has been parsed by successfully matching it with an ML_{BS} construct.

(14) If L_S is context-dependent, then the choice of syntax used by a syntax-recognizing program may be governed by the execution of a semantic construct. Fig. 2-c shows the ML_{BS} notation for a semantics-driven choice of syntax.

EXAMPLES

(1) The example in Fig. 3 is taken from the ALGOL 60 Report [3]. I have taken some small liberties in rearranging the definitions and un-naming some of the named ALGOL constructs.

The construct <for statement> was transferred from the definition of <unconditional statement> to the definition of <statement> in the revised ALGOL 60 Report [3].

(2) In Ref. [1], a "triple sequence alarm" is used consistently to demonstrate the appearance and capabilities of a collection of well-known metalanguages. Fig. 4 shows the Box Syntax representation of the triple sequence alarm.

CREDIT WHERE CREDIT IS DUE

Many of the ideas incorporated in Box Syntax were taken from already existing metalanguages.

From Backus Normal Form (Ref. [1]): The use of nonterminal constructs, and the general appearance of their definition. (See Figs. 1-a and 1-b.) Bracketing nonterminal constructs to distinguish them from terminal constructs. (See Fig. 1-c.) The use of an empty construct. (See Fig. 2-a.)

From IBM metalinguistic notation (A good example appears in pp. 51 ff. of Ref. [2]): Parenthesization. (See Fig. 1-d.) The vertical arrangement of alternative constructs. (See Fig. 1-e.)

From Regular Expressions (Ref. [1]): An asterisk to indicate arbitrarily many replications. (See Figs. 1-f and 1-g.)

New notational techniques introduced in Box Syntax: The use of box enclosure. Similar but distinct notations for one or more replications and for zero or more replications. (See Figs. 1-f and 1-g.) A special notation for delimiters. (See Figs. 1-h, 1-i, and 1-j.) Semantic description adjacent to syntactic description. (See Fig. 2-b.) Formalized coupling from semantics to syntax, for context-dependent languages. (See Fig. 2-c.)

WHY A NEW METALANGUAGE?

Each metalanguage has its drawbacks. For example, in Backus Normal Form, iteration is treated as a special case of recursion, without a special notation. I think that the frequency with which iteration occurs in mechanical languages makes it important enough to justify a special notation, such as is found in Regular Expressions. The Box Syntax notation

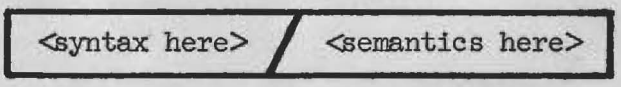
for iteration in Figs. 1-f and 1-g certainly is more apparent than the equivalent BNF notation and, in the case of Fig. 1-g, requires fewer defined constructs. (Note that the BNF in Fig. 1-g could be compressed into a single definition, but the resulting grammar would have an ambiguous parse.)

As another example, the IBM metalinguistic notation does not include the use of named nonterminal constructs. This eliminates the possibility of "subroutines" of nonterminal constructs; consequently this notation cannot be used to describe recursive languages.

In Box Syntax, I have incorporated what I consider to be the best features of several other metalanguages, with some new features. Chief among the new features is the formalism for context-dependent languages, shown in Fig. 2-c. Almost all mechanical languages in current use have some context-dependent properties, yet no formal notation previously proposed for describing these properties has found widespread acceptance. [5], [6], [7]

Box Syntax, too, is not without its drawbacks (literally). Drawing the boxes such that the resulting diagram is clearly understandable and neat in appearance, requires considerable effort. Box Syntax is best suited to a computer-assisted drafting method, with the aid of a CRT display unit.

One modification of Box Syntax, for CRT display, is due to the inability to generate both light and heavy lines. A suitable equivalent to Fig. 2-b might be a diagonal line to separate syntax from semantics:



GENERALITY

In terms of ability to describe context-free linear languages, Backus Normal Form and Box Syntax are equivalent. For each BNF construct shown in Ref. [1], there is an equivalent Box Syntax construct. For each Box Syntax construct shown in Fig. 1, there is an equivalent BNF construct or set of constructs.

Context-dependent linear languages can be described in Box Syntax, but not in Backus Normal Form.

Box Syntax is not useful in describing most 2-dimensional languages, such as flowcharts, because these languages place no significance on physical juxtaposition. The significant property of flowchart syntax is connectivity, which is topological rather than geographic. Box Syntax describes the concatenation, or geographic placement, of terminal constructs.

There is one relevant 2-dimensional language in which juxtaposition of constructs is significant....Box Syntax itself. A logical extension of Box Syntax is capable of describing unextended Box Syntax. The extension requires the ability to concatenate terminal constructs vertically, as well as horizontally. Since the class of terminal constructs includes boxes, a second color is helpful to distinguish between terminal and nonterminal boxes. Underlining the terminal boxes, in a manner analogous to Fig. 1-c, may substitute for a second color, but the resulting diagram is extremely confusing to read.

ACKNOWLEDGEMENTS

I would like to thank Robert T. Braden for his constructive criticism, Myron Ruderman for his help with the English metametalanguage, and Jerome A. Feldman for his constructive sarcasm.

REFERENCES

1. Gorn, Saul. "Specification Languages for Mechanical Languages and Their Processors -- A Baker's Dozen", Comm. ACM, Vol. 4, No. 12, Dec. 1961.

2. Supervisor and Data Management Macro-Instructions, IBM System/360 Operating System File S360-36, Form No. C28-6647-0.

3. Naur, Peter (Ed.). "Report on the Algorithmic Language ALGOL 60", Comm. ACM, Vol. 3, No. 5, May 1960; revised in Comm. ACM, Vol. 6, No. 1, Jan. 1963.

4. Burkhardt, Walter H. "Metalanguage and Syntax Specification", Comm. ACM, Vol. 8, No. 5, May 1965.

5. Leavenworth, B. M. "FORTRAN IV as a Syntax Language", Comm. ACM, Vol. 7, No. 2, Feb. 1964.

6. Caracciolo di Forino, Alfonso. "Some Remarks on the Syntax of Symbolic Programming Languages", Comm. ACM, Vol. 6, No. 8, Aug. 1963.

7. Rabinowitz, Irving N. "Report on the Algorithmic Language FORTRAN II", Comm. ACM, Vol. 5, No. 6, June 1962.

FIG. 1 EQUIVALENT STATEMENTS

	BACKUS NORMAL FORM	BOX SYNTAX	COMMENTS
(a)	<A> ::= <B>	[A] = [B]	METALINGUISTIC CONSTRUCT DEFINITION.
(b)	<A> ::= <B> <C>	[A] = [B] [C]	CONCATENATION.
(c)	<THE WORD BOY> ::= BOY	[THE WORD BOY] = BOY	TEXT OF THE LANGUAGE BEING DESCRIBED.
(d)	<A> ::= <E> <D> <E> ::= <B> <C>	[A] = [[B] [C]] [D]	PARENTHESIZATION.
(e)	<A> ::= <B> | <C> | <D>	[A] = [B] / [C] / [D]	ALTERNATIVES. EXAMPLE: [BINARY DIGIT] = [0] / [1]
(f)	<A> ::= <B> | <B> <A>	[A] = [B]*	ONE OR MORE REPLICATIONS. EXAMPLE: [UNSIGNED INTEGER] = [DIGIT]*
(g)	<A> ::= <C> | <EMPTY> <C> ::= <B> | <B> <C>	[A] = [B]⊛	ZERO OR MORE REPLICATIONS.
(h)	<A> ::= <B> | <B> <C> <A>	[A] = [B] [C]*	ONE OR MORE REPLICATIONS WITH DELIMITER. EXAMPLE: [LIST] = [LIST ELEMENT] [,]*
(i)	<A> ::= <B> | <A> <C> <B>	[A] = [C] [B]*	ONE OR MORE REPLICATIONS WITH DELIMITER. ASSOCIATION OPPOSITE FROM ABOVE. EXAMPLE: [TERM] = [[X] / [/] / [÷]] [FACTOR]*

FIG. 1 (CONT'D)

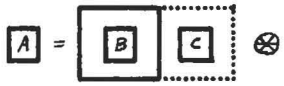
	BACKUS NORMAL FORM	BOX SYNTAX	COMMENTS
(j)	<A> ::= <D> | <EMPTY> <D> ::= <B> | <B> <C> <D>		ZERO OR MORE REPLICATIONS WITH DELIMITER.

FIG. 2 OTHER CONSTRUCTS

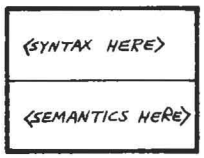
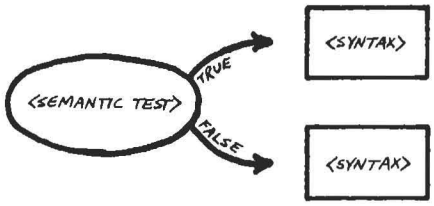
(a)		EMPTY CONSTRUCT.
(b)		ADJACENT SEMANTICS. SYNTAX ABOVE THE LIGHT LINE, SEMANTICS BELOW.
(c)		SEMANTICS-DRIVEN SYNTAX. A SEMANTIC TEST, WRITTEN IN THE SEMANTIC METALANGUAGE, APPEARS INSIDE THE ELLIPSE.

FIG. 3

ALGOL EXAMPLE

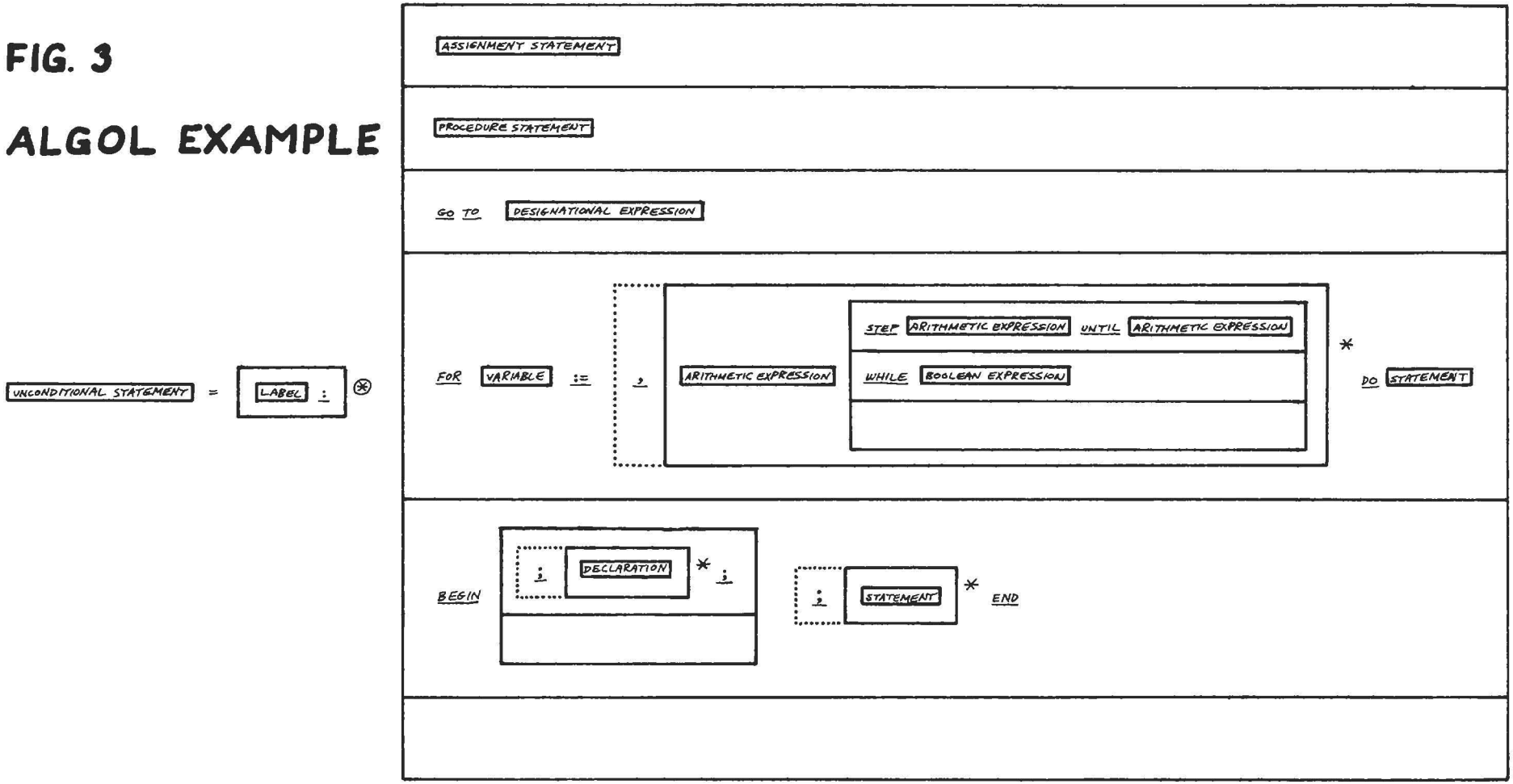
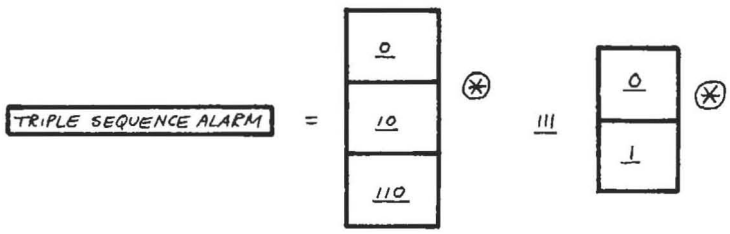


FIG. 4 TRIPLE SEQUENCE ALARM EXAMPLE



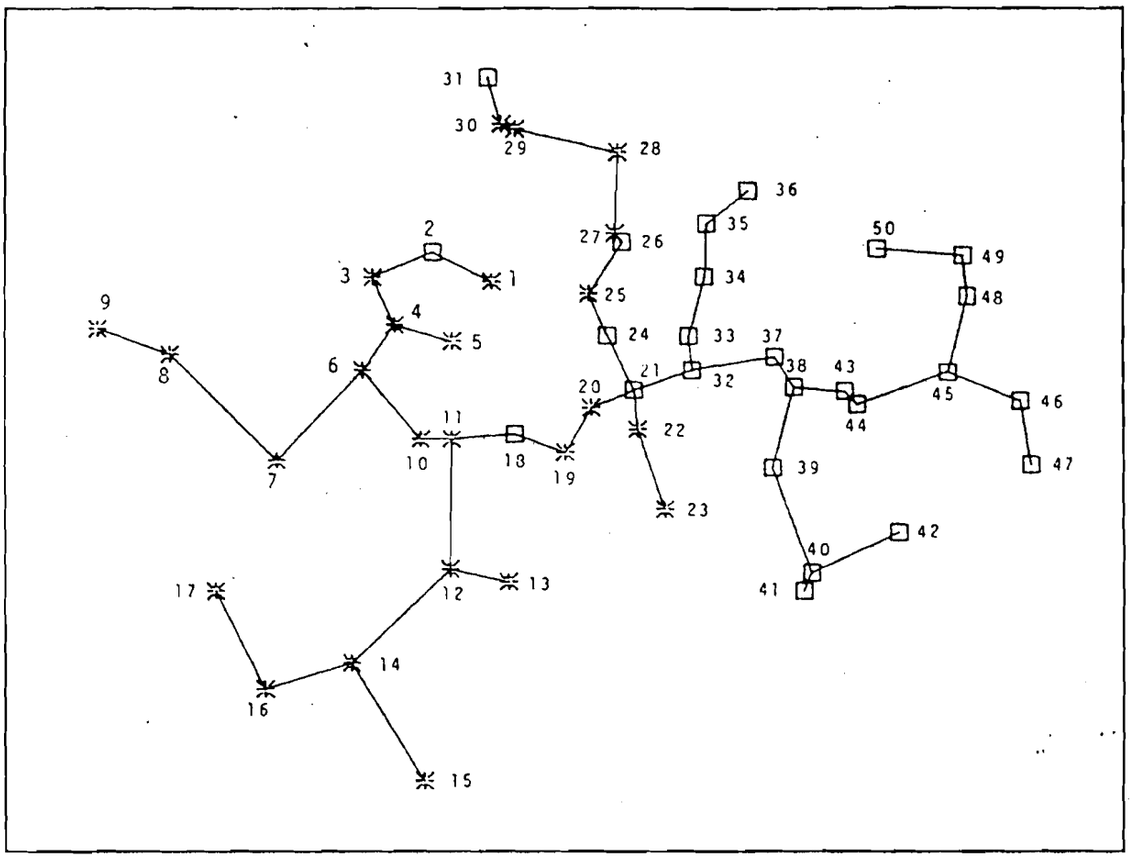


FIGURE 0

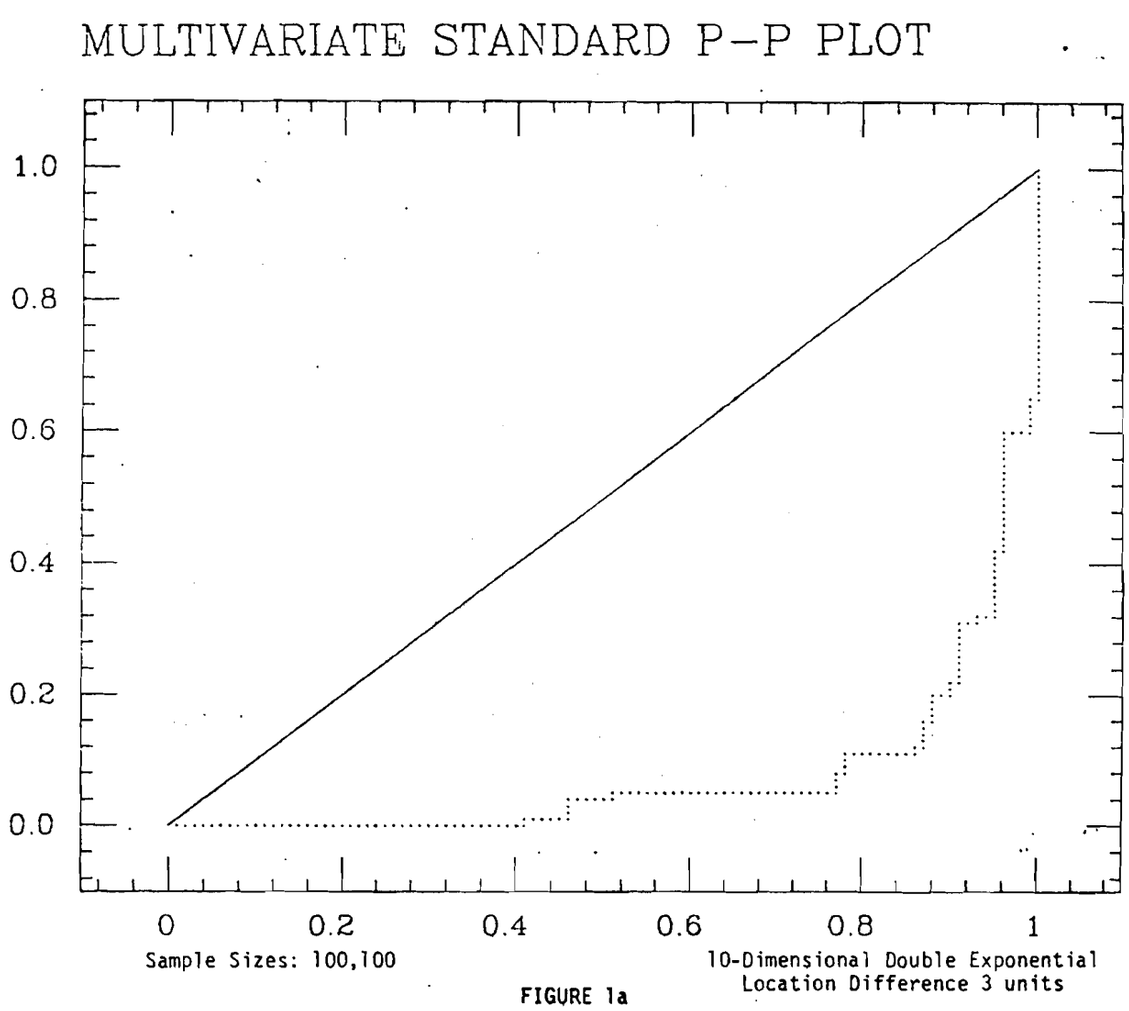


FIGURE 1a

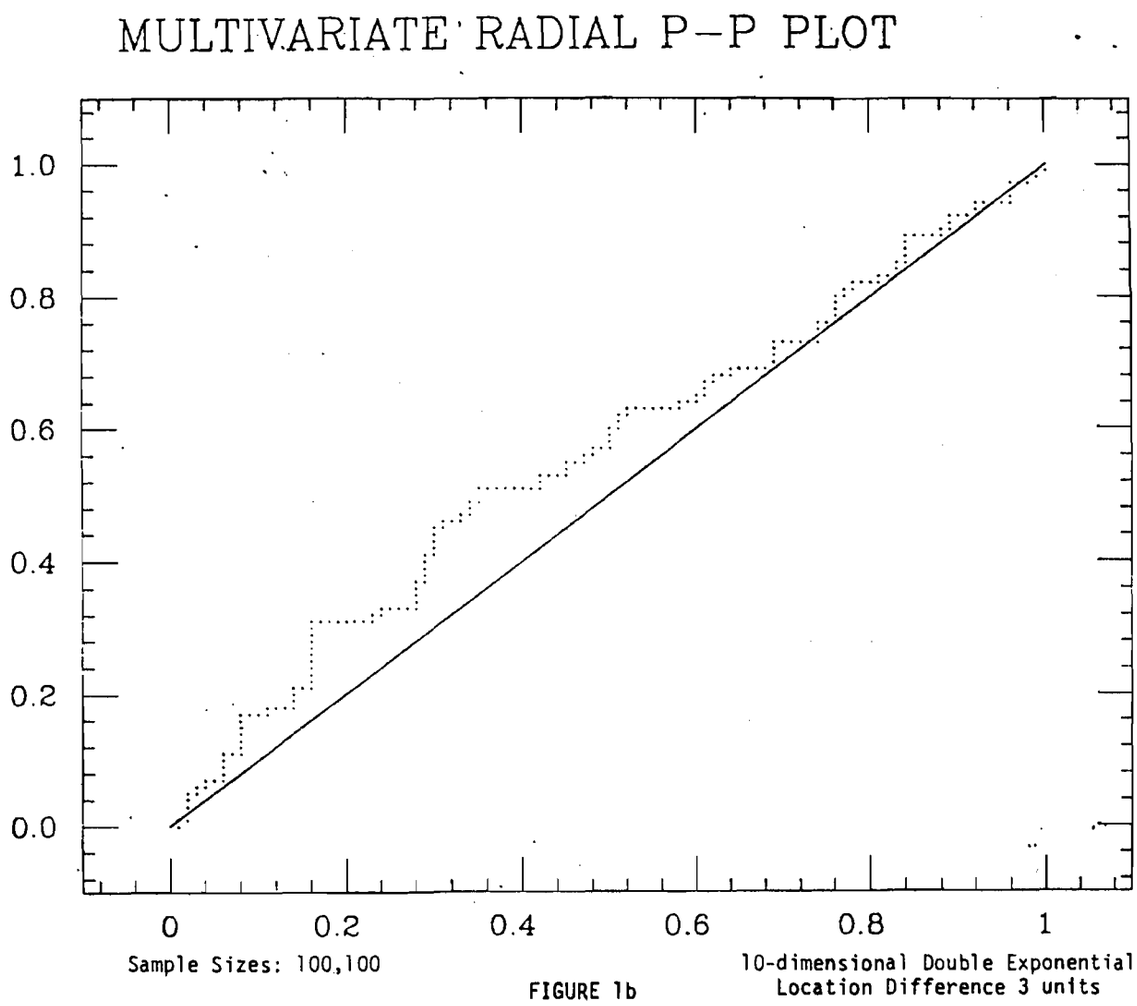


FIGURE 1b

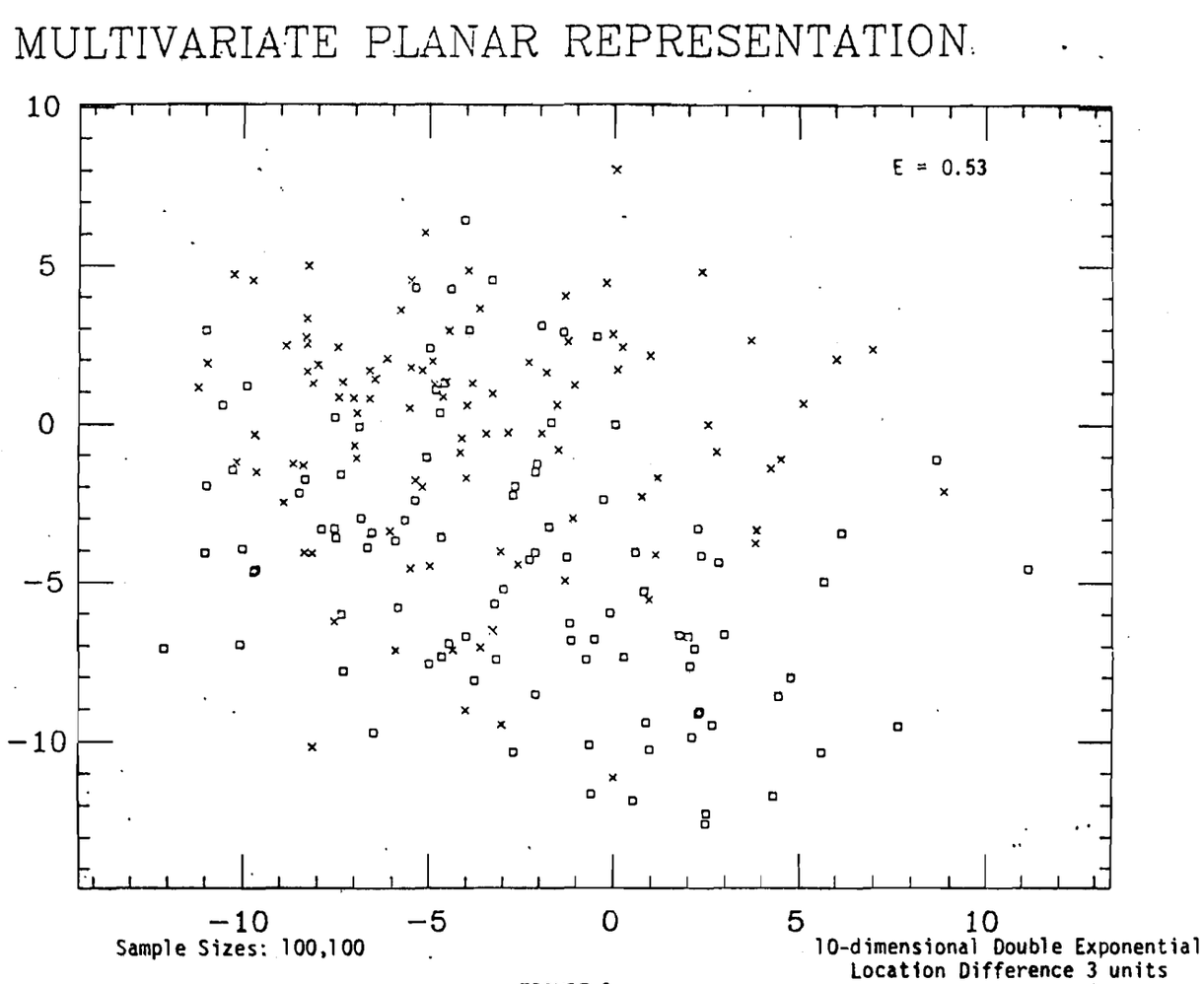


FIGURE 1c

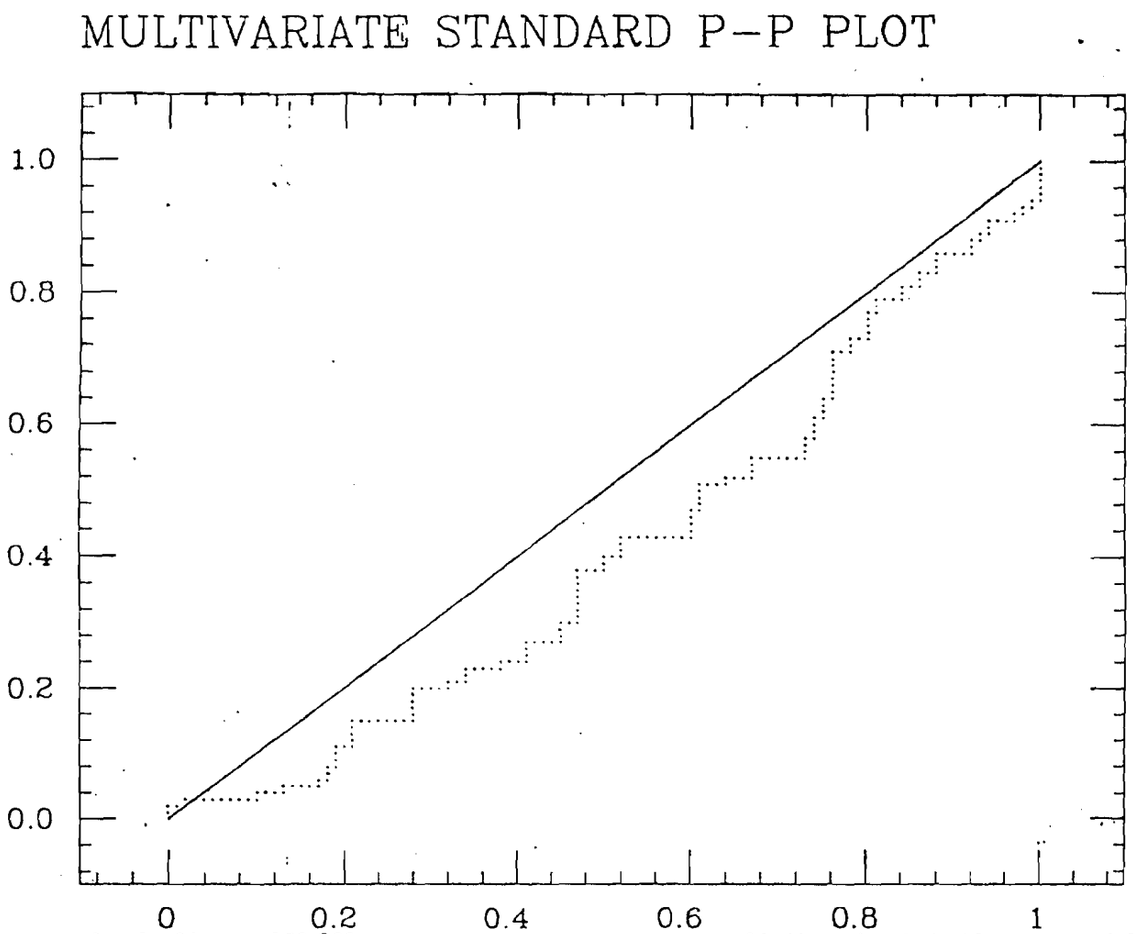


FIGURE 2a

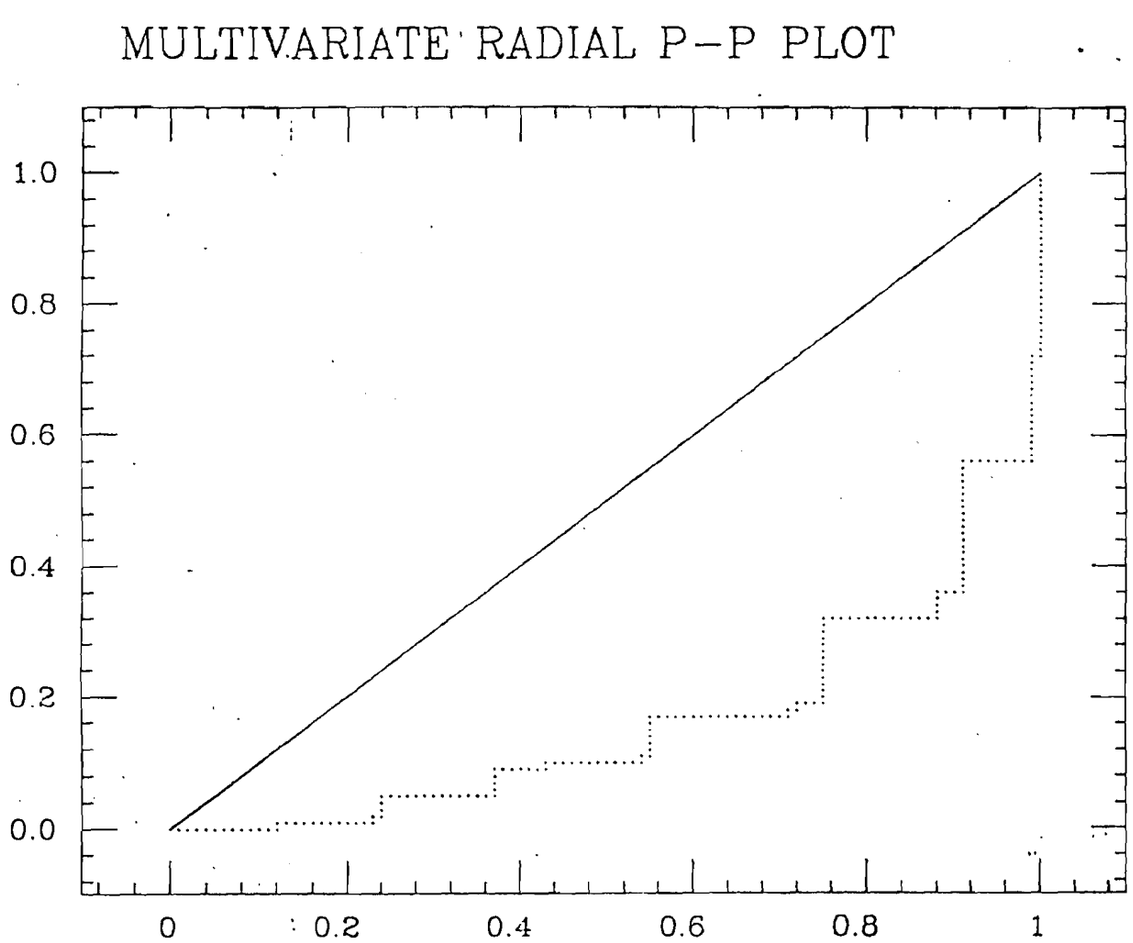
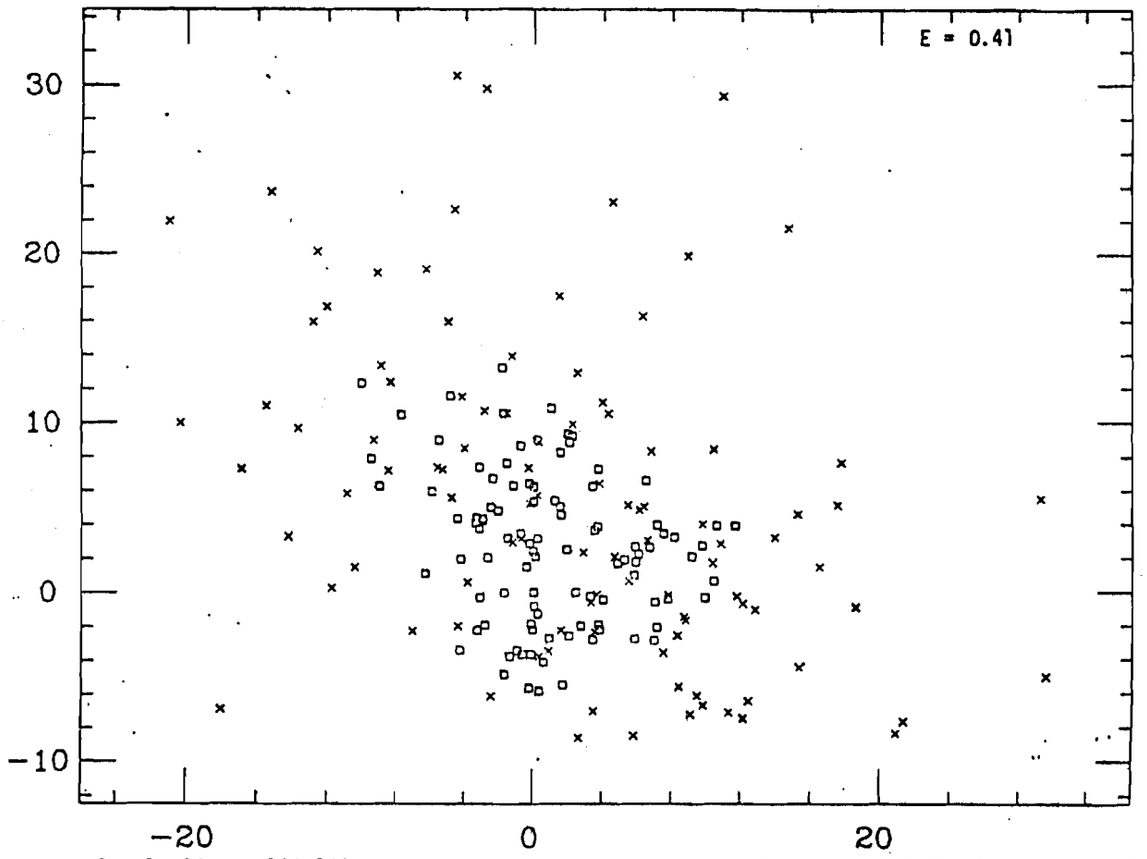


FIGURE 2b

MULTIVARIATE PLANAR REPRESENTATION

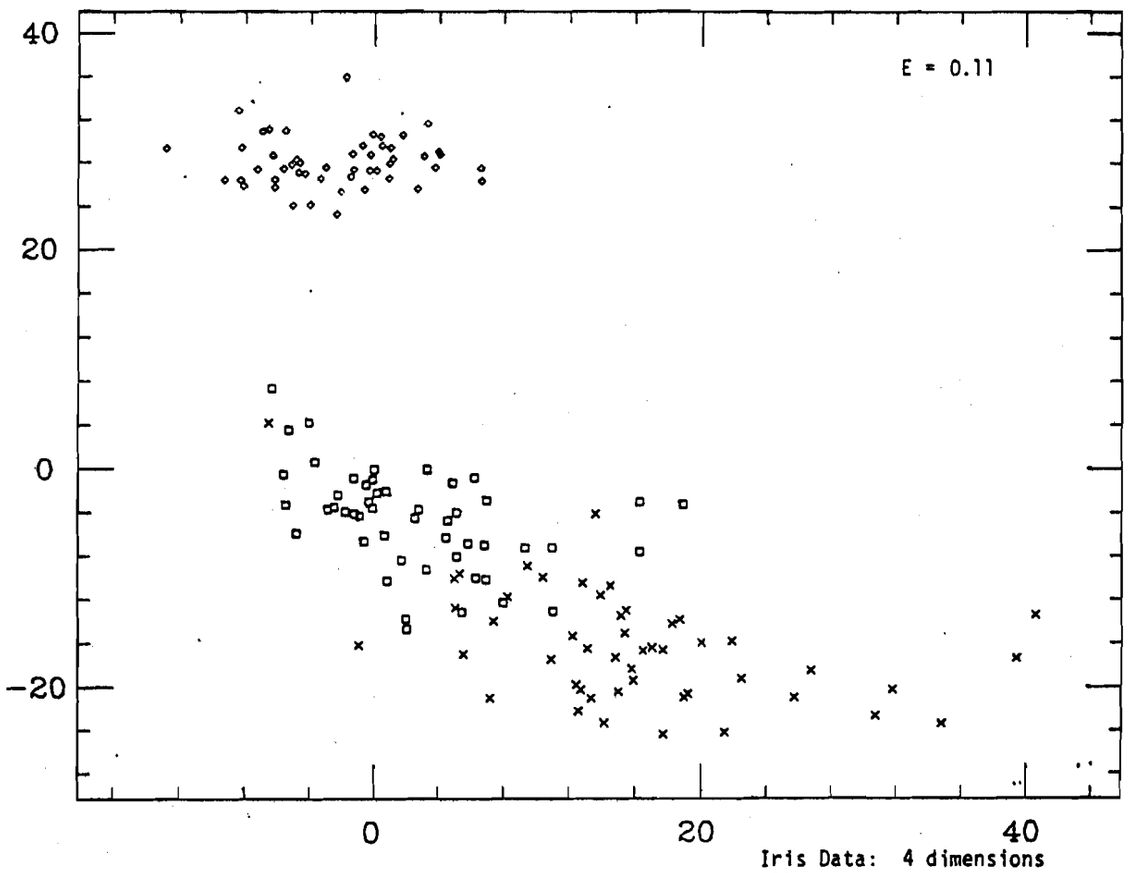


Sample Sizes: 100,100

10-dimensional double exponential
scale difference: 3 units

FIGURE 2c

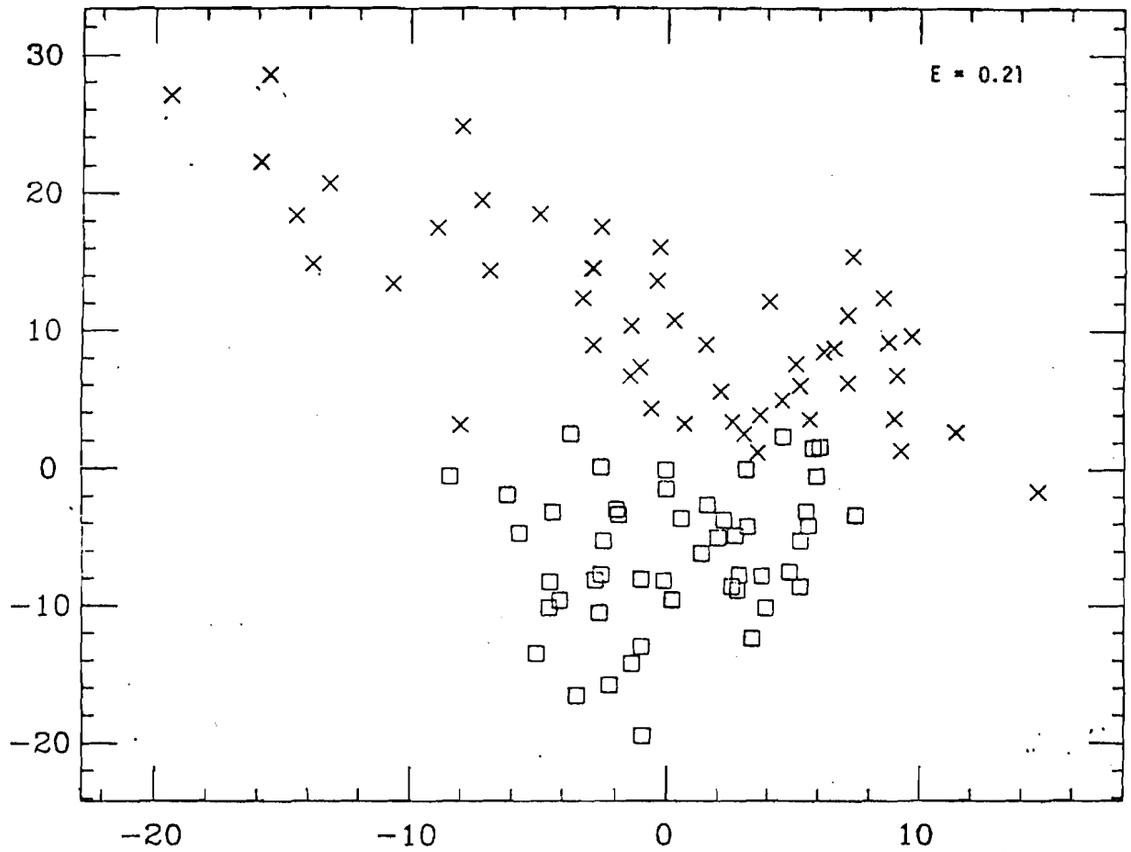
MULTIVARIATE PLANING



Iris Data: 4 dimensions
All 3 species

FIGURE 3

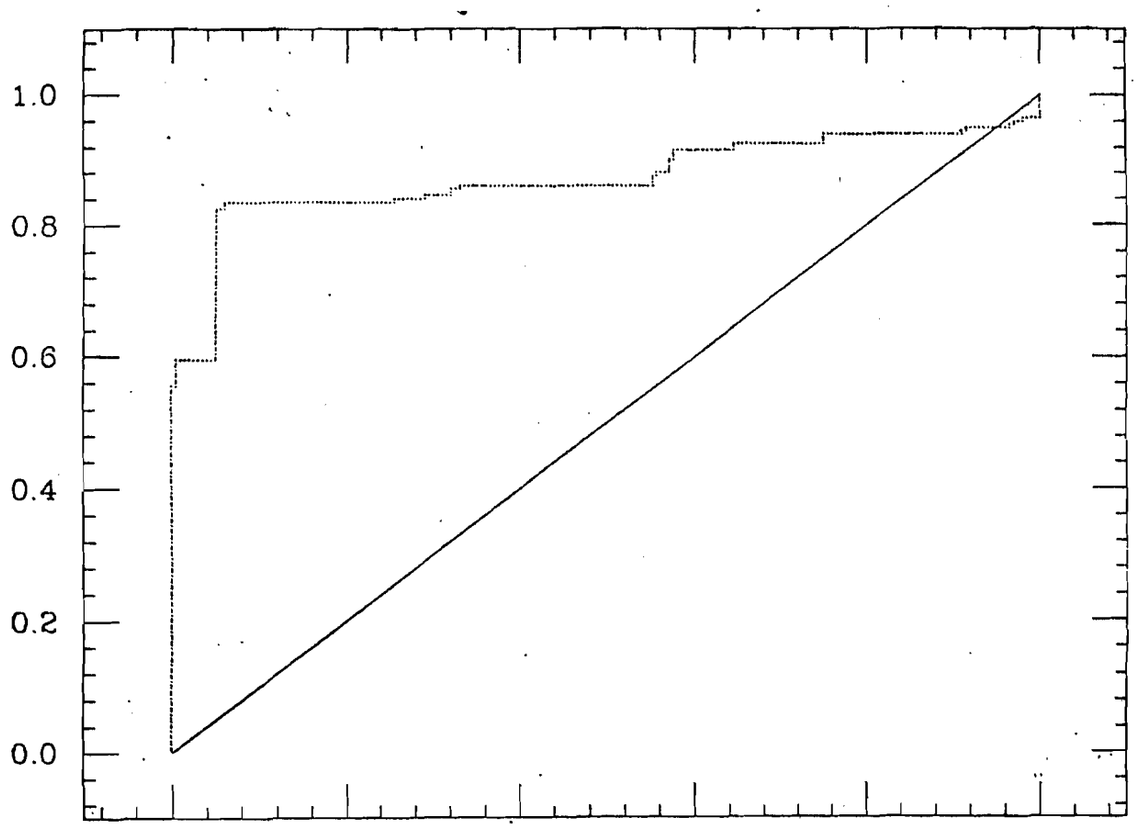
MULTIVARIATE PLANAR REPRESENTATION



Iris Data: 4 dimensions
Versicolor and Virginica

FIGURE 4

MULTIVARIATE STANDARD P−P PLOT



Sample Sizes: 200,200

Particle Physics Data
13 Dimensions

FIGURE 5a

MULTIVARIATE RADIAL P–P PLOT

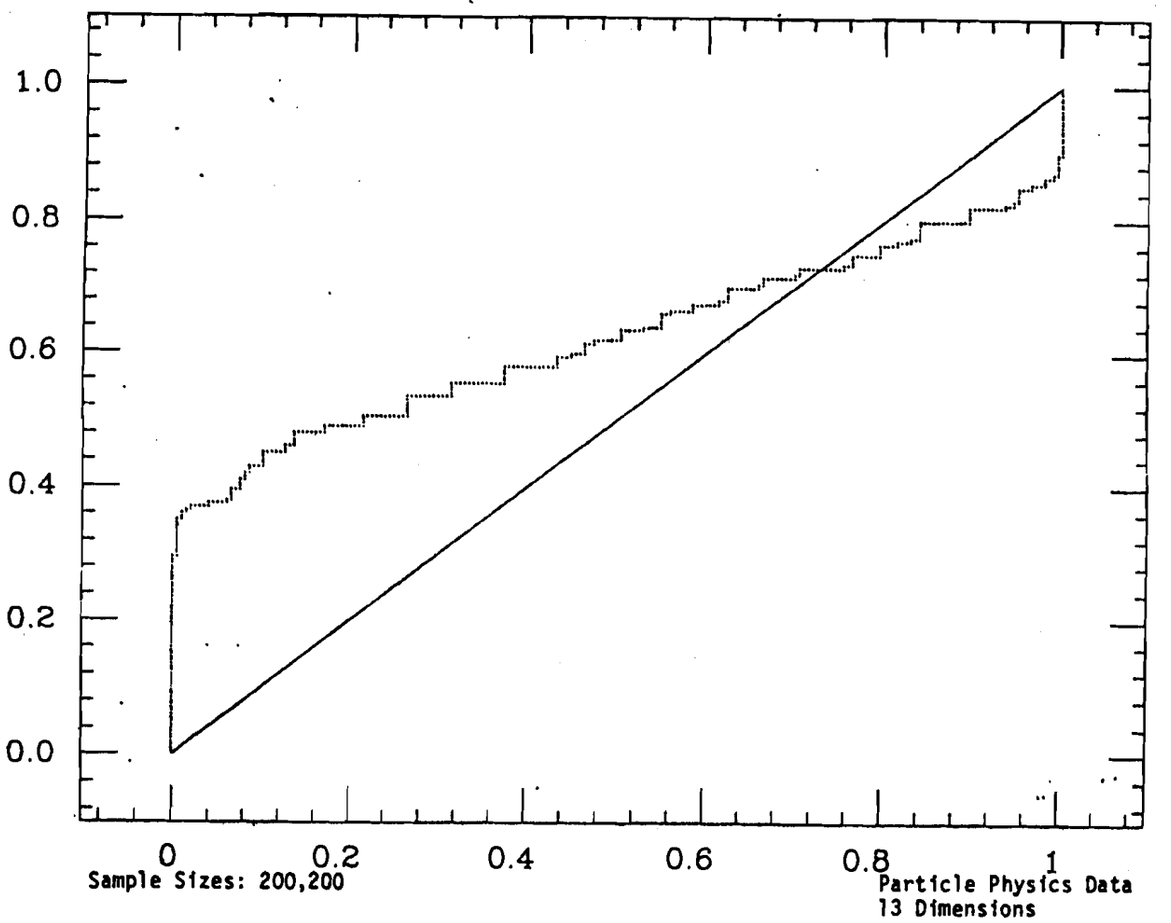


FIGURE 5b

MULTIVARIATE PLANAR REPRESENTATION

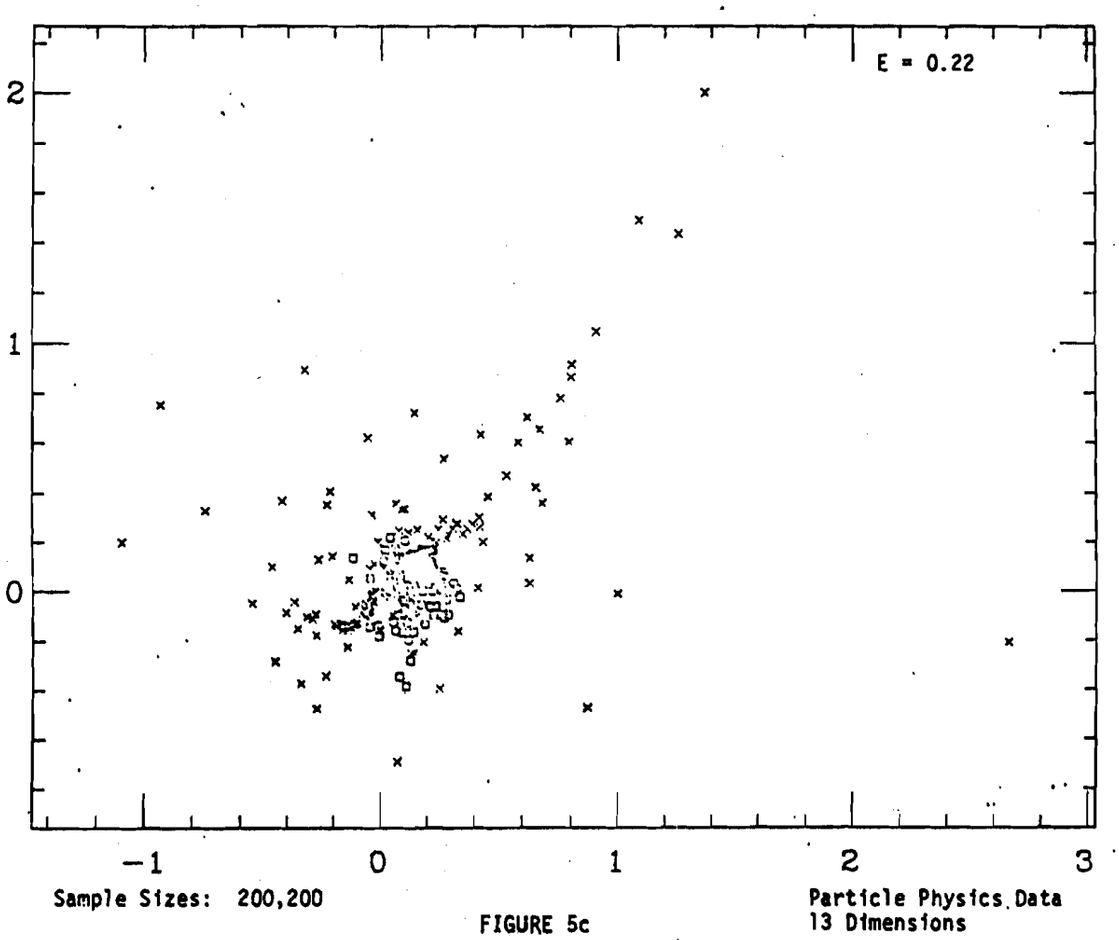


FIGURE 5c

MULTIVARIATE PLANAR REPRESENTATION

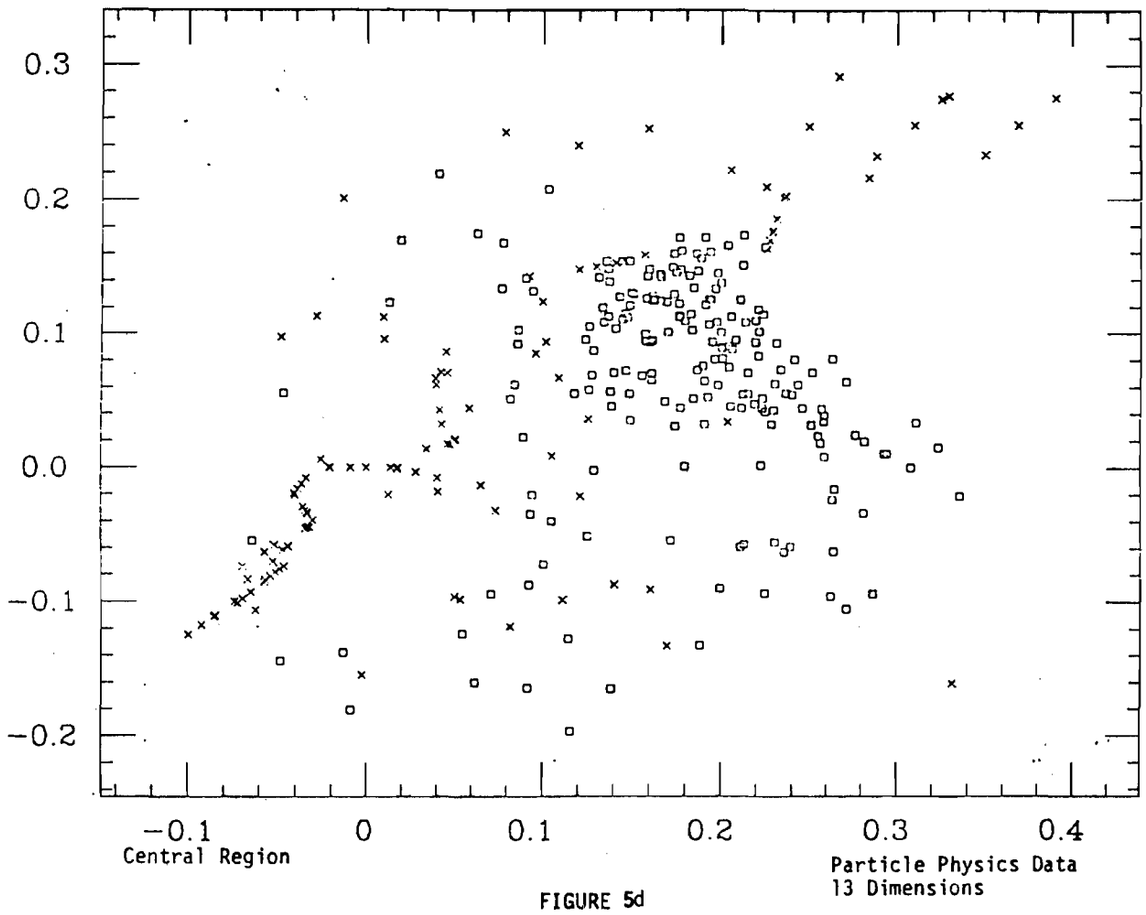


FIGURE 5d