THE UNIVERSITY OF MANITOBA

ISONEMAL ARRAYS AND TEXTILE COMPUTER GRAPHICS

by

JANET ANNE HOSKINS

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF CLOTHING AND TEXTILES

and

DEPARTMENT OF COMPUTER SCIENCE

WINNIPEG, MANITOBA

MARCH 1985

ISONEMAL ARRAYS AND TEXTILE COMPUTER GRAPHICS

BY

JANET ANNE HOSKINS

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

The binary interlacement array representation of woven textile structures is examined algebraically and computationally, and a particular class of interlacement arrays, namely those which are isonemal, is described in detail. The sub-classes of isonemal structures, such as the twills, are enumerated, both by counting arguments and by a computer sieve. Algorithms which permit various types of analysis and factorization of weave patterns into their corresponding loom set-up components are also presented and discussed in terms of their relative efficiency. The necessary features of an interactive computer graphical system for the development and rapid display of woven textile design data are examined and an implementation of such a system described.

ACKNOWLEDGEMENTS

I wish to acknowledge the financial support, in the form of a University of Manitoba graduate fellowship, of the University of Manitoba Faculty of Graduate Studies. I wish to express my gratitude to Prof. Martin King, of the Department of Clothing and Textiles, and Dr. Hugh Williams, of the Department of Computer Science, for their helpful suggestions during the preparation of this thesis. I wish to thank Dr. Anne Penfold Street of the Department of Mathematics, University of Queensland, who has served as a significant source of encouragement and inspiration over the past three years. I also wish to thank Dr. Ralph Stanton, head of the Department of Computer Science, whose foresight and support were important in establishing this program of studies and whose guidance has helped a great deal in its successful completion. Finally, I wish to thank Dr. William Hoskins for many things, but primarily for his constant challenges to me to fulfill my potential.

TABLE OF CONTENTS

INTRODUCTION

CONTENTS

1.1 HISTORICAL CONTEXT

Woven textiles hold an extremely important position in the human environment. Weaving is known to have been practiced at least as far back as 4400 B.C. [11, p. 41] and historical literary references to weaving and woven cloth are virtually limitless [28], [66], [67]. Today, woven fabrics appear in an almost infinite number of end uses, including obvious artifacts, such as clothing and household furnishings, as well as less obvious applications, such as vascular prostheses [47] and aircraft bodies [56].

It is not therefore surprising that there are a great many papers which have considered the subject of weaving, many of them attempting, with varying degrees of success, to document the significant structural characteristics of particular types of woven fabrics and to provide a description of how these fabrics can be reproduced.

A very interesting and valuable weaving resource consists of the published notebooks of the early professional weavers [46], [26] [21]. These manuscripts offer a fascinating historical insight into the types of weaving patterns which were popular during the eighteenth century, as well as the relatively high degree of technical complexity which was exhibited by these designs. The pattern structures also, in some cases, suggest the contemporary use of novel loom attachments and configurations, which are of great interest in their own right [55]. In addition, the variety and complexity of the patterns found in these books is at a level which renders them excellent sources of design inspiration for modern textile designers.

The original purpose of these notebooks was primarily as a personal documentation of the weaver's work and design ideas and as such, the pattern investigation was undertaken in a pragmatic fashion rather than for reasons of curiosity. There is no strong indication of an organized and coherent examination of weave structures or visual images, other than some attempts to create alternative designs by changing the tie-up matrices for the same standard threading. There is also a great deal of overlap between the patterns and structures found in many of these manuscripts due no doubt, largely to the strong influence of fashion in the demand for woven textiles [27].

There is some classification of weave structures present in the nineteenth century weaving literature, as for example Murphy's description and illustration of "leafed fancy tweels" [58, p. 30]; however, the emphasis at this time was directed more toward documentation and discussion of weaving machinery and engineering solutions to the problems of producing complex woven patterns. A typical example is the book written by T.F. Bell, who is said to have been a

"NATIONAL SCHOLAR IN DESIGN (1875-8) AND THIRD GRADE CERTIFICATED ART MASTER, SCIENCE AND ART DEPARTMENT, S.K.: MEDALLIST IN HONOURS AND CERTIFICATED TEACHER IN 'LINEN MANUFACTURING', AND IN 'WEAVING AND PATTERN DESIGNING', CITY AND GUILDS OF LONDON INSTITUTE" [4].

This work comprised a comprehensive treatment of both the aesthetic and practical considerations of woven textile design as related specifically to Jacquard types of looms and included extensive sections devoted to a description of the actual loom mechanics.

The attitude of many of the writers of the time is summarized by Barlow [3] thus:

> [Weaving] calls forth a greater number of mechanical appliances and ingenious contrivances than any other Art, and is on that account alone always a source of interest to the Engineer and the Mechanician, as well as to the Manufacturer and the Weaver.
>
> Formerly the Art depended almost entirely upon the handicraft skill of the weaver, and his contrivances were limited in their combinations to produce designs of any considerable extent. . . . But, by the introduction of the Jacquard Machine and the Power Loom during the present century, the whole system of weaving, with some few exceptions, has been changed, and practically become a New Art.

This interest in weaving machinery continued into the early twentieth century, with books such as Bradbury's Jacquard Mechanism and Harness Mounting [6] and Hooper's Hand-Loom Weaving: Plain & Ornamental [29] providing detailed descriptions of draw looms, jacquard heads, and other related devices.

However, a major focus of much of the twentieth century weaving literature has been the classification and analysis of weaves and woven textiles and the generation of catalogues of structures and artifacts. There have been a number of different approaches to this type of work.

The first approach has involved an exhaustive study of particular woven artifacts of a specific collection or geographic area. Coverlets have been examined extensively in this manner and an outstanding example of this type of study is found in the book Keep Me Warm One Night by Harold and Dorothy Burnham [8]. In this book, a representative sample of the coverlet holdings of the Royal Ontario Museum have been described, both as to their appearance and manufacture and as to their origins, and have been analyzed to determine their weave structures. The organization of the coverlets within the book is based on a division into the commonly observed weave classes such as overshot and double weave (for which rigorous definitions have been developed [2]).

Another important source is the book devoted to coverlets produced by weavers in the State of Indiana in the nineteenth century [57]. In this book, discussion is focused primarily on the individual weavers, with extant samples of these people's work being described in detail. The majority of the artifacts described are jacquard woven coverlets, but this is due to the fact that only the work of professional weavers was documented.

A third and more recent study of this nature is the Tennessee Textile History Project (1978-1983), which is extremely well documented in the book Of Coverlets [72]. This work is primarily a record of the coverlet weaving heritage of Tennessee and focuses on actual artifacts known to be from the area. The historical information about these coverlets is very detailed, as are the descriptions of their physical characteristics. Common use weave structure categories are indicated and there is some classification based on the appearance of recognizable motifs.

These types of studies are historical in nature, concentrating on specific woven textiles, and the information that they contain is of considerable social and cultural interest. They also serve as a valuable reference for weave structures and design motifs suited to coverlet weaving. Although some classification of the fabrics naturally appears in this type of work, this is really of secondary, rather than primary concern.

There is a considerable body of literature which has been devoted to the development of an organized and rational method of classifying weave structures into well-defined categories. This task of classification has been approached in a number of different ways.

Irene Emery, in The Primary Structure of Fabrics [14], developed a system for the classification of some of the simple fabric structures and also discussed some of the problems inherent in any classification system. It was her position that [14, p. xi]

> While there are many possible bases for classification, it is the structural make-up of the fabrics and their component parts that provides data integral to virtually all fabric studies, regardless of the nature or origin of the fabrics, the special interest of the investigator, or the special purpose of the study. Structure is never absent; it is, with negligible exceptions, determinable; it can be objectively observed; and it is varied enough for significant grouping and sub-grouping. Although the details of structure (and element make-up) do not in themselves give a complete picture of a fabric, they provide a sound factual basis for more comprehensive description and, being determinable data, for comparative studies and for classification.

She also felt that "words alone . . . . almost inevitably prove inadequate"

[14, p. xi] in gaining an understanding of the basic principles governing fabric construction and the variety of fabric structures.

Based on these premises, Irene Emery compiled a primary reference source in the area of woven textile structures. As previously stated she did however restrict the scope of her work to very simple structures. Also, although The Primary Structure of Fabrics classifies fabric structures in more rigorously defined categories than previous systems, the categories are still partially descriptive, with inherent ambiguities (eg. broken twill [14, p.129]).

Other attempts at fabric classification of which Warp and Weft: A Textile Terminology by Dorothy K. Burnham [7] and Encyclopedia of Hand-Weaving by Stanislaw Zielinski [75] are representative, have defined their categories with respect to common use definitions and terminology. This system has the major disadvantage of being heavily culturally dependant and ambiguous. D.K. Burnham made an attempt, in her book, to indicate the foreign language equivalents of the name of a particular structure but her categories are broad and sometimes overlap. Also, because of the common use nature of the classification scheme, there is no single basis for distinguishing between fabrics, such as structure in the case of Emery's work. Rather, the fabric types are sometimes defined in terms of their structure [example - satin, 7, p.113], sometimes in terms of their composite materials [example - drugget, 7, p.51], and sometimes in terms of the motifs or patterns which they generally exhibit [example - Star and Diamond, 7, p.135]. Thus, although these types of studies consider a wider range of fabrics and structures than previously, the categories into which fabrics are classified become more ambiguous and less rigorously defined.

The books written for industrial use, such as Woven Cloth Construction by A.T.C. Robinson and R. Marks [65], and Z.J. Grosicki's revised version of Watson's Advanced Textile Design: Compound Woven Structures [22] and Watson's Textile Design and Colour: Elementary Weaves and Figured Fabrics [23], offer a third system of classification. The major feature of this system is the determination of classes of weaves based on how they are constructed. Thus, in addition to structures such as twill and satin, there are categories of fabrics involving "figuring with extra threads", multi-layer fabrics (sub-divided according to how the layers are held together) and "colour and weave effects" ([22], [23], Tables of Contents). The scheme used by these industrial writers is actually a composite of the structural and traditional approaches, modified by functional considerations. Fabrics with "colour and weave effects" are classified by the motifs which they exhibit but only in so far as the motifs are produced by a specialized arrangement of coloured warp and weft yarns. Similarly, multi-layer fabrics are classified by their distinctive structure but the major focus is on the warp and weft configurations which produce these fabrics.

A fourth system of classification of fabric structures is outlined in a paper by H.J. Woods, The Geometrical Basis of Pattern Design [73]. As the title suggests, this system is concerned with classification by pattern and motif and utilizes a sophisticated application of the rules of symmetry to do this. Since all patterns can be classified according to their symmetry groups, this provides a very powerful scheme. It does not however address itself to the special characteristics of woven structures and cannot be used in its present form as the sole means of categorizing woven fabrics.

More recently, an investigation into the nature of the interlacement structure of woven textiles has been undertaken by geometers [24] [63], with the result that an additional system for the description and classification of woven fabrics has been proposed. The great benefit of this approach is that it has applied the inherent rigor and formalism of mathematics to the task at hand.

1.2 OBJECTIVE

The objective of this study was to apply the concepts and techniques of mathematics and computing to an examination of the characteristic interlacement structure of woven textiles, in order to obtain a precise description, classification, enumeration and analysis of these structures. This investigation was focused on five principal areas, as outlined in Section (1.3).

1.3 CHAPTER OUTLINE

In Chapter Two, Grünbaum and Shephard's work on the mathematical definition of some of the common weave structures [24] and the work of Hoskins and Street on the enumeration of the simple twills [44] has been examined and extended. The enumeration and corresponding theoretical development of twills with a bounded float length, balanced twills with a bounded float length, and the non-twill elementary isonemal structures is original to this study and has been published in [41], [40] and [37], respectively. The members of the remaining class of isonemal structures, the compound twillins defined by Hoskins and Thomas [45], have also been enumerated by the author and the results published in [43] and [42].

Chapter Three is concerned with algorithms for the structural analysis of woven fabrics. The process of deriving the interlacement array corresponding to a given set of threading, tie-up and shed sequence matrices is considered, as well as the inverse process of factoring a given interlacement array into its composite threading, tie-up and shed sequences. Several new factorization algorithms have been developed and implemented, and their performance compared with that of classical methods. One of these algorithms, the one based on a bucket sort technique, has appeared in [36] but the Minimal Bucket Sort Algorithm and the algorithm for factoring coloured interlacement arrays have not been published previously. The problem of factoring an interlacement array so as to have a threading matrix for multiple threading and of defining the permutation of a threading matrix so as to cause it to conform to some previous specification are also considered in this chapter.

Chapter Four examines the unique characteristics of multi-layered fabrics. The existing classification schemes for these structures are examined [22] [59], as are the extant algorithms for determining the reducibility [32] of binary interlacement arrays [10], [59], [15]. A new algorithm for determining reducibility, which is based on a systematic permutation of the rows and columns of the interlacement matrix, is introduced at this time.

Also examined in Chapter Four is the question of how to illustrate multi-layered fabric interlacement data so as to accurately and meaningfully represent the nature of these structures. Cross-sectional diagrams are one form of representation which has been frequently used in discussing multiple layered weave structures (eg. [22, p.105], [15, p. 128], [68, p. 100]), but there has been no attention paid to the development of an organized method of creating such illustrations. A technique for the automatic generation of cross-sectional representations of two, three and four layer fabrics, directly from the binary interlacement data, has been developed by the author and has appeared in [30], [33]. This process is described in this chapter.

Chapter Five further examines the procedure for representing woven fabrics. In this instance, the problems inherent in creating a graphics image which clearly illustrates the surface appearance of a woven fabric are considered. A system is described which has been developed and implemented for the automatic generation of such diagrams from a given interlacement array. Also discussed is an extension of this system to permit the representation of leno, cross woven and braided structures. This latter implementation is further discussed in terms of the graphics editor which is required for the entry and editing of the corresponding design data.

Chapter Six is concerned with the discussion and development of the principles of an interactive user-computer interface suitable for a designer involved with the design and production of textile samples. The particular system described has been designated PATTERN MASTER IV, and the architecture and internal algorithms have been published in [34], [39], [38]. The particular implementation discussed in this chapter is implemented on an APPLE II+ microcomputer and is designed to operate in conjunction with the Ahrens and Violette computer controlled dobby loom.

CHAPTER 2

ISONEMAL INTERLACEMENT ARRAYS

CONTENTS

2.1 INTRODUCTION

For clarity in the ensuing discussion, Figure (2.1.1) illustrates the salient parts of a simplified loom and shows the relationship between its moving parts and the resultant fabric.

The weave structure classification scheme developed by Grünbaum and Shephard and based on "elementary geometry, group theory, number theory and combinatorics" [24] has been further examined and extended to include other isonemal structures. By making use of the results from [44] and [45], all of the members of the classes of isonemal fabrics have been enumerated. A discussion of this investigation follows.

It is necessary first to introduce the following definitions to simplify the ensuing discussion [24], [44].

Definition 2.1.2. A binary sequence $S = (s_i)$ of period n is a sequence of zeros and ones such that

$$s_k = s_i, \qquad k \equiv i \pmod{n}.$$

Two binary sequences of period n are considered equivalent if and only if one can be transformed into the other by a cyclic shift, reversal, complementation, or any finite sequence of these operations. Cyclic shift and reversal correspond to the action of the dihedral group of order 2n on

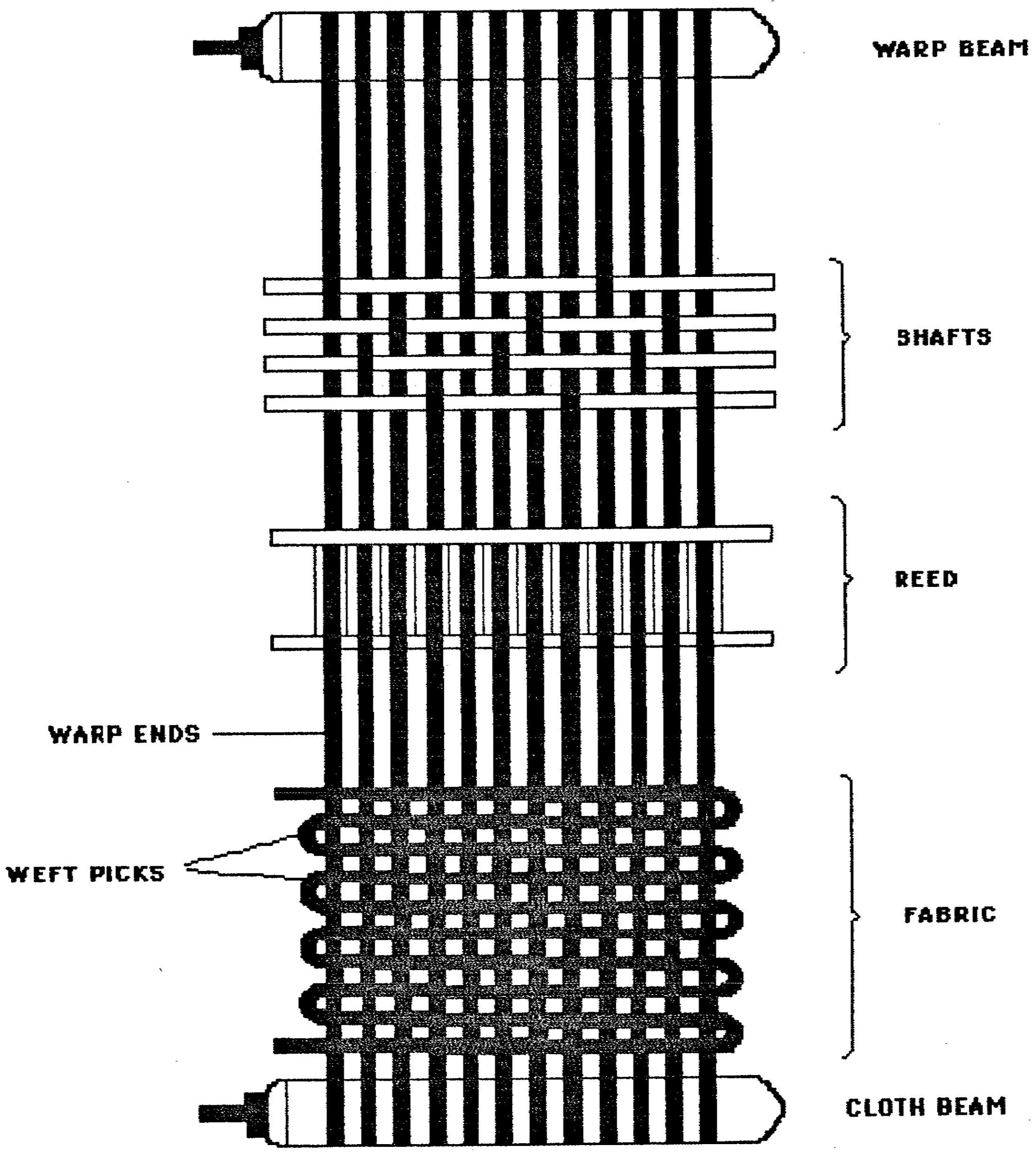

WARP BEAM
SHAFTS
REED
WARP ENDS
WEFT PICKS
FABRIC
CLOTH BEAM

FIGURE 2.1.1

the sequence, while complementation corresponds to the action of the symmetric group of order 2.

Example 2.1.3. When n = 4, the induced equivalence classes of the 16 possible binary sequences are given by the following:

{0 0 0 0, 1 1 1 1}

{1 0 1 0, 0 1 0 1}

{1 1 0 0, 0 1 1 0, 1 0 0 , 0 0 1 1}

{0 0 0 1 , 0 0 1 0, 0 1 0 0, 1 0 0 0, 1 1 1 0, 1 1 0 1, 1 0 1 1, 0 1 1 1}

Definition 2.1.4. An interlacement sequence Q of period n is a sequence of over and under crossings of orthogonal strands, and each interlacement sequence can be associated with an equivalence class of binary sequences of the same period.

Definition 2.1.5. A warp (weft) - mononemal design is one in which, for a given interlacement sequence Q, the columns (rows) are all equivalent to Q.

Definition 2.1.6. A mononemal design is one with a column interlacement sequence Q and row interlacement sequence R such that Q is equivalent to R. Implicit in this definition is that a mononemal design must be both warp and weft mononemal.

Definition 2.1.7. An elementary warp (weft) - isonemal design is warp (weft) mononemal and the operation that is applied to each column (row) to obtain the next one must be such that each column (row) maintains a constant relationship with its neighbours. In addition, the same operation must be applied to each column (row) to obtain the next one. This implies that the symmetry group is cyclic and that the transformation from one row (column) to the next is always the same.

Definition 2.1.8. An elementary isonemal design is mononemal, elementary warp isonemal, and elementary weft isonemal. Also the relationship between each column and its neighbours must be equivalent to the relationship between each row and its neighbours, that is, the symmetry group of such a design contains both the cyclic group of order n, taking one column to the next, and the group of order 2, taking columns into rows and vice versa. (For the purposes of enumeration, the warp and weft are assumed to be equivalent. In addition, the fabric is considered to be infinite, with no boundary conditions applying.)

Definition 2.1.9. Two woven fabrics are regarded as design equivalent if one can be transformed into the other by: turning the fabric over; shifting some warp (weft) threads from one side to the other (preserving their cyclic order and interlacements); taking a mirror image of the fabric parallel to either warp or weft directions; interchanging warp and weft;

any finite combination of these operations.

Definition 2.1.10. [42] A sequence for which reversal has the same effect as cyclic shift is said to be a palindrome. Such sequences are of three types, namely,

1. $a_0a_1a_2 \ldots a_{w-1}a_{w-1} \ldots a_2a_1a_0$, of period 2w,
2. $a_0a_1a_2 \ldots a_{w-1}a_wa_{w-1} \ldots a_2a_1$, of period 2w,
3. $a_0a_1a_2 \ldots a_{w-1}a_{w-1} \ldots a_2a_1$, of period 2w - 1.

Only those of type (1) occur in the present context.

Definition 2.1.11. [42] A sequence for which reversal and complementation together have the same effect as cyclic shift is said to be a co-palindrome. Such sequences are of the form

$$a_0a_1a_2 \ldots a_{w-1}\bar{a}_{w-1} \ldots \bar{a}_2\bar{a}_1\bar{a}_0,$$

with period 2w.

Definition 2.1.12. [42] The weight of a sequence is the number of ones that occur per period. The palindromes of type (1) have even weight, and we may assume without loss of generality that the weight is at most w. The co-palindromes have weight precisely w.

2.2 THE TWILLS

2.2.1 THE FUNDAMENTAL PROBLEM

The first and most simple type of isonemal structure is the twill. Twill fabrics have been woven since at least 1500-1000 BC [11] and twill weaves have long been one of the most versatile and commonly used structures:

> In the texture of plain cloth, each thread or woof [weft] passes over and under a thread of warp, alternately; and two leaves [shafts/harnesses] of heddles, only, are requisite to produce this effect. Tweeling [twills], however, takes a greater range with respect to the intervals at which the threads of warp and weft are interwoven; and these intervals increase and vary in proportion to the number of leaves employed, and the order in which they are raised and sunk. Next to plain texture, tweelling is the most extensive in its application to every branch of the cloth manufacture: it not only serves as a ground on which other decorations are woven, but it forms, purely on its own principles, some of the most beautiful patterns which can be produced in the art of weaving. [58, p.22]

Based on the principles of isonemality formulated by Grünbaum and

Shephard [24], Hoskins and Street [44] developed a theoretical formula for enumerating the twills and computed the members of the sets of twills for $n \leq 20$. For completeness, a summary of this work is included in this subsection.

Definition 2.2.1.1. A simple twill on n harnesses is a planar interlacement array in which each row (column) of the array is an interlacement sequence of period n and is obtained from the previous row (column) by displacement through one position. Thus the interlacements of a simple twill can be regarded as a binary array generated as a square tiling by a circulant binary array of period n.

The number of inequivalent twills for given n is equal to the number of equivalence classes of binary sequences of length n induced by the action of the group $D_{2n} \times S_2$, that is the direct product of the dihedral group of order $2n$ with the symmetric group of degree 2. This means that the numbers of twills could be enumerated using:

LEMMA 2.2.1.2. [9]

Let G be a finite group, of order g, of transformations acting on a finite set S and let two elements of S be equivalent if and only if one can be transformed into the other by a transformation in G. Then the number T of inequivalent elements is

$$T = (1/g) \sum_{g \in G} I(t),$$

where $I(t)$ is the number of elements of S left invariant by a transformation t belonging to G, and the sum is over all g transformations in G. □

COMPUTATIONAL ALGORITHM

The computational algorithm used to determine all of the inequivalent twills for $2 \leq n \leq 20$ is a sieve given by the following steps:

1. A vector X of bits with $2^n - 2$ components is initialized to ones. For example, if n = 4, then

   $$X = \begin{array}{cccccccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14. \end{array}$$

2. The index of the first non-zero entry in the vector gives a new twill. In our example, the first twill is given by the index 1. This corresponds to the binary sequence S of length 4, 0 0 0 1, which is the binary representation of the decimal number 1.

3. The index in X for any sequence which can be transformed into S under the action of the direct product $D_{2n} \times S_2$ is set to zero. The sequences 0 0 1 0, 0 1 0 0, 1 0 0 0, 1 1 1 0, 1 1 0 1, 1 0 1 1,

0 1 1 1, corresponding to the decimal integers 2, 4, 8, 14, 13, 11, and 7, respectively are equivalent to S.

4. Steps 2 and 3 are repeated until done. In this example, the final vector X = 1 0 1 0 1 0 0 0 0 0 0 0 0 0, indicating that there are 3 twills on 4 harnesses given by the sequences 0 0 0 1, 0 0 1 1, 0 1 0 1 which correspond to the decimal integers 1, 3 and 5, respectively.

The numbers of twills for 2 $\leq n \leq$ 20 are given in [Table 2.1.3].

TABLE 2.2.1.3

n	NUMBER OF TWILLS
2	1
3	1
4	3
5	3
6	7
7	8
8	17
9	22
10	43
11	62
12	121
13	189
14	361
15	611
16	1161
17	2055
18	3913
19	7154
20	13647

A subset of the twills, namely the balanced twills, being those structures in which each strand passes over and under those perpendicular to it equally often, were also determined and enumerated for $n \leq 18$, and are given in Table (2.2.1.4). These twills clearly correspond to the subset of binary sequences with weight equal to $n/2$.

TABLE 2.2.1.4

n	NUMBER OF BALANCED TWILLS
2	1
4	2
6	3
8	7
10	13
12	35
14	85
16	257
18	765

2.2.2 TWILLS WITH BOUNDED FLOAT LENGTH

A sequence with $s_i \neq s_{i+1} = \ldots = s_{i+k} \neq s_{i+k+1}$ is said to have a float of length k, that is, a block of k consecutive symbols which are equal. The maximum float length is closely related to the number of breaks in the sequence, where $(s_1, s_2, \ldots, s_n)$ has m breaks if and only if $s_i \neq s_{i+1}$ for precisely m distinct values of $i = 1, \ldots, n$. For example, the sequences 000111 and 00100111 both have maximum float length three, and have two and four breaks respectively. Note that the number of breaks must always be even.

The maximum float length is an important property of the twills, and indeed of any woven structure since it is a major factor in determining the functional performance and aesthetic appearance of a fabric. Long floats deprive the fabric of interlacements that provide structural stability, as well as increasing the possibility that the long exposed yarn will be snagged and damaged.

The number of equivalence classes of binary sequences of length n, with maximum float length k is denoted by F(n,k), and the number of classes of such sequences with exactly m breaks and exactly x floats of length k is denoted by F(n,k,m,x). Then

(2.2.2.1) $$F(n,k) = \Sigma\, F(n,k,m,x),$$

where the summation is over all m and x satisfying the following conditions:

(a) m is even, and $\lceil n/k \rceil + \delta \le m \le n - k + \varepsilon$,

$\delta, \varepsilon = 0$ or 1, $\delta \equiv \lceil n/k \rceil \pmod 2$, $f \equiv n - k \pmod 2$;

(b) if $n = kq + r$, $0 \le r \le k - 1$, then

$1 \le x \le q$ if $r \ge 2$,
or if $r = 1$, q odd,
or if $r = 0$, q even;

$1 \le x \le q-1$ if $r = 1$, q even,
or if $r = 0$, q odd;

(c) $x \le m \le n - kx + x - \varepsilon$, $\varepsilon \equiv n - kx + x \pmod 2$, $\varepsilon = 0$ or 1,

and

if $m = x$, then $k \mid n$, n/k is even, and $x = n/k$.

Similarly, the sum of F(n,k,m,x) over all m satisfying the conditions stated above, for given x is denoted by F(n,k,–,x).

Note that it is assumed that $k < n$, for if $k = n$, then $m = 0$, no corresponding twill exists, and condition (a) becomes $2 \leq m \leq 0$. Hence $F(n,n)$ is not defined.

Let $S(n,k,m,x)$ denote the set of binary sequences of length n, with maximum float length k, m breaks, and x floats of length k, and let $s \in S(n,k,m,x)$, where $s = \{s_1, \ldots, s_n\}$. The convention is made that $s_n \neq s_1$ (which is always possible since $k < n$), and associated with s is the sequence of positive integers

$$r(s) = \{r_1, \ldots, r_m\},$$

where $s_1 = \ldots = s_{r(1)} \neq s_{r(1)+1} = \ldots = s_{r(1)+r(2)} \neq s_{r(1)+r(2)+1} = \ldots$ (In the traditional break notation of weaving, this would be written

$$\frac{r_1 \quad r_3}{\quad r_2 \quad r_4} \ldots\ldots)$$

Thus, for example, the sequence $s = (00101) \in S(5,2,4,1)$ has associated sequence $r(s) = (2,1,1,1)$.

Let $R(n,k,m,x)$ be the set of positive integer sequences of length m, where

$$r = (r_1, \ldots, r_m) \in R(n,k,m,x)$$

satisfies

$$1 \leq r_i \leq k, \text{ for } i = 1,2, \ldots ,m,$$

$$r_i = k \text{ for exactly } x \text{ values of } i,$$

and

$$\sum_{i=1}^{m} r_i = n.$$

(Note that r is actually a composition of n into m parts, where x parts equal k, and $m - x$ parts are less than k.) Then each $s \in S(n,k,m,x)$ corresponds to $r(s) \in R(n,k,m,x)$, and conversely each $r \in R(n,k,m,x)$ corresponds to exactly two sequences $s, s' \in S(n,k,m,x)$, where one is the complement of the other. Thus these two sequences, s and s', are equivalent in $S(n,k,m,x)$.

The equivalence relation on sequences in $S(n,k,m,x)$, induced by the action of $D_{2n} \times S_2$, corresponds to an equivalence relation on $R(n,k,m,x)$, where two sequences in $R(n,k,m,x)$ are equivalent if and only if one can be transformed into the other by a cyclic shift, v , by a reversal u, or by some finite sequence of these operations.

If $r = \{r_1, r_2, \ldots, r_{m-1}, r_m\}$, then $rv = \{r_m, r_1, r_2, \ldots, r_{m-1}\}$, and $ru = \{r_m, r_{m-1}, \ldots, r_2, r_1\}$.

Hence the equivalence relation on R(n,k,m,x) is induced by the dihedral group of order 2m defined by

(2.2.2.2) $\qquad G = \{u,v\}.$

Thus F(n,k,m,x), the number of equivalence classes of binary sequences in S(n,k,m,x) under the action of $D_{2n} \times S_2$, equals the number of equivalence classes of positive integer sequences in R(n,k,m,x) under the action of G.

A formula has been developed for enumerating F(n,k,m,x)and details of the counting arguments are given in [41]. Using a sieve algorithm and the twill sequences found in [44], the values of F(n,k,-,x) for $n \leq 20$, and x = 1 to 10 have been calculated and checked against the theoretical results. These values are listed in the following tables.

TABLE 2.2.2.3

F(n,k,-,1), the number of twills on n harnesses, with maximum float length k which occurs precisely once per period for n = 4, . . . ,20, k = 1, . . . ,n-1.

n\ k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
4	0	0	1																
5	0	1	1	1															
6	0	0	1	1	1														
7	0	1	2	2	1	1													
8	0	0	3	2	2	1	1												
9	0	1	4	5	3	2	1	1											
10	0	0	7	7	5	3	2	1	1										
11	0	1	10	14	9	6	3	2	1	1									
12	0	0	16	22	17	9	6	3	2	1	1								
13	0	1	25	43	30	19	10	6	3	2	1	1							
14	0	0	40	72	58	33	19	10	6	3	2	1	1						
15	0	1	62	136	106	66	35	20	10	6	3	2	1	1					
16	0	0	101	238	205	122	69	35	20	10	6	3	2	1	1				
17	0	1	159	445	384	242	130	71	36	20	10	6	3	2	1	1			
18	0	0	257	796	740	460	258	133	71	36	20	10	6	3	2	1	1		
19	0	1	410	1476	1406	909	498	266	135	72	36	20	10	6	3	2	1	1	
20	0	0	663	2674	2710	1756	988	514	269	135	72	36	20	10	6	3	2	1	1

TABLE 2.2.2.4

F(n,k,–,2) = number of twills on n harnesses with maximum float length k which occurs precisely twice per period for n = 4, . . . ,20, k = 1, . . . ,⌊n/2⌋.

n\ k	1	2	3	4	5	6	7	8	9	10
4	0	1								
5	0	0								
6	0	2	1							
7	0	0	0							
8	0	3	2	1						
9	0	0	2	0						
10	0	4	5	2	1					
11	0	0	6	2	0					
12	0	5	15	7	2	1				
13	0	0	18	8	2	0				
14	0	6	41	23	7	2	1			
15	0	0	58	34	10	2	0			
16	0	7	113	80	25	7	2	1		
17	0	0	174	134	42	10	2	0		
18	0	8	325	291	98	27	7	2	1	
19	0	0	514	524	178	44	10	2	0	
20	0	9	929	1079	392	106	27	7	2	1

TABLE 2.2.2.5

F(n,k,–,3) = number of twills on n harnesses with maximum float length k which occurs precisely three times per period for $n = 4, \ldots, 20$, $k = 1, \ldots, \lfloor (n-1)/3 \rfloor$.

n\ k	1	2	3	4	5	6
4	0					
5	0					
6	0					
7	0	0				
8	0	0				
9	0	3				
10	0	0	1			
11	0	5	1			
12	0	0	3			
13	0	8	6	1		
14	0	0	11	1		
15	0	12	22	4		
16	0	0	46	6	1	
17	0	16	82	17	1	
18	0	0	163	32	4	
19	0	21	306	77	7	1
20	0	0	572	158	17	1

TABLE 2.2.2.6

F(n,k,-,4) = number of twills on n harnesses with maximum float length k, which occurs precisely four times per period, for $n = 4, \ldots, 20$, $k = 1, \ldots, \lfloor n/4 \rfloor$.

n\ k	1	2	3	4	5
4	1				
5	0				
6	0				
7	0				
8	0	1			
9	0	0			
10	0	3			
11	0	0			
12	0	8	1		
13	0	0	0		
14	0	16	3		
15	0	0	3		
16	0	29	11	1	
17	0	0	19	0	
18	0	47	49	3	
19	0	0	85	3	
20	0	72	211	14	1

TABLE 2.2.2.7

F(n,k,-,5) = number of twills on n harnesses with maximum float length k, which occurs precisely five times per period, for $n = 6, \ldots, 20,\ k = 1, \ldots, \lfloor (n-1)/5 \rfloor$.

n\ k	1	2	3
6	0		
7	0		
8	0		
9	0		
10	0		
11	0	1	
12	0	0	
13	0	5	
14	0	0	
15	0	16	
16	0	0	1
17	0	38	1
18	0	0	5
19	0	79	12
20	0	0	28

TABLE 2.2.2.8
F(n,k,–,6) = number of twills on n harnesses with maximum float length k, which occurs precisely six times per period, for $n = 6, \ldots, 20,\ k = 1, \ldots, \lfloor n/6 \rfloor$.

n\ k	1	2	3
6	1		
7	0		
8	0		
9	0		
10	0		
11	0		
12	0	1	
13	0	0	
14	0	4	
15	0	0	
16	0	16	
17	0	0	
18	0	50	1
19	0	0	0
20	0	126	4

TABLE 2.2.2.9

$F(n,k,-,7)$ = number of twills on n harnesses with maximum float length k, which occurs precisely seven times per period, for $n = 8, \dots, 20$, $k = 1, \dots, \lfloor (n-1)/7 \rfloor$.

n\ k	1	2
8	0	
9	0	
10	0	
11	0	
12	0	
13	0	
14	0	
15	0	1
16	0	0
17	0	8
18	0	0
19	0	38
20	0	0

TABLE 2.2.2.10

F(n,k,-,8) = number of twills on n harnesses with maximum float length k, which occurs precisely eight times per period, for $n = 8, \dots, 20$, $k = 1, \dots, \lfloor n/8 \rfloor$.

n\ k	1	2
8	1	
9	0	
10	0	
11	0	
12	0	
13	0	
14	0	
15	0	
16	0	1
17	0	0
18	0	5
19	0	0
20	0	29

TABLE 2.2.2.11

$F(n,k,-,9)$ = number of twills on n harnesses with maximum float length k, which occurs precisely nine times per period, for $n = 10,\ldots,20,\ k = 1,\ldots,\lfloor(n-1)/9\rfloor$.

n\ k	1	2
10	0	
11	0	
12	0	
13	0	
14	0	
15	0	
16	0	
17	0	
18	0	
19	0	1
20	0	0

TABLE 2.2.2.12

F(n,k,-,10) = number of twills on n harnesses with maximum float length k, which occurs precisely ten times per period, for $n = 10, \ldots, 20$, $k = 1, \ldots, \lfloor n/10 \rfloor$.

n\ k	1	2
10	1	
11	0	
12	0	
13	0	
14	0	
15	0	
16	0	
17	0	
18	0	
19	0	
20	0	1

2.2.3 BALANCED TWILLS WITH BOUNDED FLOAT LENGTH

Let $F_b(n,k)$ denote the number of equivalence classes of binary sequences of length n, with maximum float length k, which are balanced. Let $F_b(n,k,m)$ be the number of equivalence classes of balanced binary sequences of length n, with maximum float length k and exactly m breaks. Then

$$F_b(n,k) = \Sigma\, F_b(n,k,m),$$

where the summations are over all m such that

$$\lceil n/k \rceil + \delta \leq m \leq n - k - \varepsilon,$$

$\delta, \varepsilon \in \{0,1\}$, $\delta \equiv \lceil n/k \rceil \pmod 2$, $\varepsilon \equiv n - k \pmod 2$. Note that we assume that $k < n$, for if $k = n$, then $m = 0$, no corresponding balanced twill exists, and the condition becomes $2 \leq m \leq 0$. Hence $F_b(n,n)$ is not defined.

Let $S_b(n,k,m)$ denote the set of balanced binary sequences of length n, with maximum float length k and m breaks. For $s = \{s_1,\dots,s_n\} \in S_b(n,k,m)$, the convention is made that $s_1 \neq s_n$ (which is always possible since $k < n$), and s is associated with the sequence of positive integers

$$r(s) = \{r_1, \ldots, r_m\},$$

where

$$s_1 = \ldots = s_{r(1)} \neq s_{r(1)+1} = \ldots = s_{r(1)+r(2)} \neq s_{r(1)+r(2)+1} = \ldots .$$

Thus, for example, the balanced sequence

$$s = \{00100111\} \in S_b(8,3,4)$$

has associated sequence

$$r(s) = (2,1,2,3).$$

Let $R_b(n,k,m)$ be the set of positive integer sequences of length m, where

$$r = \{r_1, \ldots, r_m\} \in R_b(n,k,m)$$

satisfies $\quad 1 \leq r_i \leq k$, for $i = 1,2, \ldots, m$,

$r_i = k$ for at least one value of i,

$$\sum_{i=1}^{m} r_i = n,$$

and

$$\sum_{\text{even } i} r_i = \sum_{\text{odd } i} r_i = n/2.$$

It is noted that $r \in R_b(n,k,m)$ is a pair of compositions of $n/2$ into $m/2$ parts, where no part of either composition exceeds k and at least one part of at least one composition equals k. Further, each $s \in S_b(n,k,m)$ corresponds to $r(s) \in R_b(n,k,m)$, and conversely each $r \in R_b(n,k,m)$ corresponds to exactly two sequences $s, s' \in S_b(n,k,m)$, one of which is the complement of the other. Thus these two sequences, s and s', are equivalent in $S_b(n,k,m)$.

The equivalence relation on sequences in $S_b(n,k,m)$, induced by the action of $D_{2n} \times S_2$, corresponds to an equivalence relation on $R_b(n,k,m)$, where two sequences in $R_b(n,k,m)$ are equivalent if and only if one can be transformed into the other by a cyclic shift, u, by a reversal, v, or by some finite sequence of these operations. If $r = \{r_1, r_2, \ldots, r_{m-1}, r_m\}$, then

$$ru = \{r_m, r_1, r_2, \ldots, r_{m-1}\}$$

and

$$rv = \{r_m, r_{m-1}, \ldots, r_2, r_1\}.$$

Hence the equivalence relation on $R_b(n,k,m)$ is induced by the dihedral group of order 2m defined by

$$G = \{u,v\}.$$

Thus $F_b(n,k,m)$, the number of equivalence classes of balanced binary sequences in $S_b(n,k,m)$ under the action of $D_{2n} \times S_2$, equals the number of equivalence classes of positive integer sequences in $R_b(n,k,m)$ under the action of G.

A formula has been derived [40] to enumerate the values of $F_b(n,k,m)$, corresponding to the equivalence classes of balanced twills on n harnesses with maximum float length k and precisely m breaks. Using a sieve algorithm on the set of balanced twills found in [44], the values of $F_b(n,k,-,x)$, being the numbers of balanced twills on n harnesses with maximum float length k, where the maximum float length occurs x times per period, were computed for $x = 1, \ldots, 10$. The computed results were checked against the derived formula, using the identity

$F_b(n,k) = \Sigma\, F_b(n,k,-,x),\ x = 1,2,\ldots,n/2$

$= \Sigma\, F_b(n,k,m)$, where the summation is over all m such that $\lceil n/k \rceil + \delta \leq m \leq n - k - \epsilon$,

$\delta, \epsilon \in [0,1]$, $\delta \equiv \lceil n/k \rceil \pmod 2$, $\in\ \equiv n-k \pmod 2$.

These values are listed in the following tables.

TABLE 2.2.3.1

$F_b(n,k,-,1) : 1 \le k \le (n/2) - 1$

n\ k	1	2	3	4	5	6	7	8	9
2	0								
4	0								
6	0	0							
8	0	0	1						
10	0	0	2	1					
12	0	0	5	5	2				
14	0	0	12	14	8	2			
16	0	0	31	51	30	12	3		
18	0	0	78	164	111	46	16	3	
20	0	0	201	562	413	196	71	21	4

TABLE 2.2.3.2

$F_b(n,k,-,2) : 1 \le k \le n/2$

n\ k	1	2	3	4	5	6	7	8	9	10
2	1									
4	0	1								
6	0	1	1							
8	0	2	1	1						
10	0	2	3	1	1					
12	0	3	7	3	1	1				
14	0	3	17	8	3	1	1			
16	0	4	43	25	8	3	1	1		
18	0	4	112	77	26	8	3	1	1	
20	0	5	298	259	87	26	8	3	1	1

TABLE 2.2.3.3

$F_b(n,k,-,3) : 1 \le k \le \lfloor (n-1)/3 \rfloor$

$F_b(n,k,-,3) = 0$ for $n < 14$

n\ k	1	2	3	4	5	6
14	0	0	2	0		
16	0	0	10	0	0	
18	0	0	38	3	0	
20	0	0	138	24	0	0

TABLE 2.2.3.4

$F_b(n,k,-,4) : 1 \le k \le \lfloor n/4 \rfloor$

n\ k	1	2	3	4	5
2	0				
4	1				
6	0				
8	0	1			
10	0	2			
12	0	5	1		
14	0	9	2		
16	0	16	7	1	
18	0	24	21	2	
20	0	36	76	7	1

TABLE 2.2.3.5

$F_b(n,k,-,6) : 1 \leq k \leq \lfloor n/6 \rfloor$

n\ k	1	2	3
6	1		
8	0		
10	0		
12	0	1	
14	0	2	
16	0	9	
18	0	22	1
20	0	56	2

TABLE 2.2.3.6

$F_b(n,k,-,8) : 1 \leq k \leq \lfloor n/8 \rfloor$

n\ k	1	2
8	1	
10	0	
12	0	
14	0	
16	0	1
18	0	3
20	0	16

TABLE 2.2.3.7

$F_b(n,k,-,x)$ for $x = 5,7,9,10$

$F_b(20,3,-,5) = 4$;

$F_b(10,1,-,10) = F_b(20,2,-,10) = 1$.

All other values of $F_b(n,k,-,x)$ are zero in the ranges given, namely

$1 \leq n \leq 20$;

$1 \leq x \leq 10$

$1 \leq k \leq \lfloor(n - \varepsilon_x)/x\rfloor$ where $\varepsilon_5 = 1$,

$\varepsilon_7 = 1$,

$\varepsilon_9 = 1$,

$\varepsilon_{10} = 0$.

TABLE 2.2.3.8

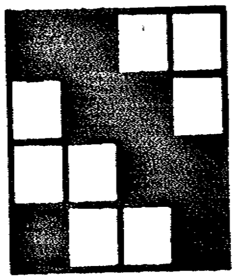
$F_b(n,k)$ for $n = 2,4,\ldots, 20$; $k = 1,2,\ldots,n/2$

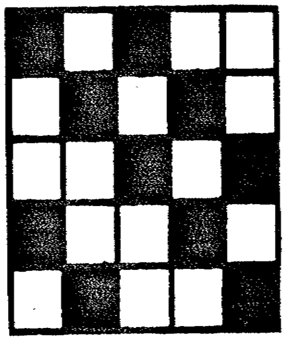
n\ k	1	2	3	4	5	6	7	8	9	10
2	1									
4	1	1								
6	1	1	1							
8	1	3	2	1						
10	1	4	5	2	1					
12	1	9	13	8	3	1				
14	1	14	33	22	11	3	1			
16	1	30	91	77	38	15	4	1		
18	1	53	250	246	137	54	19	4	1	
20	1	114	719	852	501	222	79	24	5	1

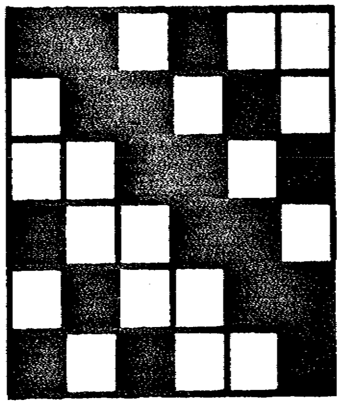
TABLE 2.2.3.9
$F_b(n)$ for $n = 2,4,\ldots,20$

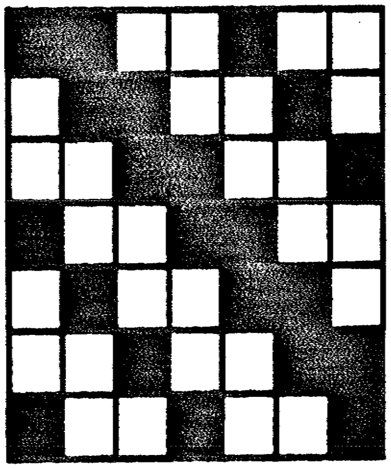
n	**$F_b(n)$**
2	1
4	2
6	3
8	7
10	13
12	35
14	85
16	257
18	765
20	2518

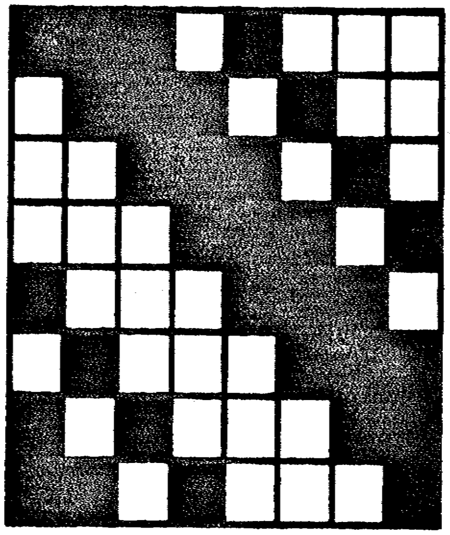
2.2.4 ILLUSTRATIONS

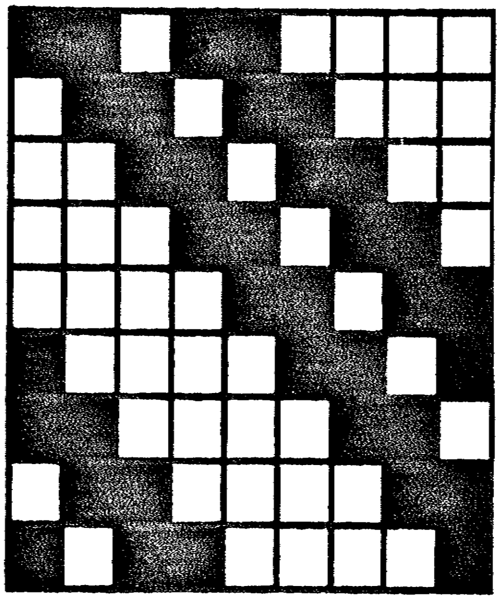

n = 4

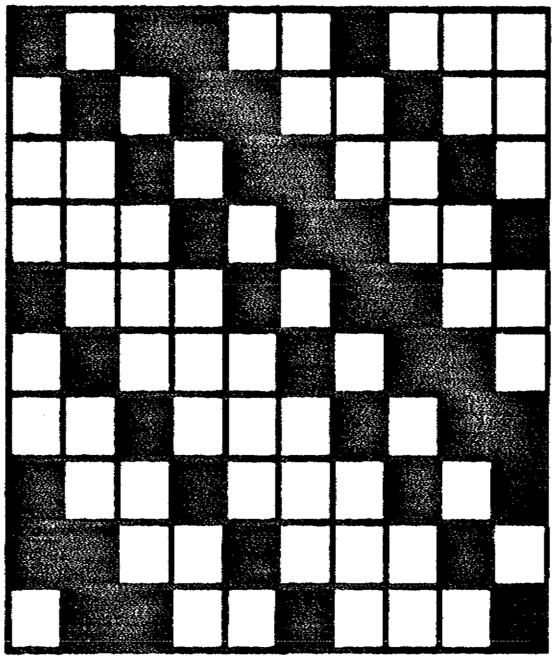

n = 5

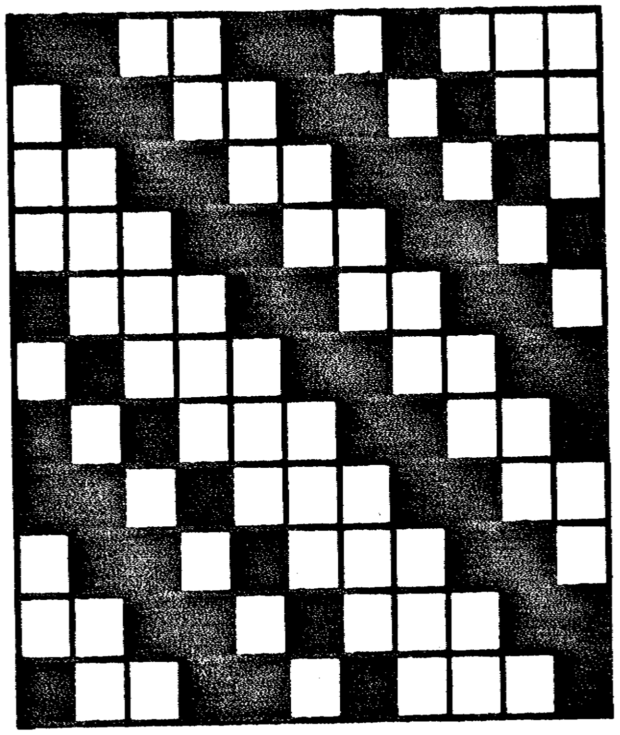

n = 6


n = 7

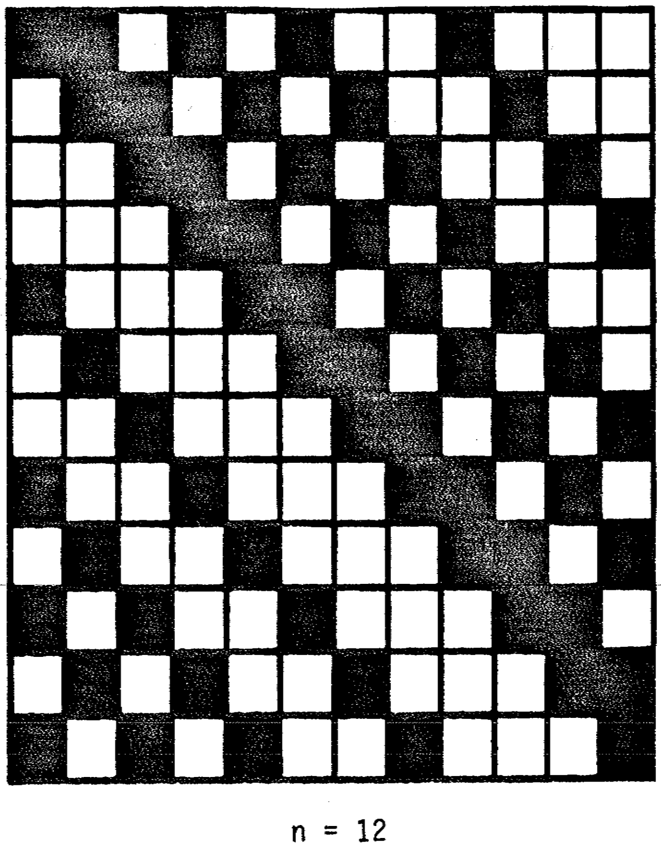

n = 8


n = 9

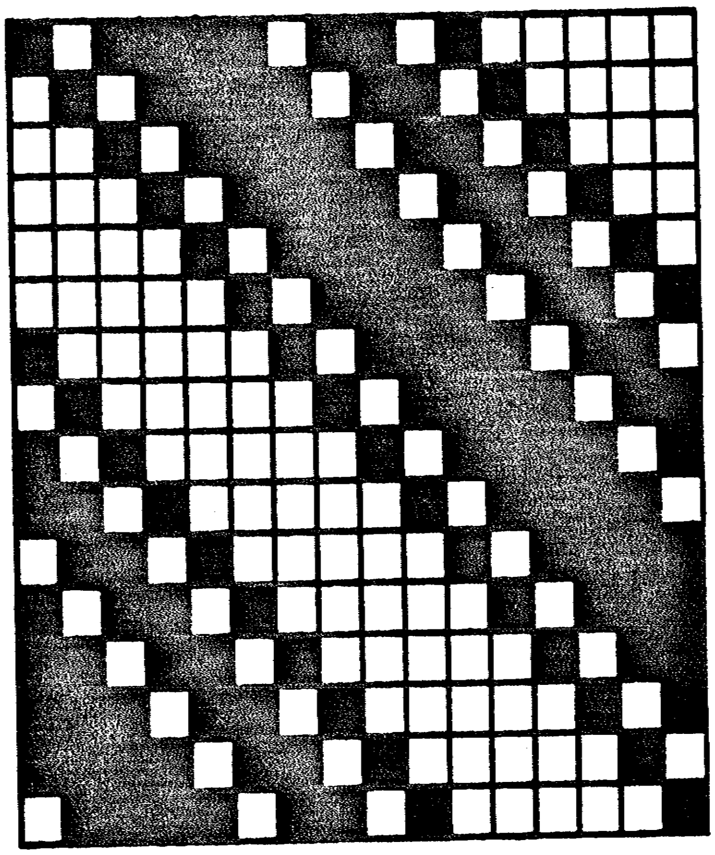

n = 10



n = 11



n = 12



n = 16

2.3 OTHER SIMPLE ISONEMAL STRUCTURES

2.3.1 INTRODUCTION AND DEFINITIONS

Having established the characteristics of isonemal planar interlacement arrays, as typified by the twills, three additional classes of simple isonemal structures [45] have been identified [24].

Definition 2.3.1.1. A twillin on n harnesses is an isonemal planar interlacement array in which each row (column) of the array is an interlacement sequence of period n and is obtained from the previous row (column) by displacement through s positions. Such a structure is called an (n,s) twillin in [24] and a classic example is the (5,2) twillin, known as a 5 end satin.

It is an obvious consequence of the definitions of interlacement sequence and design equivalence that the first row of a twillin can only be one of the first rows of a twill and that

$$\gcd(n,s) = 1.$$

Similarly, it is apparent that the (n,1) twillins are twills.

Definition 2.3.1.2. A simple alternate twill on n harnesses is an isonemal design obtained from the planar interlacement array of a simple twill with all even-numbered rows replaced by their complements.

Definition 2.3.1.3. An (n,s,~) alternate twillin is an isonemal design obtained from a planar interlacement array in which each row (column) of the array is an interlacement sequence of period n and is obtained from the previous row (column) by displacement through s positions and complementation (~).

2.3.2 ENUMERATION ALGORITHM

A result following from definitions (2.3.1.1) and (2.3.1.2) which is stated in [24] without an explicit proof is:-

THEOREM 2.3.2.1.
For an (n,s) twillin or (n,s,~) alternate twillin

$$s^2 \equiv \pm 1 \pmod n.$$

Proof. Consider the tile describing the twillin to be that member of the equivalence class such that the element in the first row and first column is a 1 (call it "black"). Hereafter note that the reference to squares in the tile will be by "row-column" names rather than the usual abscissa-ordinate forms. The black square in position (1,1) generates black squares in positions (i, 1+(i-1)s) with all numbers being taken modulo n and for convenience n being used rather than 0 for the nth row or column. The design is isonemal; so using this fact on the columns to "translate" all of the black squares to column n it is apparent that the square in position (i, 1+(i-1)s) needs to be moved through n-1-(i-1)s columns to position (x,n) and each move will either increase or decrease the row by s (say εs, where $\varepsilon = \pm 1$).

Hence $x = (i + \varepsilon s (n - 1 - (i - 1) s)$, and, since we are working modulo n, we may take $x = i - \varepsilon(i - 1)s^2 - \varepsilon s = \varepsilon s^2 - \varepsilon s + i (1 - \varepsilon s^2)$. Since $\varepsilon s^2 - \varepsilon s$ is a constant, it can be ignored as far as the column pattern is concerned. Let

$$\varepsilon s^2 - \varepsilon s = t,$$
$$1 - \varepsilon s^2 = u.$$

Then column n has black squares in row positions t, t+u, t+2u, t+3u, . . ., and these are generated by the original square in position (1,1). This of course implies that any square does not occur in isolation but as a member of a sequence of squares in every uth position. If the gcd (u,n) is designated by the symbol a, then this sequence of squares has n/a elements. Clearly, this number should be less than n or we would have no mix of 0's and 1's in the array.

Now the remark in the last paragraph shows that the twillin splits up into subsquares of size n/a, and the whole tile is patched together from these sub-twillins. If we agree to exclude this possibility of having a "decomposable" twillin, then we must have $n/a = 1$, $a = n$; this implies that $u = 0$ and

$$1 - \varepsilon s^2 = 0. \quad \square$$

<u>THEOREM 2.3.2.2.</u>

Non-trivial simple alternate twills and $(n,s,\tilde{})$ twillins on n harnesses only exist when

$$n \equiv 0 \pmod 4.$$

<u>Proof.</u> In an isonemal planar interlacement array, the row and column sequences have the same weight. But every second element of the row sequence has been complemented when forming the column sequence. Hence, among these $n/2$ elements, half must be zeros and half ones. Thus $4|n$. $\square$

A consequence of definitions [2.3.1.1], [2.3.1.2], [2.3.1.3] and the preceding two results is that the color alternate twills, twillins, and color alternate twillins can be determined by the following algorithm. The twillins and alternate twillins counted and listed are those for which $s \neq$

± 1 (i.e. not twills or alternate twills) and $s^2 \equiv 1 \pmod n$.

1. Given r,s,n, and the set of twills on n harnesses.
2. Take the kth twill Q whose rows are $\{q_i\}$, $i = 1,2,\ldots,n$.
3. Generate the planar interlacement array C with rows $\{c_j\}$, $j = 1,2,\ldots,n$ where

$$c_j = (\sim)^{r\,(j-1)} q_{i+1}$$

where i is computed from

$$s\,(j-1) \equiv i \pmod n \quad j = 1,2,\ldots,n.$$

4. Suppose that C and C^* (the * denoting the transposed array) are design equivalent.
   If $r = 1$, $s = 1$, then C is an alternate twill.
   If $r = 0$, $s > 1$, $s^2 \equiv \pm 1 \pmod n$, then C is a twillin.
   If $r = 1$, $s > 1$, $s^2 \equiv \pm 1 \pmod n$, then C is an alternate twillin.
   If $r = 0$, $s = 1$, then C is a twill.

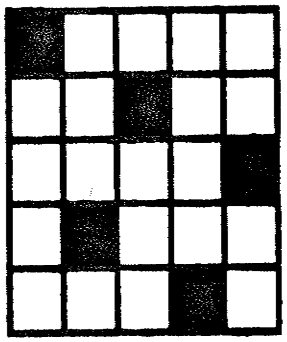
Using the twills determined in [44] and the above algorithm, the members of the classes of alternate twills, twillins and alternate twillins were found for $n \leq 20$. The number of members in each class was also computed for $n \leq 20$ and cross-checked by counting arguments. These results appear in Table (2.3.2.3).

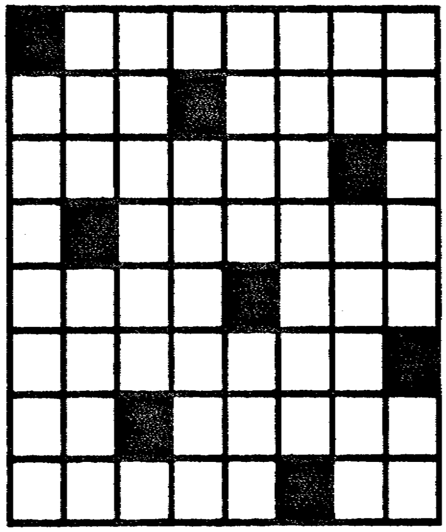
TABLE 2.3.2.3
Number of Alternate Twills, Twillins and Alternate Twillins

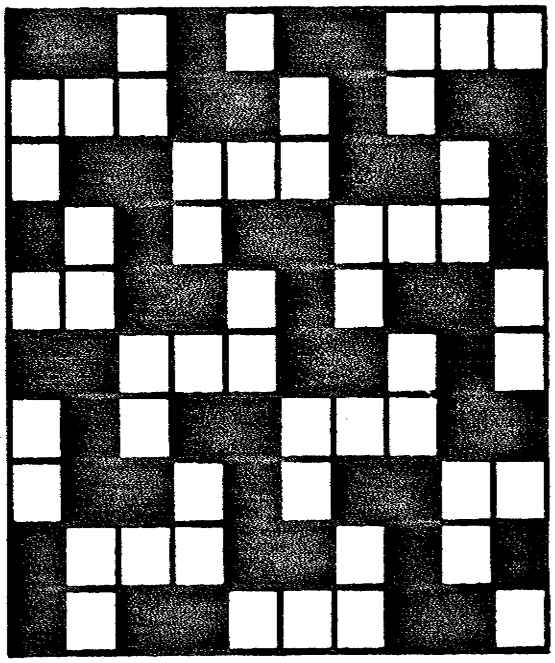
n	No. of Alternate Twills	No. of Twillins	No. of Alternate Twillins
4	2	0	0
5	0	1	0
6	0	0	0
7	0	0	0
8	6	9	10
9	0	0	0
10	0	7	0
11	0	0	0
12	20	53	32
13	0	7	0
14	0	0	0
15	0	95	0
16	74	197	198
17	0	15	0
18	0	0	0
19	0	0	0
20	284	1263	956

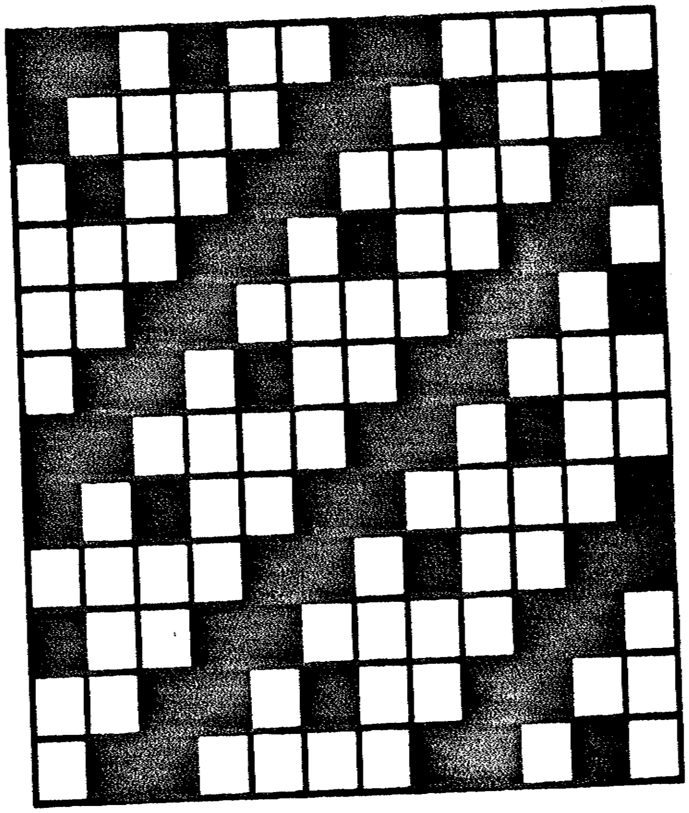
2.3.3 ILLUSTRATIONS

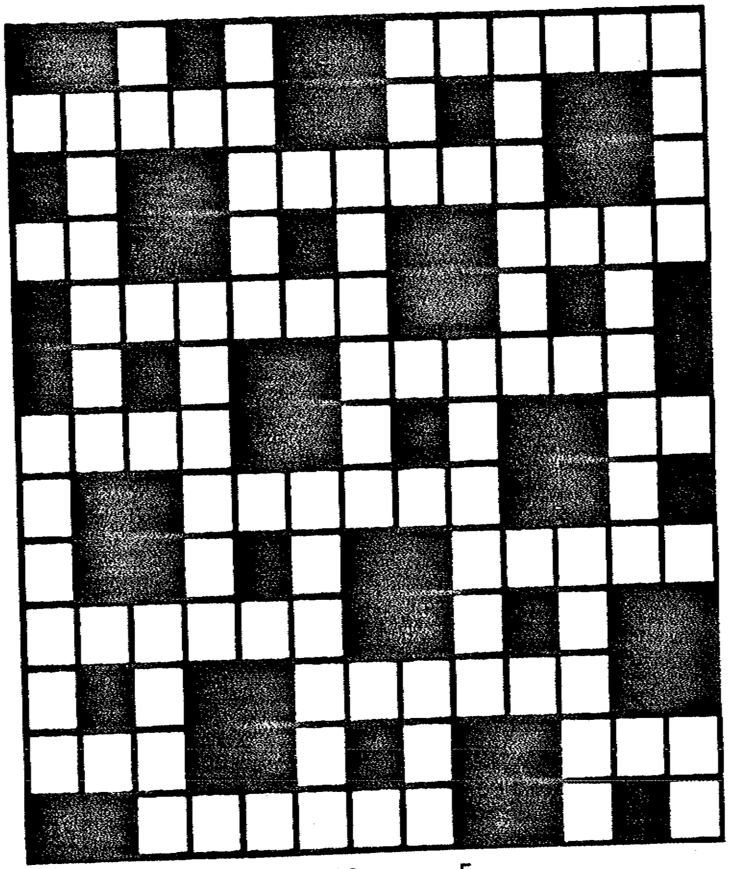
TWILLINS

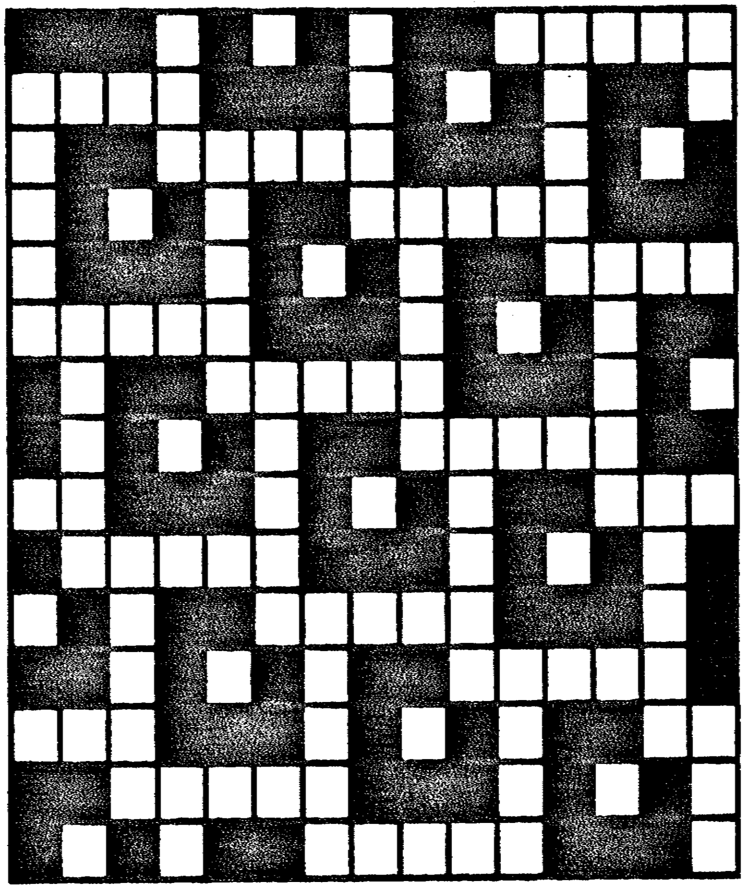

n = 5
s = 2

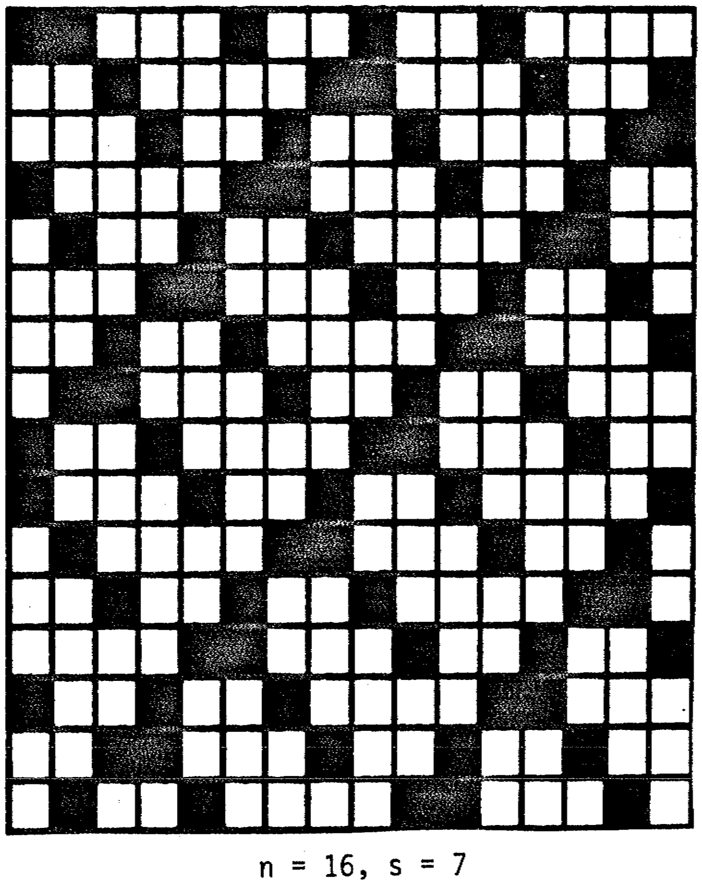

n = 8
s = 3


n = 10, s = 3

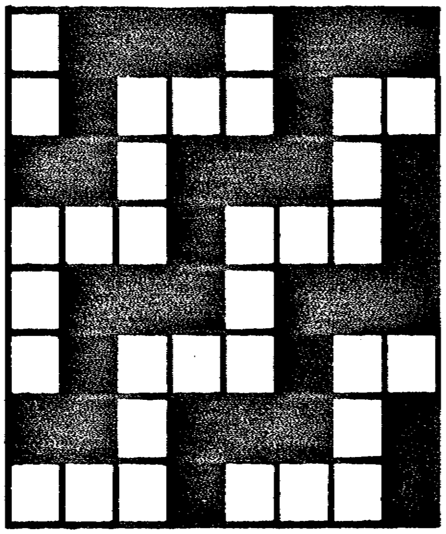

n = 12, s = 5

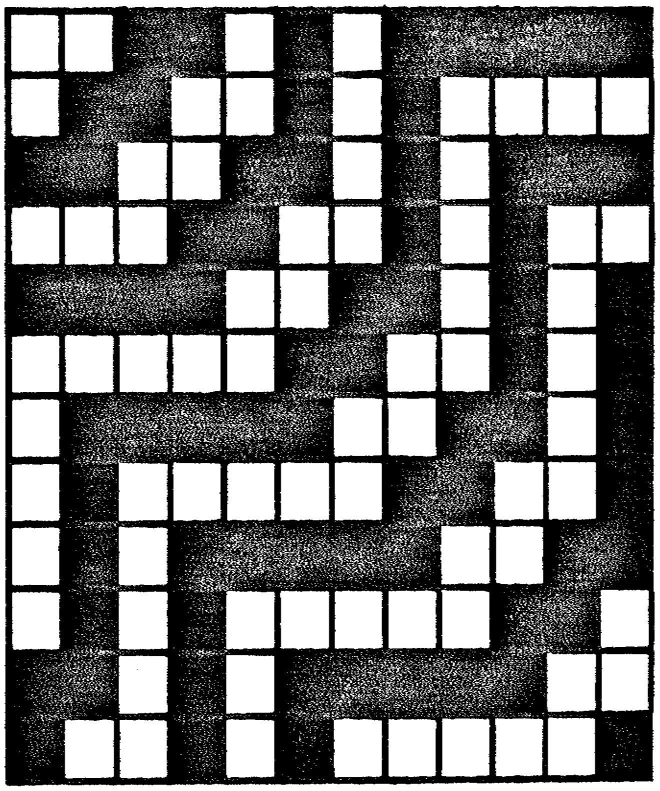

n = 13, s = 5

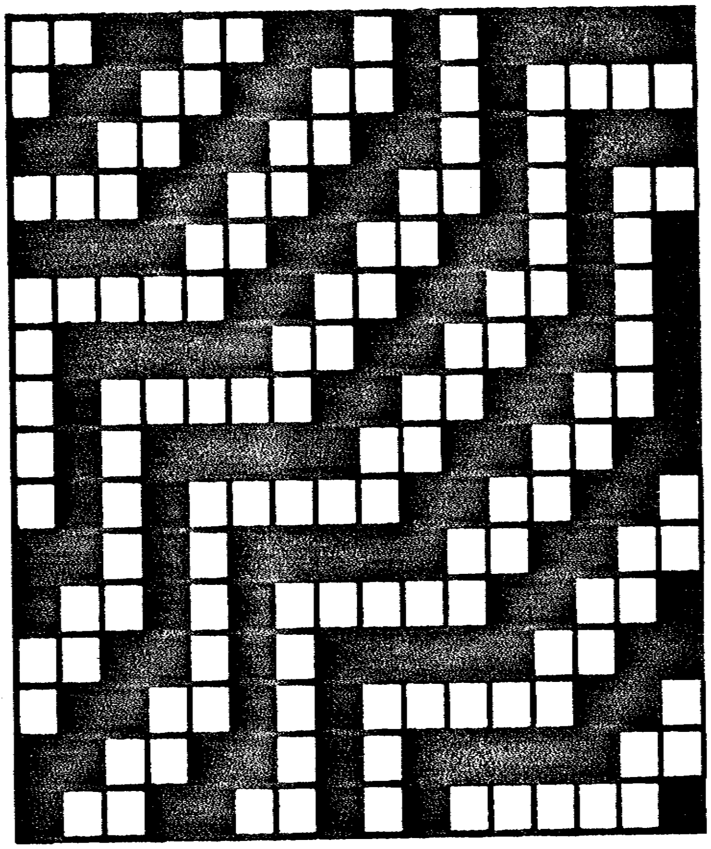

n = 15, s = 4


n = 16, s = 7

COLOR ALTERNATE TWILLS

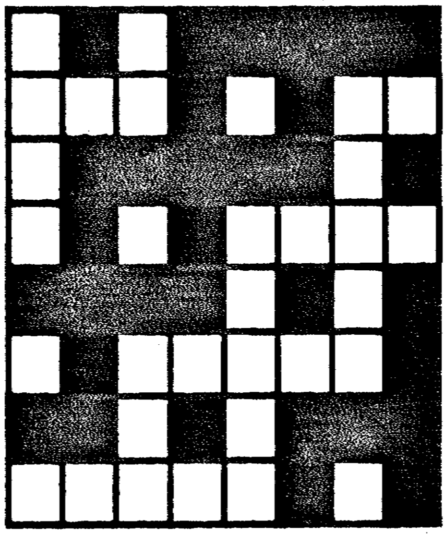

n = 8

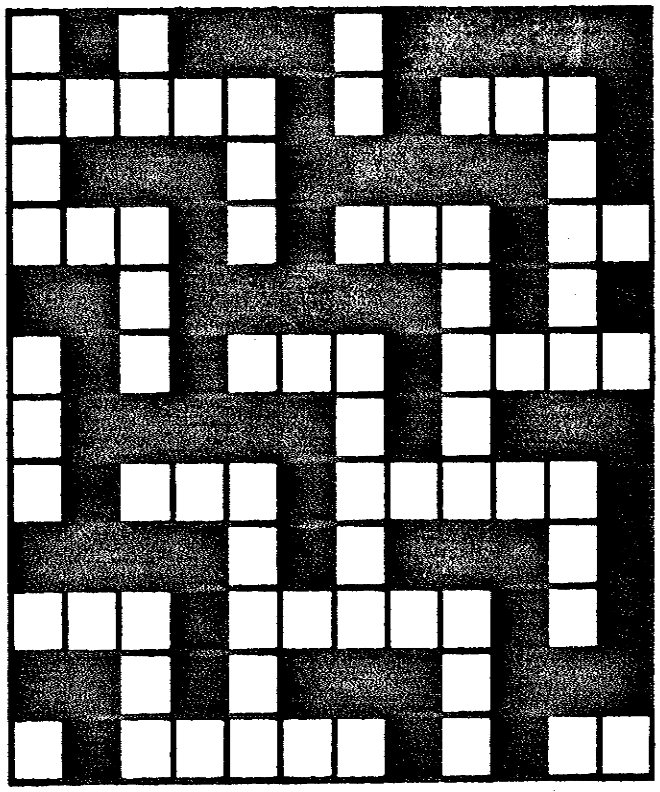

n = 12

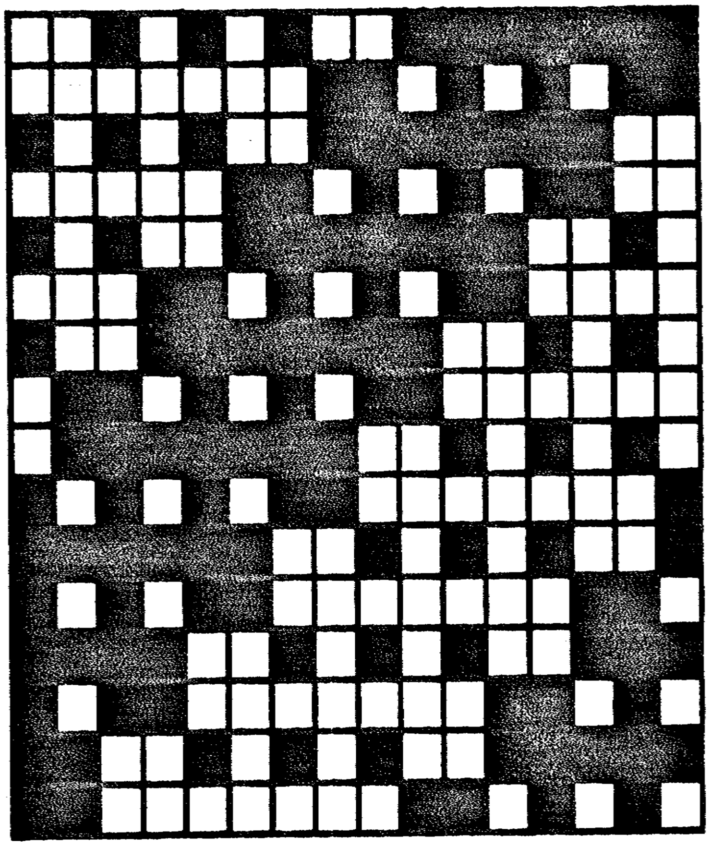


n = 16

COLOR ALTERNATE TWILLINS


n = 8, s = 3


n = 12, s = 5



n = 16, s = 7

2.4 COMPOUND TWILLINS

2.4.1 INTRODUCTION AND DEFINITIONS

Discussion to this point has been limited to simple isonemal structures having one rule for deriving each row (and hence each column) sequence from the previous one. Hoskins and Thomas [45] introduced the term compound twillin to describe the only other possible type of isonemal binary interlacement array, which they defined as:-

Definition 2.4.1.1 A compound twillin is an isonemal binary interlacement array in which one rule is used to derive even row or column sequences from the immediately previous row or column, and another rule is used to derive odd row or column sequences from the immediately previous row or column. These two rules involve reflection about a fixed point, with this point being different for each of the rules. Complementation can also be present in one or both of the rules.

The point of reflection of the sequence $S = \{a_0, a_1, \ldots, a_{n-1}\}$ can be either at a symbol or midway between two symbols and, for convenience, the point midway between a_s and a_{s+1} is considered to be the $(s + \frac{1}{2})^{th}$ position of the sequence. (Subscripts are added modulo n as usual.) Similarly, if s is an odd multiple of $\frac{1}{2}$, then the element $a_{s+\frac{1}{2}}$ is the next sequence element after s. The reflection that fixes position s is denoted by Φ_S: thus if the sequence is

$$S = (0,1,2,3,4,5),$$

then $\qquad \Phi_{1/2}\, S = (1,0,5,4,3,2)$

and $\qquad \Phi_0 S = (0,5,4,3,2,1).$

In all the cases of interest here, n is even. So Φ_S fixes two positions: s and s + n/2.

In this notation, the transition rules of [45] can be represented as

$$P = (\tilde{\ })^x\, \Phi_S \text{ and } Q = (\tilde{\ })^y\, \Phi_t,$$

where $(\tilde{\ })$ denotes complementation and $x,y \in \{0,1\}$.

In developing subsequent results we make use of the following additional definitions.

Definition 2.4.1.2. A dyadically derived row isonemal array is a row isonemal array which has been generated using two different rules applied alternately, where the first rule involves reflection about a fixed point s and the second rule involves reflection about a different fixed point t.

If the sequence is binary, complementation may or may not be present in one or both of the rules; in general, permutation of the symbols may or may not be present.

Definition 2.4.1.3. An array $A_{n \times n}$ is block isonemal if, when it is partitioned into s × s blocks ($n \equiv 0 \pmod{s}$), the resulting blocks themselves form an isonemal array.

Clearly this property applies to all isonemal sequences and not just to the compound twillins. It is convenient to choose an example from the twills for illustration.

Example 2.4.1.4. Let $A_{6 \times 6}$ be a twill with first row [1,1,0,1,0,0]; then

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

If A is partitioned into 3 × 3 blocks then

$$\mathbf{A} = \begin{bmatrix} \alpha & \beta \\ \beta & \alpha \end{bmatrix}$$

where

$$\alpha = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix};$$

and

$$\beta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Since there are only two blocks, we can set $\alpha = 0$ and $\beta = 1$. Then

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

which is clearly an isonemal structure.

2.4.2 DETERMINING THE COMPOUND TWILLINS

In the development of an algorithm for determining the compound twillins, we make use of a number of results. Firstl, a computationally simplified method of generating a compound twillin arises from the the recognition that it can be constructed by specifying a (1,1) mapping from the first row onto each subsequent row.

THEOREM 2.4.2.1. Consider a compound twillin with transition rules Φ_s from row 2k to row 2k + 1, and Φ_t from row 2k + 1 to row 2k + 2. If the top row of the matrix is given by

$$a_{0,j} = a_j,\ j = 0, 1, \ldots, n-1, \text{ then}$$

$$a_{2k,j} = a_{0,j-2k(s-t)} \text{ and } a_{2k+1,j} = a_{0,2s+2k(s-t)-j},$$

for $k = 0, 1, \ldots, (n-2)/2.$

Proof. If a_s occupies position s' in row 2k, then applying Φ_s to row 2k takes the element in position (s' + w) to position (s' − w), that is

$$a_{2k+1,s'-w} = a_{2k,s'+w}.$$

Similarly, if a_t occupies position t' in row 2k + 1, then

$$a_{2k+2,t'-w} = a_{2k+1,t'+w}.$$

Thus

$$\begin{aligned} a_{2k+2,j} &= a_{2k+2,t'-(t'-j)} \\ &= a_{2k+1,t'+(t'-j)} \\ &= a_{2k+1,2t'-j} \\ &= a_{2k+1,s'-(s'-(2t'-j))} \\ &= a_{2k,s'+(s'-(2t'-j))} \\ &= a_{2k,2(s'-t')+j} \, . \end{aligned}$$

But, in row 2k, the element a_t occupies the position $s'+(t-s)$; so $t' = s' - (t-s)$ and $s' - t' = t - s$. Hence

$$a_{2k+2,j} = a_{2k,j-2(s-t)} \, ;$$

in other words, the even-numbered rows of the matrix form a cyclic array, where each even-numbered row is just the preceding even-numbered row shifted $2(s-t)$ places to the right. The odd-numbered rows form a similar cyclic array.

This justifies the first statement, namely

$$a_{2k,j} = a_{0,j-2k(s-t)} \, .$$

Now the position s' of a_s in row 2k is just $s + 2k(s-t) = s'$; so

$$\begin{aligned} a_{2k+1,j} &= a_{2k+1,s'-(s'-j)} \\ &= a_{2k,s'+(s'-j)} \\ &= a_{2k,2s+4k(s-t)-j} \\ &= a_{0,2s+2k(s-t)-j}\,, \end{aligned}$$

giving the second statement of the theorem.□

From [45] it follows that the reflection points s and t may always be chosen to satisfy certain restrictions, summarized as follows:

1. $0 \le |s-t| < n/4$, where n is the number of symbols in the sequence;
2. $\gcd(n/2, (s-t)) = 1$;
3. $(s-t)^2 \equiv \pm 1 \pmod{n/2}$.

Further, ϕ_s and $\phi_{s+n/2}$ are equivalent reflections, for $\phi_{s+n/2}$ takes position $s + w = (s+n/2) + (w+n/2)$ to position $(s+n/2) - (w+n/2) = s - w$, just as ϕ_s does. Hence we also require:

4. $0 \le s,t < n/2$.

In addition, several observations can be made concerning the periods of compound twillins. Any compound twillin must obviously have period $n \equiv 0 \pmod 2$. Any compound twillin with single complementation must have period $n \equiv 0 \pmod 4$, since it has a four-row repeat. In fact, any compound twillin with double complementation must also have period $n \equiv 0 \pmod 4$, a less obvious fact proved in the following.

<u>THEOREM 2.4.2.2.</u> A compound twillin with double complementation has period $n \equiv 0 \pmod 4$.

<u>Proof.</u> In a compound twillin, the row and column sequences are equal (up to cyclic shifts and reflection). But every second element of the row sequence has been complemented when forming the column sequence. Hence, among these $n/2$ elements, half must be zeros and half ones. Thus $4 \mid n$. □

<u>COROLLARY 2.4.2.3.</u> If a compound twillin with complementation has y 1's in its initial sequence, then $n/4 \leq y \leq 3n/4$.

<u>Proof.</u> For both single and double complementation, $n/2$ of the elements will be complemented. Of these $n/2$ elements, precisely half (or $n/4$) must be 1's. The remaining $n/2$ elements can be all 1's, all 0's or some combination of the two. Therefore y, the number of 1's, must be in the range

$n/4 \leq y \leq 3n/4$, as required. $\square$

At this point, it is convenient to divide discussion of the compound twillins into two sections, namely

- compound twillins with reflection points s and t between elements, that is, where s, t are odd multiples of ½ ;
- compound twillins with reflection points s and t at elements, that is, where s, t are even multiples of ½ .

2.4.2.1 COMPOUND TWILLINS WITH REFLECTION BETWEEN ELEMENTS

THEOREM 2.4.2.1.1. Consider a dyadically derived row isonemal array A, with no complementation or permutation, and with reflection points s and t, where $|s-t| = 1$ and s,t are odd multiples of $\frac{1}{2}$. Then there are two elements, $a_{0,h}$ and $a_{0,h+n/2}$, in row zero of A, such that there exist arrays B and C, design equivalent to A, with $a_{0,h}$ and $a_{0,h+n/2}$, respectively, lying on their principal diagonals.

Proof. Suppose that $t = s + 1$. Let $h = s + \frac{1}{2}$. Then $a_{0,h}$ moves $2(s-t) = -2$ places to the right, or in other words two places to the left, over each two rows. Also

$$a_{1,h-1} = a_{1,s-\frac{1}{2}} = a_{0,2s-(s-\frac{1}{2})} = a_{0,s+\frac{1}{2}} = a_{0,h},$$

so that $a_{0,h}$ moves one place to the left, from row to row.

Define the $n \times n$ arrays B and C to be:

$b_{i,j} = a_{i,h-j}, \quad i,j = 0,1,\ldots,n-1;$

$c_{i,j} = a_{i,h+(n/2)-j}, \quad i,j = 0,1,\ldots,n-1.$

Since s-t = -1, the formulae of Theorem [2.4.2.1] now show that $b_{i,i} = a_{0,h}$ and that $c_{i,i} = a_{0,h+n/2}$, as required. A similar (but shorter argument deals with the case where t-s = 1, and $b_{i,j} = a_{i,h+j}$, $c_{i,j} = a_{i,h+(n/2)+j}$. □

<u>Example 2.4.2.1.2</u> Let A be an array, as in Theorem 2.4.2.1.1, with $s = 5\frac{1}{2}$, $t = \frac{1}{2}$, $n = 6$, $a_{0,j} = j$, $j = 0, \ldots, 5$.

Then

$$A = \begin{bmatrix} 0&1&2&3&4&5 \\ 5&4&3&2&1&0 \\ 2&3&4&5&0&1 \\ 3&2&1&0&5&4 \\ 4&5&0&1&2&3 \\ 1&0&5&4&3&2 \end{bmatrix}, \quad B = \begin{bmatrix} 0&5&4&3&2&1 \\ 5&0&1&2&3&4 \\ 2&1&0&5&4&3 \\ 3&4&5&0&1&2 \\ 4&3&2&1&0&5 \\ 1&2&3&4&5&0 \end{bmatrix}, \quad C = \begin{bmatrix} 3&2&1&0&5&4 \\ 2&3&4&5&0&1 \\ 5&4&3&2&1&0 \\ 0&1&2&3&4&5 \\ 1&0&5&4&3&2 \\ 4&5&0&1&2&3 \end{bmatrix}.$$

Note that, in B and C, $s = \frac{1}{2}$, $t = -\frac{1}{2}$, relative to the sequence in the initial row. Note also that, in a compound twillin, which is a binary array, only two distinct values can in fact occur.

<u>COROLLARY 2.4.2.1.3.</u> If $s = \frac{1}{2}$, $t = -\frac{1}{2}$, and the initial sequence of A is $a_{0,j} = j$, $j = 0, 1, \ldots, n-1$, then $a_{i,j} = (-1)^i (j-i)$.

<u>Proof.</u> Since $a_{2k,j} = a_{0,j-2k(s-t)}$, and s-t = 1, we have

$$a_{2k,j} = a_{0,j-2k} = j - 2k.$$

Since $\qquad a_{2k+1,j} = a_{0,2s+2k(s-t)-j}$

$$= a_{0,1+2k-j},$$

we have $\qquad a_{2k+1,j} = 2k + 1 - j.$

The corollary follows. □

THEOREM 2.4.2.1.4. Let A be a dyadically derived row isonemal array with no complementation or permutation, with reflection points s and t where $|s-t| = 1$ and s,t are odd multiples of $\frac{1}{2}$. Then A is column isonemal.

Proof. By Corollary (2.4.2.1.3), we may assume without loss of generality that $s = \frac{1}{2}$, $t = -\frac{1}{2}$, and

$$a_{i,j} = (-1)^i (j-i) \pmod n,$$

for $i,j = 0,1,\ldots,n-1$. Then

$$a_{i+1,j} = (-1)^{i+1} (j-(i+1)) = -(-1)^i ((j-1)-i) = a_{i,j-1}$$

and

$$a_{i,j+1} = (-1)^i ((j+1)-i) = -(-1)^{i-1} (j-(i-1)) = -a_{i-1,j}.$$

The sequence in the initial column is 0,1,-2,3,-4, . . . and subsequent columns are obtained by reflection in $s' = \frac{1}{2}$ and $t' = -\frac{1}{2}$ alternately. Thus A is also column isonemal. □

To discuss binary arrays, we interpret the notation slightly differently: the initial sequence consists of 0's and 1's, and the term $a_{i,j}$ gives the subscript of the term of this sequence occurring in position (i,j).

Example 2.4.2.1.5. If $a_{i,j} = (-1)^i (j-i)$ as before, and the initial sequence is $(1,0,1,0,0,0)$, then the corresponding binary array has rows

$$(1,0,1,0,0,0),\ (0,1,0,0,0,1),\ (0,0,1,0,1,0),$$
$$(0,1,0,1,0,0),\ (1,0,0,0,1,0),\ (0,0,0,1,0,1).$$

THEOREM 2.4.2.1.6. Let A be a binary dyadically derived row isonemal array, with complementation in one or both of the transition rules, with reflection points s and t where $|s-t| = 1$ and s,t are odd multiples of $\frac{1}{2}$. Then A is column isonemal.

Proof. As in Theorem [2.4.2.1.4], we may assume that $s = \frac{1}{2}$, $t = -\frac{1}{2}$, and that

$$a_{i,j} = (\sim)^h (-1)^i (j-i).$$

If we have single complementation, so that the transition rules for rows are $(\sim)\phi_{\frac{1}{2}}$ and $\phi_{-\frac{1}{2}}$, then $h \equiv \lfloor (i+1)/2 \rfloor \pmod 2$; if we have double complementation, so that the transition rules are $(\sim)\phi_{\frac{1}{2}}$ and $(\sim)\phi_{-\frac{1}{2}}$, then $h \equiv i \pmod 2$.

This gives the initial column sequences and transition rules for columns as follows: for single complementation, the sequence

$$0,\bar{1},-\bar{2},3,-4,\bar{5},-\bar{6},7,-8,\ldots$$

and transition rules $(\sim)\,\phi_{\frac{1}{2}}$ and $\phi_{-\frac{1}{2}}$; for double complementation, the sequence

$$0,\bar{1},-2,\bar{3},-4,\bar{5},-6,\bar{7},-8,\ldots$$

and transition rules $(\sim)\phi_{\frac{1}{2}}$ and $(\sim)\phi_{-\frac{1}{2}}$.

In either case, the array is column isonemal. □

<u>THEOREM 2.4.2.1.7.</u> Let A be a binary dyadically derived row isonemal array, with or without complementation, and with reflection points s and t, both odd multiples of $\frac{1}{2}$, and satisfying requirements (i) - (iv) of Section (2.4.2). Then A is column isonemal.

<u>Proof</u>. (a) If s-t is odd, then the points midway between s and t are occupied by sequence elements $a_h, a_{h+n/2}$ say, where $h = (s+t)/2$. By requirement (ii), $\gcd(n,(s-t)) = 1$; so s-t has an inverse r (modulo n).

By Theorem (2.4.2.1), a_h moves s-t spaces to the right in each successive row; so, after r rows, it has moved r (s-t) = 1 space to the right. Thus the array formed by taking

$$\begin{bmatrix} a_{0,h} & a_{0,h+1} & \cdots & a_{0,h-1} \\ a_{r,h} & a_{r,h+1} & \cdots & a_{r,h-1} \\ a_{2r,h} & a_{2r,h+1} & \cdots & a_{2r,h-1} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{(n-1)r,h} & a_{(n-1)r,h+1} & \cdots & a_{(n-1)r,h-1} \end{bmatrix}$$

is an array of the form given in Theorem (2.4.2.1.1). Hence the original array is column isonemal by Theorem (2.4.2.1.4).

(b) If s-t is even, by requirement (ii), $n \equiv 2 \pmod 4$ and gcd (n,(s-t)) = 2. Here the mid-point between s and t (position (s+t)/2 again) moves s-t spaces to the right in each successive row, and occurs alternately to the left and to the right of the point s. We may assume without loss of generality that $t = \frac{1}{2}$, $s = 2m + \frac{1}{2}$, and that there exists an integer r such that $(s-t)\,r \equiv 2 \pmod n$. Then the array formed by taking rows 0, r, 2r, . . . is again row and column isonemal, and hence the original array is column isonemal. □

<u>COROLLARY 2.4.2.1.8.</u> For the array A of Theorem [2.4.2.1.7], the terms of any column must be a permutation of the terms of the original row sequence.

<u>Proof.</u> The result follows immediately. □

<u>ENUMERATION ALGORITHM.</u>

An algorithm has been developed, based on the preceding results, which can be applied to the twill sequences of [44] to determine all of the possible compound twillins for a given sequence length. This algorithm is given by the following sequence of steps:

1. Start with n and the twill sequences on n shafts.
2. Choose appropriate values for s and t.
3. Taking each sequence in turn, generate a dyadically derived row isonemal array, C, with rows C_k, $k = 1,2,\dots,n$, where the original sequence S is permuted to give C_k using the following computational formula:

   For k even: $C_k = S_h$,

   where $h = 1 + ((ks - (t(k-2)+n+1) - r) \bmod n)$, $r = 0,1,\dots,n-1$.

For k odd: $C_k = S_h$,

where $h = 1 + (((k-1)(t-s)+r) \bmod n)$, $r = 0,1,\ldots,n-1$.

This array will be column isonemal.

4. Determine whether the row and column sequences are design equivalent. If they are design equivalent, then determine whether the rules for generating the columns are the same as the rules for generating the rows. If they are the same, then the array is isonemal.
5. Ensure that each of the compound twillins which has been found is unique.
6. Introduce complementation into the first rule and repeat steps 1 to 5.
7. Introduce complementation into both rules and repeat steps 1 to 5.

Using an implementation of the above algorithm, the numbers of compound twillins with reflection points between elements for $n \leq 16$ shafts were determined and are listed in Table [2.4.2.1.9].

TABLE 2.4.2.1.9

NUMBER OF COMPOUND TWILLINS WITH

REFLECTION POINTS BETWEEN ELEMENTS

n	s	t	NO COMPLEMENTATION	SINGLE COMPLEMENTATION	DOUBLE COMPLEMENTATION
6	½	1½	9	0	0
8	½	1½	29	10	16
10	½	1½	67	0	0
10	½	2½	19	0	0
12	½	1½	216	1	95
14	½	1½	519	0	0
16	½	1½	1645	86	640
16	½	3½	509	306	16

2.4.2.2 COMPOUND TWILLINS WITH REFLECTION AT AN ELEMENT

In the case of compound twillins with reflection points which occur at elements of the array, the situation is more complicated than the one previously discussed, as the following example shows.

EXAMPLE 2.4.2.2.1. If the reflection points s and t are integers, with $|s-t| = 1$, then row isonemality of the array need not imply column isonemality. For instance, if $n = 8$, $s = 0$, $t = 1$, then the initial sequence 00010011 for the rows leads to a row isonemal array which is not even column isonemal, as illustrated.

$$\begin{array}{cccccccc}
0&0&0&1&0&0&1&1\\
0&1&1&0&0&1&0&0\\
0&1&0&0&1&1&0&0\\
1&0&0&1&0&0&0&1\\
0&0&1&1&0&0&0&1\\
0&1&0&0&0&1&1&0\\
1&1&0&0&0&1&0&0\\
0&0&0&1&1&0&0&1
\end{array}$$

The following results enable the algorithm of [2.4.2.1] to be simplified and used for the determination of the compound twillins with reflection points occurring at elements of the array.

Theorem 2.4.2.2.2. Any compound twillin which is partitioned into four $n/2 \times n/2$ blocks is block isonemal.

<u>Proof.</u> A consequence of property (iv) in (2.4.1) is that Φ_S and $\Phi_{S+n/2}$ are equivalent reflections; further, Φ_t and $\Phi_{t+n/2}$ are equivalent reflections. Thus, any element $a_{i+n/2}$, $i \in \{0,1, \ldots ,(n/2)-1\}$, in the first row of a dyadically derived row isonemal array will be mapped to the same position as element a_i (mod n/2). The array A can therefore be considered to be two contiguous arrays of size $n \times n/2$. But, if the array is isonemal, then the elements in a column of the array must exhibit the same properties as did the elements of the rows. Thus, the array A can be partitioned into four blocks of size $n/2 \times n/2$.

The array A considered in a block sense is now

$$A = \begin{bmatrix} \alpha & \beta \\ \delta & \gamma \end{bmatrix}$$

If we consider two repeats of A in both the x and the y direction,

$$A^* = \begin{bmatrix} \alpha & \beta & \alpha & \beta \\ \delta & \gamma & \delta & \gamma \\ \alpha & \beta & \alpha & \beta \\ \delta & \gamma & \delta & \gamma \end{bmatrix}$$

but A is an isonemal structure. Therefore $\alpha = \gamma$ and $\beta = \delta$. Assigning

$\alpha = 0$ and $\beta = 1$,

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

a block isonemal array, and the result is proved. □

Theorem 2.4.2.2.3. Consider a dyadically derived row isonemal array A with reflection points s and t, where s and t are elements for which $|s-t| = 1$, with no complementation and with $n \not\equiv 0 \pmod 4$. In order for this array to be isonemal, it must be symmetric.

Proof. Let $n = 2m$. Since $n \not\equiv 0 \pmod 4$, m is odd. From Theorem [2.4.2.1], the elements of row m are given by:

$$a_{m,j} = a_{0,2s+(m-1)+j} \pmod n.$$

But, for this array to be isonemal, it must be block isonemal; see Figure [2.4.2.2.4.]

$$A = \begin{bmatrix} a_{0,0} & \cdots & a_{0,m-1} & a_{0,m} & \cdots & a_{0,n-1} \\ \cdot & & \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot & & \cdot \\ a_{m-1,0} & \cdots & a_{m-1,m-1} & a_{m,m} & \cdots & a_{m-1,n-1} \\ a_{m,0} & \cdots & a_{m,m-1} & a_{m,m} & \cdots & a_{m,n-1} \\ \cdot & & \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot & & \cdot \\ a_{n-1,0} & \cdots & a_{n-1,m-1} & a_{n,m} & \cdots & a_{n-1,n-1} \end{bmatrix}$$

FIGURE 2.4.2.2.4

Thus $a_{0,0} = a_{m,m}$, $a_{m-1,m-1} = a_{n-1,n-1}$, $a_{0,m} = a_{m,0}$, etc.

We can assume, with no loss of generality, that $s = 0$ in the formulae of Theorem (2.4.2.1), since the first row can always be cycled so that s will lie in the zero position.

Apply Theorem (2.4.2.1) to the matrix $(a_{i,j})$, $i,j = 0,1,\ldots,n-1$.

For i,j even, $a_{i,j} = a_{0,i+j}$, and $a_{j,i} = a_{0,i+j}$, so $a_{i,j} = a_{j,i}$.

For i,j odd, $a_{i,j} = a_{0,1-(i+j)}$, $a_{j,i} = a_{0,1-(i+j)}$, so $a_{i,j} = a_{j,i}$.

For i even, j odd, $a_{i,j} = a_{0,i+j}$, and $a_{j,i} = a_{0,1-(i+j)}$,

but since m is odd, $a_{i,j} = a_{i+m,j+m} = a_{0,1-(i+j)}$,

and hence $a_{i,j} = a_{j,i}$.

For i odd, j even, $a_{i,j} = a_{0,1-(i+j)}$, and $a_{j,i} = a_{0,i+j}$.

But $a_{i,j} = a_{i+m,j+m} = a_{0,i+j}$,

and hence $a_{i,j} = a_{j,i}$,

and the result is proved. □

Corollary 2.4.2.2.5. Any compound twillin with reflection at an element, with $|s-t| = 1$, and with order not a multiple of 4, is also a twill.

Proof. This follows as a direct consequence of Theorem (2.4.2.2.3) and Definition (2.2.1.1.) □

Example 2.4.2.2.6. Let A be a dyadically derived row isonemal array whose top row is given by $a_{0,j} = j$, $j = 0,1,\ldots,5$, with $s= 0$, $t = 1$.

$$A = \left[\begin{array}{ccc|ccc} 0&1&2&3&4&5 \\ 0&5&4&3&2&1 \\ 2&3&4&5&0&1 \\ \hline 4&3&2&1&0&5 \\ 4&5&0&1&2&3 \\ 2&1&0&5&4&3 \end{array}\right]$$

For this array to be isonemal, the element in cells containing 0 must be the same as the element in a "1" cell; "2" and "5" must correspond, as well as "3" and "4". A could thus be written

$$A = \left[\begin{array}{ccc|ccc} 0&0&2&3&3&2 \\ 0&2&3&3&2&0 \\ 2&3&3&2&0&0 \\ \hline 3&3&2&0&0&2 \\ 3&2&0&0&2&3 \\ 2&0&0&2&3&3 \end{array}\right]$$

and the symmetry is apparent.

There are further restrictions that can be placed on the sequences which can possibly occur in compound twillins with reflection occurring at elements of the array, which we now examine.

Consider first the case with no complementation, and suppose that the reflections that generate the compound twillins occur at the elements a_0 and a_t of the initial sequence $a_0, a_1, a_2, \ldots, a_{2k-1}$ (A) where $n = 2k$ and t is chosen to satisfy the restrictions (i) - (iv) in Section [2.4.2], so that

$$0 < t < k/2, \ \gcd(k,t) = 1, \ t^2 \equiv \pm 1 \pmod k.$$

Then column zero will be occupied by the sequence

$$a_0, a_0, a_{2t}, a_{-2t}, a_{4t}, a_{-4t}, \ldots, a_{4t}, a_{-2t}, a_{2t} \qquad \text{(E)}$$

and column one by the sequence

$$a_1, a_{-1}, a_{2t+1}, a_{-2t-1}, a_{4t+1}, a_{-4t-1}, \ldots, a_{4t-1}, a_{-2t+1}, a_{2t-1} \qquad \text{(O).}$$

The restrictions on t ensure that the subscripts 0, 2t, 4t, . . . , -4t, -2t run through all the even integers modulo n, and thus that (E) consists of all the elements from even-numbered positions of (A), twice each. Every even-numbered column is occupied by cyclic shifts of (E) and every odd-numbered column by cyclic shifts of (O). Since the array is isonemal, these sequences are equivalent; hence (A) is a palindrome of type (1) and even weight, say 2h. Further, h of the ones in (A) must occur in even-numbered and h in odd-numbered positions, so that (E) and (O) contain equal numbers of ones.

Consider the particular case where $t = 1$, which satisfies restrictions (i) - (iv) of Section [2.4.2] for any value of n. If we choose i and j so that $i + j \equiv 1 \pmod n$ the sequence given by

$$a_i = a_j = 1$$
$$a_w = 0, w \neq i, w \neq j$$

is a palindrome which, on reflection at 0 and 1 alternately, gives a twill. If $k = 2m$ so that $n = 4m$, then

$$a_i = a_j = 1, a_w = 0, w \neq i, w \neq j \qquad (*)$$

and

$$a_{i+k} = a_{j+k} = 1, a_w = 0, w \neq i+k, w \neq j+k \qquad (*)$$

are distinct but equivalent sequences, so we have m such twills; if $k = 2m + 1$, so that $n = 4m + 2$, the sequences (*) are distinct except in the single case $i = m + 1$, $j = 3m + 2$; so we have altogether $m + 1$ such twills.

Further if we choose pairs $(i_1, j_1), \ldots, (i_p, j_p)$ where

$i_w + j_w \equiv 1 \pmod{n}$, $w = 1,2,\ldots,p$, and let

$$a_{i[w]} = a_{j[w]} = 1,\ w = 1,2,\ldots,p$$
$$a_h = 0,\ h \notin \{i_1, j_1, \ldots, i_p, j_p\},$$

we have a palindrome of weight $2p$, which again gives a twill when reflected at 0 and 1 alternately.

If $k = 2m + 1$, we choose p so that $1 \leq p \leq m$, and the weight of the palindrome is at most $2m$. This can be done in

$$\tfrac{1}{2}\left(\left[{2m+1 \atop p}\right] + \left[{m \atop [p/2]}\right]\right) \text{ inequivalent ways,}$$

using Burnside's lemma as described in [37], where the group is generated by the reflection that maps i to $2m + 1 - i \pmod{n}$. Hence the number of inequivalent twills which occur as compound twillins of this type is

$$\tfrac{1}{2}\sum_{p=1}^{m}\left(\left[{2m+1 \atop p}\right] + \left[{m \atop [p/2]}\right]\right).$$

A similar, but more lengthy argument gives the number of inequivalent twills which occur as compound twillins when $k = 2m$. The detailed development is given in [42].

The only case which has been considered theoretically is the one with no complementation and with the 1's in the first row sequence appearing in positions i,j for i and j such that $i + j \equiv 1$. In other words, only compound twillins which are isomorphic with twills have been considered. Thus, the formulae developed give lower bounds, as shown in Table (2.4.2.2.7).

We now consider the possible sequences for the compound twillins with complementation in one or both of the generating rules. We may assume that $n = 4w$, and that reflections occur at points a_0 and a_t, where $0 < t < w$, $\gcd(2w,t) = 1$, $t^2 \equiv \pm 1 \pmod{2w}$, and our initial sequence is

$$a_0, a_1, a_2, \ldots, a_{4w-1}.$$

If complementation occurs when we reflect at a_0, but not when we reflect at a_t, then column zero has the sequence

$$a_0, \bar{a}_0, \bar{a}_{2t}, a_{-2t}, a_{4t}, \bar{a}_{-4t}, \bar{a}_{6t}, a_{-8t} \ldots, \bar{a}_{4t}, \bar{a}_{-2t}, a_{2t},$$

and column one the sequence

$$a_1, \bar{a}_{-1}, \bar{a}_{2t+1}, a_{-2t-1}, a_{4t+1}, \bar{a}_{-4t-1}, \ldots, \bar{a}_{4t-1}, \bar{a}_{-2t+1}, a_{-2t-1}.$$

If complementation occurs with both reflections, then column zero has the sequence

$$a_0, \bar{a}_0, a_{2t}, \bar{a}_{-2t}, a_{4t}, \bar{a}_{-4t}, \ldots, \bar{a}_{4t}, a_{-2t}, \bar{a}_{-2t},$$

and column one the sequence

$$a_1, \bar{a}_{-1}, a_{2t+1}, \bar{a}_{-2t-1}, a_{4t+1}, \bar{a}_{-4t+1}, \ldots, a_{4t-1}, \bar{a}_{-2t+1}, a_{2t-1}.$$

Hence for a compound twillin with either single or double complementation, the initial sequence must in fact be of the form

$$a_0, a_1, a_2, \ldots, a_{2w-1}, \bar{a}_{2w-1}, \ldots, \bar{a}_2, \bar{a}_1, \bar{a}_0,$$

that is a copalindrome of weight 2w, where w ones occur in even and w in odd positions in the sequence.

The algorithm of Section [2.4.2.2] can be applied to the twill sequences of [44] to determine all of the possible compound twillins for a given sequence length, with reflection points at an element. Based on the preceding results, the following filters can be applied to the twill sequences prior to invoking the sieve algorithm, to obtain a rapid reduction of those sequences which can potentially be used to create these compound twillins.

Filtering Algorithm:

When no complementation is to take place:

1. discard all sequences which have a weight for the odd numbered elements not equal to the weight for the even numbered elements;
2. of the remaining sequences, discard those which are not palindromes.

When complementation is to occur in either one or both rules:

1. discard all sequences whose weight is not precisely equal to one half the sequence length;
2. of the remaining sequences, discard those which are not copalindromes.

Using an implementation of the above algorithm, the numbers of compound twillins with reflection at an element have been determined for $n \leq 16$ and are listed in Table (2.4.2.2.7).

TABLE 2.4.2.2.7

NUMBER OF COMPOUND TWILLINS

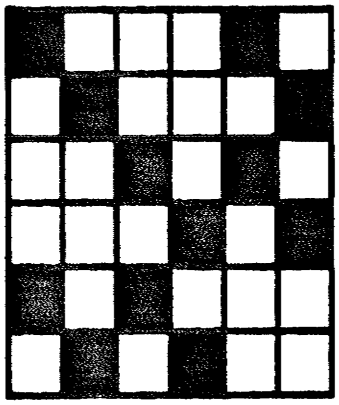
WITH REFLECTION POINTS AT ELEMENTS

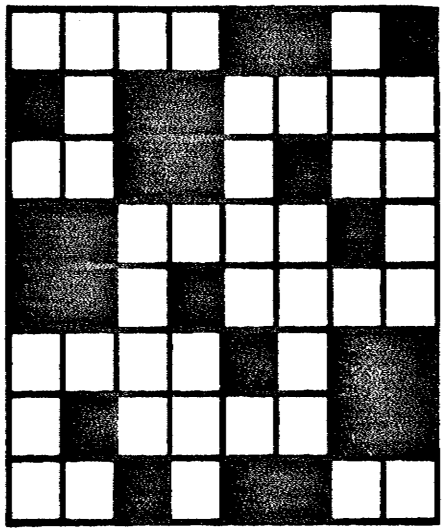
			No Complementation			
n	s	t	Actual Number	Theoretical Lower Bound	Single Complementation	Double Complementation
6	0	1	2	2	0	0
8	0	1	5	5	2	6
10	0	1	9	9	0	0
10	0	2	1	-	0	0
12	0	1	23	18	0	24
14	0	1	35	35	0	0
16	0	1	69	69	2	70
16	0	3	5	-	10	6

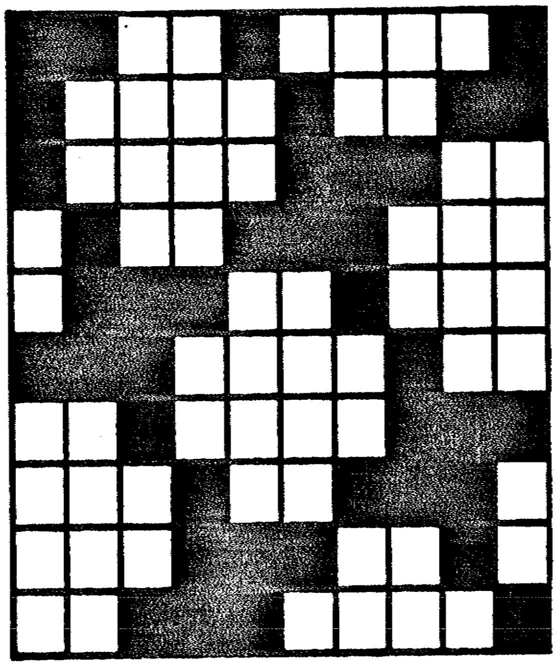
The programs which were developed to enumerate and identify all of the isonemal structures were written in APL and run on an Amdahl 5850. The CPU time required to obtain these tables was of course exponential with n and, even for the small cases considered, involved in excess of fifty hours.

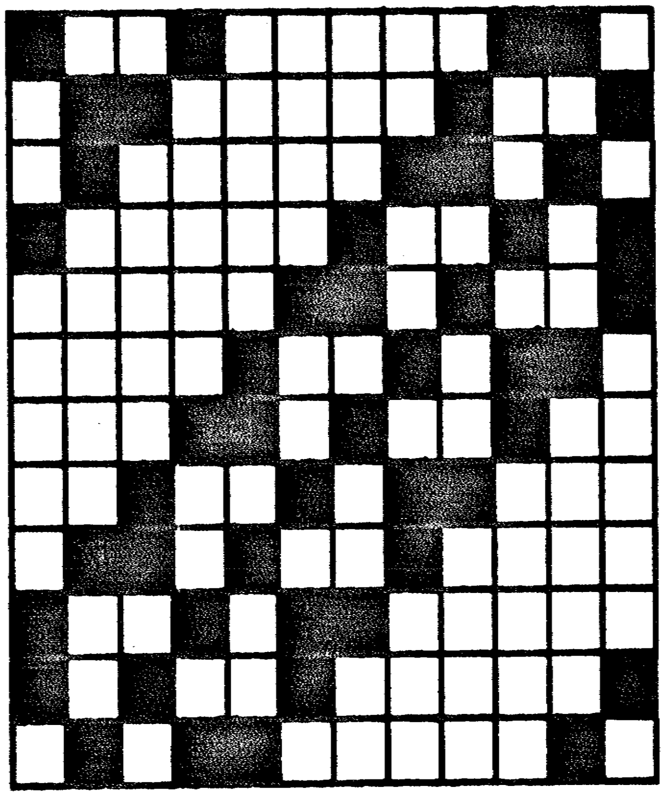
2.4.3 ILLUSTRATIONS

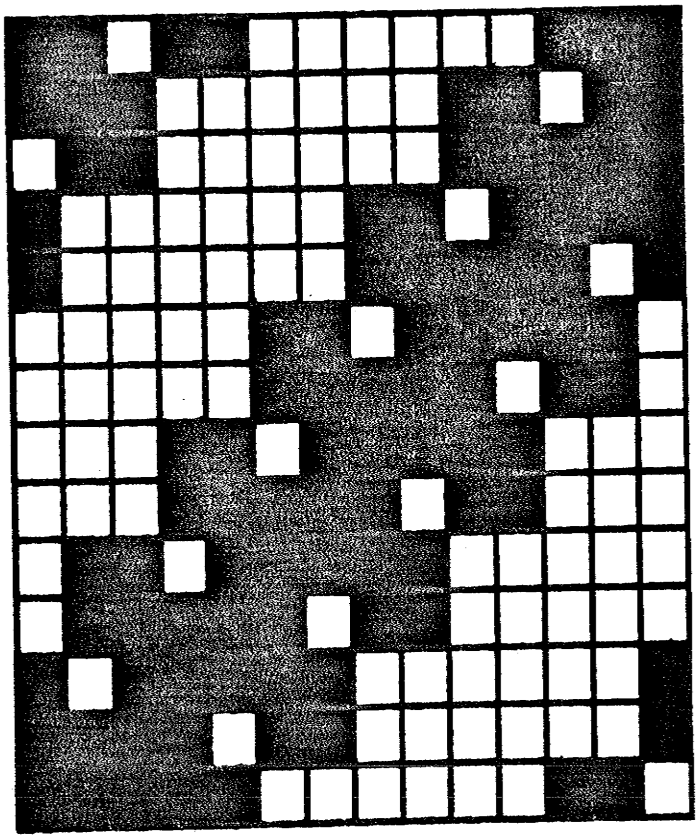
COMPOUND TWILLINS
NO COMPLEMENTATION

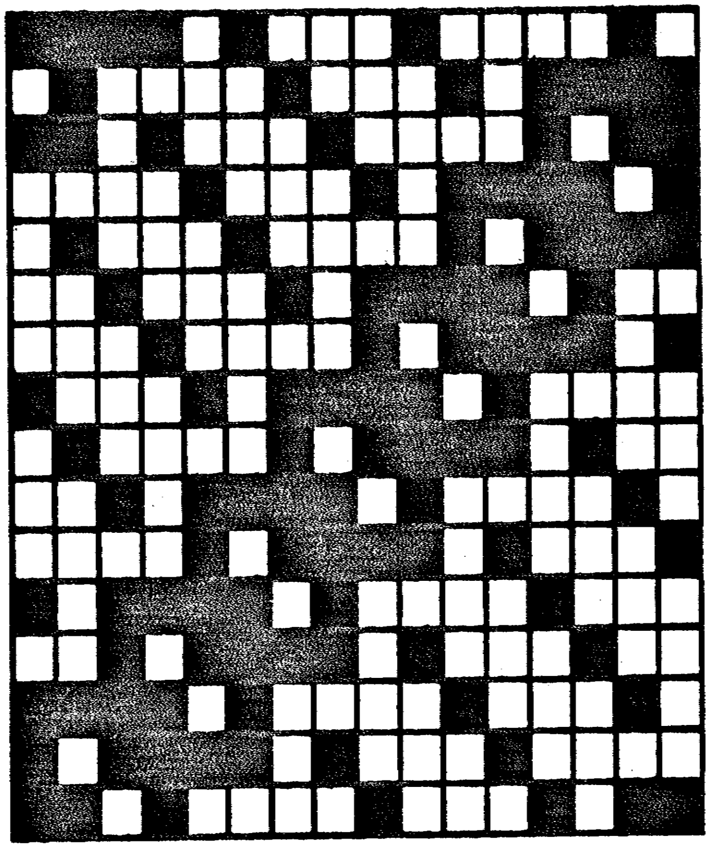

n = 6, s = ½, t = 1½

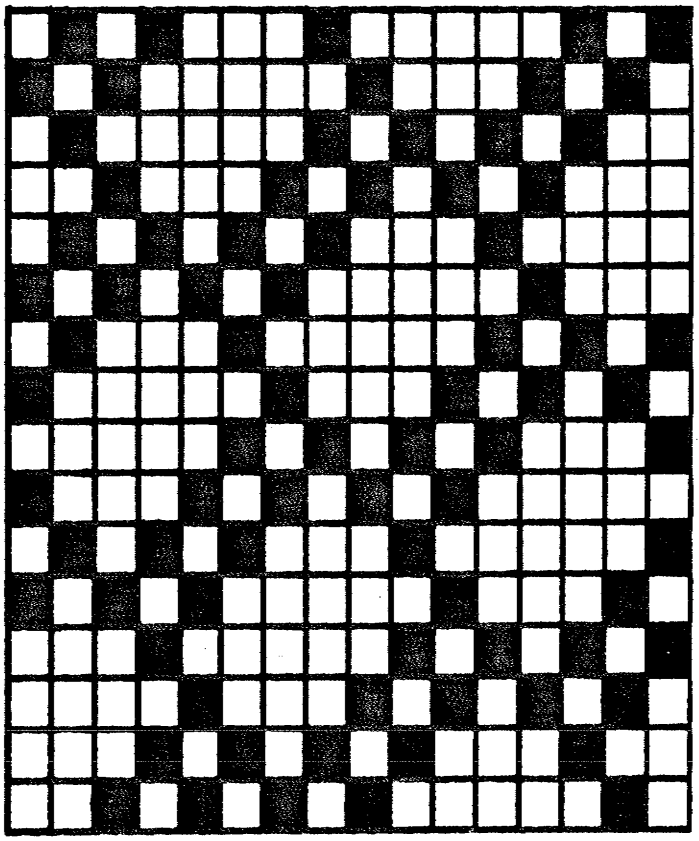

n = 8
s = ½
t = 1½


n = 10, s = ½, t = 2½


n = 12, s = ½, t = 1½

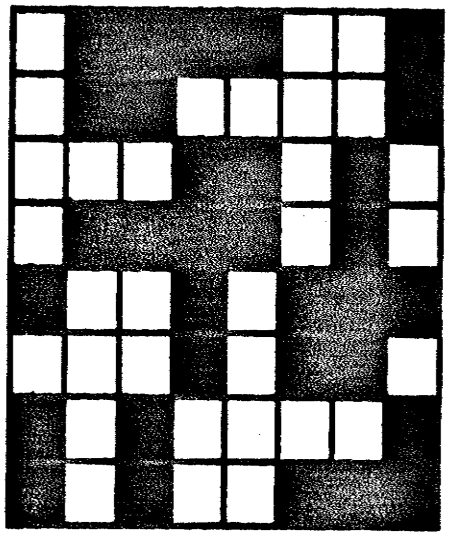

n = 14, s = ½, t = 1½


n = 16, s = ½, t = 1½

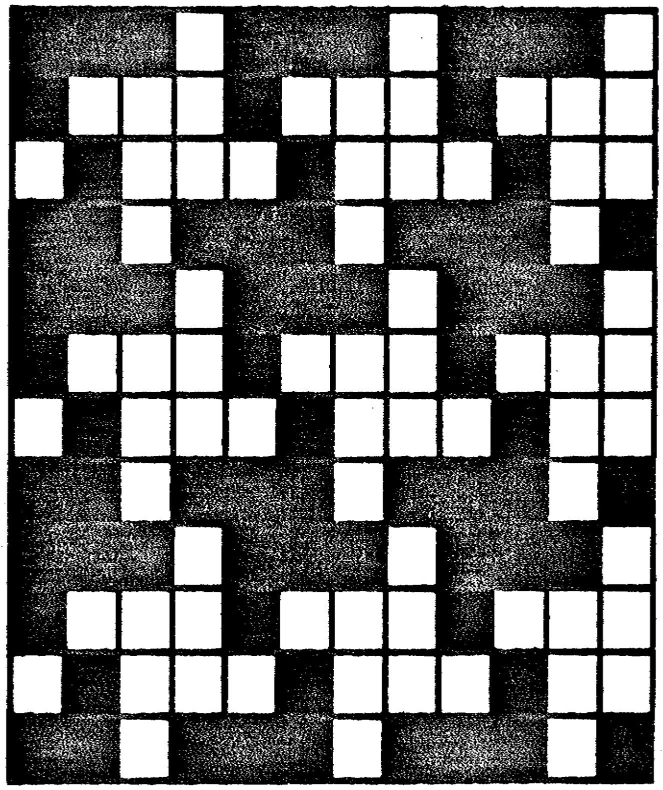

n = 16, s = ½, t = 3½

COMPOUND TWILLINS

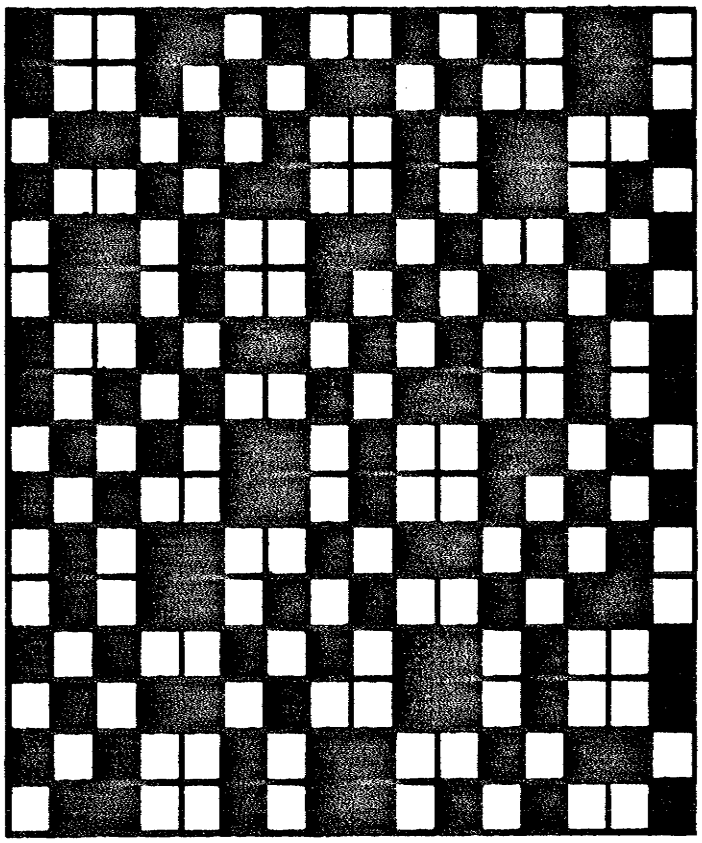
SINGLE COMPLEMENTATION



n = 8, s = ½, t = 1½



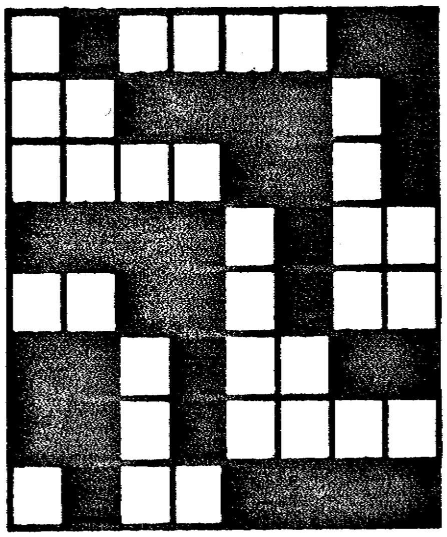
n = 12, s = ½, t = 1½



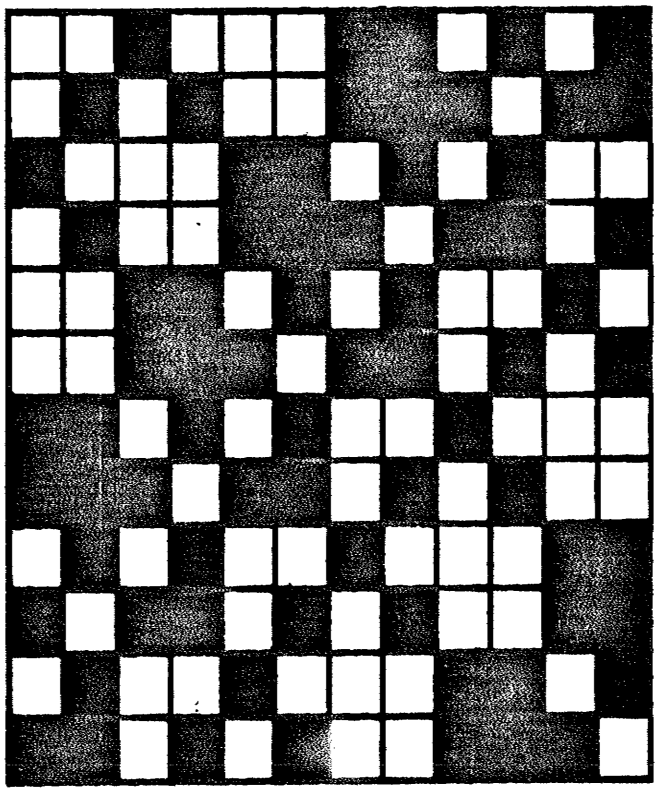
n = 16, s = ½, t = 3½

COMPOUND TWILLINS

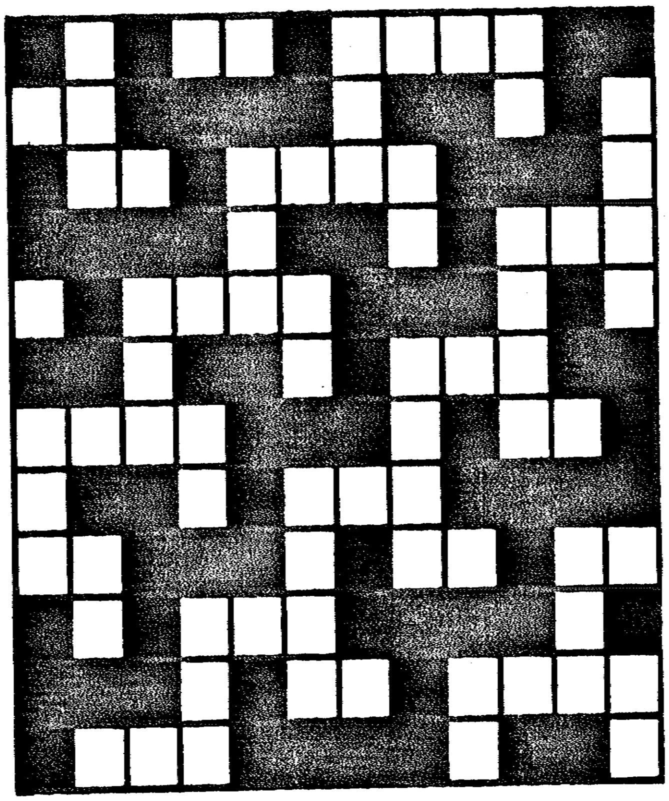
DOUBLE COMPLEMENTATION



n = 8, s = ½, t = 1½



n = 12, s = ½, t = 1½


n = 12, s = ½, t = 1½

CHAPTER 3

ALGORITHMS FOR FABRIC ANALYSIS

CONTENTS

3.1 <u>INTRODUCTION</u>

An interlacement array is actually the product of three matrices which describe the physical set up and operation of a loom, namely the threading, tie-up and shed sequence matrices. On all but the most simple of looms, each lengthwise yarn, or warp end, is threaded through a heddle on a particular shaft, with the result that every warp end threaded on the same shaft makes exactly the same interlacements with the crosswise yarn, or weft pick, as every other end on that shaft. The threading matrix has the same number of columns as warp ends and the same number of rows as shafts on which the ends can be threaded.

The weaving process involves raising one or more of these shafts at a time. All of the warp ends threaded on this shaft, or these shafts, will then lie on top of the weft pick inserted at this time, while the remaining warp ends will lie underneath the weft pick. In the case of the most common loom configuration, the single harness system, each warp end is threaded on only one shaft. This places the restriction on the binary threading matrix that there be precisely one 1 in every column, with the remaining elements being 0.

The tie-up matrix indicates in which combinations the shafts will be raised. The number of rows is equal to the number of shafts and the number of columns is equal to the number of different combinations of shafts to be used. Each set of shafts to be raised in a particular combination is physically tied to a treadle which, when depressed, causes the shafts to rise. Thus, the number of columns in the tie-up matrix also represents the number of treadles.

The shed sequence matrix indicates which treadle will be used for each weft pick. The number of columns is equal to the number of treadles and there are as many rows in this matrix as there are picks in the corresponding interlacement array. Normally, only one treadle is depressed for any weft pick. The shed sequence matrix therefore contains precisely one 1 per row, with the remaining elements being 0.

Because of the inherent association between the woven design and the set up and operation of the loom, it is essential to be able to determine the interlacement array which will result from a specified threading, tie-up and shed sequence matrix and conversely, to factor a known interlacement array into its three matrix components. This chapter will discuss algorithms for both processes.

3.2 DETERMINING AN INTERLACEMENT ARRAY

The traditional method of computing an interlacement array (Figure {3.2.1}) requires that one start with the first row of the shed sequence matrix (C) and find the column which has a non-zero entry. This corresponds to the treadle which is to be used for the corresponding row of the interlacement array. This is projected upward to the tie-up matrix (B), which indicates the shaft or shafts tied up to this particular treadle. Projecting across to the threading matrix (A) gives the precise warp ends which are threaded on this (these) shaft(s). If these shafts are raised, the warp ends which are threaded on them will be raised and will therefore lie on top of the inserted weft pick. The first row of the interlacement array will thus contain 1's where these raised warp ends are located and 0's in the other positions. This process is continued until each row of the shed sequence matrix has been used and all of the rows of the interlacement array have been filled in. The resulting interlacement array is obviously isomorphic to the conventional point-paper diagram [69] form of representation, as described in Section {3.3}.

The relationship between a given interlacement array D and its corresponding threading, tie-up and shed sequence matrices can be more succintly formulated as a matrix equation.

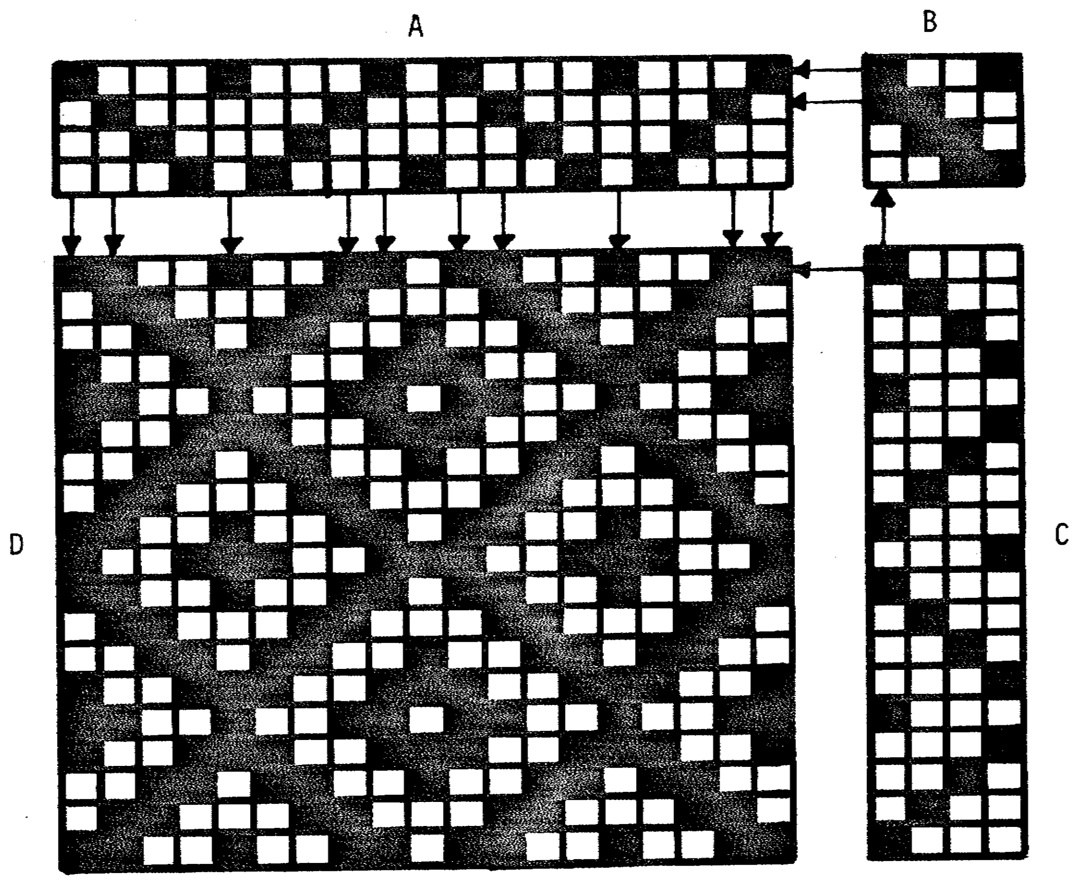


FIGURE 3.2.1

Specifically, if the threading matrix is denoted by A, the tie-up matrix by B and the shed sequence matrix by C (Figure [3.2.2]), then the i,jth element in the interlacement array, is given by

$$d_{i,j} = \bigcup_{r=1}^{m} \bigcup_{k=1}^{n} c_{i,k} \wedge b_{r,k} \wedge a_{r,j} \tag{3.2.3}$$

$$i = 1,2,\ldots,s$$
$$j = 1,2,\ldots,t$$

where the logical operators "and" and "or" replace the conventional matrix operations, multiplication and summation, respectively.

More conveniently, using the notation developed in APL, Equation [3.2.3] can be rewritten as [35]

$$D = (C \vee . \wedge (⍉B)) \vee . \wedge A \tag{3.2.4}$$

where ⍉ denotes the operation of transposition.

Although the preceding matrix equation completely and unambiguously specifies the relationship between the interlacement array and its three factors, this approach is not computationally very efficient. Each element of D is obtained as the result of (r + 1) s "multiplications".

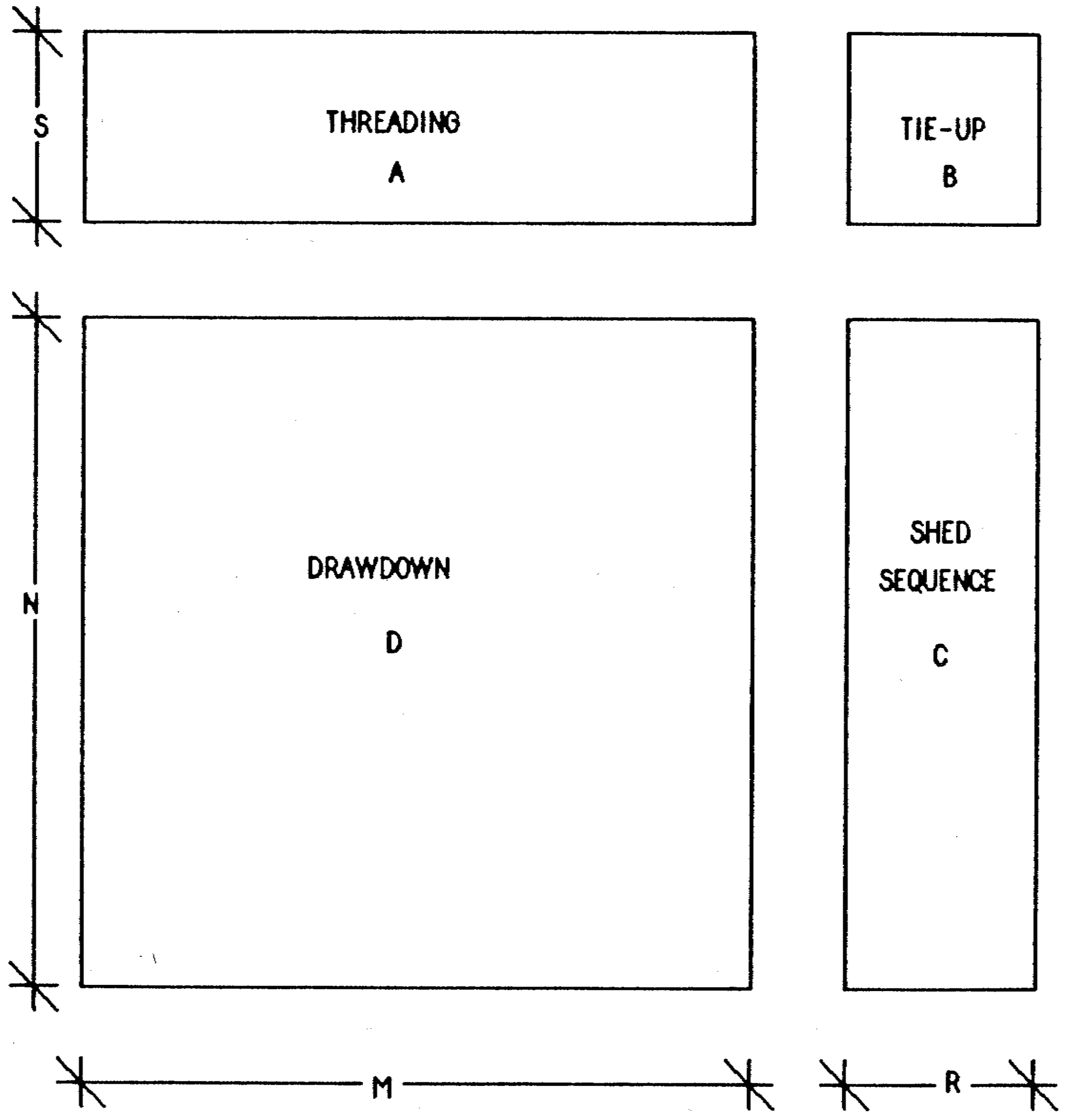


FIGURE 3.2.2

A more efficient and considerably faster process makes use of indirect addressing, as in the following equation:

(3.2.5) $d_{i,j} = b_{q,p}$

where p is the column index of the single 1 in the ith row of the shed sequence matrix C, and where q is the row index of the single 1 in the jth column of the threading matrix A.

This formulation arises from the observation that the tie-up matrix is, in fact, a tile which is used to tessellate the plane defined by the dimensions of the interlacement array. The threading and shed sequence matrices specify the rules according to which this tile is placed. These rules are the placement and orientation of the tie-up matrix tile, as well as whether the entire matrix is to be placed in a given position, or rather some submatrix.

An implementation of this indirect addressing algorithm is given by the following Pascal procedure.

VARIABLE DICTIONARY:

M	NUMBER OF COLUMNS IN THE THREADING MATRIX AND RESULTING INTERLACEMENT ARRAY. (CORRESPONDS TO THE NUMBER OF WARP ENDS IN THE FABRIC SEGMENT)
N	NUMBER OF ROWS IN THE SHED SEQUENCE MATRIX AND RESULTING INTERLACEMENT ARRAY. (CORRESPONDS TO THE NUMBER OF WEFT PICKS IN THE FABRIC SEGMENT)
NS	NUMBER OF ROWS IN THE THREADING AND TIE-UP MATRICES. (CORRESPONDS TO THE NUMBER OF SHAFTS USED)
NT	NUMBER OF COLUMNS IN THE SHED SEQUENCE AND TIE-UP MATRICES. (CORRESPONDS TO THE NUMBER OF TREADLES USED)
THREAD	THREADING MATRIX. BINARY MATRIX OF SIZE NS BY M WITH PRECISELY ONE ONE IN EVERY COLUMN.
SHEDSEQ	SHED SEQUENCE MATRIX. BINARY MATRIX OF SIZE N BY NT WITH PRECISELY ONE ONE IN EVERY ROW.
TIEUP	TIEUP MATRIX. BINARY MATRIX OF SIZE NS BY NT.
INTARRAY	INTERLACEMENT ARRAY. BINARY MATRIX OF SIZE N BY M.
SIZE	VARIABLE TYPE - **PACKED ARRAY** [1 .. 120,1 .. 120] **OF INTEGER**

PASCAL ALGORITHM:

```
PROCEDURE INTAR (VAR THREAD,SHEDSEQ,TIEUP,INTARRAY:SIZE;M,N,NS,NT:INTEGER);

   (* THE ARRAY TEMTIE CORRESPONDS TO THE ARRAY TIEUP WITH AN ADDITIONAL ROW
   AND COLUMN OF ZERO VALUES ADDED THIS ALLOWS THE POSSIBILITY OF A COMPLETELY
   ZERO COLUMN OF THREAD OR ROW OF SHEDSEQ THE VECTOR SHAFTS CONTAINS THE ROW
   INDEX OF THE ONE NON-ZERO VALUE IN EACH COLUMN OF THREAD THE VARIABLES I,J
   AND K ARE LOOP INDICES THE VARIABLE TREADLE CONTAINS THE COLUMN INDEX OF THE
   ONE NON-ZERO VALUE IN THE CURRENT ROW OF THE SHED SEQUENCE MATRIX *)

  VAR
     TEMTIE:SIZE;
     SHAFTS:ARRAY[120] OF INTEGER;
    I,J,K,TREADLE: INTEGER;

  BEGIN

    FOR I:=1 TO NS DO
    BEGIN
       FOR J:=1 TO NT DO TEMTIE[I,J]:=TIEUP[I,J];
       TEMTIE[I,NT+1]:=0;
    END;

    FOR J:=1 TO NT+1 DO TEMTIE[NS+1,J]:=0;

    FOR J:=1 TO M DO
    BEGIN
       K:=1;
       SHAFTS[J]:=NS+1;
       WHILE (K<=NS) AND (SHAFTS[J] > NS) DO
          IF THREAD[K,J] <> 0 THEN SHAFTS[J]:=K ELSE K:=K+1;
    END;

    FOR I:=1 TO N DO
    BEGIN
       K:=1;
       TREADLE:=NT+1;
       WHILE (K<=NT) AND (TREADLE> NT) DO
          IF SHEDSEQ[I,K] <> 0 THEN TREADLE:=K ELSE K:=K+1;
       FOR J:=1 TO M DO
          INTARRAY[I,J]:=TEMTIE[SHAFTS[J],TREADLE];
    END;

  END;
```

3.3 DETERMINING A COLOURED INTERLACEMENT ARRAY

Traditionally, weavers have represented interlacement arrays, and thus the corresponding fabric structures, diagrammaticaly as a matrix of black and white squares, where a black square represents a warp over weft intersection (a value of 1 in our binary matrix notation) and a white square indicates a weft over warp intersection (a value of 0 in our binary matrix notation). This representation is known as a point-paper diagram or draw-down [5].

For a fabric with black warp and white weft yarns, the black and white squares of the point-paper diagram represent the colouring of the fabric, as well as the intersections. There are however, some instances in which it is advantageous that the colour and intersection representations not correspond directly. This occurs when the warp or weft, or both, contain stripes of colour.

Sometimes a particular motif or coloured pattern is required which would not be structurally stable if it were woven with a solid colour warp and weft. In this case, the colour of sections of warp and weft yarns can be changed such that interlacement sequences altered to produce a more structurally stable fabric will result in the same motif as the original. That is, the altered interlacement array will reproduce the original colour array but the correspondence between colour and interlacement array will no longer be one to one. This technique, know to weavers as colour and weave effects [23], [25], is commonly used to produce a wide variety of motifs in fabrics which are structurally stable.

When considering coloured interlacement arrays, the corresponding threading and shed sequence matrices are now integer matrices while the tie-up remains a binary matrix.

The threading matrix still corresponds to the shafts on which each of the warp ends is threaded. The one non-zero entry in each column is however now not necessarily equal to one. This entry instead is some integer value which represents an encoded yarn colouring.

Similarly, the shed sequence matrix represents, not only which treadle is to be depressed for a given weft pick, but also the encoded colour of that yarn. As before, the shed sequence contains precisely one non-zero value per row.

The traditional method of computing a coloured interlacement array (Figure [3.3.1]) requires that one start with the first row of the shed sequence matrix (C) and find the column which has a non-zero entry. This is projected upward to the tie-up matrix (B), and then across to the threading matrix (A), as before. The precise warp ends which are to be raised when the current weft pick is inserted have been determined. The

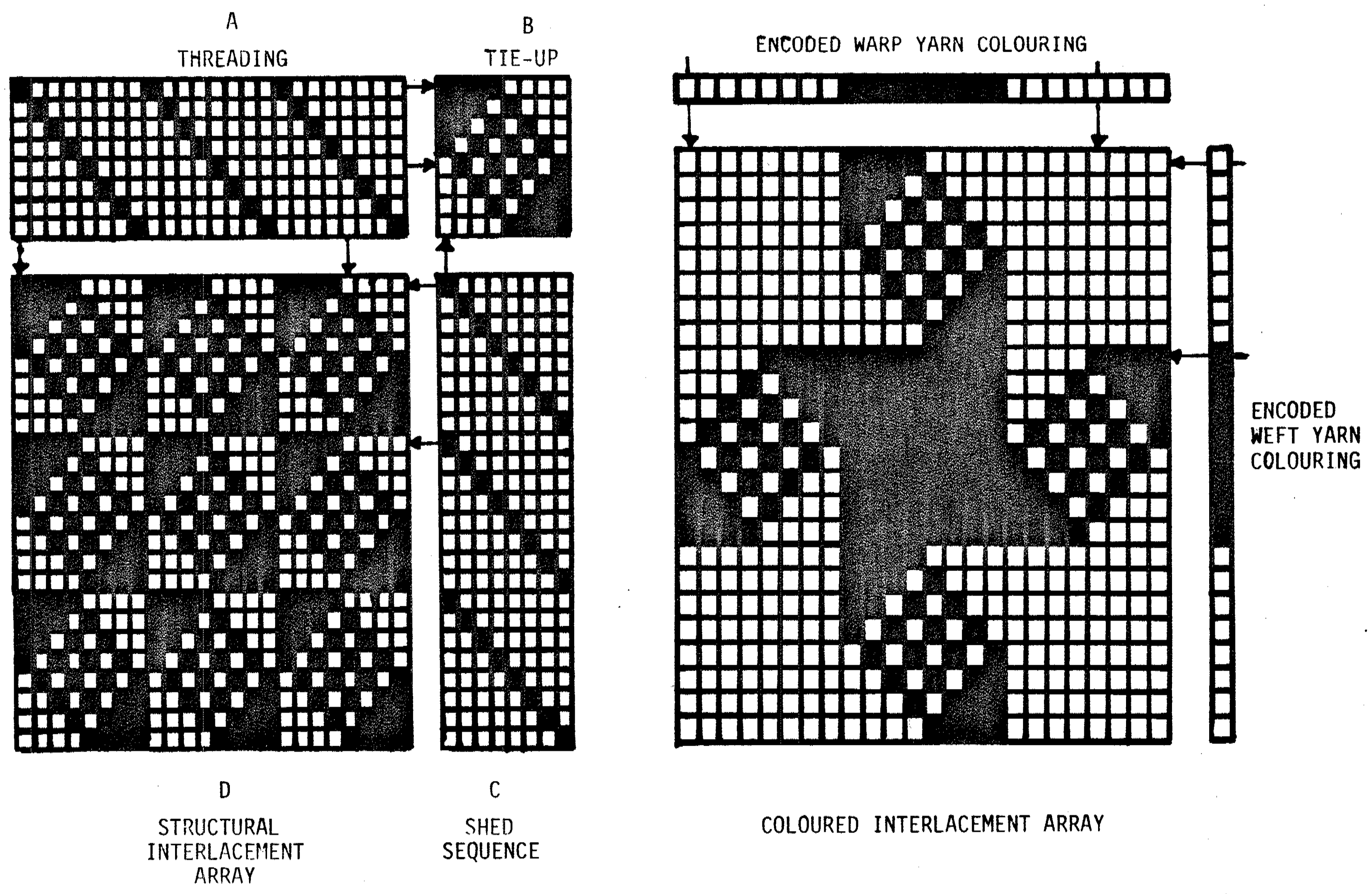

A
THREADING
B
TIE-UP
ENCODED WARP YARN COLOURING
ENCODED WEFT YARN COLOURING
D
STRUCTURAL INTERLACEMENT ARRAY
C
SHED SEQUENCE
COLOURED INTERLACEMENT ARRAY

FIGURE 3.3.1

visible colour in the first row of the coloured interlacement array will thus be the colour of each of the raised warp ends where they lie on top of the weft pick, and the colour of the current weft yarn everywhere else. The coloured interlacement array, in its encoded form, is thus a discrete matrix whose entries correspond to the encoded colours which will appear on the surface of the fabric at each intersection.

The relationship between a given coloured interlacement array and its corresponding threading, tie-up and shed sequence matrices can be formulated using matrix notation. If the threading matrix is denoted by $A = \{a_{i,j}: i=1,2, \ldots ,s; j=1,2, \ldots ,m\}$, the tie-up matrix by $B = \{b_{i,j}: i=1,2, \ldots ,s; j=1,2, \ldots ,r\}$ and the shed sequence matrix by $C = \{c_{i,j}: i=1,2, \ldots ,n; j=1,2, \ldots r\}$, then the i,j^th elements in the coloured interlacement array, denoted by $D = \{d_{i,j}: i=1,2, \ldots ,n; j=1,2, \ldots ,m\}$, is given by the following:

(3.3.2) $$D = Z \propto A$$

where

(3.3.3) $$Z = C \,.\, B',$$

with ' denoting the transpose and the operation being conventional matrix multiplication, and where the operation $\propto$ is defined as

(3.3.4) $$d_{i,j} = (w_{i,j} \times \underset{k=1,2,\ldots,s}{\mathrm{MAX}}\, a_{k,j}) + (\neg w_{i,j} \times \mathrm{MAX}\, z_{i,k})$$

where

(3.3.5) $$w_{i,j} = \bigcup_{k=1}^{s} (z_{i,k} > 0) \wedge (a_{k,j} > 0),$$

and where $\neg w$ denotes the complement of w.

EXAMPLE 3.3.6

$$A = \begin{matrix} 0&0&0&2&0&0 \\ 0&0&3&0&3&0 \\ 0&3&0&0&0&3 \\ 2&0&0&0&0&0 \end{matrix} \qquad B = \begin{matrix} 0&0&1&1 \\ 0&1&1&0 \\ 1&1&0&0 \\ 1&0&0&1 \end{matrix}$$

$$C = \begin{matrix} 2&0&0&0 \\ 0&3&0&0 \\ 0&0&3&0 \\ 0&0&0&2 \\ 3&0&0&0 \\ 0&2&0&0 \\ 0&0&2&0 \\ 0&0&0&3 \end{matrix} \qquad Z = \begin{matrix} 0&0&2&2 \\ 0&3&3&0 \\ 3&3&0&0 \\ 2&0&0&2 \\ 0&0&3&3 \\ 0&2&2&0 \\ 2&2&0&0 \\ 3&0&0&3 \end{matrix}$$

$$W = \begin{matrix} 1&1&0&0&0&1 \\ 0&1&1&0&1&1 \\ 0&0&1&1&1&0 \\ 1&0&0&1&0&0 \\ 1&1&0&0&0&1 \\ 0&1&1&0&1&1 \\ 0&0&1&1&1&0 \\ 1&0&0&1&0&0 \end{matrix} \qquad D = \begin{matrix} 2&3&2&2&2&3 \\ 3&3&3&3&3&3 \\ 3&3&3&2&3&3 \\ 2&2&2&2&2&2 \\ 2&3&3&3&3&3 \\ 2&3&3&2&3&3 \\ 2&2&3&2&3&2 \\ 2&3&3&2&3&3 \end{matrix}$$

An implementation of this algorithm is given in the following Pascal procedure.

VARIABLE DICTIONARY:

M	NUMBER OF COLUMNS IN THE THREADING MATRIX AND RESULTING COLOURED INTERLACEMENT ARRAY. (CORRESPONDS TO THE NUMBER OF WARP ENDS IN THE FABRIC SEGMENT)
N	NUMBER OF ROWS IN THE SHED SEQUENCE MATRIX AND RESULTING COLOURED INTERLACEMENT ARRAY. (CORRESPONDS TO THE NUMBER OF WEFT PICKS IN THE FABRIC SEGMENT)
NS	NUMBER OF ROWS IN THE THREADING AND TIE-UP MATRICES. (CORRESPONDS TO THE NUMBER OF SHAFTS USED)
NT	NUMBER OF COLUMNS IN THE SHED SEQUENCE AND TIE-UP MATRICES. (CORRESPONDS TO THE NUMBER OF TREADLES USED)
THREAD	THREADING MATRIX. DISCRETE MATRIX OF SIZE NS BY M WITH PRECISELY ONE NON-ZERO VALUE IN EVERY COLUMN.
SHEDSEQ	SHED SEQUENCE MATRIX. DISCRETE MATRIX OF SIZE N BY NT WITH PRECISELY ONE NON-ZERO VALUE IN EVERY ROW.
TIE-UP	TIE-UP MATRIX. BINARY MATRIX OF SIZE NS BY NT.
COLARRAY	COLOURED INTERLACEMENT ARRAY. DISCRETE MATRIX OF SIZE N BY M.
SIZE	VARIABLE TYPE - **PACKED ARRAY**[1 .. 120,1 .. 120] **OF INTEGER**

PASCAL ALGORITHM:

```
PROCEDURE COLAR(VAR THREAD,SHEDSEQ,TIEUP,COLARRAY:SIZE;M,N,NS,NT:  INTEGER);

 (* THE ARRAY TEMTIE CORRESPONDS TO THE ARRAY TIEUP WITH AN ADDITIONAL ROW AND
 COLUMN OF ZERO VALUES ADDED. THIS ALLOWS THE POSSIBILITY OF A COMPLETELY ZERO
 COLUMN OF THREAD OR ROW OF SHEDSEQ. THE VECTOR SHAFTS CONTAINS THE ROW INDEX
 OF THE ONE NON-ZERO VALUE IN EACH COLUMN OF THREAD. THE VECTOR COLOURS
 CONTAINS THE ENCODED COLOUR OF EACH OF THE CORRESPONDING WARP THREADS. THE
 VARIABLES I,J AND K ARE LOOP INDICES. THE VARIABLE TREADLE CONTAINS THE COLUMN
 INDEX OF THE ONE NON-ZERO VALUE IN THE CURRENT ROW OF THE SHED SEQUENCE MATRIX.
 THE VARIABLE COLTREAD CONTAINS THE ENCODED COLOUR OF THE CORRESPONDING WEFT
 THREAD *)

 VAR
    TEMTIE:SIZE;
    COLOURS,SHAFTS: ARRAY[1..20] OF INTEGER;
    I,J,K,COLTREAD,TREADLE: INTEGER;

 BEGIN

    FOR I:=1 TO NS DO
    BEGIN
       FOR J:=1 TO NT DO TEMTIE[I,J]:=TIEUP[I,J];
       TEMTIE[I,NT+1]:=0;
    END;

    FOR J:=1 TO NT+1 DO TEMTIE[NS+1,J]:=0;

    FOR J:=1 TO M DO
    BEGIN
       K:=1;
       SHAFTS[J]:=NS+1;
       COLOURS[J]:=0;
       WHILE (K<=NS) AND (SHAFTS[J] > NS) DO
          IF THREAD[K,J] <> 0 THEN
          BEGIN
             SHAFTS[J]:=K;
             COLOURS[J]:=THREAD[K,J];
          END
          ELSE K:=K+1;
    END;
    FOR I:=1  TO N DO
    BEGIN
        K:=1;
        TREADLE:=NT+1;
        COLTREAD:=0;
        WHILE (K<=NT) AND (TREADLE> NT) DO
            IF SHEDSEQ[I,K] <> 0 THEN
            BEGIN
                TREADLE:=K;
                COLTREAD:=SHEDSEQ[I,K];
            END
            ELSE K:=K+1;

        FOR J:=1  TO M DO
        IF TEMTIE[SHAFTS[J],TREADLE]=1  THEN COLARRAY[I,J]:=COLOURS[J]
        ELSE COLARRAY[I,J]:=COLTREAD;
    END;

END
```

3.4 DETERMINING AN INTERLACEMENT ARRAY USING MULTIPLE THREADING

Before beginning a discussion of multiple threading, we must first consider the different types of sheds which can be created by moving shafts up and down from their normal rest position (Figure [3.4.1]). These are basically of three types, namely rising sheds, sinking sheds and those sheds created by the simultaneous raising and lowering of warp ends. Rising sheds are created when a shaft which is tied up to a treadle is raised when that treadle is depressed. All of the remaining shafts remain in their rest position (Figure [3.4.2]). A sinking shed, on the other hand, is created when a shaft which is tied up to a treadle is lowered when that treadle is depressed (Figure [3.4.3]). All of the remaining shafts rise in a counter-balance effect. A rising and sinking shed is produced in a single system when some shafts are tied up so as to rise when a particular treadle is depressed, while other shafts are tied up to sink when the same treadle is depressed (Figure [3.4.4]). All of the shafts are normally tied either to rise or to sink, with the tie-up configuration corresponding to the particular design being woven.

The idea of multiple threading is very old, dating back to at least the eleventh century in England and even earlier than that in Asia [29, p.199]. Numerous references appear throughout the literature describing different types of double threading schemes [75, p.170], [16], and to a

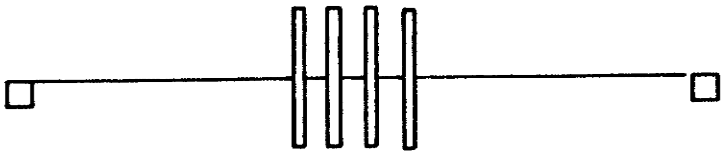


FIGURE 3.4.1 REST POSITION

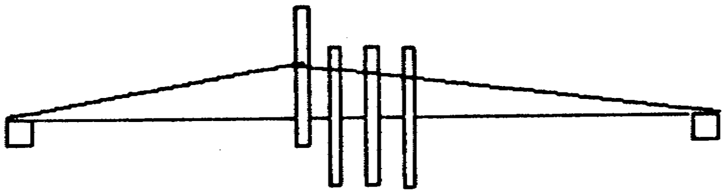


FIGURE 3.4.2 RISING SHED

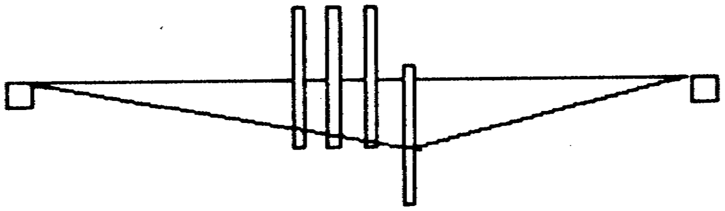


FIGURE 3.4.3 SINKING SHED

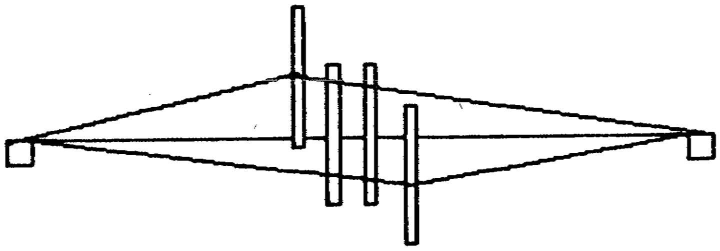


FIGURE 3.4.4 RISING AND SINKING SHED

discussion of various applications [74], [49]. These references however, refer primarily to two particular schemes, the double presser harness system [60, p.375] used on looms such as the counter-marche loom which employ the rising and sinking shed system , and a modification of this system for use on a rising shed loom [19]. In future the former will be referred to as Type 1 and the latter as Type 2.

The advantage of multiple threading is that a smaller number of shafts may be required to produce a particular weave structure than if that same fabric had been woven on a loom which was singly threaded. Structures which successfully employ this technique are generally those which can be partitioned into recognizable blocks and counter-blocks. In other words, the macro-structure of the fabric consists of only two different design elements. Each of these two blocks is, in itself, an interlacement array. A multiply threaded loom then has shafts which fulfill one of two functions. A given warp end is threaded through a heddle on one shaft which controls the gross pattern of the fabric and then through a second, and possibly a third, shaft which controls the detailed or ground structure of the fabric (Figure [3.4.5]).

The Type 1 system uses long-eyed heddles on the ground shafts (x) at the front of the loom and regular heddles on the pattern shafts (y), located at the back of the loom. Each of the heddles on the ground shafts can take one of three positions -- down, neutral or up, which can be

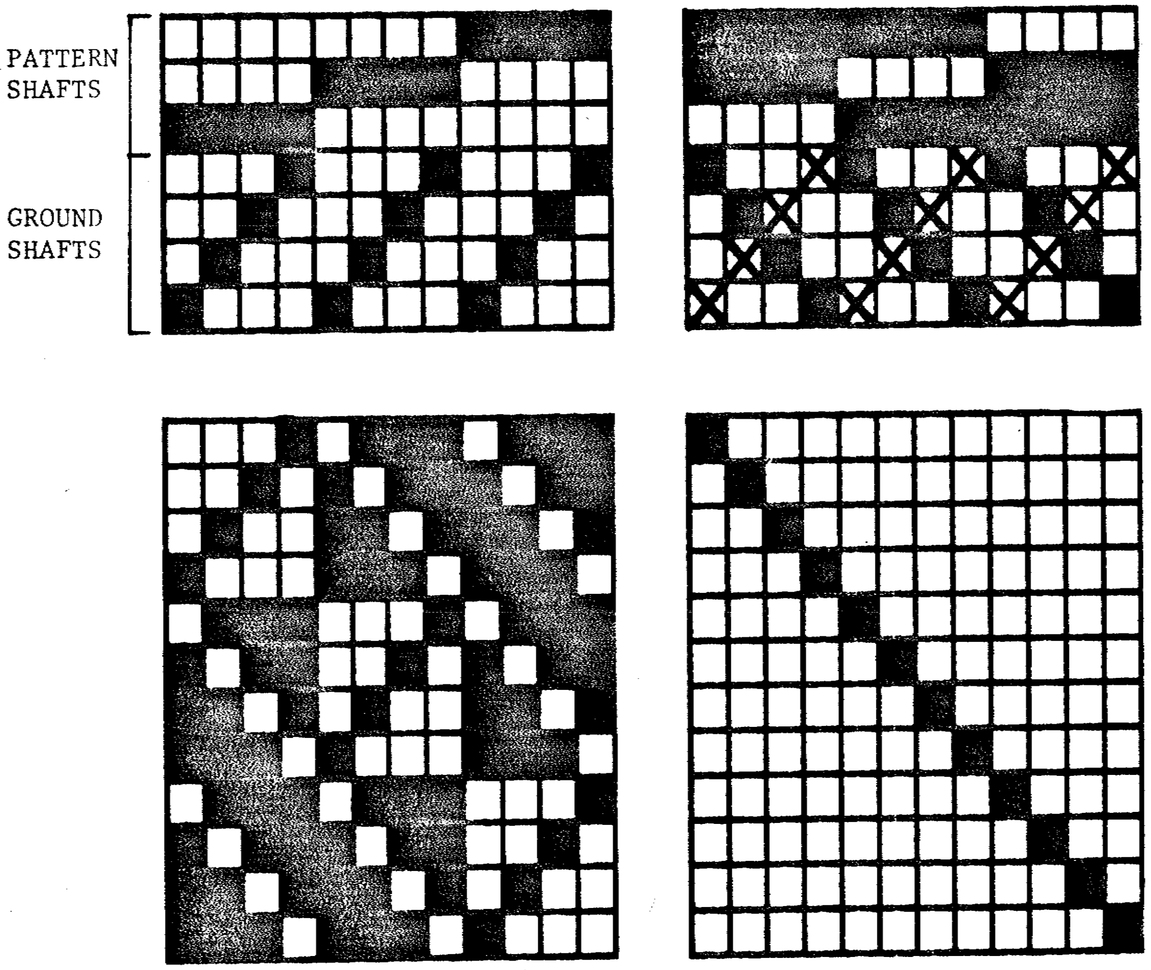


FIGURE 3.4.5

represented by 0, 1 and 2 respectively, while each of the heddles on the pattern shafts can only take one of two positions -- down and up or 0 and 1. All possible configurations of these two groups of shafts are illustrated in Figure (3.4.6), along with an indication of which groupings will produce a shed, in the position of the shuttle race marked with an arrow. From this diagram, it is clear that the presence or absence of a shed formed between any single warp end (w) and the remaining warp ends, which are threaded on pattern shafts in the neutral position and ground shafts in the down position, can be modelled by the values of the expression

(3.4.7) $$(x_w + y_w) \geq 2$$

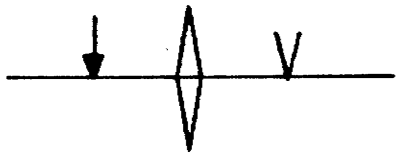
where $x_w \in x$

and $y_w \in y$.

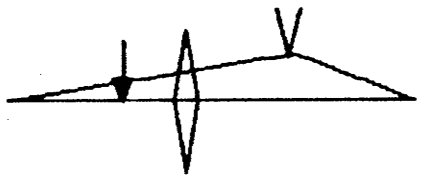
One method of determining an interlacement array (D) in these circumstances is to evaluate

(3.4.8) $$D = C\,B'\,A$$

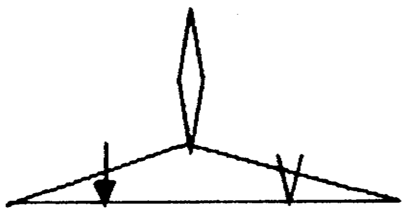
where ' indicates the usual matrix transpose and the operation is



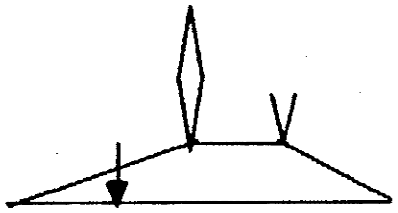
X = NEUTRAL	1
Y = DOWN	0
NO SHED	1



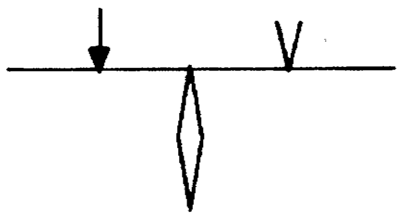
X = NEUTRAL	1
Y = UP	1
SHED	2



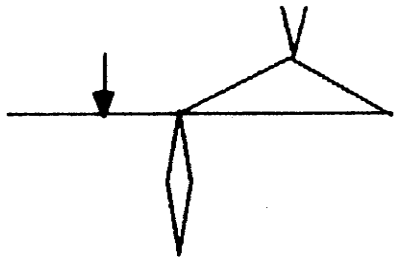
X = UP	2
Y = DOWN	0
SHED	2



X = UP	2
Y = UP	1
SHED	3



X = DOWN	0
Y = DOWN	0
NO SHED	0



X = DOWN	0
Y = UP	1
NO SHED	1

FIGURE 3.4.6

conventional matrix multiplication. The threading matrix (A) now contains precisely two 1's per column, with one of these 1's appearing in the area corresponding to the ground, or x shafts, and the other appearing in the area corresponding to the pattern, or y shafts. The tie-up matrix (B) now contains values of 0, 1 or 2 in the area of the tie-up corresponding to the ground shafts and values of 0 or 1 in the area of the tie-up corresponding to the pattern shafts. The shed sequence matrix contains precisely one 1 per row with the rest of the elements being 0. The resulting matrix D contains elements of value 0, 1, 2, or 3. In order that this matrix be interpreted as an interlacement array, each of the elements is divided by 2 using integer division, to obtain the required binary matrix.

An alternative algorithm for determining an interlacement array makes use of the indirect addressing concept discussed in Section (3.2), using the following equation:

$$(3.4.9) \qquad d_{i,j} = ((b_{r,p} = 1) \vee (b_{r,q} = 2)) \wedge (b_{r,q} \neq 3),$$

where r is the column index of the single 1 in the ith row of the shed sequence matrix C, and where p and q are the row indices of the first and second 1, respectively, in the jth column of the threading matrix A.

This formulation corresponds to the requirement that, in order to have a warp end lie on top of a given weft pick, either the ground or the pattern shaft (or both) on which the warp end is threaded must be raised and the ground shaft must not be in the down position.

An implementation of this algorithm is given by the following Pascal procedure.

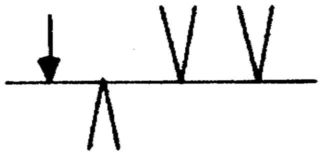
VARIABLE DICTIONARY:

M	NUMBER OF ROWS IN THE THREADING MATRIX AND RESULTING INTERLACEMENT ARRAY. (CORRESPONDS TO THE NUMBER OF WARP ENDS IN THE FABRIC SEGMENT)
N	NUMBER OF ROWS IN THE SHED SEQUENCE MATRIX AND RESULTING INTERLACEMENT ARRAY. (CORRESPONDS TO THE NUMBER OF WEFT PICKS IN THE FABRIC SEGMENT)
NS	NUMBER OF ROWS IN THE THREADING AND TIE-UP MATRICES. (CORRESPONDS TO THE NUMBER OF SHAFTS USED)
NT	NUMBER OF COLUMNS IN THE SHED SEQUENCE AND TIE-UP MATRICES. (CORRESPONDS TO THE NUMBER OF TREADLES USED)
THREAD	THREADING MATRIX. BINARY MATRIX OF SIZE NS BY M WITH PRECISELY TWO ONES IN EVERY COLUMN.
SHEDSEQ	SHED SEQUENCE MATRIX. BINARY MATRIX OF SIZE N BY NT WITH PRECISELY ONE ONE IN EVERY ROW.
TIEUP	TIEUP MATRIX. DISCRETE MATRIX OF SIZE NS BY NT.
INTARRAY	INTERLACEMENT ARRAY. BOOLEAN MATRIX OF SIZE N BY M.
SIZE	VARIABLE TYPE - **PACKED ARRAY** [1 .. 120,1 .. 120] **OF INTEGER**

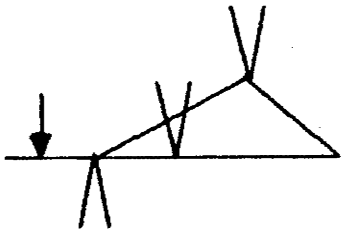
PASCAL ALGORITHM:

```
PROCEDURE DOUBLETH(VAR THREAD,SHEDSEQ,TIEUP,I NTARRAY:SIZE;M,N,NS,NT:  INTEGER);

   (* THE ARRAY TEMTIE CORRESPONDS TO THE ARRAY TIEUP WITH AN ADDITIONAL  ROW
   AND COLUMN OF ZERO VALUES ADDED. THIS ALLOWS THE POSSIBILITY OF A COMPLETELY
   ZERO COLUMN OF THREAD OR ROW OF SHEDSEQ. THE VECTOR PATTERN CONTAINS THE ROW
   INDEX OF THE FIRST NON-ZERO VALUE IN EACH  COLUMN OF THREAD. THE VECTOR GROUND
   CONTAINS THE SECOND NON-ZERO  VALUE IN EACH COLUMN OF THREAD. THE VARIABLES I,J
   AND  K ARE LOOP INDICES. THE VARIABLE TREADLE CONTAINS THE COLUMN INDEX OF THE
   ONE NON-ZERO VALUE IN THE CURRENT ROW OF THE SHED SEQUENCE  MATRIX. *)

  VAR
     TEMTIE:SIZE;
     GROUND,PATTERN: ARRAY[1..20] OF INTEGER;
     I,J,K,TREADLE: INTEGER;

  BEGIN
       FOR I:=1  TO NS DO
       BEGIN
          FOR J:=1  TO NT DO TEMTIE[I,J]:=TIEUP[I,J];
          TEMTIE[I,NT+1]:=0;
       END;
       FOR J:=1  TO NT+1 DO TEMTIE[NS+1,J]:=0;

       FOR J:=1  TO M DO
       BEGIN
          K:=1;
          GROUND[J]:=NS+1;
          PATTERN[J]:=NS+1;
          WHILE (K<=NS) AND (GROUND[J] > NS) DO
             IF THREAD[K,J] <> 0 THEN
               IF PATTERN[J] > NS THEN
               BEGIN
                  PATTERN[J]:=K;
                  K:=K+1;
               END
               ELSE GROUND[J]:=K
             ELSE K:=K+1;
       END;

       FOR I:=1  TO N DO
       BEGIN
          K:=1;
```

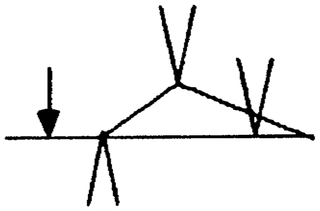
```
        TREADLE:=NT+1;
        WHILE (K<=NT) AND (TREADLE> NT) DO
           IF SHEDSEQ[I,K] <> 0 THEN TREADLE:=K ELSE K:=K+1;
        FOR J:=1 TO M DO
        IF((TEMTIE[PATTERN[J],TREADLE]=1)  OR (TEMTIE[GROUND[J],TREADLE]=2))
        AND (TEMTIE[GROUND[J],TREADLE]<>3)  THEN
           INTARRAY[I,J]:=1
        ELSE
           INTARRAY[I,J]:=0;
     END;

END;
```



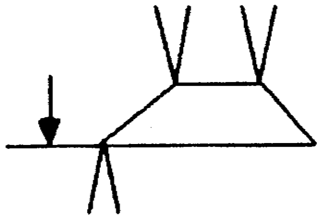
L = DOWN	0
M = DOWN	0
N = DOWN	0
NO SHED	0



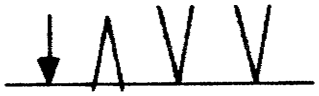
L = DOWN	0
M = DOWN	0
N = UP	1
NO SHED	0



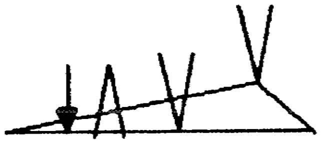
L = DOWN	0
M = UP	1
N = DOWN	0
NO SHED	0



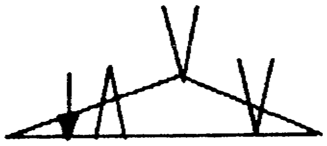
L = DOWN	0
M = UP	1
N = UP	1
NO SHED	0



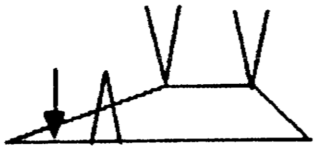
L = UP	1
M = DOWN	0
N = DOWN	0
NO SHED	0



L = UP	1
M = DOWN	0
N = UP	1
SHED	1



L = UP	1
M = UP	1
N = DOWN	0
SHED	1



L = UP	1
M = UP	1
N = UP	1
SHED	1

FIGURE 3.4.10

The second system (Type 2) is meant to be used where only a rising shed is available and involves triple threading of each warp end [19] (Figure (3.4.10)). The first set of ground shafts at the front of the loom (L) contain push down heddles, the second group of ground shafts (M) contain push up heddles and the pattern harnesses (N) contain either regular or push up heddles. Each type of heddle can take one of two positions -- down, where down is equivalent to neutral in the Type 1 system, and up, these positions being represented by 0 and 1 respectively as in the figure. Since this is a completely binary system, the possible sheds that can be formed between a single warp end (w) and the remaining warp ends, which are considered to be in the down position, can be represented by the values of the logical expression

(3.4.11) $$L_w \wedge (M_w \vee N_w)$$

where $L_w \in L$, $M_w \in M$ and $N_w \in N$.

The algorithm for determining an interlacement array D in this instance is given by the following steps:

(1) $S = C\,B'$, where $'$ indicates the usual matrix transpose and the operation is conventional matrix multiplication

(2) K is assigned a value of 1

(3) The threading matrix A is partioned row-wise into 3 sets, where the bottom partition is the area of the threading corresponding to the first ground shafts, the second partition is the area of the threading corresponding to the second set of ground shafts, and the top partition is the area of the threading corresponding to the pattern shafts.

(4) The K^th row of S is "multiplied" by every column of A with no summation taking place.

(5) The elements of the K^th row of D are determined by evaluating the logical expression

(3.4.12) $$L \beta (M \gamma N)$$

for every column of the result of (4).

The dyadic operation β takes two binary matrices as arguments and returns a result with one row. The i^th element in this row is 1 if either argument has a 1 in its i^th column.

The dyadic operation γ takes two binary matrices as arguments and returns a result with one row. The i^{th} element in this row is 1 if both arguments have a 1 in their i^{th} columns.

(6) K is assigned a value of K + 1.

(7) Steps (4), (5) and (6) are repeated for every row of S.

This algorithm is clearly very similar to the second algorithm discussed for the Type 1 system and, in fact we have the following result:

<u>THEOREM 3.4.13</u>. The double presser harness threading system (Type 1) can be simulated on a triply threaded rising shed loom (Type 2).

<u>Proof</u>. From the preceding discussion, it is evident that the triply threaded rising shed system can be represented by

$$R: \; L_W \wedge (M_W \vee N_W)$$

and that the double presser harness system can be represented by

$$C: \; X_W + Y_W$$

where it may be recalled that L_W, M_W, N_W, Y_W are binary variables and X_W is a ternary variable.

As previously shown, the two expressions R and C describe whether or not a given warp end will be allowed to rise to create a shed. R takes the values 0 and 1 while C can take the values 0, 1, 2 and 3.

The values of the ternary variable X_W may be represented by the following arithmetic expression:

(3.4.14) $$X_W = Z_1 \times (Z_1 + Z_2)$$

where Z_1 and Z_2 are binary valued variables. Then

(3.4.15) $$X_W + y_W = Z_1 \times (Z_1 + Z_2) + Y_W.$$

But (3.4.15) is "true" if and only if $X_W + Y_W = 2$ or 3. That is, Z_1 <u>must</u> be true (=1) <u>and, either</u> Z_2 must be true (= 1) <u>or</u> Y_W must be true (=1), therefore $Z_1 \wedge (Z_2 \vee Y)$, as required. □

3.5 <u>FACTORING INTERLACEMENT ARRAYS</u>

3.5.1 <u>THE FUNDAMENTAL PROBLEM</u>

Weavers frequently wish to know what loom set up and what operating sequence they should use in order to weave a particular structure. This involves them in problems of textile analysis in which the method of construction has to be determined from a sample of cloth or from an abstract interlacement array. For shuttle woven structures (i.e. those in which the weft picks run from one side or selvedge completely to the other side or selvedge), this means the determination of three binary matrices (A, B, C) referred to as the threading, tie-up and shed sequence (or treadling) matrices, respectively in the literature [60]. Numerous algorithms for this form of analysis have appeared in the past [13, p.130-133], [18]. However they all assume that the analysis is being performed directly on the physical sample, and hence rely solely on a slow and tedious manual approach.

The analysis process can however be divided into two distinct phases. The first phase involves translating the intersections which every warp yarn in the fabric sample makes with every weft yarn into the corresponding interlacement array. The second phase of the analysis can now take place entirely with respect to this interlacement array, with no further reference to the actual fabric itself. This task now becomes an interesting problem in binary matrix factorization and is ideally suited to machine algorithms.

Five such algorithms will be described, namely the Mathematical, the Classical, a new algorithm called the Bucket Sort, and variations of this latter process called the Alternating Direction and the Minimal Bucket Sort Algorithms. Theoretical comparisons between these processes will be made, so as to determine estimates of the order of the number of operations performed. A brief discussion of the practical considerations of implementation will also be included.

3.5.2 THE MATHEMATICAL ALGORITHM

The Mathematical Algorithm for factoring a binary interlacement array into its threading, shed sequence and tie-up components proceeds with absolutely no regard to the data which is being analyzed and, as a result, much information is discarded. The process basically requires that every element of the first column be compared with every element of all the remaining columns. This establishes which columns of the interlacement array are identical to the first column. Next, every element of the first of the columns not belonging to this equivalence class is compared with every element of all the remaining columns not in the equivalence class. This process is continued until all of the columns are partitioned into equivalence classes. The threading matrix will thus contain as many columns as there are columns in the interlacement array and as many rows as there are equivalence classes. Each column of the threading matrix contains precisely one "1" value located in the row corresponding to the number of the equivalence class to which that column belongs.

Once the columns of the array have been analyzed, and the threading matrix thus determined, the identical procedure is applied to the rows of the matrix, to give the shed sequence matrix. The shed sequence matrix has as many rows as the interlacement array has rows and as many columns as there are equivalence classes of rows. Each row of the shed sequence matrix contains precisely one "1" value, in the column corresponding to the number of the equivalence class to which the row belongs.

The tie-up matrix contains as many columns as there are equivalence classes of rows and as many rows as there are equivalence classes of columns. For each distinct row of the interlacement array, the tie-up contains "1" values in positions corresponding to the equivalence classes of columns with a "1" value in that row.

THEOREM 3.5.2.1. There is no binary interlacement array for which this algorithm is not maximal.

Proof. Every element of two columns which are being compared for equivalence is examined in order to detect corresponding positions in which they disagree. The only circumstance under which every pair of elements need be checked for disagreement is in the case of equivalent columns. Therefore, all of the columns are equivalent and the fabric structure corresponding to this array is certainly reducible (see Chapter 4). Furthermore, as will be discussed in Section [3.5.4], no comparisons need be performed on the rows, in order to determine the distinct rows of this array. There are only two possible rows, namely all ones or all zeros. The Mathematical algorithm ignores this information and requires that all of the first row be compared with all of the elements of all the remaining rows to determine the rows which lie in the first equivalence class. If there are two equivalence classes of columns, all of the elements of the rows in that class are compared with a representative row of the class. □

An example of this algorithm is discussed in detail in Section (3.5.7).

3.5.3 THE CLASSICAL ALGORITHM

The first stage in the Classical Algorithm can be construed simply as performing the identification of the number of distinct columns in an interlacement array D and determining which columns are identical. Identical columns in the interlacement array correspond to warp yarns which intersect with all of the weft yarns in the fabric, in precisely the same way. Each set of distinct warp yarns must be threaded on a separate shaft.

Similarly, in the second stage of this algorithm, the distinct rows of this interlacement array must be identified and the identical rows determined. All identical rows in the interlacement array correspond to weft yarns which intersect with all of the warp yarns in the fabric, in exactly the same way. Each set of distinct weft picks must be assigned to a separate treadle.

The third stage of the algorithm, which can in fact be executed simultaneously with stage two, involves determining, for each distinct row of the interlacement array, precisely which combination of shafts must be raised in order to produce this row sequence.

There are numerous variations of this algorithm [70], [20] which have traditionally been performed by hand. Since the order of the interlacement arrays can be quite large (10^5 to 10^6 elements in one of these matrices is not unreasonable), these processes become extremely tedious and prone to error. Even a straightforward computer implementation of the Classical Algorithm therefore represents an improvement in the speed and accuracy of fabric analysis.

One such computer implementation of the hand algorithm for fabric analysis involves the following steps [31]:

(3.5.3.1) <u>To obtain the threading matrix (A)</u>

1. Put a 1 in $A_{1,1}$ and 0 in $A_{k,1}$, $k = 2, 3, \ldots, s$, where s is the maximum number of rows of A (corresponding to the maximum number of available shafts).

2. Compare the first column of D to all other columns of D.

3. For every column which <u>does not</u> match the first one in every corresponding position, put a 0 in the corresponding column of the first row of A.

4. For every column which <u>does</u> match the first one exactly, put a 1 in the corresponding position of the first row of A and 0 in the rest of that column of A, as was done with the first column.

5. Choose the first column of D which does not already have a 1 in its corresponding column of A. Put a 1 in this column of A, in the next row, and 0 in the rest of the column.

6. Compare this column with the remaining unused columns, assigning 1 and 0 to the A matrix as before.

7. Repeat this process until all columns of D have a 1 in some row of the corresponding column of A.

(3.5.3.2) <u>To obtain the shed sequence or treadling matrix (C)</u>

1. Repeat the preceding 7 steps for the rows of D, with the corresponding entries being made in the matrix C.

(3.5.3.3) <u>To obtain the tie-up matrix (B)</u>

1. For each distinct row of D, determine which elements of this row are equal to 1. For each of these positions, scan the corresponding column of A to find the row index of the single 1 element. Place a 1 in this row of B, in the column associated with the single 1 in the row of C corresponding to this row of D.

Examples of this algorithm are discussed in detail in Section [3.5.7], where it is shown that the order of the number of comparisons for only the first pass of the threading is $n(m-1)$, with n the number of rows of D and m the number of columns. The computer implementation of this algorithm performs quite satisfactorily, although it takes no advantage of being machine based.

3.5.4 THE BUCKET SORT ALGORITHM

The historic algorithm outlined in Section (3.5.3) involves a simple identification of the distinct columns and rows of an interlacement array D. However, the number of rows and columns in D are typically large and it is generally known a priori that the number of distinct columns will be much smaller (eg. of order 16 - 50). In practice, rows may differ by as little as a single element and the different element may be in any position. The threading analysis compares columns of D while the shed sequence analysis compares rows of D so that, although it is possible to use special hardware features which will automatically perform extended memory comparisons as part of a sorting algorithm for the threading, the interleaving of memory that this will imply for the corresponding row-wise sort prohibits the simple use of this type of hardware feature.

It is also possible to note that in comparing two columns for distinctness and accepting their difference, a considerable amount of information is ignored if their first encountered position of difference is discarded. This is the case with the classical algorithm.

Let us now introduce the following terms.

Let $$D_j = \{d_{1,j}, d_{2,j}, \ldots d_{n,j}\}^T$$
$$j = 1, 2, \ldots m,$$

be the set of column sequences considered, where m is the number of columns, n is the the number of rows, and

$$Q_{j,k} = \min_{r \in \{1, 2, \ldots n\}} r \ni \{d_{r,j} \neq d_{r,k}\}.$$

The following procedure can now be applied:

<u>Definition 3.5.4.1</u>.

Define a bucket $B_{s,j:q}$ as the set

$$B_{s,j:q} = \{s: s = Q_{j,k}\}$$
$$s = 0, 1, 2, \ldots n$$
$$k = 1, 2, \ldots$$
and $$q = 1, 2, 3, \ldots$$

corresponds to the qth stage of the bucket determination.

Obviously, use of the column D_j has generated a set of <u>distinctness</u> classes determined by their first element of difference, and we have the two results.

Theorem 3.5.4.2. Every set $B_{s,j:q}$ contains all columns which are possibly equal.

Proof. Follows from Definition (3.5.4.1). □

THEOREM 3.5.4.3. The number of distinct columns is greater than or equal to the number of non-null buckets $B_{s,j:q}$.

Proof. Follows from Definition (3.5.4.1). □

COROLLARY 3.5.4.4. In any given bucket $B_{s,j:q}$, two sequences can only differ in position k where $k > s + 1$.

Proof. Follows from Definition (3.5.4.1). □

A convenient method of implementing the algorithm implied by these observations is therefore:

1. j is assigned a value of 1; q is assigned a value of 1.

2. Determine $B_{s,j:q}$ for all columns not identified.

3. All columns identical to D_j are in bucket $B_{n+1,j:q}$. Identify them and $B_{n+1,j:q}$ is assigned this set of columns.

4. All columns in $B_{n,j:q}$ are identical (sequences are binary). Identify them and $B_{n,j:q}$ is assigned this set of columns.

5. Select any element in $B_{k,j:q}$ where $k = \max r$ $(k \in 0 \ \ldots m)$. If none exist, then $B_{r,j:q}$ is assigned the null set.

6. j is assigned the value k. q is assigned the value q + 1. Repeat from step 2.

<u>COROLLARY 3.5.4.5.</u> If $B_{s,j:q}$ over the unidentified columns $(q > 1)$ is being computed then it need only be determined for the contents of the bucket $B_{k,j:q-1}$.

<u>Proof.</u> All other buckets already have a lower index of disagreement Q and will be unchanged. □

<u>COROLLARY 3.5.4.6.</u> If a bucket contains a single column, it is a distinct column.

<u>Proof.</u> Follows immediately. □

<u>COROLLARY 3.5.4.7.</u> If $B_{s,j:q}$ over the unidentified columns is being determined then $Q_{j,k:q}$ (where k belongs to the set of {unidentified column

indices] is such that

$$n + 1 \geq Q_{j,k:q} \geq Q_{j,k:q-1} \quad .$$

<u>Proof</u>. The index of disagreements between the columns of a bucket cannot decrease and cannot exceed n + 1. □

The shed sequence matrix determination part of the algorithm proceeds in an exactly similar manner. However, it should be apparent that, in determining the distinct rows, <u>having</u> examined the columns, further information is available, viz.

<u>THEOREM 3.5.4.8.</u> The distinctness of row sequences cannot differ over columns which are identical.

<u>Proof</u>. Consider two rows r_1 and r_2 and two identical columns c_1 and c_2, with intersections $e_{1,1}$, $e_{1,2}$, $e_{2,1}$ and $e_{2,2}$.

<u>Case 1</u>: If $e_{1,1} = e_{2,1}$ then $e_{1,2} = e_{2,2}$, and thus r_1 and r_2 are not distinct over both c_1 and c_2.

<u>Case 2</u>: If $e_{1,1} \neq e_{2,1}$ then $e_{1,2} \neq e_{2,2}$, and thus r_1 and r_2 are

distinct over both c_1 and c_2. □

COROLLARY 3.5.4.9. If k columns are identical, then for the row-wise algorithm k-1 columns can be discarded.

Proof. Follows directly. □

THEOREM 3.5.4.10. If the first pass of the Bucket Sort Algorithm has been applied to the columns of a two dimensional array then columns which agree to k places may be replaced by one of their number and the others ignored in their first k places since they do not contribute to the distinctness of rows.

Proof. Without loss of generality we can assume the first column defines the first bucket classification of columns and that by permutation of the columns each column in the array agrees to fewer or the same number of places with column 1.

Case 1. A column is identical to column 1: This implies that if two rows are distinct over the identical columns then they will still be distinct if all except one of the identical columns is ignored. In addition, if two rows are identical then the corresponding elements in the identical columns are all identical even if all except one of the identical columns is ignored.

Case 2. The r^{th} column is identical to the first column as far as the k^{th} position: This implies that, for the first k rows, Case 1 applies and the theorem follows. □

Examples of an algorithm based on the preceding results are discussed in detail in Section (3.5.7) and a machine implementation of this algorithm is given in the following Pascal procedures.

VARIABLE DICTIONARY:

N	NUMBER OF ROWS IN THE BINARY INTERLACEMENT ARRAY
M	NUMBER OF COLUMNS IN THE BINARY INTERLACEMENT ARRAY
SIZE	VARIABLE TYPE - **PACKED ARRAY** [1 .. 120,1 .. 120] **OF INTEGER**

PASCAL ALGORITHM:

```
PROCEDURE SHUFFLE(VAR VECTOR:VECSIZE;START,FINISH: INTEGER);
 (* THIS PROCEDURE ACCEPTS ONE VECTOR AND TWO INTEGERS AS ARGUMENTS. THE TWO
 INTEGERS SPECIFY A RANGE OF ELEMENTS WITHIN THE VECTOR. THE VECTOR IS PASSED
 BACK WITH THE ELEMENTS WITHIN THE SPECIFIED RANGE MOVED ONE POSITION TO THE LEFT.
 *)

VAR
   I:INTEGER;

BEGIN
   FOR I:=START TO FINISH-1 DO VECTOR[I]:=VECTOR[I+1];
END;

PROCEDURE INSERT(VAR POINTERS,SORT:VECSIZE;TARGET,POSITION, LAST:INTEGER);
 (* THE PROCEDURE PERFORMS A BISECTION SEARCH ON A VECTOR OF DISAGREEMENTS,
 SORTED IN ASCENDING ORDER (SORT), TO FIND THE POSITION WHERE THE CURRENTLY
 COMPUTED DISAGREEMENT (TARGET) BELONGS. THE INSERTION IS MADE AND THE
 CORRESPONDING VECTOR OR POINTER (POINTERS) IS ALTERED ACCORDINGLY. THE
 VARIABLE POSITION INDICATES TO WHICH COLUMN OR ROW THIS DISAGREEMENT
 CORRESPONDS. THE VARIABLE LAST CORRESPONDS TO THE INDEX OF THE LAST
 DISAGREEMENT WHICH MUST BE CHECKED. THE INSERTION WILL BE MADE BETWEEN
 POSITION AND LAST. *)

VAR
   INDEX,FIRST,MID: INTEGER;
   FOUND:BOOLEAN;

BEGIN
   IF POSITION<LAST THEN
   BEGIN
      INDEX:=POINTERS[POSITION];
      IF TARGET<SORT[LAST] THEN
      BEGIN
         FIRST:=POSITION;
         FOUND:=FALSE;
         WHILE (FIRST<=LAST) AND NOT FOUND DO
         BEGIN
            MID:=(FIRST+LAST) DIV 2;
            IF (TARGET>=SORT[MID]) AND (TARGET<=SORT[MID+1]) THEN
               FOUND:=TRUE
            ELSE
               IF TARGET>SORT[MID] THEN FIRST:=MID+1
               ELSE LAST:=MID-1;
         END;
         SHUFFLE(SORT,POSITION,MID);
         SHUFFLE(POINTERS,POSITION,MID);
         SORT[MID]:=TARGET;
         POINTERS[MID]:=INDEX;
      END
      ELSE
      BEGIN
         SHUFFLE(SORT,POSITION,LAST);
         SHUFFLE(POINTERS,POSITION,LAST);
         SORT[LAST]:=TARGET;
         POINTERS[LAST]:=INDEX;
      END
   END
   ELSE SORT[POSITION]:=TARGET;
END;

FUNCTION MATCH(BUCKET,N,M,R1,R2:  INTEGER;ROW: BOOLEAN;VAR
BNEW:INTEGER;VAR INTARRAY:SIZE;ROWSORT:VECSIZE): BOOLEAN;
   (* THIS FUNCTION COMPARES TWO ROWS OR COLUMNS (R1 AND R2) OF A BINARY
 INTERLACEMENT ARRAY TO DETERMINE WHETHER THEY ARE IDENTICAL IN ALL
 CORRESPONDING POSITIONS. IF THEY ARE IDENTICAL THEN THE FUNCTION RETURNS A
 VALUE OF TRUE. IF THEY ARE NOT IDENTICAL THEN THE VALUE OF THE FUNCTION IS FALSE,
 AND THE POINT OF DISAGREEMENT IS RETURNED IN THE VARIABLE BNEW. COMPARISONS
 BEGIN AT THE VALUE PASSED DOWN IN THE VARIABLE BUCKET. THE VARIABLE ROW IS TRUE
 IF ROW COMPARISONS ARE TO BE MADE AND FALSE IF COLUMN COMPARISONS ARE TO BE
 MADE. *)

VAR
   STATE:BOOLEAN;
   INDEX:INTEGER;

BEGIN
   INDEX:=BUCKET;
   IF ROW THEN
   BEGIN
      REPEAT
      INDEX:=INDEX+1;
      STATE:=INTARRAY[R1,ROWSORT[INDEX]]=INTARRAY[R2,ROWSORT[INDEX]];
      UNTIL (NOT STATE) OR (INDEX=M);
   END
   ELSE
   BEGIN
      REPEAT
      INDEX:=INDEX+1;
      STATE:=INTARRAY[INDEX,R1]=INTARRAY[INDEX,R2];
      UNTIL (NOT STATE) OR (INDEX=N);
    END;
    BNEW:=INDEX-1;
   MATCH:=STATE;
END;



PROCEDURE ANALYSIS(VAR INTARRAY:SIZE;VAR ROWSORT, SHAFTNUM: VECSIZE;M,N:
INTEGER; VAR NS:INTEGER);

(* THIS PROCEDURE DETERMINES THE THREADING MATRIX WHICH CORRESPONDS TO A GIVEN
BINARY INTERLACEMENT ARRAY *)

VAR
   SORT,POINTERS:VECSIZE;
   BNEW,BUCKET,I,LAST,R1,R2,SHAFT,P1,P2:  INTEGER;
   CONDITION,ROW: BOOLEAN;

BEGIN
   SHAFT:=1;
   R2:=M-1;
   LAST:=M;
   ROW:=FALSE;
   NS:=1;
   FOR I:=1 TO M DO
   BEGIN
      SORT[I]:=0;
      POINTERS[I]:=M-I+1;
   END;

   WHILE LAST>1 DO
   BEGIN
      SHAFTNUM[POINTERS[LAST]]:=SHAFT;
      ROWSORT[SHAFT]:=POINTERS[LAST];
      R1:=LAST;
      BUCKET:=SORT[R1];
      LAST:=LAST-1;
      R2:=LAST;
      CONDITION:=(BUCKET=SORT[R2]) AND (R2>0);
      WHILE CONDITION DO
      BEGIN
         P1:=POINTERS[R1];
         P2:=POINTERS[R2];
         IF MATCH(BUCKET,N,M,P1,P2,ROW,BNEW,INTARRAY,ROWSORT)   THEN
         BEGIN
            SHAFTNUM[POINTERS[R2]]:=SHAFT;
            LAST:=LAST-1;
            SHUFFLE(SORT,R2,LAST+1);
            SHUFFLE(POINTERS,R2,(LAST+1));
```

```
          END
          ELSE INSERT(POINTERS,SORT,BNEW,R2,LAST);
          R2:=R2-1;

          IF R2=0 THEN CONDITION:=FALSE
          ELSE CONDITION:=BUCKET=SORT[R2];
       END;
       SHAFT:=SHAFT+1;
    END;
    IF LAST=1 THEN
    BEGIN
       SHAFTNUM[POINTERS[LAST]]:=SHAFT;
       ROWSORT[SHAFT]:=POINTERS[LAST];
       NS:=SHAFT;
    END
    ELSE NS:=SHAFT-1;
END;


PROCEDURE TIE(VAR INTARRAY,TIEUP:SIZE;SHAFTNUM,ROWSORT,POINTERS:
VECSIZE;NS,LAST,TREADLE  :INTEGER);

(* THIS PROCEDURE DETERMINES THE TIE-UP MATRIX CORRESPONDING TO A GIVEN BINARY
INTERLACEMENT ARRAY *)

VAR
    I:INTEGER;

BEGIN
    FOR I:=1  TO NS DO
    TIEUP[NS+1-SHAFTNUM[ROWSORT[I]],TREADLE]:=INTARRAY[POINTERS
    [LAST],ROWSORT[I]]
END;


PROCEDURE TREADAN(VAR INTARRAY,TIEUP:SIZE;VAR  TREADNUM:VECSIZE; VAR
NT:INTEGER;ROWSORT:VECSIZE;N: INTEGER);

(* THIS PROCEDURE DETERMINES  THE SHED SEQUENCE MATRIX CORRESPONDING TO A GIVEN
BINARY INTERLACEMENT ARRAY. *)

VAR
    POINTERS,SORT:VECSIZE;
    BNEW,BUCKET,I,R1,R2,LAST,P1,P2,TREADLE:   INTEGER;
    CONDITION,ROW: BOOLEAN;

BEGIN
    TREADLE:=1;
```

```
  R2:=N-1;
  LAST:=N;
  ROW:= TRUE;
  NT:=1;
  FOR I:=1 TO N DO

  BEGIN
     SORT[I]:=0;
     POINTERS[I]:=N-I+1;
  END;
  WHILE LAST>1 DO
  BEGIN
     TREADNUM[POINTERS[LAST]]:=TREADLE;
     TIE(INTARRAY,TIEUP,SHAFTNUM,ROWSORT,POINTERS,NS,LAST,TREADLE);
     R1:=LAST;
     BUCKET:=SORT[R1];
     LAST:=LAST-1;
     R2:=LAST;
     CONDITION:=(BUCKET=SORT[R2]) AND (R2>0);
     WHILE CONDITION DO
     BEGIN
        P1:=POINTERS[R1];
        P2:=POINTERS[R2];
        IF MATCH(BUCKET,N,NS,P1,P2,ROW,BNEW,INTARRAY,ROWSORT)   THEN
        BEGIN
           TREADNUM[POINTERS[R2]]:=TREADLE;
           LAST:=LAST-1;
           SHUFFLE(SORT,R2,LAST+1);
           SHUFFLE(POINTERS,R2,LAST+1);
        END
        ELSE INSERT(POINTERS,SORT,BNEW,R2,LAST);
        R2:=R2-1;
        IF R2=0 THEN CONDITION:= FALSE
        ELSE CONDITION:=BUCKET=SORT[R2];
     END;
     TREADLE:=TREADLE+1;
  END;
  IF LAST=1 THEN
  BEGIN
     TREADNUM[POINTERS[LAST]]:=TREADLE;
     NT:=TREADLE;
     TIE(INTARRAY,TIEUP,SHAFTNUM,ROWSORT,POINTERS,NS,LAST,TREADLE);
  END
  ELSE NT:=TREADLE-1;
END;
```

3.5.5 THE ALTERNATING DIRECTION ALGORITHM

The Bucket Sort Algorithm represents a considerable saving over the Classical Algorithm in terms of the time required to factor a given binary interlacement array. As will be shown in Section [3.5.6], the largest reduction in the number of operations performed appears in the second phase of the algorithm, when the shed sequence and tie-up matrices are being determined. The greatest reduction in the number of operations required to determine the threading matrix can be realized when the buckets are well distributed at each stage. In a worst case situation all of the columns of the interlacement array which are being examined at any stage lie in the same bucket. In this instance, the Bucket Sort Algorithm threading computation becomes identical to the threading determination process used in the Classical Algorithm.

The Alternating Direction Algorithm performs the first phase of the factorization process (column distinctness determination) using the Bucket Sort Algorithm. This determination of the threading is not however, continued to completion but rather, is suspended at the k^{th} stage. In other words, only k columns of the interlacement array are assigned threadings. As before, any columns within these k columns which are identical are eliminated from consideration. At this point, processing switches to the shed sequence determination stage (row distinctness determination) of the Bucket Sort Algorithm and continues to the r^{th} stage.

Threading determination now resumes on the array with identical rows excluded. This alternation of processing continues until the threading, shed sequence and tie-up matrices have been computed in their entirety.

Specifically, the basic alternating direction algorithm is given by the following steps:

1. Determine the entries in the threading matrix for k_1 columns. These will be the first k_1 columns encountered in the execution of the Bucket Sort Algorithm and will not necessarily be contiguous.

2. Determine row distinctness over those columns which have been determined to be distinct and the $m - k_1$ columns not completely analyzed, for the first r_1 rows encountered in the execution of the Bucket Sort Algorithm.

3. Make the corresponding entries in the r_1 rows of the shed sequence matrix.

4. For each distinct row encountered, compute the corresponding column of the tie-up matrix.

5. Resume threading determination at the $k_1 + 1^{st}$ stage. Determine column distinctness over all of the distinct rows encountered in step 4 and all of the $n - r_1$ rows which were not completely analyzed. Continue this process to the k_2^{th} stage, when k_2 entries of the threading matrix have been determined.

6. Repeat steps 4 and 5 until the entire binary interlacement array has been factored into its corresponding threading, tie-up and shed sequence matrices.

In the algorithm which has just been described, the various values of k and r are not chosen with respect to the data being processed but rather, are selected a priori. A k-vector and r-vector where the elements are multiples of 10 is one example of such a scheme. The threading analysis continues until threading entries are assigned for 10 columns, at which time the shed sequence analysis begins. After 10 rows of the shed sequence matrix have been determined, processing of the columns resumes and continues until an additional 10 columns have been completely analyzed, and so on until all of the columns and all of the rows have been examined. At each stage of course, the analysis takes place over a sub-matrix of the previous interlacement array.

The greatest homogeneity of processing is obviously achieved by a one - one alternation between column and row processing, where the dimensions of the active interlacement array decrease by one every time that a row or column is determined to be equivalent to a previously examined row or column.

An alternative form of this algorithm has the values of k and r being determined dynamically based on the character of the interlacement data. One particular variation of this approach is given by the following steps:

1. Compare the first column with every other column, up to the point where they disagree. This will establish the coarse buckets.

2. Choose the first bucket and determine the threading for all of the columns in it.

3. Compare the first row with every other row, up to the point where they disagree and ignoring positions corresponding to non-distinct columns. This will establish the coarse buckets for the rows.

4. Choose the first bucket and determine the treadles for all of the

rows in it.

5. At the same time, determine the corresponding tie-up entries for every distinct row of the shed sequence matrix.

6. Resume examination of the columns. Determine the threading for all of the columns in the second bucket.

7. Determine the shed sequence for all of the rows in the second row bucket and make the appropriate entries in the tie-up matrix.

8. Repeat steps 6 and 7 until the binary interlacement array has been completely factored into its threading, tie-up and shed sequence components.

The Alternating Direction Algorithm can provide a high degree of homogeneity of processing. This may be desirable if, for example, processing is not to continue if the number of shafts or treadles required exceeds a given value. In the case where the number of shafts used is small relative to the number of treadles, phase 1 of the Bucket Sort Algorithm (the time-consuming stage) would be executed in its entirety and the factorization attempt would be aborted part of the way through the second phase of the algorithm. Using the Alternating Direction

Algorithm, the end condition on the shed sequence matrix would be detected at an earlier stage.

It should be noted however that, if the factorization process is to be continued to completion, the Alternating Direction Algorithm represents <u>no</u> saving in operations over the Bucket Sort Algorithm. This is illustrated in Section (3.5.7) where a test case, constructed to give the maximum advantage to the Alternating Direction Algorithm, is examined.

3.5.6 THE MINIMAL BUCKET SORT ALGORITHM

Although the Bucket Sort Algorithm represents a substantial saving of operations, particularly in the determination of the shed sequence matrix, this saving is still not maximal. Having compared the columns of an interlacement array and determined the number of leading positions of agreement, this information can be used to eliminate row-wise comparisons. Specifically, we have the following results:

Definition 3.5.6.1. Two columns are said to be k - equivalent if they agree in the first k positions.

THEOREM 3.5.6.2. If n columns are k - equivalent then the elements in n - 1 of these columns need not be examined in the comparison of the first k rows.

Proof. Without loss of generality, we can consider the n k-equivalent columns to be contiguous. The matrix with these columns will be of the form

$$\begin{array}{c}
|\!\longleftarrow n \longrightarrow\!| \\
\left.\begin{array}{cccccc}1&1&1&1&1&1\\2&2&2&2&2&2\\ \cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\ k&k&k&k&k&k\end{array}\right\updownarrow k \\
\begin{array}{cccccc}*&*&*&*&*&*\\ \cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\ *&*&*&*&*&*\end{array}
\end{array}$$

where all elements represented by the same integer i ($1 \leq i \leq k$) are the same, and where (*) indicates elements about which we have no information. Clearly, all of these columns are identical up to and including the k^{th} element, and only one of these columns need be considered with respect to the equivalence of the first k rows. □

If the complete Bucket Sort Algorithm for the columns has been applied then the columns are sorted as follows:

<u>Definition 3.5.6.3</u>. The Bucket Sort Algorithm <u>k-sorts</u> the columns in the sense that:

(i) all distinct columns are adjacent;

(ii) all columns which are k-equivalent with the first column are contiguous;

(iii) each set of k-equivalent columns is sorted with respect to the first column of the k-equivalent set, etc.

<u>THEOREM 3.5.6.4</u>. At the termination of the column analysis, the bucket value for each of the distinct columns, with the exception of the first column examined in each set of k-equivalent columns, contains the number of leading elements in the column which have been accessed.

<u>Proof</u>. If, in the comparison of one column with any other column, a disagreement occurs then the last position of agreement is recorded as the bucket value for each column. Since we are only considering the distinct columns, this value will always be less than the length of the column. □

<u>COROLLARY 3.5.6.5</u>. At the termination of the Bucket Sort Algorithm for the columns, the first column of each k-equivalent set of columns will have a bucket value corresponding to the last position in which it agreed with the column to which it was found to be k-equivalent.

<u>COROLLARY 3.5.6.6</u>. The very first column which is examined in the Bucket Sort Algorithm for the columns will have a final bucket value of 0.

<u>COROLLARY 3.5.6.7</u>. The final bucket vector gives the number of rows for which each column can be ignored in performing the Bucket Sort Algorithm for the rows.

An algorithm which utilizes these results involves the following steps:

1. Perform the Bucket Sort Algorithm on the columns of a binary interlacement array D;
2. compare the second row with the first row, only over the distinct columns, only in positions corresponding to bucket values less than two, and only until a disagreement is encountered;
3. set the bucket value for the second row to the last position of agreement with row one;
4. determine the appropriate entries in the tie-up matrix for each distinct row, as in the Bucket Sort Algorithm;
5. repeat steps two, three and four until the shed sequence and tie-up matrices has been completely determined.

An example of this algorithm is discussed in detail in Section (3.5.7).

3.5.7 THEORETICAL COMPARISONS

The factorization algorithms discussed in this section have been presented in order of increased potential efficiency. It is clear however, that the efficiency of any of the algorithms is very data specific and that, in some circumstances, two algorithms do in fact become identical. For that reason, any analysis of the relative efficiency of these processes must be based on examination of specific non-trivial test cases.

The following four test cases have been constructed to present the the factorization algorithms with worst case situtations, so as to ensure that these algorithms retain their distinct character.

TEST CASE 1:

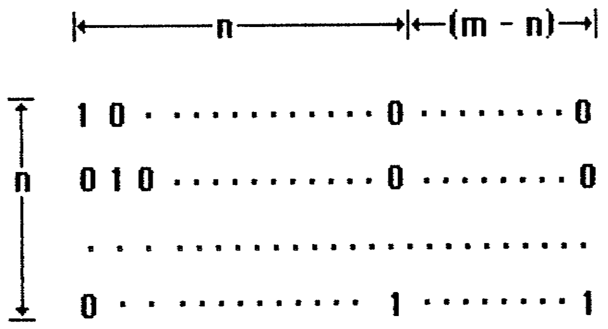
Let A be an $n \times m$ array with $m \geq n$ and such that

(1) A is symmetric (excluding the $(m - n)$ last columns of A);

(2) $m - n + 1$ columns are identical;

(3) the columns of A, c_i, are such that, if c_i and c_j are columns with $i < j$ and $j \leq n$, then c_i is identical with c_j to $(i - 1)$ places.

EXAMPLES:



GENERAL

$$\begin{array}{cccccccc}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&1&1&1\end{array}$$

SPECIFIC
(n=5, m=8)

<u>TEST CASE 2:</u>

Let A be an $n \times m$ array with $m \geq n$ and such that

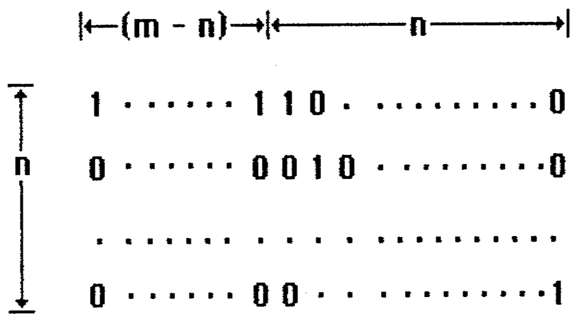
(1) $m - n + 1$ columns are identical;

(2) the columns of A are such that, if c_i and c_j are columns with $i < j$ and $i,j > (m - n)$, then c_i is identical with c_j to $(i - (1 + m - n))$ places;

(3) the n rows of A, r_i, are distinct and such that, if r_i and r_j are rows with $1 < i < j$, then r_i is identical with r_j to $(i - 1 + (m - n))$ places;

(4) r_1 is identical with all r_j to 0 places.

EXAMPLES:



GENERAL

$$\begin{matrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

SPECIFIC
($n=5$, $m=8$)

<u>TEST CASE 3:</u>

Let A be an $n \times m$ array with $m \geq n$ and such that

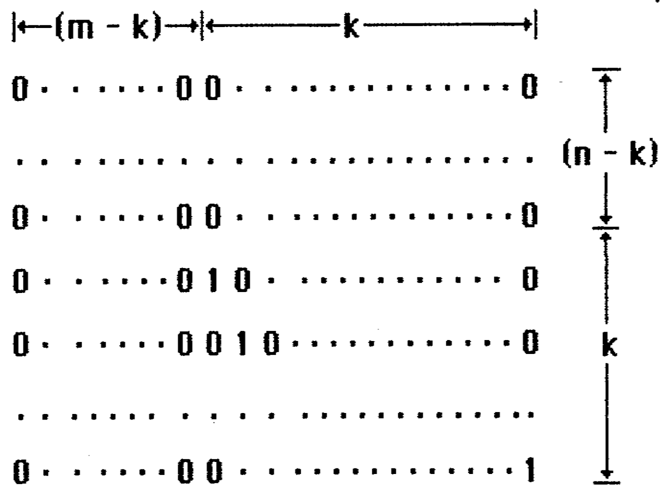
(1) $m - k$ columns are identical;

(2) $n - k$ rows are identical;

(3) the columns of A are such that, if c_i and c_j are columns and $i < j$ and $i > (m - k)$, then c_i is identical with c_j to $(i - (1+m-n))$ places;

(4) the rows of A are such that, if r_i and r_j are rows and $i < j$ and $i > (n - k)$, then r_i is identical with r_j to $(i - 1 + (m-n))$ places.

EXAMPLES:



GENERAL

$$\begin{matrix}
0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0\\
0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&1
\end{matrix}$$

SPECIFIC
(n=6, m=8, k=4)

TEST CASE 4:

Let A be an $n \times m$ array with $n \geq m$ and such that

(1) all m of the columns are distinct;

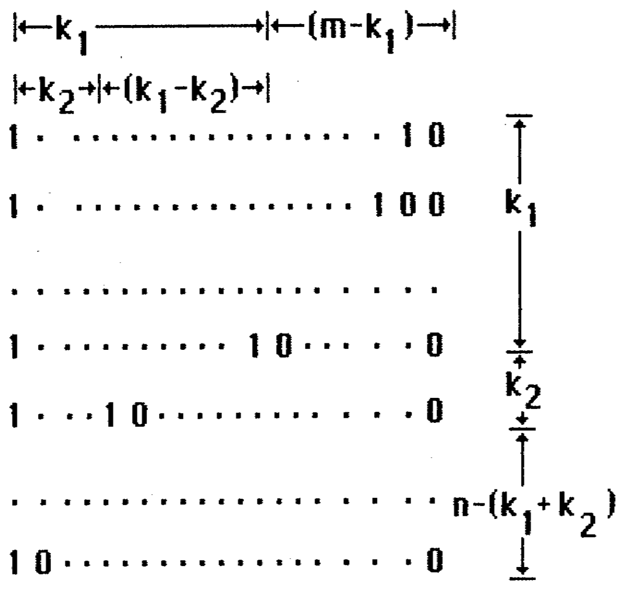
(2) the first k_1 of the columns agree to k_1 positions;

(3) of these k_1 columns, the first k_2 of them agree to an additional k_2 positions;

(4) $(m - k_1)$ of the columns agree to k_i positions, with $i < k_1$;

(5) the $(m - k_1)$ remaining columns are sorted in descending order of the value of k_i.

EXAMPLES:



GENERAL

$$\begin{matrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

SPECIFIC
($m=6$, $n=7$, $k_1=3$, $k_2=2$)

The number of operations required in performing the Mathematical Algorithm, the Classical Algorithm and the Bucket Sort Algorithm on the interlacement array given by Test Case 1 are now determined, both for the general and for the specific examples. Since the tie-up matrix determination is constant in all of these algorithms, the number of operations required to compute this array are not included.

MATHEMATICAL ALGORITHM:

Part 1 - Threading Determination:

step i: compares column i with all others.

* of comparisons in general and specific examples.

step 1:	$n \times (m - 1)$	35
step 2:	$n \times (m - 2)$	30
.		
.		
.		
step n:	$n \times (m - n)$	15

At the nth step all of the remaining columns are determined to be identical.

Total:	$n^2 \times (2m - n - 1)/2$	125.

Part 2 - Shed Sequence Determination:

step i:	compares row i with all others.

* of comparisons in general and specific examples.

step 1:	$m \times (n - 1)$	32
step 2:	$m \times (n - 2)$	24
.		
.		
.		
step n - 1:	$m \times 1$	8
Total:	$m \times n (n - 1)/2$	80.

Total number of operations - part 1 and part 2:

$(3mn^2 - n (m + n^2 + n))/2$ 205.

CLASSICAL ALGORITHM:

Part I - Threading Determination:

step i: compares column i with all others

# of comparisons in general and specific examples

step 1:	$(m - 1)$	7
step 2:	$2 \times (m - 2)$	12
.		
.		
.		
step n - 1:	$(n - 1) \times (m - n + 1)$	16
step n:	$n (m - n)$	15

At the n^{th} step all of the remaining columns are determined to be identical to column n.

Total:	$(3mn - 2n^2 - n)(n + 1)/6$	65.

Part 2 - Shed Sequence Determination:

step i: compares row i with all others.

# of comparisons in general and specific examples.

step 1:	$(n - 1)$	4
step 2:	$2 \times (n - 2)$	6
.		
.		
.		
step $n - 1$:	$(n - 1)$	4
Total:	$(n^2 - n)(n + 1)/6$	20.

Total number of operations - part 1 and part 2:

$n(n + 1)(3m - n - 2)/6$ 85.

BUCKET SORT ALGORITHM:

Part 1 - Threading Determination:

step i: compares column i with all others.

* of comparisons in general and specific examples.

step 1:	$(m - 1)$	7
step 2:	$(m - 2)$	6
.		
.		
.		
step $n - 1$:	$(m - n + 1)$	4
step n:	$(m - n)$	3

At the nth step all of the remaining columns are determined to be identical to column n.

Total:	$mn - n(n + 1)/2$	25.

Part 2 - Shed Sequence Determination:

step i: compares row i with all others.

* of comparisons in general and specific examples.

step 1:	$(n - 1)$	4
step 2:	$(n - 2)$	3
.		
.		
.		
step $n - 1$:	1	1
Total:	$n(n - 1)/2$	10.

Total number of operations - part 1 and part 2:

$n(m - 1)$	35.

Summary and Comparison for Test Case 1:

Mathematical Algorithm:

Part 1:	$n^2(2m - n - 1)/2$	125
Part 2:	$mn(n - 1)/2$	80.

The difference in the number of operations for part 1 and part 2 of this algorithm is based solely on the difference in size between m and n, as illustrated when we set $m = n$.

Part 1:	$n^2(n - 1)/2$	50
Part 2:	$n^2(n - 1)/2$	50.

Total number of operations – part 1 and part 2:

$(3mn^2 - mn - n^3 - n)/2$	205.

Classical Algorithm:

Part 1:	$(3mn - 2n^2 - n)(n + 1)/6$	65
Part 2:	$(n^2 - n)(n + 1)/6$	20.

For this test case, the difference in the number of operations required in parts 1 and 2 of the Classical Algorithm is based solely on the difference in size between m and n, as illustrated when we set $m = n$.

Part 1:	$(n^2 - n)(n + 1)/6$	20
Part 2:	$(n^2 - n)(n + 1)/6$	20.

Total number of operations - part 1 and part 2:

$(3mn^2 + 3mn - n^3 - 3n^2 - 2n)/6$	85.

Clearly the ratio of the number of operations performed in the Mathematical Algorithm to the number of operations performed in the Classical Algorithm is d:1, where d is asymptotically 3. The ratio in the specific example is 205/85, or 2.4.

Bucket Sort Algorithm:

Part 1:	$mn - n(n + 1)/2$	25
Part 2:	$n(n - 1)/2$	10.

For this test case, the difference in the number of operations required in part 1 and part 2 of the Bucket Sort Algorithm is also based solely on the difference between m and n, as illustrated when we set $m = n$.

Part 1:	$n(n - 1)/2$	10
Part 2:	$n(n - 1)/2$	10.

Total number of operations - part 1 and part 2:

$n(m - 1)$ 35.

The Classical Algorithm has a dominant term of $(3mn^2 - n^3 + 3mn - 3n^2)/6$, while the Bucket Sort Algorithm has a dominant term mn. Clearly the ratio of the number of operations in the Classical Algorithm to the number of operations in the Bucket Sort Algorithm will be greater than 1, and will increase with n. The ratio in the specific example is 85 to 35, or 2.4.

The number of operations (comparisons) performed in the analysis of the binary interlacement array of Test Case 2 is now considered with respect to the Classical and Bucket Sort Algorithms. The Mathematical Algorithm is dependent only on the number of rows and columns in the matrix and takes no account of the data itself. For this reason, it is not included in the comparison of this, or the next test case.

NUMBER OF COMPARISONS
TEST CASE 1
SPECIFIC EXAMPLE

	ALGORITHM		
	MATHEMATICAL	CLASSICAL	BUCKET
PART 1	125	65	25
PART 2	80	20	10
TOTAL	205	85	35

TEST CASE 2:

CLASSICAL ALGORITHM:

Part 1 - Threading Determination:

step 1_r:	step 1 compares column r+1 with all others (r=m-n).

# of comparisons in general and specific examples.

step 1_0:	$(m - n)n + (n - 1)$	19
step 2_r:	$2(n - 2)$	6
step 3_r:	$3(n - 3)$	6
.		
.		
.		
step $(n - 1)_r$:	$1(n - 1)$	4
Total:	$(6mn + n^3 - 6n^2 - n)/6$	35.

Part 2 - Shed Sequence Determination:

step i: compares row i with all others.

* of comparisons in general and specific examples.

step 1:	$n - 1$	4
step 2:	$(n - 2)(m - n + 2)$	15
.		
.		
.		
step n - 1	$m - 1$	7
Total:	$((n - 2)(2n + nm - m)/2) - (n(2n^2 - 3n - 5)/6)$	38.

Total number of operations - part 1 and part 2:

$$(n(-n^2 - 3n + 6m + 4)/6) + (n - 2)(2n + nm - m)/2 \qquad 73.$$

BUCKET ALGORITHM:

Part 1 - Threading Determination:

step i_r: step i compares column i+r with all others (r=m-n).

# of comparisons in general and specific examples.

step 1_0:	$(m - n)\,n + (n - 1)$	19
step 2_r:	$(n - 2)$	3
.		
.		
.		
step $(n - 1)_r$:	1	1
Total:	$(m - n)\,n + (n - 1)\,n/2$	25.

Part 2 - Shed Sequence Determination:

step i: compares row i with all others ignoring the first (m-n) columns.

# of comparisons in general and specific examples.

step 1:	(n - 1)	4
step 2:	(n - 2)	3
.		
.		
.		
step n - 1	1	1
Total:	n (n - 1)/2	10.

Total number of operations - part 1 and part 2:

n (m - 1)	35.

Summary and Comparisons for Test Case 2:

Classical Algorithm:

$1/2 [(n^2 - n + 2) m - (n^2 - 5 n - 6) n]$ 73

Bucket Sort Algorithm:

$n (m - 1)$ 35.

For $n \geq 5$, the ratio between the number of operations required in the Classical Algorithm to the number of operations required in the Bucket Sort Algorithm is ≥ 2. The ratio in the specific example is 73/35, or 2.1.

NUMBER OF COMPARISONS
TEST CASE 2
SPECIFIC EXAMPLE

	ALGORITHM	
	CLASSICAL	BUCKET
PART 1	35	25
PART 2	38	10
TOTAL	73	35

The Bucket Sort and Alternating Direction Algorithms are examined in terms of the number of operations required to factor the binary interlacement array represented by Test Case 3 into its constituent threading and shed sequence matrices.

TEST CASE 3:

BUCKET SORT ALGORITHM:

Part 1 - Threading Determination:

step i: compares column i with all others

* of comparisons in general and specific examples.

step 1: $n(m - 1) - (k/2)(k - 1)$ 36

At this point, the first $(m - k)$ columns are found to be equivalent, while each of the remaining columns is in a separate bucket and is therefore distinct.

Total: $n(m - 1) - (k/2)(k - 1)$ 36.

Part 2 - Shed Sequence Determination:

step i:	compares row i with all others, ignoring the first (m-k-1) columns	
step 1:	$[(2n - k)(k + 1)/2] - 1$	19

At this point, the first $(n - k)$ rows are found to be equivalent, while each of the remaining rows is in a different bucket and is therefore distinct.

Total:	$[(2n - k)(k + 1)/2] - 1$	19.

Total number of operations - Part 1 and Part 2:

$$n(m + k) - (k^2 + 1) \qquad 55.$$

ALTERNATING DIRECTION ALGORITHM:

Part 1 - Threading Determination:

# of comparisons in general and in specific examples.

step 1: compares first column with all others

$n(m-1) - k(k-1)/2$ 36

step 2 Compare first row with all others, ignoring the first $(m-k-1)$ columns

$[(2n-k)(k+1)/2] - 1$ 19

At this point, the equivalent rows and columns have been identified while all of the remaining rows and columns lie in different buckets and are therefore distinct.

Total: $n(m+k) - (k^2+1)$ 55.

NUMBER OF COMPARISONS
TEST CASE 3
SPECIFIC EXAMPLE

	ALGORITHM	
	BUCKET	ALTERNATING DIRECTION
PART 1	36	36
PART 2	19	19
TOTAL	55	55

The Bucket Sort and Minimal Bucket Sort Algorithms are now considered with respect to the interlacement array represented by Test Case 4. However, since the character of this test case depends critically on the specific elements in all of the positions, the specific case only will be analyzed as a representative of this type of array.

<u>TEST CASE 4:</u>

BUCKET SORT ALGORITHM:

Part 1 - Threading Determination:

# of comparisons in general and specific examples.

step 1:	compares first column with all others
	16

At this point, all of the remaining columns are different buckets.

Total:	16

Part 2 - Shed Sequence Determination:

step i:	compares row i with all others.

# of comparisons in general and specific examples.

step 1:	6
step 2:	14
step 3:	1
Total:	21.

Total number of operations - Part 1 and Part 2:

37.

MINIMAL BUCKET SORT ALGORITHM:

Part 1 of the Minimal Bucket Sort Algorithm is identical to the Bucket Sort Algorithm, so that the number of operations required for Part 1 is 16.

Part 2 - Shed Sequence Determination:

# of comparisons in general and specific examples.

step 1:	compares first row with all other rows
	6
step 2:	compares second row with all other rows r_i, $3 \leq i \leq 7$, only in positions corresponding to columns whose bucket value b_j is less than i
	10
step 3:	compares fifth and seventh rows
	1
Total:	17.

Total number of operations - Part 1 and Part 2:

33.

Clearly, the Minimal Bucket Sort Algorithm requires fewer operations than the Bucket Sort Algorithm for this test case and this increased efficiency is realized entirely in the performance of the shed sequence factorization. The ratio of operations for the Bucket Sort and the Minimal Bucket Sort Algorithms for Part 2 is 21/17, or 1.2, in this specific case. Since the Minimal Bucket Sort Algorithm is so data specific however, it is not possible to generalize the degree of improved efficiency. The only claim which can be made is that there exist binary interlacement arrays for which this algorithm represents a theoretical improvement in the amount of processing required and in all other cases it is no worse, since it includes the Bucket Sort Algorithm as a special case.

NUMBER OF COMPARISONS
TEST CASE 4
SPECIFIC EXAMPLE

	ALGORITHM	
		MINIMAL
	BUCKET	BUCKET
PART 1	16	16
PART 2	21	17
TOTAL	37	33

3.5.8 <u>PRACTICAL CONSIDERATIONS</u>

Although the Mathematical Algorithm is of some interest because of its simplification of the factorization problem, it is not a reasonable algorithm to be applied, either by hand or by machine. As shown in Section [3.5.7], the Classical Algorithm represents a considerable saving in the number of operations, while still maintaining a relatively simple form to the process. The Bucket Sort Algorithm represents a further saving still, but the processing becomes considerably more complex.

In order to ascertain whether or not the overhead involved in the complexity of the Bucket Sort Algorithm would eliminate the theoretical advantage to be realized, the Bucket Sort Algorithm was implemented in Applesoft Basic and run on an Apple 2+ microcomputer. Timing comparisons were made between this algorithm and a similar implementation of the Classical Algorithm, with respect to the binary interlacement arrays shown in Figures [3.5.8.1] and [3.5.8.2]. These arrays were selected as being representative of typical multi-shaft woven structures. The results are summarized as follows:

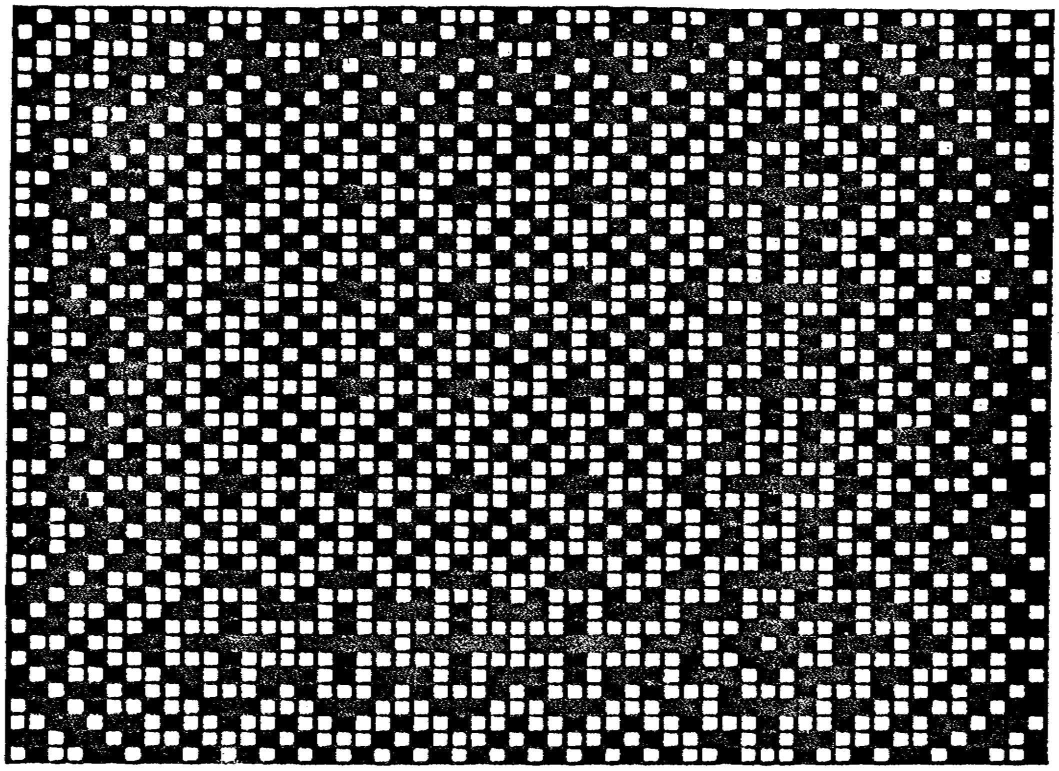


FIGURE 3.5.8.1

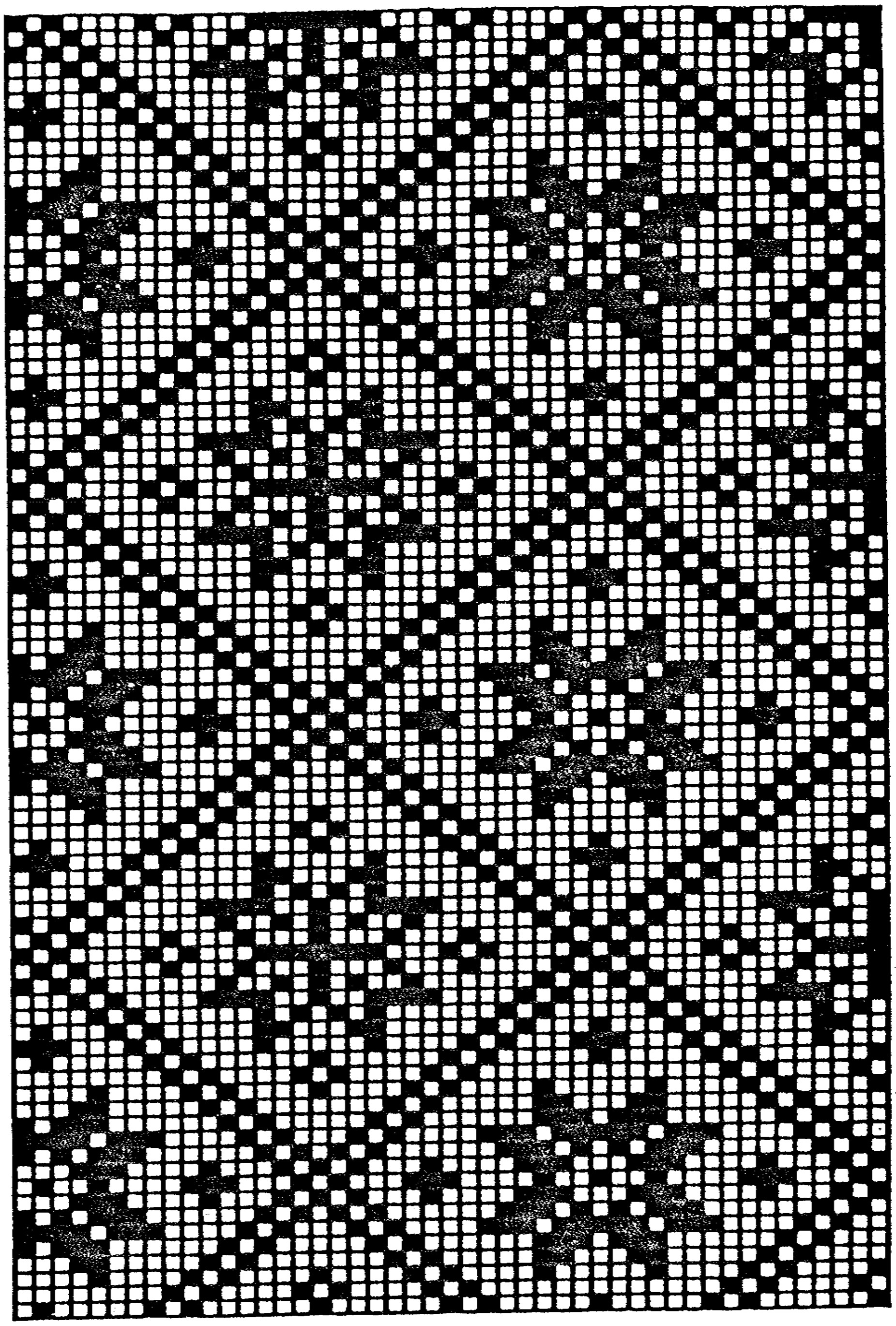


FIGURE 3.5.8.2

FIGURE	ARRAY SIZE	TIME (MIN. : SEC.)					
		CLASSICAL ALGORITHM			BUCKET SORT ALGORITHM		
		1	2	TOTAL	1	2	TOTAL
3.5.8.1	8 x 8	:05	:29	:34	:04	:08	:12
	16 x 16	:25	3:12	3:37	:10	:30	:40
	32 x 32	1:08	6:37	7:45	:37	:52	1:29
	47 x 84	3:54	17:19	21:13	2:41	1:12	3:53
3.5.8.2	12 x 12	:15	1:25	1:40	:07	:18	:25
	27 x 27	:53	3:30	4:23	:28	:32	1:00
	54 x 54	2:51	11:59	14:50	1:48	1:17	3:05

where 1 indicates Part 1 (threading determination) of the algorithm and 2 indicates Part 2 (shed sequence determination) of the algorithm plus the tie-up computation.

Clearly, the Bucket Sort Algorithm exhibits a practical as well as a theoretical, advantage over the Classical Algorithm.

The Alternating Direction Algorithm is identical with the Bucket Sort Algorithm with the exception that processing switches back and forth between Parts 1 and 2. The speed of processing should thus be comparable in these two situtations, except that the switching overhead is included in the Alternating Direction Algorithm. This is not great but, since the Alternating Direction has no advantage except in partial processing, this algorithm is actually less efficient if the entire array is always going to be processed.

The Minimal Bucket Sort Algorithm is never theoretically less efficient than the Bucket Sort Algorithm. However, the structure of this algorithm is so complex that it is not clear that these theoretical savings can in fact be realized.

3.6 FACTORING A COLOURED INTERLACEMENT ARRAY

Normally the analysis of a woven textile fragment begins with the development of the corresponding binary interlacement array, with some notation being made as to the colours of the various warp and weft yarns. At this point, any of the preceding factorization algorithms can be applied to the interlacement array to determine the appropriate threading, tie-up and shed sequence matrices. It should be recalled that this factorization is unique modulo permutation of the rows of the threading and tie-up matrices and permutation of the columns of the shed sequence and tie-up matrices.

Sometimes however, a given coloured design is to be analyzed to determine how, or in fact whether it can be produced on a loom. This design then must correspond to a multi-valued array which is a coloured interlacement array if and only if it can be decomposed into a vector of warp colours, a vector of weft colours and a binary interlacement array, all of which are self-consistent. In other words, we require that the resulting factors reproduce the original coloured design when the coloured interlacement array computation algorithm of Section (3.3) is applied to them.

The first stage of this analysis therefore involves determining a vector of warp colours (WARP), a vector of weft colours (WEFT) and a binary interlacement array ($D = \{d_{i,j}\}$) corresponding to a given colour interlacement array ($K = \{k_{i,j}\}$). Implicit in this description is the verification that the array K is, indeed a colour interlacement array. We will make use of the following results.

<u>THEOREM 3.6.1</u>. At least one unambiguous warp and weft colouring can always be determined for any given coloured interlacement array.

<u>Proof</u>. Let us assume that the n x m array K under consideration is a coloured interlacement array. We can now assume that weft colours will appear at least once per row and that warp colours will appear at least once per column. □

The following finite series of steps will produce the required vectors of colours (encoded as integers):

1. Set $WARP_j = K_{1,j}$ for all columns where $K_{1,j} = K_{i,j}$, for all i.
2. Set $WEFT_i = K_{i,1}$ for all rows where $K_{i,1} = K_{i,j}$, for all i.
3. Terminate the process if the WARP and WEFT vectors are complete.
4. Choose a value for j (j = 1,2,...,m) such that there exists some $K_{i,j} \neq K_{1,j}$ (i = 1,2,...,n).

5. Let $K_{i,j}$ be a warp over weft intersection and set

   $WARP_j = K_{i,j}$.

6. All $K_{i,j} \neq WARP_j$ are weft over warp intersections. Set

   $WEFT_i = K_{i,j}$ for all of these values of i.

7. For all $WEFT_i$ determined in step 6, locate all $K_{i,j} \neq WEFT_i$

   and set $WARP_j$ equal to these $K_{i,j}$.

8. Repeat steps 6 and 7 until all values for WARP and WEFT have been defined.

* If, at any point in this process, an inconsistency develops, the array K <u>is not</u> a coloured interlacement array.

<u>THEOREM 3.6.2</u>. Any rectangular region in a coloured interlacement array corresponding to intersections between warp and weft yarns of the same colour is structurally indeterminate.

<u>Proof</u>. The coloured interlacement array represents the colour which appears at a given intersection. If the warp and weft yarns are of the same colour, then it is impossible to tell whether this is a warp over weft or weft over warp intersection. □

<u>THEOREM 3.6.3</u>. The representation of the intersection between individual warp and weft strands remains invariant under row and column

permutations of the binary interlacement array.

<u>COROLLARY 3.6.4</u>. The colour representation of the intersection between individual warp and weft strands remains invariant under row and column permutations of the coloured interlacement array.

<u>EXAMPLE 3.6.5</u>.

This example shows a coloured interlacement array K with two colours (encoded as 2 and 3) and its corresponding warp and weft colour vectors. The same array is also shown with its rows and columns permuted, with corresponding changes being made to the colour vectors. It is clear that, although the colour <u>sequences</u> have changed, the interlacement relationship between <u>individual</u> warp and weft strands has not.

$$
K = \begin{array}{cccccccccc}
3&3&2&2&3&3&3&3&3&3\\
3&3&2&3&3&3&3&2&3&3\\
3&3&3&3&3&3&3&3&3&3\\
3&3&3&2&3&3&2&3&3&3\\
2&2&2&2&3&2&2&2&3&3\\
2&3&2&2&3&3&2&2&3&2\\
3&3&2&2&2&3&2&2&2&2\\
3&3&3&2&3&3&2&3&3&3\\
2&3&2&2&2&3&2&2&2&2\\
2&3&2&2&3&3&2&2&3&2\\
2&2&2&2&2&3&2&2&2&3
\end{array}
\qquad
\mathrm{WEFT} = \begin{array}{c}
3\\3\\3\\3\\2\\2\\2\\3\\2\\2\\2
\end{array}
$$

$$
\mathrm{WARP} = \begin{array}{cccccccccc}
3&3&2&2&3&3&2&2&3&3
\end{array}
$$

$$K' = \begin{array}{cccccccccc}
3&3&3&3&3&3&2&2&3&3\\
3&3&3&3&3&3&2&3&3&2\\
3&3&3&3&3&3&3&3&3&3\\
3&3&3&3&3&3&3&2&2&3\\
3&3&3&3&3&3&3&2&2&3\\
2&2&3&2&3&3&2&2&2&2\\
2&3&3&3&3&2&2&2&2&2\\
3&3&2&3&2&2&2&2&2&2\\
2&3&2&3&2&2&2&2&2&2\\
2&3&3&3&3&2&2&2&2&2\\
2&2&2&3&2&3&2&2&2&2
\end{array} \qquad WEFT' = \begin{array}{c}
3\\3\\3\\3\\3\\2\\2\\2\\2\\2\\2
\end{array}$$

$$WARP' = \begin{array}{cccccccccc}
3&3&3&3&3&3&2&2&2&2
\end{array}$$

<u>THEOREM 3.6.6</u>. A coloured interlacement array is structurally determinate if and only if the sets of warp and weft colourings are disjoint, as determined by the algorithm of Theorem (3.6.1).

<u>Proof</u>. Let us first assume that the sets of warp and weft colourings are disjoint; that is that there are no weft strands of the same colour as any of the warp strands. With no loss of generality, we can substitute values of 1 in the coloured interlacement array wherever a warp colour appears at a given intersection and values of 0 where a weft colour appears. The resulting matrix is a binary interlacement array.

Let us next assume that the sets of warp and weft colours for a given coloured interlacement array K are not disjoint. By Corollary (3.6.4), we can permute the rows and columns of K to produce a coloured array K' with solid coloured rectangular blocks lying on the principal diagonal. These regions correspond to the intersections of warp and weft yarns of the same colour and, from Theorem (3.6.2), we know that these regions are structurally indeterminate. Therefore our original coloured interlacement array K is also structurally indeterminate. □

Based on the preceding results, we now have an algorithm for factoring coloured interlacement arrays, as follows:

1. Obtain the warp and weft colour vectors, using the process outlined in Theorem (3.6.1).
2. If the sets of warp and weft colours are disjoint, set elements in the coloured interlacement array corresponding to warp colours equal to 1 and weft colour elements equal to 0. Go to step 4.
3. If the sets of warp and weft colours are not disjoint, partition the coloured interlacement array into regions which are colour disjoint and regions which are not colour disjoint. Assign 1's and 0's in disjoint regions, as in step 2. Areas which are undefined are structurally indeterminate and intersections can be freely specified according to whatever criteria you choose (for example, reducibility -- see 3.8). Assign these values.
4. Determine the corresponding threading, tie-up and shed

sequence matrices for the resulting binary interlacement array, using one of the algorithms of 3.5.

<u>EXAMPLE 3.6.7.</u>

This example shows a coloured interlacement array K, with disjoint warp and weft colouring, along with its corresponding binary interlacement array D.

$$K = \begin{array}{cccccccc} 5&2&5&2&3&3&3&5 \\ 5&2&5&2&2&4&4&4 \\ 3&2&5&2&2&5&3&3 \\ 4&4&5&2&2&5&2&4 \\ 3&3&3&2&2&5&2&5 \\ 5&4&4&4&2&5&2&5 \\ 5&2&3&3&3&5&2&5 \\ 5&2&5&4&4&4&2&5 \end{array} \qquad \text{WEFT} = \begin{array}{c} 3\\4\\3\\4\\3\\4\\3\\4 \end{array}$$

$$\text{WARP} = \begin{array}{cccccccc} 5&2&5&2&2&5&2&5 \end{array}$$

$$\begin{array}{lccccccccr}
 & 1&1&1&1&0&0&0&1 & \text{WEFT} = 3\\
 & 1&1&1&1&1&0&0&0 & 4\\
 & 0&1&1&1&1&1&0&0 & 3\\
 & 0&0&1&1&1&1&1&0 & 4\\
D = & 0&0&0&1&1&1&1&1 & 3\\
 & 1&0&0&0&1&1&1&1 & 4\\
 & 1&1&0&0&0&1&1&1 & 3\\
 & 1&1&1&0&0&0&1&1 & 4\\
 & & & & & & & & & \\
\text{WARP} = & 5&2&5&2&2&5&2&5 &
\end{array}$$

EXAMPLE 3.6.8.

This example shows a coloured interlacement array K with warp and weft colours which are not disjoint, along with the corresponding partially determinate binary interlacement array D. The array D' is D, with its rows and columns permuted to show the indeterminate intersections in rectangular regions.

$$\begin{array}{lccccccccr}
 & 4&2&4&3&2&2&2&3 & \text{WEFT} = 2\\
 & 4&2&4&3&4&4&4&4 & 4\\
 & 2&2&4&3&4&2&2&2 & 2\\
 & 3&3&4&3&4&2&4&3 & 3\\
K = & 2&2&2&3&4&2&4&3 & 2\\
 & 4&4&4&4&4&2&4&3 & 4\\
 & 4&2&2&2&2&2&4&3 & 2\\
 & 4&2&3&3&3&3&4&3 & 3\\
 & & & & & & & & & \\
\text{WARP} = & 4&2&4&3&4&2&4&3 &
\end{array}$$

```
                1  110  01            WEFT = 2
                 1  1  0  0                  4
                0  111  00                   2
                001  111                     3
      D =       0  011  11                   2
                 0  0  1  1                  4
                1  000  11                   2
                110  001                     3

      WARP =    42434343


                   |1 0|1 0           WEFT' = 4
                   |0 1|0 1                   4
                ______________                -
                1 1 0 0|  |1 1                2
                0 1 1 0|  |1 0                2
      D' =      0 0 1 1|  |1 1                2
                1 0 0 1|  |                   2
                ________________              -
                0 1 1 1|0 0|                  3
                1 0 0 1|1 0|                  3

      WARP' =   4 4 4 4|2 2|3 3
```

An implementation of this colour factorization process is given in the following Pascal procedures.

PASCAL ALGORITHM:

```
FUNCTION COLOR(VAR K:SIZE;M,N: INTEGER):BOOLEAN;
(*THIS FUNCTION DETERMINES A VECTOR OF WARP COLORS (WARP) AND A VECTOR OF WEFT
COLORS (WEFT) FOR A GIVEN COLOURED ARRAY, IF POSSIBLE.  IF THE GIVEN COLOURED ARRAY IS
NOT A COLOURED INTERLACEMENT ARRAY, THIS FUNCTION WILL ASSUME A VALUE OF FALSE. *)

VAR
   WCOUNT,FCOUNT,I,J: INTEGER;
   FLAG:BOOLEAN;

BEGIN
   COLOR:= TRUE;
    WCOUNT:=0;
    FCOUNT:=0;
    FOR I:=1  TO N DO WEFT[I]:=-1;
    FOR J:=1  TO M DO WARP[J]:=-1;

    FOR J:=1  TO M DO
    BEGIN
       I:=1;
       WHILE (I<=N) AND (K[I,J]=K[1,J])  DO I:=I+1;
       IF I>N THEN
       BEGIN
          WARP[J]:=K[1,J];
          WCOUNT:=WCOUNT+1;
       END;
    END;

    FOR I:=1  TO N DO
    BEGIN
       J:=1;
       WHILE (J<=M)  AND (K[I,J]=K[I,1])  DO J:=J+1;
       IF J>M  THEN
       BEGIN
          WEFT[I]:=K[I,1];
          FCOUNT:=FCOUNT+1;
       END;
    END;
```

```
IF WCOUNT<M THEN
    BEGIN
        J:=1;
        WHILE WARP[J]>-1 DO J:=J+1;
        WARP[J]:=K[1,J];
        WCOUNT:=WCOUNT+1;
    END;

    WHILE (WCOUNT<M) OR (FCOUNT<N) DO
    BEGIN
        J:=1;
         FLAG:= TRUE;
        WHILE J<=M  DO
        BEGIN
            IF WARP[J]<0 THEN J:=J+1
            ELSE
            BEGIN
                FOR I:=1  TO N DO
                BEGIN
                    IF (K[I,J]<>WARP[J])  THEN
                    BEGIN
                        IF WEFT[I]<0 THEN
                        BEGIN
                            FLAG:=FALSE;
                            WEFT[I]:=K[I,J];
                            FCOUNT:=FCOUNT+1;
                        END
                        ELSE IF K[I,J]<>WEFT[I]  THEN
                        BEGIN
                            COLOR:=FALSE;
                            WCOUNT:=M;
                            FCOUNT:=N;
                            J:=M;
                        END;
                    END;
                END;
            J:=J+1;
            END;
        END;
    I:=1;
    WHILE I<=N DO BEGIN
        IF WEFT[I] < 0 THEN I:=I+1
        ELSE
        BEGIN
            FOR J:=1  TO M DO
            BEGIN
                IF (K[I,J]<>WEFT[I])  THEN
                BEGIN
                    IF WARP[J]<0 THEN
                    BEGIN
                        FLAG:=FALSE;
                        WARP[J]:=K[I,J];
                        WCOUNT:=WCOUNT+1;
                    END
                    ELSE IF K[I,J]<>WARP[J]  THEN
                    BEGIN
                        COLOR:=FALSE;
                        WCOUNT:=M;
                        FCOUNT:=N;
                        I:=N;
                    END;
                END;
            END;
        I:=I+1;
        END;
    END;

    IF FLAG THEN
     BEGIN
        IF (WCOUNT<M) THEN
        BEGIN
           J:=1;
           WHILE WARP[J]>-1 DO J:=J+1;
           WARP[J]:=K[1,J];
           WCOUNT:=WCOUNT+1;
        END
        ELSE
        BEGIN
           I:=1;
           WHILE WEFT[I]>-1 DO I:=I+1;
           WEFT[I]:=K[I,1];
           FCOUNT:=FCOUNT+1;
        END;
   END
   ELSE FLAG:=TRUE:
  END;
END;
```

```
PROCEDURE BINARRAY(VAR K,D:SIZE;WARP,WEFT:VECSIZE;M,N:  INTEGER);

(*THIS PROCEDURE COMPUTES A BINARY INTERLACEMENT ARRAY D FOR A GIVEN COLOURED
INTERLACEMENT ARRAY K, BASED ON THE WARP AND WEFT COLOUR VECTORS COMPUTED IN THE
PREVIOUS PROCEDURE. ALL ELEMENTS OF D WHICH ARE INDETERMINATE ARE ASSIGNED A VALUE OF -1.
*)

VAR
   I,J: INTEGER;

BEGIN
   FOR I:=1  TO N DO FOR J:=1  TO M DO D[I,J]:=-1;
   FOR J:=1  TO M DO
   BEGIN
      FOR I:=1  TO N DO
      IF WARP[J]<>WEFT[I] THEN
         IF WARP[J]=K[I,J]  THEN D[I,J]:=1  ELSE D[I,J]:=0;
   END;
END;
```

3.7 A FACTORING ALGORITHM FOR MULTIPLE THREADING

When a conventionally woven fabric is produced on a loom with only one harness, a separate shaft is required for each equivalence class of strands. However, in some instances the nature of the weave structure is such that the number of shafts can be reduced if multiple threading is used. In this case, each warp strand is threaded through more than one shaft as described in Section (3.4). This corresponds to a factorization of a binary interlacement array into a threading matrix with more than one 1 per column and a tie-up matrix which is multi-valued, as described in Section (3.4). The shed sequence matrix is binary with precisely one 1 per row, as in the case of the single harness system.

Not all binary interlacement arrays can be factored in this manner. In order to determine whether or not a given structure can be multiply threaded, we consider the single harness factors.

Definition 3.7.1. A single harness factorization of a given binary interlacement array is one which produces a binary threading matrix with precisely one entry of one per column, a binary shed sequence matrix with precisely one entry of one per row, and a binary tie-up matrix.

Definition 3.7.2. A compound factorization of a given binary interlacement array is one which produces a binary threading matrix with more than one entry of 1 per column, a binary shed sequence matrix with precisely one entry of 1 per row, and a tie-up matrix which is multi-valued.

__THEOREM 3.7.3__. The ability for a given binary interlacement array to be compoundl7 factored is dependent solely on the properties of the single harness tie-up matrix.

This follows from the following theorem.

__THEOREM 3.7.4__. A given binary interlacement array can be compoundly factored if and only if the corresponding single harness factorization is such that there exists a one-to-one mapping of the ns shafts onto ns/d shafts, with ns/d an integer.

__Proof__. If a binary interlacement array is compound factored then the ground structure is controlled by the long-eyed heddles at the front of the loom, where these heddles can take one of three positions, namely down, neutral or up. The determination of precisely which of the threads which can potentially be raised by the front shafts is controlled by the pattern shafts at the back of the loom, where the heddles on these shafts can take one of two positions, namely down or up. Therefore, p x q shafts have been mapped onto q shafts, where p is the number of pattern shafts and q is the number of ground shafts.

If the single harness tie-up for a given binary interlacement array is such that there exists a one-to-one mapping from ns shafts to ns/d shafts, with ns/d an integer, then all of the strands can be threaded on ns/d ground shafts. In addition, all of the strands must be threaded on one of d pattern shafts. □

THEOREM 3.7.5. In order for compound factorization to reduce the number of shafts required, the number of ground shafts (ns/d) must be greater than 2.

Proof. There are only two inequivalent binary tie-up matrices of size 2 × 2, namely:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} .$$

The four shafts required for this tie-up could be reduced to two, by combining these tie-up matrices, however two pattern shafts would also be required and there would be no reduction in the total number of shafts required. □

EXAMPLE 3.7.6.

Single Harness Factorization:

0 0 0 0 0 1	1 0 0 1 1 0
0 0 0 0 1 0	0 1 0 0 1 1
0 0 0 1 0 0	0 0 1 1 0 1
0 0 1 0 0 0	1 1 0 1 0 0
0 1 0 0 0 0	0 1 1 0 1 0
1 0 0 0 0 0	1 0 1 0 0 1
1 0 1 0 0 1	1 0 0 0 0 0
0 1 1 0 1 0	0 1 0 0 0 0
1 1 0 1 0 0	0 0 1 0 0 0
0 0 1 1 0 1	0 0 0 1 0 0
0 1 0 0 1 1	0 0 0 0 1 0
1 0 0 1 1 0	0 0 0 0 0 1

Factorization Using Multiple Threading:

0 0 0 1 1 1	0 0 0 1 1 1	Pattern Shafts
1 1 1 0 0 0	1 1 1 0 0 0	0 = down; 1 = up
0 0 1 0 0 1	2 1 0 2 1 0	
0 1 0 0 1 0	0 2 1 0 2 1	Ground Shafts
1 0 0 1 0 0	1 0 2 1 0 2	0 = down; 1 = neutral; 2 = up
1 0 1 0 0 1	1 0 0 0 0 0	
0 1 1 0 1 0	0 1 0 0 0 0	
1 1 0 1 0 0	0 0 1 0 0 0	
0 0 1 1 0 1	0 0 0 1 0 0	
0 1 0 0 1 1	0 0 0 0 1 0	
1 0 0 1 1 0	0 0 0 0 0 1	

An algorithm based on the preceding results has the following steps:

1. Obtain a single harness factorization for the given binary interlacement array.
2. Choose the minimum block size b, such that $b \mid ns$, the total number of shafts required in the single harness factorization, and such that $b > 2$.
3. Partition the rows of the single harness tie-up matrix B into sets of b rows, {b'}, and sum the columns of each of the {b'}.
4. Choose the set of rows, {b'}, whose first column sum is the smallest.
5. Construct a one-to-one mapping of the shafts in column one of this {b'} onto the shafts of the row set whose first column sum is the next largest. Repeat for all columns of {b'}.
6. If an inconsistency develops at any point, return to step 5 and choose a different initial mapping.
7. Construct a one-to-one mapping of this combined set onto the row set whose first column sum is the next largest. Repeat for all columns.
8. Repeat step 7 until all ns rows have been mapped onto ns/b rows.
9. If no self-consistent mapping exists then return to step 2 and

choose the next largest block size.

10. If no further block sizes are possible then the given binary interlacement array cannot be compoundly factored.

11. Re-write the threading matrix A on b shafts, according to the mapping developed in the previous steps.

12. Choose a second shaft s_j for each warp thread a_j,

$$\ni \; s_j = b + \lceil (a_j - 1) \rceil + 1 \, .$$

EXAMPLE 3.7.7.

TIE-UP

FIGURE 3.7.8

INTEGER REPRESENTATION

```
0 1 1 1 0 1 1 1 0 0 0 1
1 0 1 1 1 0 1 1 0 0 1 0
1 1 0 1 1 1 0 1 0 1 0 0
1 1 1 0 1 1 1 0 1 0 0 0
0 1 1 1 0 0 0 1 0 0 0 1
1 0 1 1 0 0 1 0 0 0 1 0
1 1 0 1 0 1 0 0 0 1 0 0
1 1 1 0 1 0 0 0 1 0 0 0
0 0 0 1 0 1 1 1 0 1 1 1
0 0 1 0 1 0 1 1 1 0 1 1
0 1 0 0 1 1 0 1 1 1 0 1
1 0 0 0 1 1 1 0 1 1 1 0
```

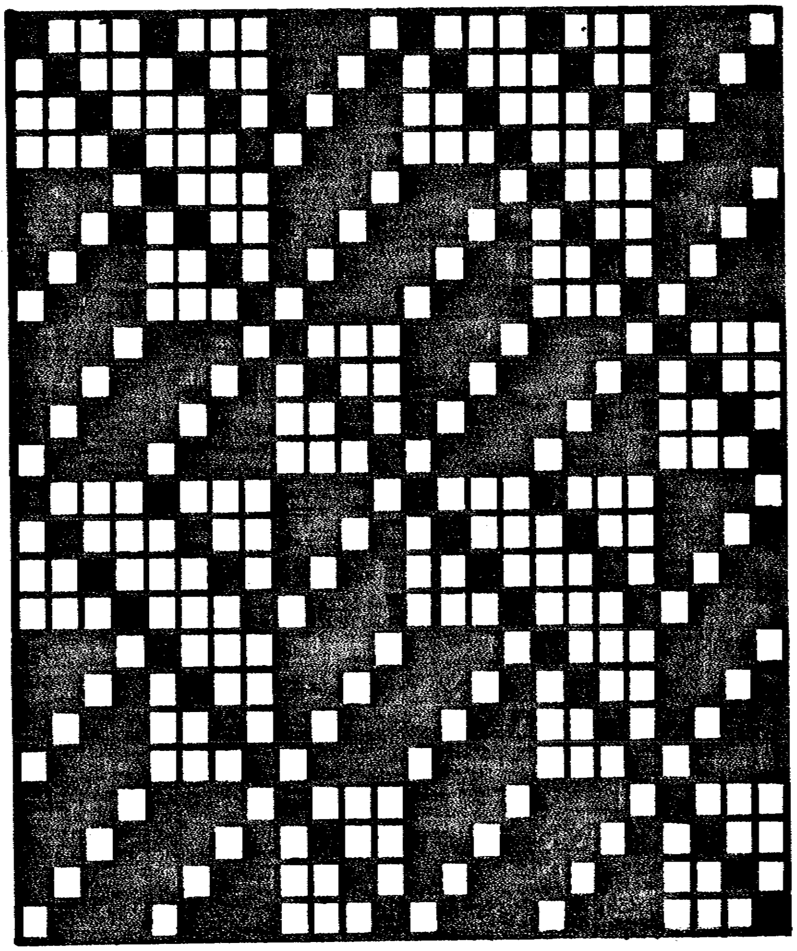


FIGURE 3.7.8

<u>b = 3:</u>

COLUMN SUMS FOR (b')

```
2 2 2 3 2 2 2 2 3 0 1 1 1
2 2 3 2 1 1 2 1 1 0 1 1
2 2 1 2 1 2 1 1 1 2 1 1
1 1 1 0 3 2 2 2 2 3 2 2 2
```

MAPPING 1

b'_1:	1	2	3 }	
b'_2:	5	6	5 }	inconsistent

MAPPING 2

b'_1:	1	2	3 }	
b'_2:	6	5	5 }	inconsistent

b = 4:

COLUMN SUMS FOR (b')

```
3 3 3 3 3 3 3 3 1 1 1 1
3 3 3 3 1 1 1 1 1 1 1 1
1 1 1 1 3 3 3 3 3 3 3 3
```

MAPPING 1

b'_1:	1	2	3	4
b'_2:	5	6	7	8
b'_3:	9	10	11	12

TIE-UP

FIGURE 3.7.8

COMPOUND FACTORIZATION

1 1 1 1 1 1 1 1 0 0 0 0	Pattern Shafts
1 1 1 1 0 0 0 0 0 0 0 0	0 = down; 1 = up
0 0 0 0 1 1 1 1 1 1 1 1	
0 1 1 2 0 1 1 2 0 1 1 2	
1 0 2 1 1 0 2 1 1 0 2 1	Ground Shafts
1 2 0 1 1 2 0 1 1 2 0 1	0 = down; 1 = neutral;
2 1 1 0 2 1 1 0 2 1 1 0	2 = up

THREADING

FIGURE 3.7.8

INTEGER REPRESENTATION

```
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
```

THREADING

FIGURE 3.7.8

COMPOUND FACTORIZATION

0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1

0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0

1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0

0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2

0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0

0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0

2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0 2 0 0 0

where 1 = regular heddles and 2 = long-eyed heddles.

3.8 COERCIVE ANALYSIS

In the interest of efficiency, it is frequently desirable to determine whether a new structure can be woven on a loom which has already been threaded on a specified number of shafts.

Definition 3.8.1. Coercive analysis is defined as the process whereby an attempt is made to force the threading matrix for a given interlacement array to conform to a previously formulated configuration, with appropriate changes being made to the tie-up matrix.

It should be noted that coercion of a threading is not always possible, and the following results apply:

Definition 3.8.2. A single harness threading $T_{k,r}$ on r shafts is defined as:

$$T_{k,r} = \{\ldots, T_k, T_k, T_k, \ldots\}$$

where

$$T_k = \{a_1, a_2, \ldots, a_k\}$$

and

$$a_i \in \{1,2, \ldots ,r\}$$

with

$$\bigcup_{i=1}^{k} \{a_i\} = \{1,2,\ldots,r\}.$$

<u>Definition 3.8.3</u>. The <u>set of interlacement arrays</u> $\alpha(T_{k,r})$ is that set of interlacement arrays whose threading factor is $T_{k,r}$.

<u>Definition 3.8.4</u>. The threading $T_{k,r}$ is <u>subordinate</u> to $T_{n,s}$ if and only if $\alpha(T_{k,r})$ is contained in or equal to $\alpha(T_{n,s})$ and $s \geq r$.

<u>Definition 3.8.5</u>. The <u>period</u> p of a threading $T_{k,r}$ is such that

$$p = \min q$$

where

$$a_j = a_{j+q} \qquad \forall j.$$

<u>Theorem 3.8.6</u>. If the period of the threading $T_{k,r}$ is p_1, and the period of the threading $T_{k,n}$ is p_2 then, in order that $T_{k,r}$ be subordinate to $T_{k,n}$, p_1 must divide p_2, for non-trivial threadings.

Proof. Let $\{a_i\}$ be the elements of $T_{k,r}$ and $\{b_i\}$ be the elements of $T_{k,n}$. If $T_{k,r}$ is subordinate to $T_{k,n}$ then there is an equivalence relation between the two threadings, such that

$$a_x \text{ is equivalent to } b_x \quad x = 1,2,\ldots,k.$$

Consider

$$a_1\ a_2 \ldots a_x \ldots a_{p[1]}\ a_{p[1]+1} \ldots a_{p[1]+x} \ldots a_{p[2]} \ldots a_{p[2]+x} \ldots a_k$$

$$b_1\ b_2 \ldots b_x \ldots b_{p[1]}\ b_{p[1]+1} \ldots b_{p[1]+x} \ldots b_{p[2]} \ldots b_{p[2]+x} \ldots b_k$$

where $p[1] = p_1$

$$a_x \text{ is equivalent to } b_x$$

$$a_{p[2]+x} \text{ is equivalent to } b_{p[2]+x}.$$

But p_2 is the period of $T_{k,n}$, so that

$$b_x \text{ is equivalent to } b_{p[2]+x}.$$

Therefore a_x is equivalent to $a_{p[2]+x}$.

But p_1 is the period of $T_{k,r}$ (with $p_1 \leq p_2$). Therefore $a_{x \text{ (mod } p[1])}$ is equivalent to $a_{p[2]+x \text{ (mod } p[1])}$. In order that $T_{k,r}$ be a non-trivial threading, p_1 must divide p_2 as required. □

Theorem 3.8.7. For $T_{k,r}$ to be subordinate to $T_{k,n}$, there must exist a mapping

$$a_s, a_{s+1}, \ldots, a_{s+t}, \ldots, a_{s+k-1}$$
$$b_1, \ b_2, \ldots, b_{t+1}, \ldots, b_k$$

from $T_{k,r}$ to $T_{k,n}$, such that

$$\forall j \ni \{a_{s+j} \neq b_{j+1}\},$$

if $\quad a_{s+p} = b_{t+1}$

and $\quad a_{s+q} = b_{t+1}$,

then $\quad a_{s+p} = a_{s+q}$.

Proof. If $T_{k,r}$ is subordinate to $T_{k,n}$, then the mapping from $T_{k,r}$ to $T_{k,n}$ can be constructed by the following sequence of steps:

1. Given $T_{k,r}$ with $T_k = (a_1, a_2, \ldots, a_k)$ and $T_{k,n}$ with

$T_k = \{b_1, b_2, \ldots, b_k\}$,

2. Set $i = 1$.

3. If $a_i = b_i$ then set $c_{b[i]} = a_i$, set $i = i + 1$ and return to step 3.

4. If $a_i \neq b_i$ and $c_{b[i]}$ is undefined, then set $c_{b[i]} = a_i$, set $i = i + 1$ and return to step 3.

5. If $a_i \neq b_i$ and $c_{b[i]} = x$, where $x \neq a_i$, then shift the sequence $\{a_1, a_2, \ldots, a_k\}$ one position to the right, with cyclic wrap-around, and return to step 2.

6. If the sequence $\{a_1, a_2, \ldots, a_k\}$ has been shifted through k positions, then reverse the order of the sequence and return to step 2.

7. The algorithm terminates, either when the mapping $C = \{c_1, c_2, \ldots, c_k\}$ has been constructed, or when the 2k variations of the sequence $\{a_1, a_2, \ldots, a_k\}$ have been unsuccessfully compared with the sequence $\{b_1, b_2, \ldots, b_k\}$. In the latter case, the

threading $T_{k,r}$ is <u>not</u> subordinate to the threading $T_{k,n}$.

EXAMPLE 3.8.8.

Given two threading sequences

$$T_{10,4} = \{2,3,2,1,4,1,2,3,2,1\}$$

and

$$T_{10,6} = \{1,2,3,4,5,6,5,4,3,2\},$$

we wish to determine whether the threading $T_{10,4}$ is subordinate to the threading $T_{10,6}$.

In attempting to construct a mapping between the two threading sequences using the algorithm in Theorem (3.8.7), an inconsistency is encountered when $i = 7$. The threading sequence $T_{10,4}$ is shifted one position to the right, with cyclic wrap-around, and a mapping between the two threading sequences $T_{10,4}$ and $T_{10,6}$ is successfully constructed, as follows:

$$1 = 1;\ 1 = 5;\ 2 = 2;\ 2 = 4;\ 3 = 3;\ 4 = 6.$$

Figure {3.8.9} illustrates the effect of this coercion on the corresponding tie-up matrix.

```
          4                         4 4
    3           3                 3 3
  2   2       2   2             2 2
1       1   1                   1     1
```

$T_{10,4}$ THREADING AND CORRESPONDING TIE-UP

```
          6                         6 6
        5   5                   5     5
      4       4                 4 4
    3           3                 3 3
  2               2             2 2
1                               1     1
```

$T_{10,4}$ THREADING AND CORRESPONDING TIE-UP
AFTER COERCION TO $T_{10,6}$ THREADING

FIGURE 3.8.9

CHAPTER 4

ALGORITHMS FOR STRUCTURAL CROSS-SECTIONS

CONTENTS

4.1 INTRODUCTION

Multi-layered cloths represent a class of textile structures which exhibit a number of important utilitarian and aesthetic features. Fabrics can be made thicker and heavier without increasing yarn size. The volume and weight of a fabric can thus be increased while still maintaining the visual appearance and surface tactile properties of a finely woven structure. Much use of multi-layer cloth techniques can also be made in creating interesting visual effects. Extra sets of warp or weft yarns can be introduced to create blocks of design in a solid colour. Alternatively, extra warp and weft layers can be woven simultaneously to produce two completely different fabrics which are "stitched" together to form a reversible textile with surface interest on both faces.

Classically, these multi-layer structures have been classified according to whether the layers are point stitched together at single intersections or are held together by the interchange of entire fabric areas. The point stitched fabrics have been subdivided further, based on which set or sets of yarns do the stitching. This classification system is well defined in [22, p.103] as follows:

> (1) Self-stitched double cloths. These fabrics contain only the two series of threads in both directions and the stitching of the face cloth layer to the back layer is accomplished by occasionally dropping a face end under a back pick, or, by lifting

a back end over a face pick, or, by utilising both of the above systems in different portions of the cloth . . .

(2) Centre-stitched double cloths. In these fabrics a third series of threads is introduced either in the warp or in the weft direction whose entire function is to stitch the two otherwise separate layers of cloth together. The centre threads lie between the face and the back cloth and for the purpose of stitching oscillate at regular intervals between the face and the back thus achieving the required inter-layer cohesion . . . .

(3) Double cloths stitched by thread interchange. These structures are similar to the first category inasmuch as they do not contain an additional series of stitching threads. However, they are distinguished from the self-stitched fabrics by the fact that the stitching of the face and the back cloth is achieved by frequent and continuous interchange of some thread elements between the two cloth layers. Thus, in some portions of the cloth the face ends may be made to interweave with the back picks and the back ends with the face picks . . . .

(4) Double cloths stitched by cloth interchange. In this class of constructions the principle of the interchange is taken one stage further than in the third category and complete layers are made to change places. . . .

A more recent scheme proposed by Newton and Sarkar [59] divides <u>all</u> structures into four types, namely

(1) single-layer;
(2) multi-layer without stitching;
(3) single-direction multi-layer with stitching where, either the ends form two or more layers with the picks remaining in a single layer or conversely, where the picks form two or more layers and the ends remain in a single layer;
(4) two-directional multi-layer with stitching where the warp ends and weft picks form two fabric layers connected by stitching points.

Stitching in this sense is taken to include areas of cloth interchange as well as the more restrictive form involving stitching points or intersections.

In order to classify a fabric by either of these systems one relies on being able to observe the structure or physically analyze the fabric. If however, one is examining a point diagram rather than the actual fabric, the type of fabric cannot be readily classified because a homogenous point diagram may conceal the fact that a structure is multi-layered and/or reducible. Weavers have therefore needed an identification process to aid in classifying point diagrams into these different types of structures and to determine their level of reducibility. Two types of analytical tools have been developed to facilitate this process.

The first of these classes of tools comprises the algorithms for identifying binary interlacement arrays which correspond to reducible fabrics. These textile structures do not form a cohesive fabric but can be separated into two or more completely disjoint layers of strands and/or fabrics, corresponding to Newton and Sarkar's category 2. Partial reducibility, corresponding to Newton and Sarkar's categories 3 and 4 and all of Grosicki's four types, can also be identified from the binary interlacement array, by means of a computational algorithm. These algorithms are discussed in Section (4.2).

The second type of analytical tool is that characterized by algorithms for mapping binary interlacement data to an alternative representation corresponding to cross-sectional cuts through the warp or weft yarns of a fabric, between successive weft or warp yarns. This form of display provides a great deal of insight into the relative planar positions of the constituent strands and is particularly useful for examining reducible and partially reducible fabrics. Cross-sectional diagrams and the appropriate mapping algorithms are discussed in Section (4.3).

4.2 ALGORITHMS FOR DETERMINING REDUCIBILITY

A reducible weave structure is a binary interlacement array corresponding to a cloth which separates into two or more disjoint layers of strands and/or fabrics. An irreducible weave structure is a binary interlacement array corresponding to a cloth in which all of the constituent warp and weft strands are interlaced. A partially reducible weave structure is a binary interlacement array corresponding to a cloth which consists of two or more fabric or strand layers held together at a small number of intersections.

In some instances, fabric structures are designed to be reducible to fulfill particular purposes. This does provide, for example, a technique for producing two fabrics simultaneously or for weaving a tubular cloth or a fabric which is twice the loom width [22, p. 104]. It is far more common however, to design woven textiles which are intended to be irreducible or only partially reducible, in which case structural reducibility is a property which must be identified and eliminated. This is especially important with the development of interactive computer graphical systems which permit the rapid design of point diagrams representing novel and unfamiliar interlacement arrays. As discussed in Section (4.3.2), it is not always intuitively obvious from a point diagram that a given structure is in fact reducible. We now consider four algorithms for determining reducibility, namely an algorithm based on row and column sums [10], an algorithm involving row and column permutation, an algorithm based on a graph theoretic approach [15], and an algorithm based on the identification of circuits [51], [59]. The last mentioned algorithm can also be used in identifying structures which are only partially reducible.

4.2.1 ALGORITHM BASED ON ROW AND COLUMN SUMS

Clapham's algorithm [10] relies entirely on two principles.

1. The reducibility of a weave structure depends entirely on the "set of row-sums and the set of column-sums" of the corresponding interlacement array.

2. If a fabric separates into disjoint layers of strands or fabrics "then it does so by taking a set of weft strands corresponding to a certain number of rows with row-sums as small as possible and a set of warp strands corresponding to a certain number of columns with column-sums as large as possible".

This mathematically simple process is defined by the following steps.

1. For a given binary interlacement array

   $D = \{d_{i,j} \mid i = 1,2, \ldots, n;\ j = 1, 2, \ldots, m\}$

   compute the row sums

   $\{r_i \mid i = 1,2, \ldots, n\}$

   and the column sums

$\{c_j \mid j = 1,2,\ldots,m\}$.

2. Sort the row sums into ascending order and the column sums into descending order.
3. Let $j = 1$.
4. Choose the largest value of i such that

   $r_i < j$.
5. Calculate

   $E_{i,j} = r_1 + \ldots + r_i + (n - c_1) + \ldots + (n - c_j) - (i \times j)$.
6. If $E_{i,j}$ is not equal to 0, calculate $E_{i,j}$ for $j = j + 1$.
7. Continue until $E_{i,j} = 0$ or until $j = m$.
8. When $E_{i,j} = 0$, the weft strands corresponding to

   $r_1 \ldots r_i$

   and the warp strands corresponding to

   $c_1 \ldots c_j$

   form one reducible layer which can be separated from the remaining fabric.
9. If $E_{i,j}$ is not equal to 0 for all values of j, $j = 1,2,\ldots,m$ then the interlacement array D represents a single layer irreducible fabric structure.

It should be noted that this algorithm does not differentiate between irreducible and partially reducible structures. A cloth which consists of two fabric layers joined at only one intersection will therefore be designated irreducible by this analysis. The algorithm also does not directly determine whether a reducible structure is composed of more than two layers.

In order to determine whether or not the two layers of a reducible fabric are themselves reducible, it is necessary to construct two sub-matrices and re-apply the algorithm on each of these, as follows:

1. Given $A = \{a_{i,j}\}$, an $n \times m$ reducible array in which the first r rows and the first c columns separate from the remaining $(n - r)$ rows and $(m - c)$ columns of A,
2. Create two sub-arrays, X and Y, where

$$X = \{a_{i,j} \mid i = 1,2,\ldots,r\,;\ j = 1,2,\ldots,c\}$$

and $$Y = \{a_{i,j} \mid i = r+1,r+2,\ldots,n;\ j = c+1,c+2,\ldots,m\}.$$

3. Apply the preceding algorithm to X and Y.
4. If neither X nor Y is reducible, then the structure consists of only two layers. If either, or both of these matrices is reducible, then apply steps 1 through 3, as before.

Clearly a major component of this algorithm is the sorting of the integer valued row and column sums, and the evaluation of the table of values $E_{i,j}$, thus making it ideally suited to computer processing.

4.2.2 <u>ALGORITHM INVOLVING ROW AND COLUMN PERMUTATIONS.</u>

<u>THEOREM 4.2.2.1</u>. The reducibility of an interlacement array depends only on the row and column sums.

<u>Proof</u>. From the previous algorithm. □

<u>LEMMA 4.2.2.2</u>. The reducibility of an interlacement array is invariant under row and/or column permutations.

<u>Proof</u>. If A is an interlacement array and

$$u = \begin{bmatrix} 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}$$

with $$A\,u = r,$$

then r is the vector of row sums of A. Applying a row permutation P to A implies that

$$P\,A\,u = P\,r$$

and the values in P r are the same as those in r, but permuted in order. A similar argument applies to columns and hence, using Theorem [4.2.2.1], the result follows. □

Janice Lourie [50] used a similar result to effect the synthesis of a multi-layered fabric from two or more single layer structures. She constructed an initial matrix P, where

$$P = \begin{bmatrix} W_1 & 0 \\ 1 & W_2 \end{bmatrix} ,$$

with: W_1 and W_2 interlacement arrays,

0 a matrix of 0's of appropriate dimensions,

1 a matrix of 1's of appropriate dimensions;

and "with the '1' submatrix insuring that all the columns of W_1 are 'over' all the rows of W_2, and the '0' submatrix insuring that the columns of W_2 are 'under' all rows of W_1".

She then developed an algorithm for systematically permuting the rows and columns of P so that the rows and columns of the arrays W_1 and

W_2 were interleaved. This was analogous to "vertically stacking" the corresponding single layer fabrics.

<u>THEOREM 4.2.2.3</u>. An $n \times m$ array A is reducible if and only if

$$A = P \left[\begin{array}{c|c} E & C \\ \hline D & 0 \end{array}\right] Q$$

where P, Q are permutation matrices, E is an $r \times s$ matrix of 1's, 0 is an $(n - r) \times (m - s)$ matrix of 0's and C, D are matrices of appropriate dimension.

<u>Proof</u>. If

$$A = P \left[\begin{array}{c|c} E & C \\ \hline D & 0 \end{array}\right] Q$$

then, from Lemma (4.2.2.2), A is reducible. Similarly, if A is reducible, this implies that the interlacement array consists of at least two layers which separate. These layers may be written, with no loss of generality, as

$$L_1 = \left[\begin{array}{c|c} 1 & C \\ \hline N & 0 \end{array}\right]$$

$$L_2 = \left[\begin{array}{c|c} 1 & M \\ \hline D & 0 \end{array}\right]$$

where L_1 and L_2 have the same partitioning as A and the dimensions of 1, C, 0, and D are changed appropriately. The arrays N and M are dummy arrays, whose entries fulfill no purpose other than that of enabling L_1 and L_2 to be written as square arrays. Now, if L_1 overlays L_2, the array

$$\left[\begin{array}{c|c} 1 & C \\ \hline D & 0 \end{array}\right]$$

is obtained. □

A consequence of this theorem is that it may be used to determine the reducibility of an interlacement array, and in detail is given by the following steps, where it is assumed that the array $A = (a_{i,j})$ is not trivially reducible (i.e. has no rows which are all zeros and no columns which are all ones):

1. Permute the rows of A such that, if w is the number of ones per row, then

$$w_{i+1} \geq w_i \qquad i = 1,2,\ldots,n-1.$$

2. k is assigned 1; p is assigned m.
3. If the kth row has a zero in the rth position, then insert the rth column between the columns in positions p and $(p+1)$ (or, if $p = m$, then concatenate the rth column on the right).
4. Repeat step 2 for all the zeros in the first p columns of the kth row.
5. s is assigned the number of ones in the first p columns of the kth row.
6. A is irreducible if $s = 0$, and the algorithm terminates.
7. Otherwise, if

$$a_{i,j} = 1 \qquad (i = 1, 2, \ldots, k;\ j = 1, 2, \ldots, s)$$

and $$a_{i,j} = 0 \qquad (i = k+1, k+2, \ldots, n;\ j = s+1, s+2, \ldots, m),$$

then the array is reducible and the first k rows lift off with the $(m - s)$ last columns. The algorithm terminates.

8. Otherwise, if $k = n$ then A is irreducible and the algorithm

terminates.

9. Otherwise, k is assigned (k + 1) and steps 2 through 7 are repeated.

Having applied this algorithm to a matrix A and found it to consist of two separate layers X and Y, it is a simple matter to determine whether either of these submatrices is itself reducible. For example, to test the k x (m-s) array X for reducibility, one must check whether X can be partitioned into the form

$$\left[\begin{array}{c|c} E & C \\ \hline D & 0 \end{array}\right]$$

No permutations are required to put X into this form if it is reducible.

Clearly this algorithm has the same mathematical structure as Clapham's algorithm. Clapham permutes the row and column sums and applies his computational formula to determine reducibility, whereas this algorithm involves sorting the rows and columns themselves. Due to the increased data handling, this approach leads to a much less efficient implementation than Clapham's algorithm. It does however provide a useful insight into the structural nature of reducible fabrics.

4.2.3 ALGORITHM BASED ON A GRAPH THEORETIC APPROACH

Enns' algorithm [15] is "based on the observation that a fabric hangs together if and only if the lifting of each strand v causes each other strand to lift. The lifting of a strand v causes a strand w to lift if and only if there is a sequence of strands, beginning with w and ending with v, such that each element of the sequence lies under the next element of the sequence." This behaviour can be modeled using a bipartite directed graph [48].

The steps involved in the execution of this process can be summarized as follows:

1. For a given binary interlacement array

   $D = \{ d_{i,j} \mid i,j = 1, 2, \ldots, k \}$, where $k = n + m$,

   number the rows of D $\{1, 2, \ldots, n\}$ and
   the columns of D $\{n+1, n+2, \ldots, k\}$.

2. Use D to construct an adjacency list representation for G, where

   v_i is adjacent to v_{j+n} if $d_{i,j} = 1$

   and v_{i+n} is adjacent to v_j if $d_{i,j} = 0$

   $(i = 1,2, \ldots, n;\ j = 1,2, \ldots, m)$.

3. Determine the strongly connected components of G:

i. Order the vertices of G using a depth-first search algorithm [1, p. 200 - 226], as follows.

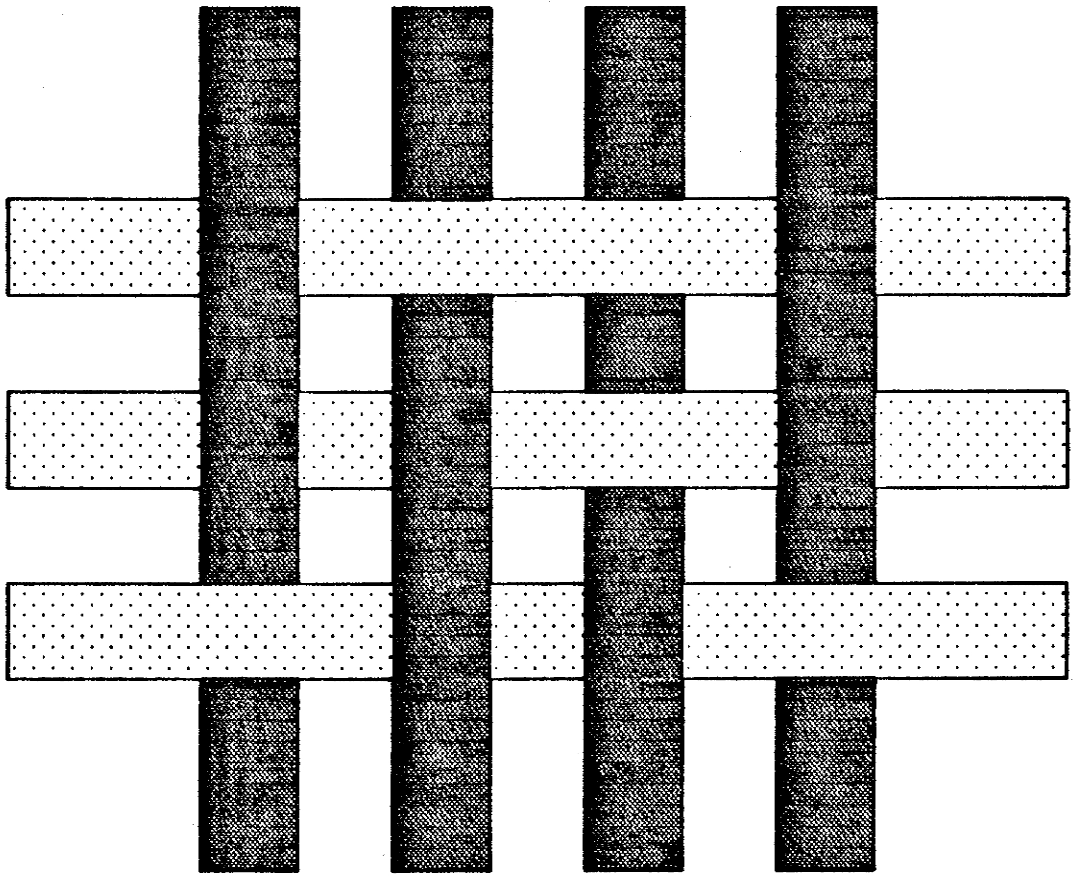
> Suppose we have a directed graph G in which all vertices are initially marked unvisited. Depth-first search works by selecting one vertex v of G as a start vertex; v is marked visited. Then each unvisited vertex adjacent to v is searched in turn, using depth-first search recursively. Once all vertices that can be reached from v have been visited, the search of v is complete. If some vertices remain unvisited, we select an unvisited vertex as a new start vertex. We repeat this process until all vertices of G have been visited. [1, p. 215]

ii. Construct an adjacency list for a new graph G', formed from G by reversing the direction of all the edges of G.

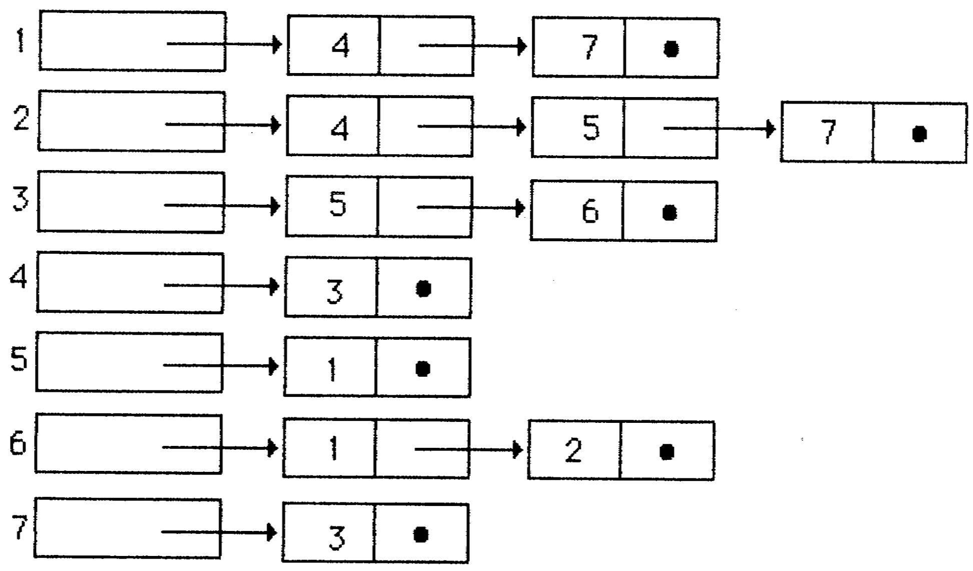
iii. Starting with the highest numbered vertex from (i) as a root, perform a depth-first search on G' to locate all of the depth-first spanning trees of G'. Each tree in this depth-first spanning forest forms a strongly connected component of the original graph G.

4. If G is strongly connected, then the corresponding fabric is not reducible. Otherwise, each of the strongly connected components of G corresponds to a disjoint layer of the corresponding fabric.

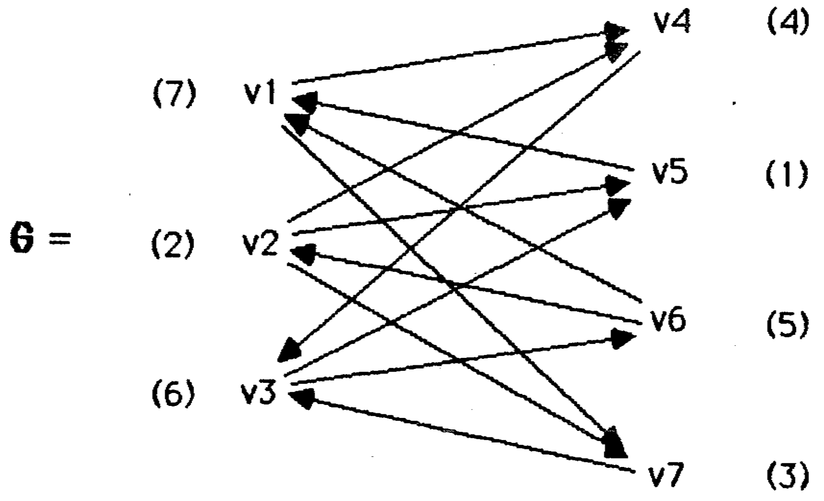
EXAMPLE 4.2.3.1



$$D = \begin{array}{cccc|c} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 & 3 \\ \hline 4 & 5 & 6 & 7 & \end{array}$$



ADJACENCY LIST REPRESENTATION

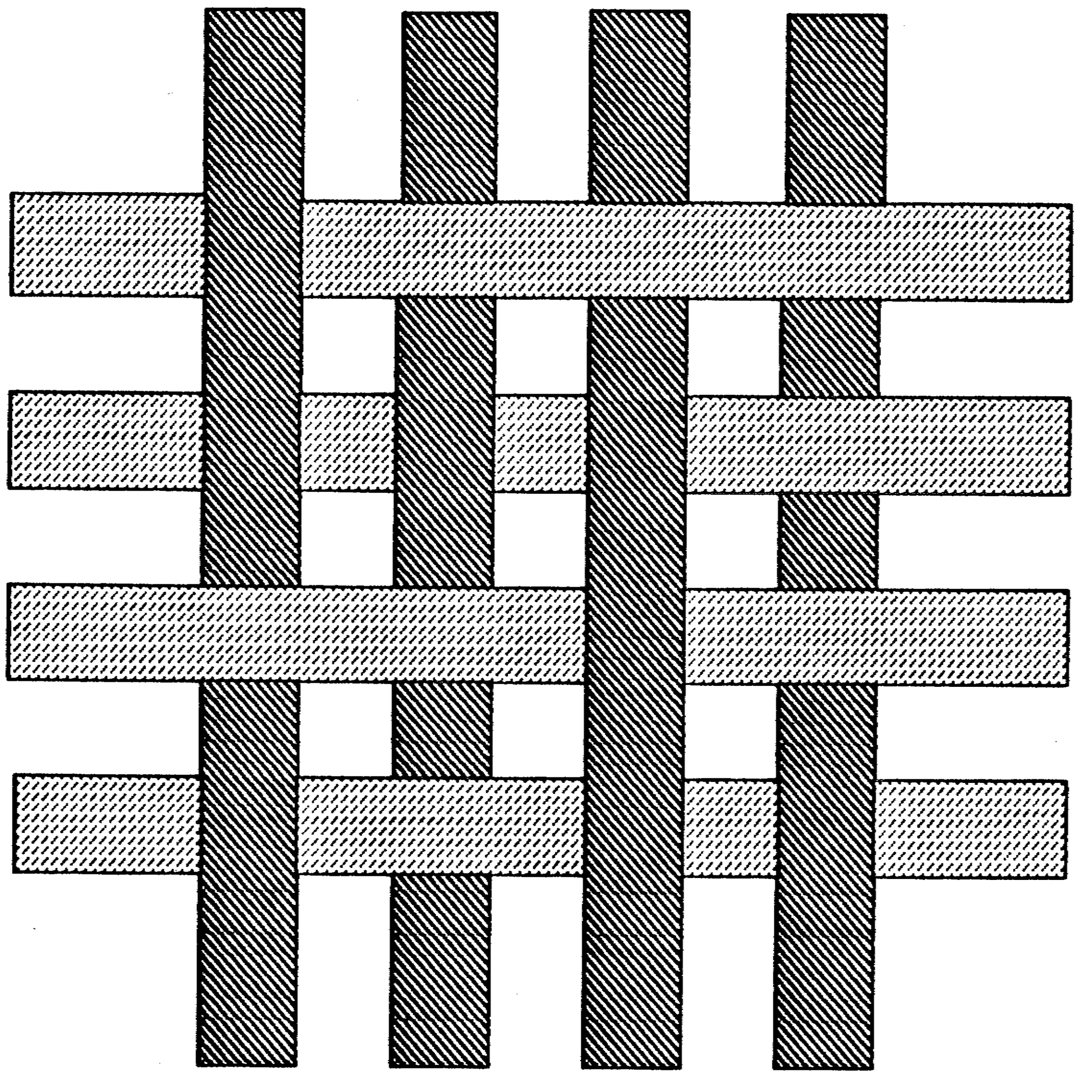


The numbers in brackets indicate the ordering of the vertices after the depth-first search in step (3 - i).

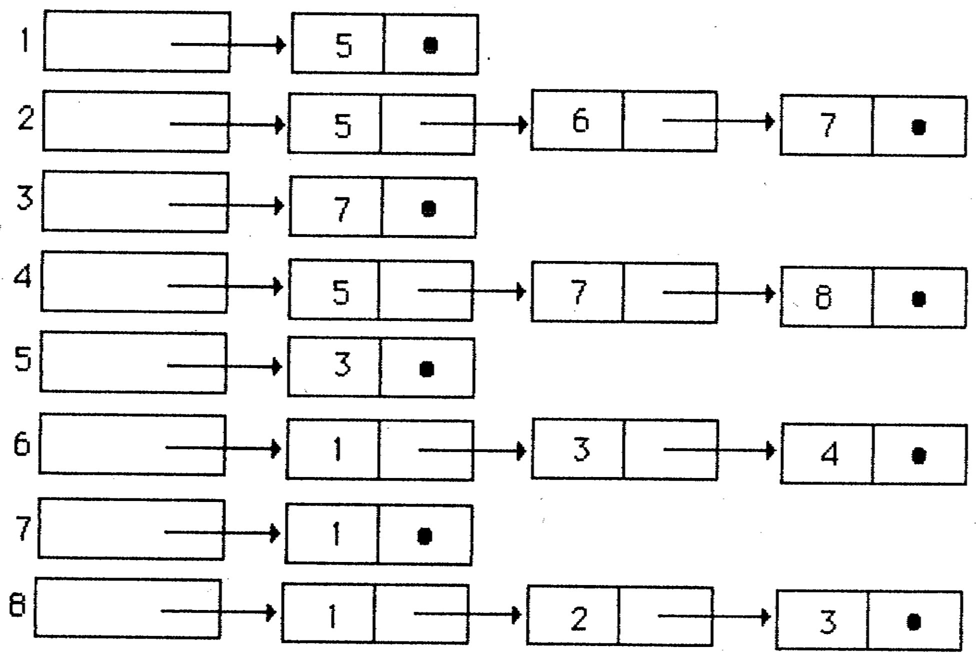
$$\begin{tikzcd}
 & & & v4 \arrow[dl] \arrow[dddl] & (4) \\
 & (7) & v1 \arrow[dr] \arrow[dddr] & & \\
 & & & v5 \arrow[dl] \arrow[dddl] & (1) \\
G' = & (2) & v2 \arrow[dr] & & \\
 & & & v6 \arrow[dl] & (5) \\
 & (6) & v3 \arrow[uuuuur] \arrow[dr] & & \\
 & & & v7 \arrow[uuuuul] \arrow[uuul] & (3)
\end{tikzcd}$$

G is strongly connected and corresponds to an irreducible fabric.

EXAMPLE 4.2.3.2

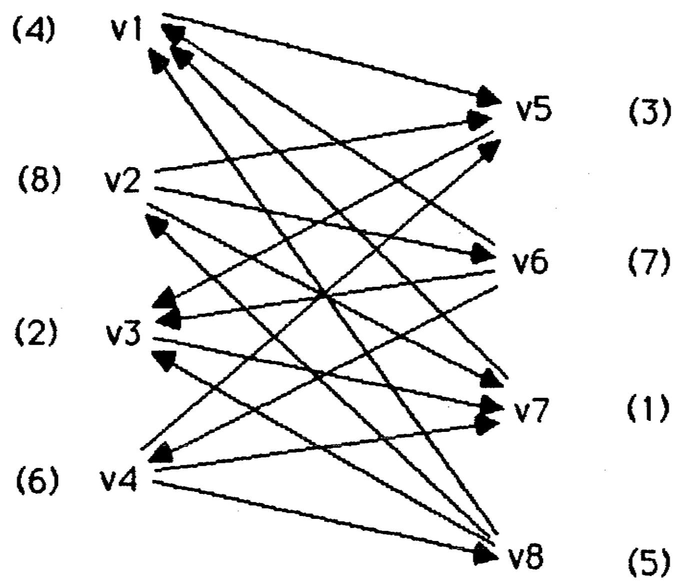


$$D = \begin{array}{ccccc} 1&0&0&0&1\\ 1&1&1&0&2\\ 0&0&1&0&3.\\ 1&0&1&1&4\\ \\ 5&6&7&8& \end{array}$$

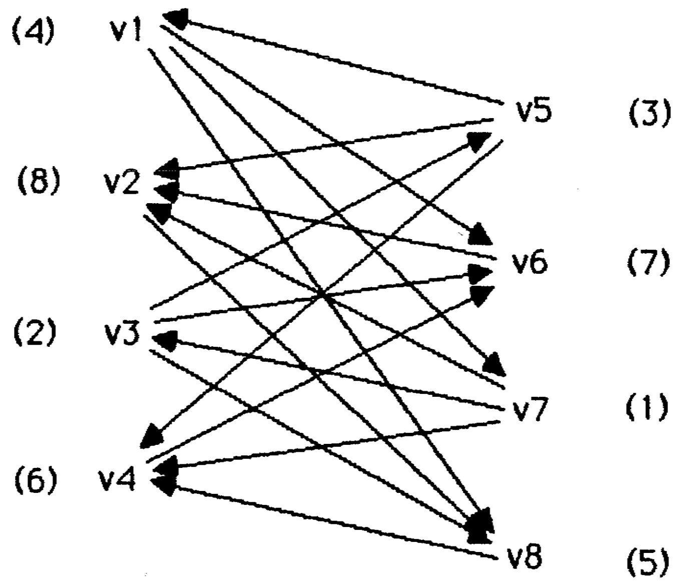


ADJACENCY LIST REPRESENTATION

G =



G' =



The graph G has two strongly connected components, with the vertices V2, V8, V4 and V6 being in one and the vertices V1, V7, V3 and V5 being in the other. This corresponds to a fabric in which the second and fourth warp strands interlace with the second and fourth weft strands to form one fabric layer, while the first and third warp strands interlace with the first and third weft strands to form a second and disjoint fabric layer.

The time complexity of this algorithm is O(e), where e is the number of edges of G. Since e is always <u>precisely equal</u> to m x n, the number of elementary operations required is the same as for Clapham's algorithm. This algorithm does however possess two distinct advantages over those previously discussed.

1. All of the fabric layers can be determined in one pass of the algorithm.

2. This process can be easily generalized to deal with fabrics with more than two sets of strands.

4.2.4 <u>ALGORITHM INVOLVING IDENTIFICATION OF CIRCUITS</u>

This algorithm, developed by Newton and Sarkar [59], involves considerably more operations than the previous three processes and is based on the principle of a circuit, as defined by Lourie [50]:

> A set of rows and columns represents an interwoven structure, if from any row or column some circuit can be found consisting alternately of 0 and 1 corners such that all the rows and columns contribute at least one corner point. Alternately or equivalently stated, for any pair of columns (or rows) in a woven structure or layers, . . . there exists a circuit which contains alternate 0's and 1's.

It should be noted that the approach that Enns [15] used in his work on reducibility (that is, that in order for a fabric to be irreducible, the lifting of one strand of the fabric causes the lifting of all the remaining strands as well) is an alternative formulation of Lourie's notion of a circuit.

The steps involved in the execution of this algorithm, when applied to a binary interlacement array $A = \{a_{i,j} \mid i = 1,2, \ldots ,n;\ j = 1,2, \ldots ,m\}$, can be summarized as follows.

1. Label each column of A with a layer number L_j such that $L_j = j$ $(j = 1,2, \ldots , m)$.

2. Label each row of A with a layer number LL_i such that $LL_i = i$ $(i = 1,2 \ldots, n)$.
3. Examine all possible combinations of pairs of rows and columns to identify circuits.
4. If a circuit is found between column p and column q, with $L_p < L_q$, set L_q equal to L_p.
5. Similarly, if a circuit is found between row x and row y, with $LL_x < LL_y$, set LL_y equal to LL_x.
6. At the termination of this stage, complete reducibility has been determined. The fabric structure separates into as many layers as there are different layer numbers, with these layer numbers indicating to which layer a given strand belongs.
7. If the array A has been found to be irreducible, the algorithm now looks for partial reducibility.
8. Remove a column of A and repeat steps 1 through 5 on the remaining submatrix.
9. Repeat step 8 until the submatrix which is being examined is reducible.
10. Repeat steps 8 and 9 on the rows of A.
11. If there is no reducible submatrix, then the array A is genuinely irreducible and the algorithm terminates.
12. Otherwise, the array A corresponds to a partially reducible structure, i.e. a multi-layer fabric with stitching points.

13. Add one of the original columns of A to the smallest submatrix obtained in step 9 and test for reducibility.
14. Repeat step 13 until the structure is no longer reducible.
15. Identify the circuit between two different layers and the stitching point within the circuit. (This is related to the row and column sums.)
16. The stitching point can be verified by changing it from 0 to 1 or vice versa. If this change renders the array reducible, then that intersection is indeed, a stitching point.
17. Repeat steps 13 through 16 for the remaining columns of the original array A, with previously found stitching points removed.

4.3 ALGORITHMS FOR STRUCTURAL CROSS-SECTIONS

Some woven fabric structures rely almost entirely on the tension created by special interlacement sequences to produce visual or textural patterning [51], [22, p.274]. These tension effects cannot be meaningfully form represented using the traditional point diagram interpretation of the corresponding binary interlacement array and therefore some alternative form of graphical display is required to identify and visualize these effects. A useful and meaningful of display is the sectional drawdown which represents a cross-section of the fabric structure cut between adjacent weft picks or warp ends.

The automatic display of these sectional drawdowns requires the mapping of the binary interlacement array data onto an appropriate set of graphic output primitives. The ensuing discussion presents newly developed mapping algorithms and optimal sets of graphic primitives for four primary types of fabric.

4.3.1 THE PRIMARY FABRIC TYPE F[1]

Definition 4.3.1.1. The primary fabric type F[1] is defined as that structure corresponding to a single strand t, passing either over or under in turn, a succession of strands k, orthogonal to it, where both t and k lie in the same plane.

The crucial part of this definition is that the warp and weft strands are co-planar. An example of this fabric type is given by the binary interlacement sequence

$$R = \{1,1,0,0,1,1,0,0,\ldots\}$$

which corresponds visually to

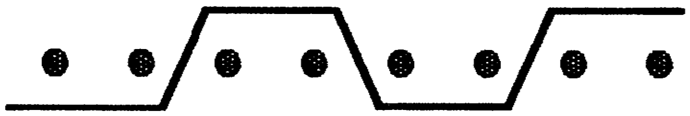


FIGURE4.3.1.2

For convenience, this and subsequent examples will consider the sequence in the weft direction, with the cross-sections cut through the warp ends, although it is understood that the sequence could equally be taken in the warp direction, with the cross-sections cut through the weft picks.

Definition 4.3.1.3. A mid-point centered graphic primitive is an element of a basis set of graphic tiles, centered on the point midway between two adjacent ends (or sets of ends). When drawn, mid-point centered graphic primitives are intended to overlap to the extent that the first ends (or set of ends) of one primitive overlaps the second end (or ends) of the previous one.

Definition 4.3.1.4. An end centered graphic primitive is an element of the basis set of graphic tiles, centered on an end (or set of ends). There will be no overlap when these tiles are drawn.

THEOREM 4.3.1.5. For the primary fabric type F[1], four mid-point centered graphic primitives are sufficient (without rotation or reflection allowed).

Proof. The elements of this set are:

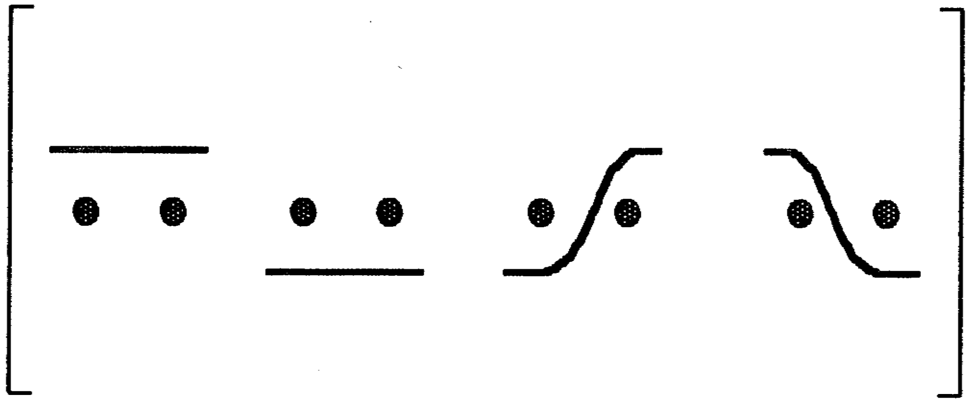


FIGURE4.3.1.6

<u>**COROLLARY 4.3.1.7.**</u> The mapping from the interlacement sequence R to the corresponding set of graphic primitives S for the primary fabric type F[1] is obtained by ordering the set S and computing the index i, where

(4.3.1.8) $\quad i = 2\,r_j + r_{j+1} \qquad\qquad j = 1, 2, \ldots, n-1.$

<u>Proof</u>. The ordering of the graphic primitives S can be defined as

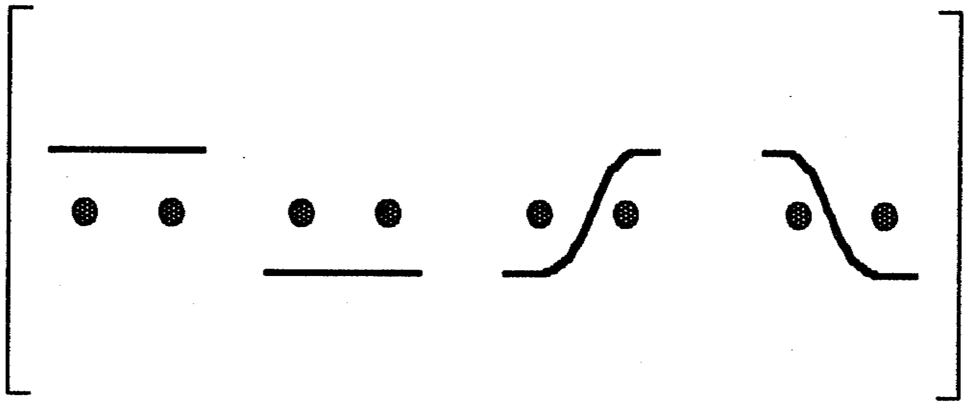


FIGURE4.3.1.9

Interpreting each tile as corresponding to a 2 place binary interlacement sequence and computing the index i for each of them gives the set of decimal integers {0,1,2,3} which will uniquely identify each of the four graphic primitives.

<u>THEOREM 4.3.1.10.</u> For the primary fabric type F[1], eight end-centered graphic primitives are required (without rotation or reflection allowed).

<u>Proof</u>. The elements of this set are:

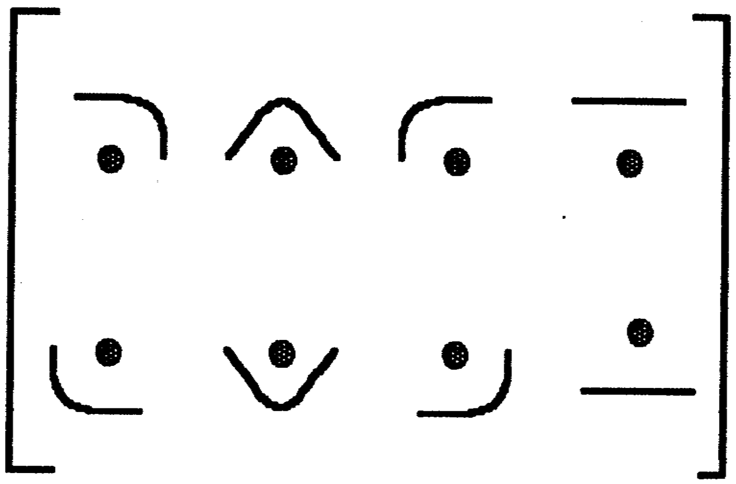


FIGURE4.3.1.11

<u>COROLLARY 4.3.1.12.</u> The mapping from the interlacement sequence R to the corresponding set of graphic primitives Z for the primary fabric type F[1] is obtained by ordering the set Z and computing the index i, where

(4.3.1.13)

$$
\begin{array}{ll}
i = 4 + 2r_j + r_{j+1} & j = 1 \\
i = 4r_{j-1} + 2r_j + r_{j+1} & j = 2,3,\ldots,n-1 \\
i = 4r_{j-1} + 2r_j + 1 & j = n.
\end{array}
$$

<u>Proof</u>. The ordering of the graphic primitives Z can be defined as

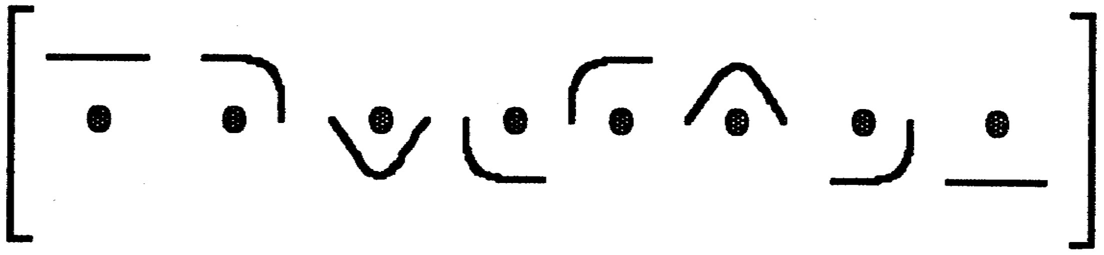


FIGURE4.3.1.14

Computing an index value i for all possible binary interlacement sequences of length three produces the eight integers, 0 through 7, inclusive. These can be used to uniquely identify each of the eight graphic primitives.

The set of end centered graphic primitives contains twice as many elements as the set of mid-point centered primitives. The end centered system also requires the computation of one extra term in determining the index value, as well as requiring that the first and last elements of the binary interlacement sequence R be handled as exceptional cases. For these reasons the mid-point centered graphic system is considered to be computationally more efficient and only primitives using this system will be discussed in the subsequent sections.

An example of a structural cross-section for a primary fabric type F[1] is given in Figure [4.3.1.15].

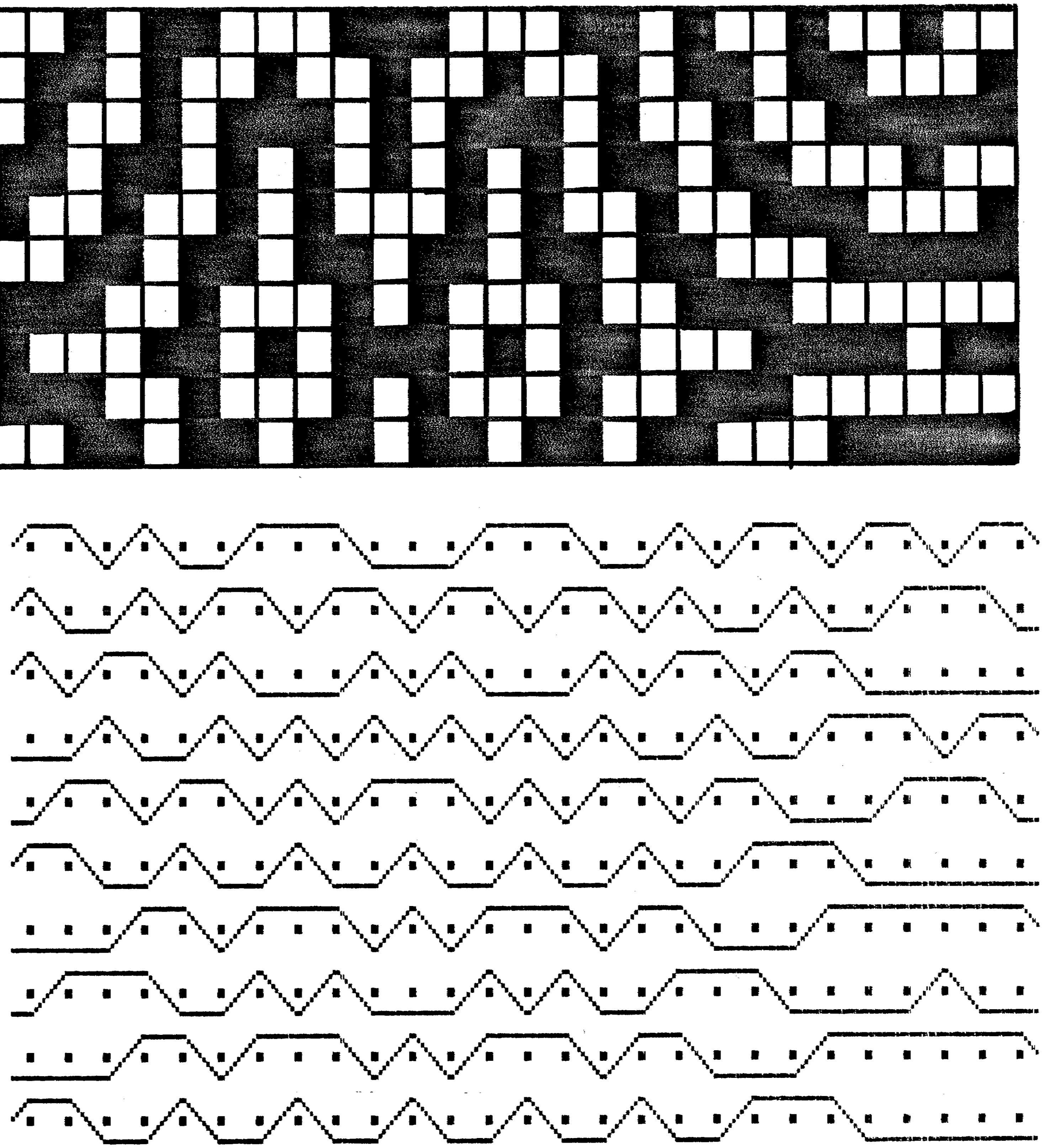


FIGURE 4.3.1.15

INTERLACEMENT ARRAY INTERPRETED AS AN F[1] FABRIC

4.3.2 THE SECONDARY FABRIC TYPE F[2]

Definition 4.3.2.1. The secondary fabric type F[2] is defined as that structure corresponding to a single strand t passing either over or under in turn, a set k of strands orthogonal to it. This structure is composed of two parallel fabric planes with the elements of k, and all of the t-strands, lying in these two planes or oscillating between them. For the fundamental type F[2:1:1], each fabric plane contains precisely half of the k strands and half of the t strands at any one time. In general, F[2:i:k] is a secondary fabric type F[2] such that the relative proportion of strands per unit length, in the two layers, is uniformly i:k (for example, F[2:1:2]). Since F[2:1:1] is the most frequently occurring structure of this type, it is convenient to abbreviate F[2:1:1] to F[2].

THEOREM 4.3.2.2. For the secondary fabric type F[2], nine mid-point centered graphic primitives are sufficient (without rotation or reflection allowed).

Proof. The ordered set S of these elements is:

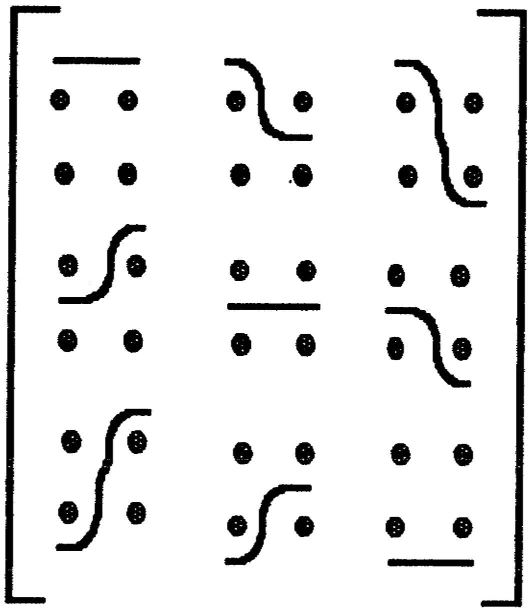


FIGURE4.3.2.3

<u>COROLLARY 4.3.2.4.</u> The mapping from the interlacement sequence R to the corresponding ordered set of graphic primitives S for the secondary fabric type F[2] is obtained by computing the index i where

(4.3.2.5) $$i = 3(r_j + r_{j+1}) + r_{j+2} + r_{j+3} \qquad j = 1,3,5,\ldots,n-3$$

The number of elements in the sequence R must be divisible by 2. If it is not, the final element is dropped from R.

<u>Proof</u>. The proof follows <u>mutatis mutandis</u> from Corollary [4.3.1.7].

Two facts about this mapping can be noted immediately. The first observation is that the values of i are not distinct for all of the possible binary interlacement sequences of length four. The second observation is that this mapping assumes that the left-hand intersection of a pair is always in the top layer and the right-hand intersection is always in the bottom layer. This leads to a discussion of what will be termed a coercive interaction.

Definition 4.3.2.6. A coercive interaction in the secondary fabric type F[2] occurs when two successive intersections unambiguously define the planar relationship between the corresponding warp ends.

If a weft pick lies between two warp ends, one of which is in the top plane and the other of which is in the bottom plane, then this weft pick constrains these two ends in their current positions. If, on the other hand, a weft pick lies either on top of or underneath two ends belonging to different fabric planes, then there exists some ambiguity as to which of these two ends belongs in the top layer and which belongs in the bottom layer. For example, in Figure (4.3.2.7), the F[2] representation of the interlacement sequence (1 0 0 0), illustrates a coercive interaction between the first pair of intersections but not between the second pair. Warp end number 1 must be on top of warp end number 2, whereas the relative planar positions of ends 3 and 4 are not fixed by this interlacement sequence and form a non-coercive interaction.

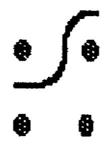


FIGURE 4.3.2.7

Definition 4.3.2.8. An exchange interaction is defined as a special case of coercive interaction in which natural yarn tension would cause two successive warp ends to exchange their relative positions. That is, the physical constraints of yarn tension in a woven fabric structure would render this interaction impossible.

Figure {4.3.2.9}, an F[2] representation of the interlacement sequence {0 1 0 0}, illustrates such an interaction.

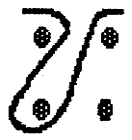


FIGURE 4.3.2.9

The effect of tension in this interaction would be to exchange the relative positions of warp ends 1 and 2 without affecting the relationship between ends 3 and 4. The tile required to illustrate this altered sequence is the one illustrated in Figure {4.3.2.7}.

It is convenient to label individual warp ends which occur in a coercive interaction so as to identify planar position changes as a result of an exchange interaction. Figure {4.3.2.10}, for example, illustrates the F[2] representation of the interlacement sequence {1 0 0 0 0 1}, where intersections 1 and 2 form a coercive interaction, intersections 3 and 4 form a non-coercive interaction and intersections 5 and 6 form a coercive exchange interaction.

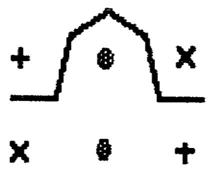


FIGURE 4.3.2.10

THEOREM 4.3.2.11. Labels to identify warp ends occurring in a coercive interaction in the secondary fabric type F[2] can be indexed by computing

(4.3.2.12) $$i = 2r_j + r_{j+1} \qquad j = 1,3,5,\ldots,n-1.$$

Proof. $i \in \{0,1,2,3\}$.

If $i \in \{0,3\}$, then the F[2] representation of the interaction corresponds to Figures {4.3.2.13} and {4.3.2.14}, respectively. Since neither of these is a coercive interaction, no labelling changes are necessary.

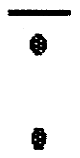


FIGURE 4.3.2.13

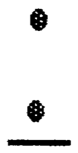


FIGURE 4.3.2.14

If i = 1, then the corresponding F[2] interaction is given by Figure {4.3.2.15}. Since this is a coercive interaction, the warp ends must be re-labelled to indicate that the ends have exchanged positions. Label 1 could be given by Figure {4.3.2.16}.

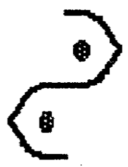


FIGURE 4.3.2.15

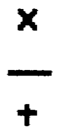


FIGURE 4.3.2.16

If i = 2, then the corresponding F[2] interaction is given by Figure {4.3.2.17}. Since this is a coercive interaction, the warp ends must be re-labelled to indicate that warp end number 1 must lie on top of warp end number 2. Label 2 would then correspond to Figure {4.3.2.18}.

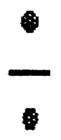


FIGURE 4.3.2.17

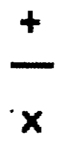
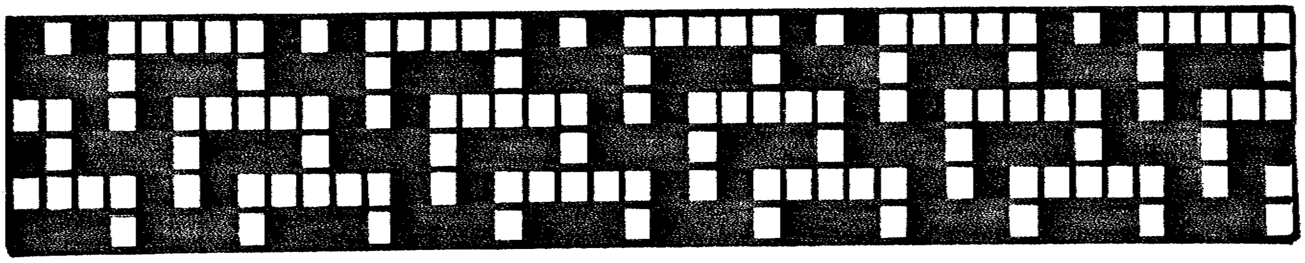


FIGURE 4.3.2.18

An example of a structural cross-section for the secondary fabric type F[2] is given in Figure {4.3.2.19}.



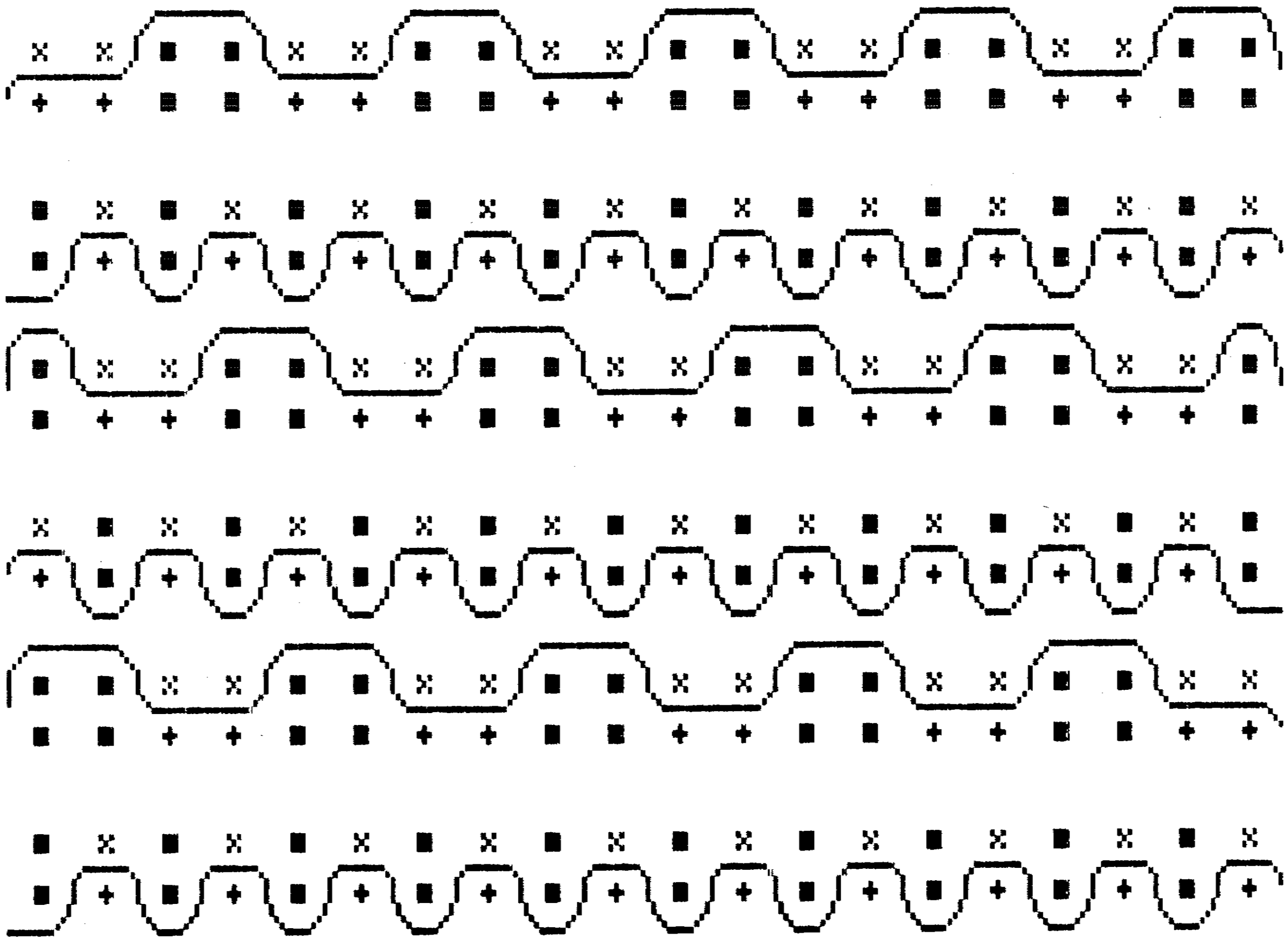


FIGURE 4.3.2.19

INTERLACEMENT ARRAY INTERPRETED AS AN F[2] CROSS-SECTION

DISJOINT LAYERS

4.3.3 <u>THE SECONDARY FABRICS</u> F[2:1:2] <u>AND</u> F[2:1:3]

<u>**THEOREM 4.3.3.1**</u>. For the secondary fabric F[2:1:2], eighteen mid-point centered graphic primitives are sufficient (without rotation or reflection allowed).

<u>Proof</u>. The ordered set S of these elements is shown in Figure (4.3.3.2).

<u>**COROLLARY 4.3.3.3**</u>. The mapping from the interlacement sequence R to the corresponding ordered set of graphic primitives S for the secondary fabric F[2:1:2] is obtained by computing the index i where

$$(4.3.3.4) \qquad i = 6(r_j + r_{j+1}) + 3r_{j+2} + r_{j+3} + r_{j+4} \qquad j = 1,4,7,\ldots,n-4.$$

The interlacement sequence R must consist of, or be reduced to, 3k + 2 elements, where k is an integer.

Proof. The proof follows <u>mutatis mutandis</u> from Corollary (4.3.1.7).

<u>**COROLLARY 4.3.3.5**</u>. Labels to identify warp ends occurring in a coercive interaction in the secondary fabric F[2:1:2] can be indexed by computing

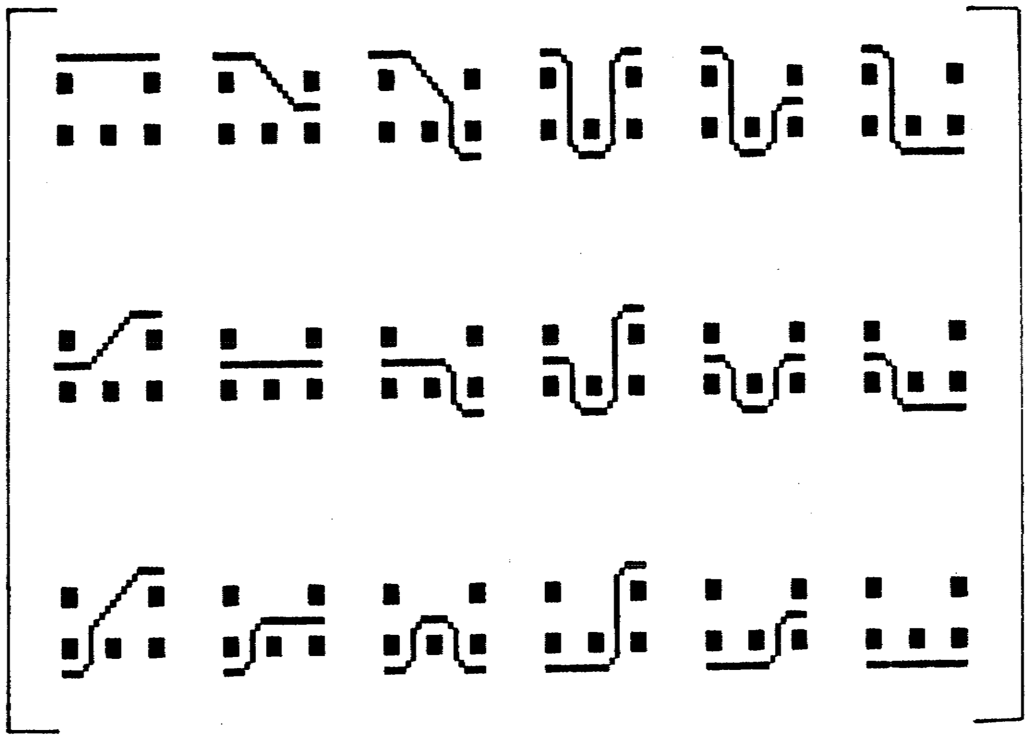


FIGURE 4.3.3.2

(4.3.3.6) $i = 2r_j + r_{j+1}$ $j = 1,4,7,\ldots,n-1.$

Proof. Labels for interactions can be indexed by i, as in Theorem (4.3.2.11). Since the assumption is made that the relative proportion of yarns per layer remains constant, the warp ends corresponding to r_j, $j = 3,6,9,\ldots,n-2$ never occur as part of a pair of intersections. These warp ends will never occur in a coercive interaction and thus require no re-labelling. Therefore, only the pairs of intersections need to be examined for coercion.

THEOREM 4.3.3.7. For the secondary fabric F[2:1:3], thirty-six mid-point centered graphic primitives are sufficient (without rotation or reflection allowed).

Proof. The ordered set S of these elements is shown in Figure (4.3.3.8).

COROLLARY 4.3.3.9. The mapping from the interlacement sequence R to the corresponding ordered set of graphic primitives S for the secondary fabric F[2:1:3] is obtained by computing the index i where

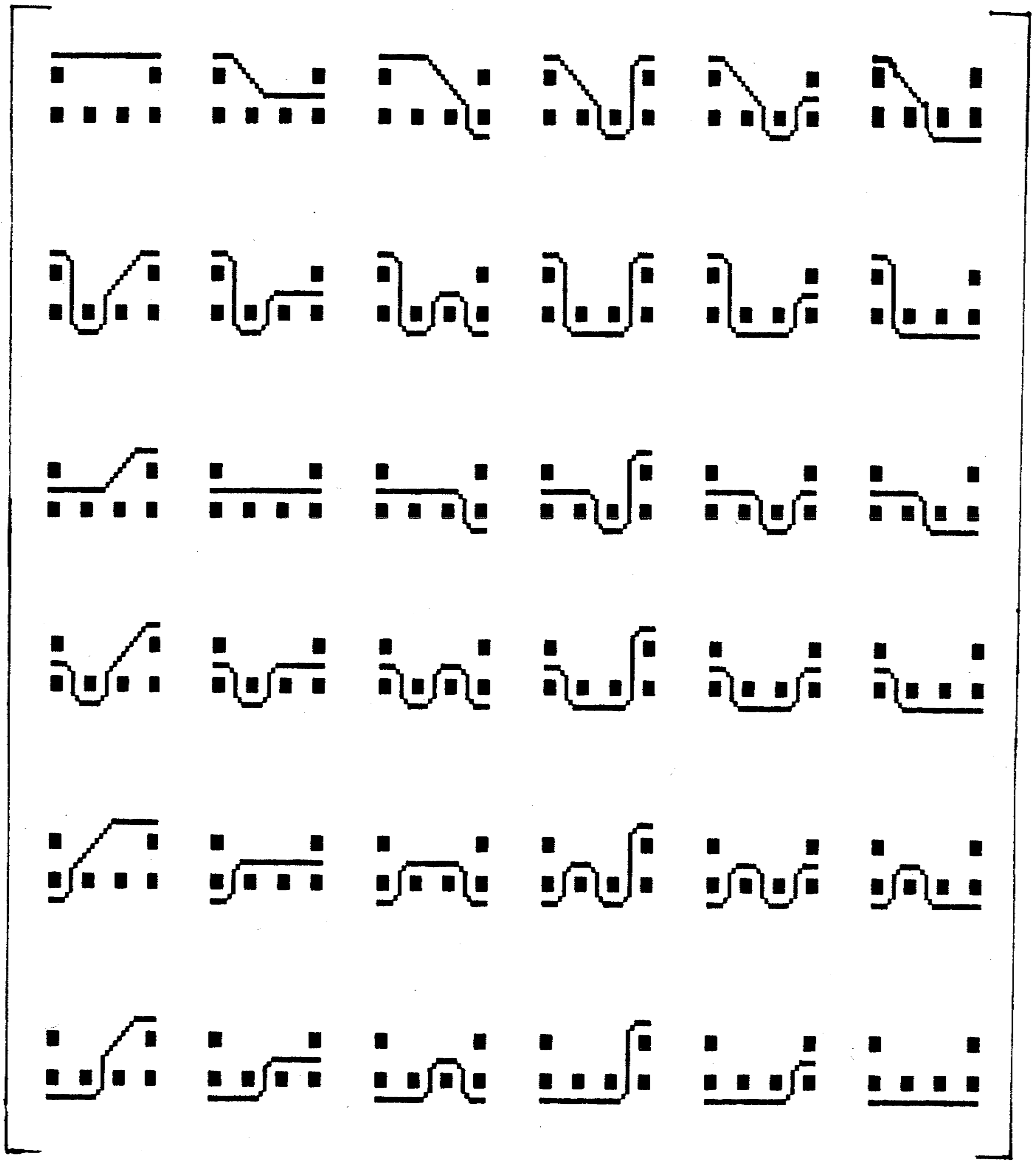


FIGURE 4.3.3.8

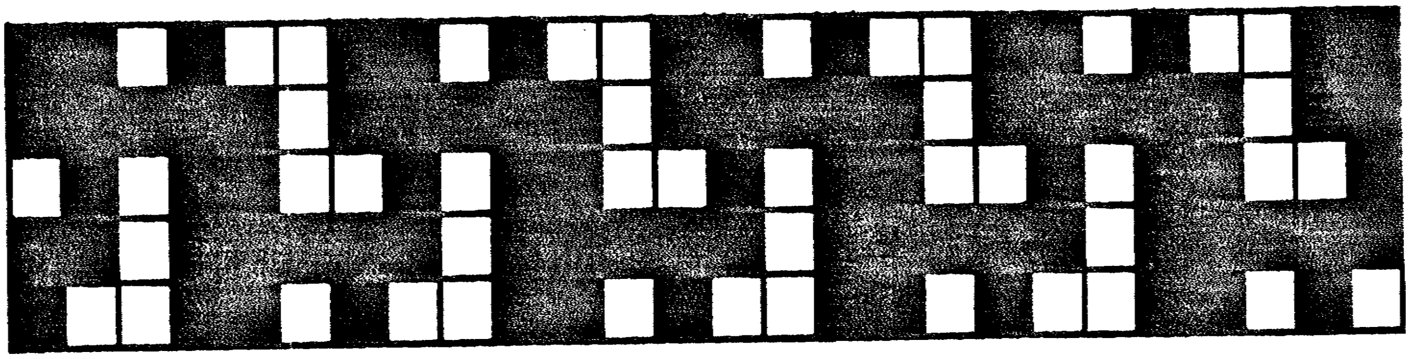
(4.3.3.10) $$i = 12(r_j + r_{j+1}) + 6r_{j+2} + 3r_{j+3} + r_{j+4} + r_{j+5}$$

$$j = 1,5,9,\ldots,n-5.$$

The interlacement sequence must consist of, or be reduced to, 4k + 2 elements, where k is an integer.

Proof. The proof follows mutatis mutandis from Corollary (4.3.3.5).

An example of a structural cross-section of the secondary fabric type F[2:1:2] is given in Figure (4.3.3.11).



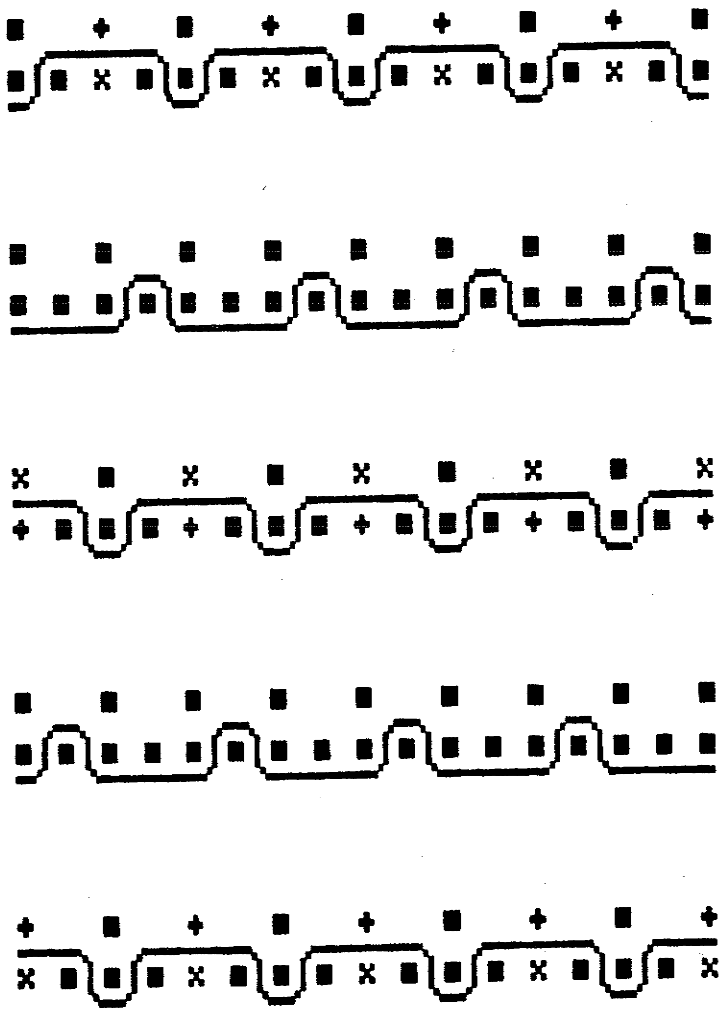


FIGURE 4.3.3.11

INTERLACEMENT ARRAY INTERPRETED AS AN F[2:1:2] CROSS-SECTION

4.3.4 THE TERNARY FABRIC TYPE F[3]

Definition 4.3.4.1. The ternary fabric type F[3] is defined as that structure corresponding to a single strand t passing either over or under in turn, a set k of strands, orthogonal to it. This structure is composed of three parallel fabric planes with the elements of k, and all of the t-strands, lying in these three planes or oscillating between them. For the fundamental type F[3:1:1:1], each fabric plane contains precisely one third of the k strands and one third of the t-strands at any one time. In general F[3:i:k:m] is a ternary fabric type F[3] such that the relative proportion of strands per unit length, in the three layers, is uniformly i:k:m (for example: F[3:1:2:3]). The only structure of this type which will be considered is F[3:1:1:1], which will be abbreviated to F[3].

THEOREM 4.3.4.2. For the ternary fabric type F[3], sixteen mid-point centered graphic primitives are sufficient (without rotation or reflection allowed).

Proof. The ordered set S of these elements is shown in Figure (4.3.4.3).

COROLLARY 4.3.4.4. The mapping from the interlacement sequence R to the corresponding ordered set of graphic primitives S for the ternary fabric type F[3] is obtained by computing the index i where

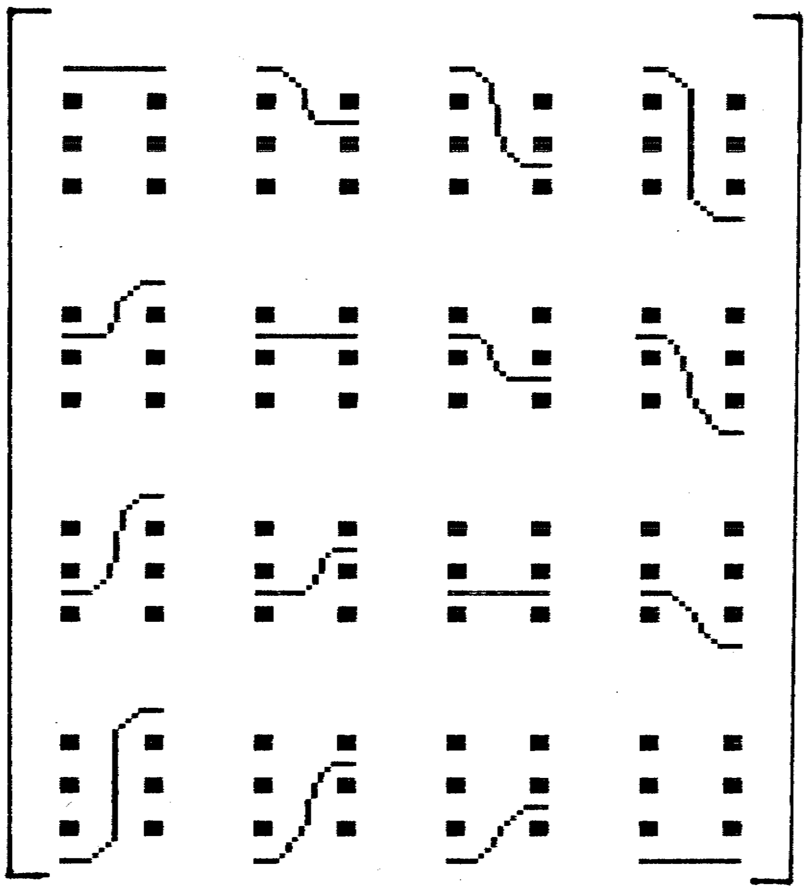


FIGURE 4.3.4.3

(4.3.4.5) $i = 4(r_j + r_{j+1} + r_{j+2}) + r_{j+3} + r_{j+4} + r_{j+5} \qquad j = 1,4,7,\ldots,n-5.$

The number of elements in the sequence R must be divisible by 3. If it is not, the final elements(s) are dropped from R.

Proof. The proof follows mutatis mutandis from Corollary (4.3.1.7}.

COROLLARY 4.3.4.6. Four distinct labels are required to identify warp ends occurring in a coercive interaction in the ternary fabric F[3]. These labels can be indexed by computing

(4.3.4.7) $$i = 4r_j + 2r_{j+1} + r_{j+2} \qquad j = 1,4,7,\ldots,n-2$$

as an index to the ordered set of label tiles Z in Figure (4.3.4.8}. If i belongs to (0,7} no re-labelling is required.

$$\begin{bmatrix} x & + & \bullet & = & = & \bullet \\ \bullet & = & \bullet & \bullet & x & \bullet \\ \bullet & x & = & \bullet & + & x \end{bmatrix}$$

FIGURE4.3.4.8

<u>Proof.</u> i $\in$ {0,1,...,7}.

If i $\in$ {0,7}, then the F[3] representation of the interaction is not coercive and no re-labelling is required.

If i = 1, then the bottom layer intersection moves to the top layer. The relative positions of the remaining two intersections, with respect to each other, are indeterminate.

If i = 2, then the relative positions of all of the intersections are constrained and an exchange of the first and second layer intersections takes place.

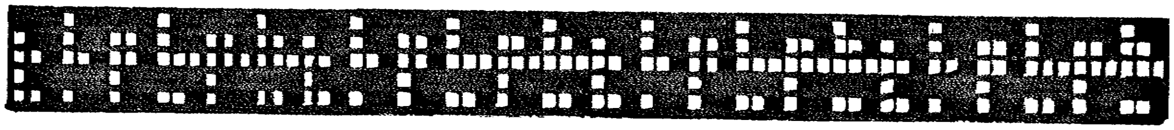
If i = 3, then the top layer intersection moves to the bottom layer. The relative positions of the remaining two intersections with respect to each other are indeterminate.

If i = 4, then the position of the top layer intersection is constrained but no exchange takes place. The relative positions of the remaining intersections are indeterminate.

If i = 5, then the relative positions of all of the intersections are constrained and an exchange of the second and third layer intersections takes place.

If i = 6, then the position of the bottom layer intersection is constrained but no exchange takes place. The relative positions of the remaining intersections are indeterminate.

An example of the ternary fabric type F[3] is given in Figure {4.3.4.9}.



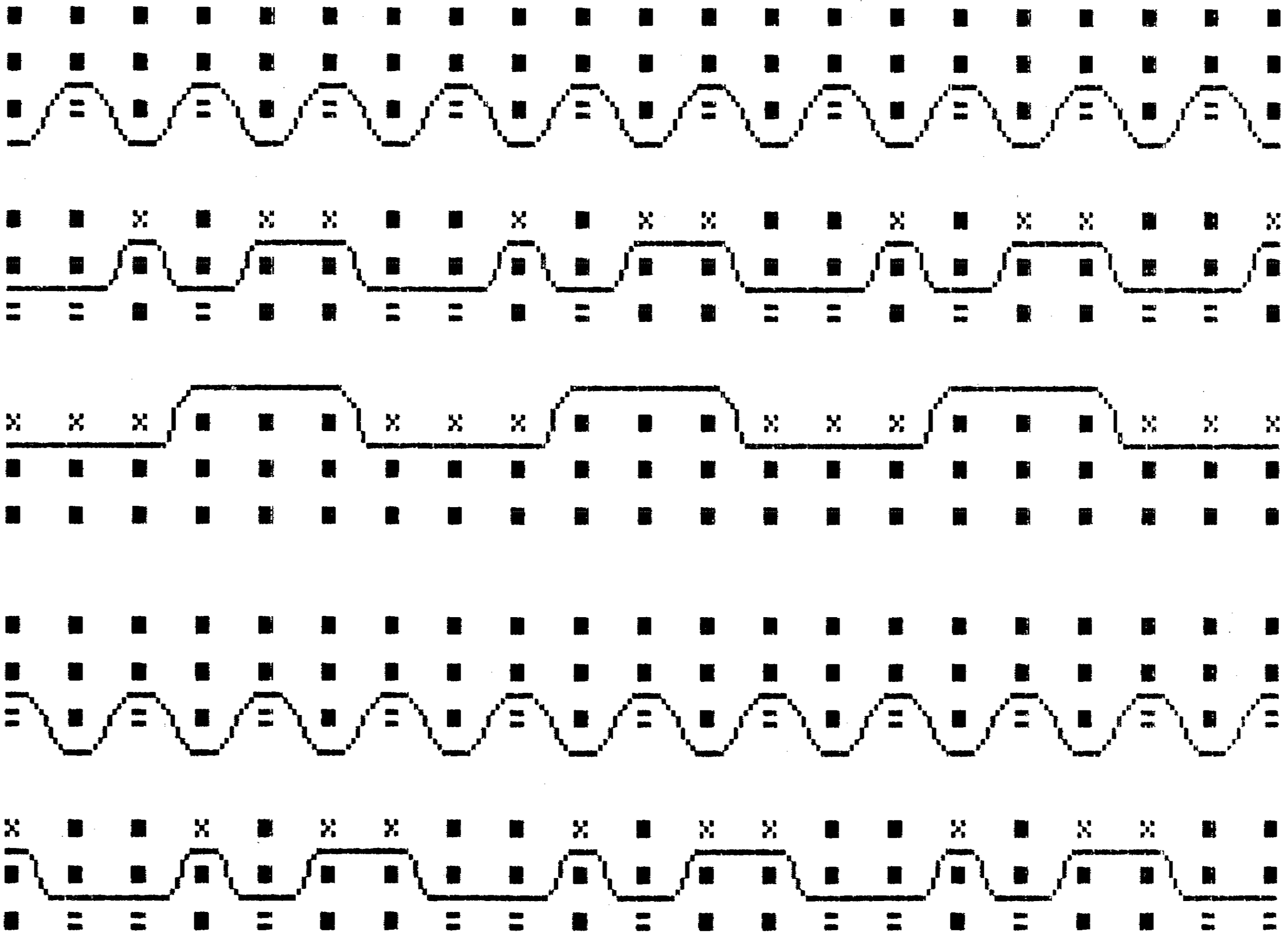


FIGURE 4.3.4.9

INTERLACEMENT ARRAY INTERPRETED AS AN F[3] CROSS-SECTION

DISJOINT LAYERS

4.3.5 THE QUATERNARY FABRIC TYPE F[4]

Definition 4.3.5.1. The quaternary fabric type F[4] is defined as that structure corresponding to a single strand t passing either over or under in turn, a set k of strands orthogonal to it. This structure is composed of four parallel fabric planes with the elements of k, and all of the t-strands, lying in these four planes or oscillating between them. For the fundamental type F[4:1:1:1:1], each fabric plane contains precisely one fourth of the k strands and one fourth of the t-strands at any one time. In general F[4:i:k:m:n] is a quaternary fabric type F[4] such that the relative proportion of strands per unit length, in the four layers, is uniformly i:k:m:n (such as F[4:1:2:3:4]). The only structure of this type which will be considered is F[4:1:1:1:1], which will be abbreviated to F[4].

THEOREM 4.3.5.1. For the quaternary fabric type F[4], twenty-five mid-point centered graphic primitives are sufficient (without rotation or reflection allowed).

Proof. The ordered set S of these elements is shown in Figure (4.3.5.2).

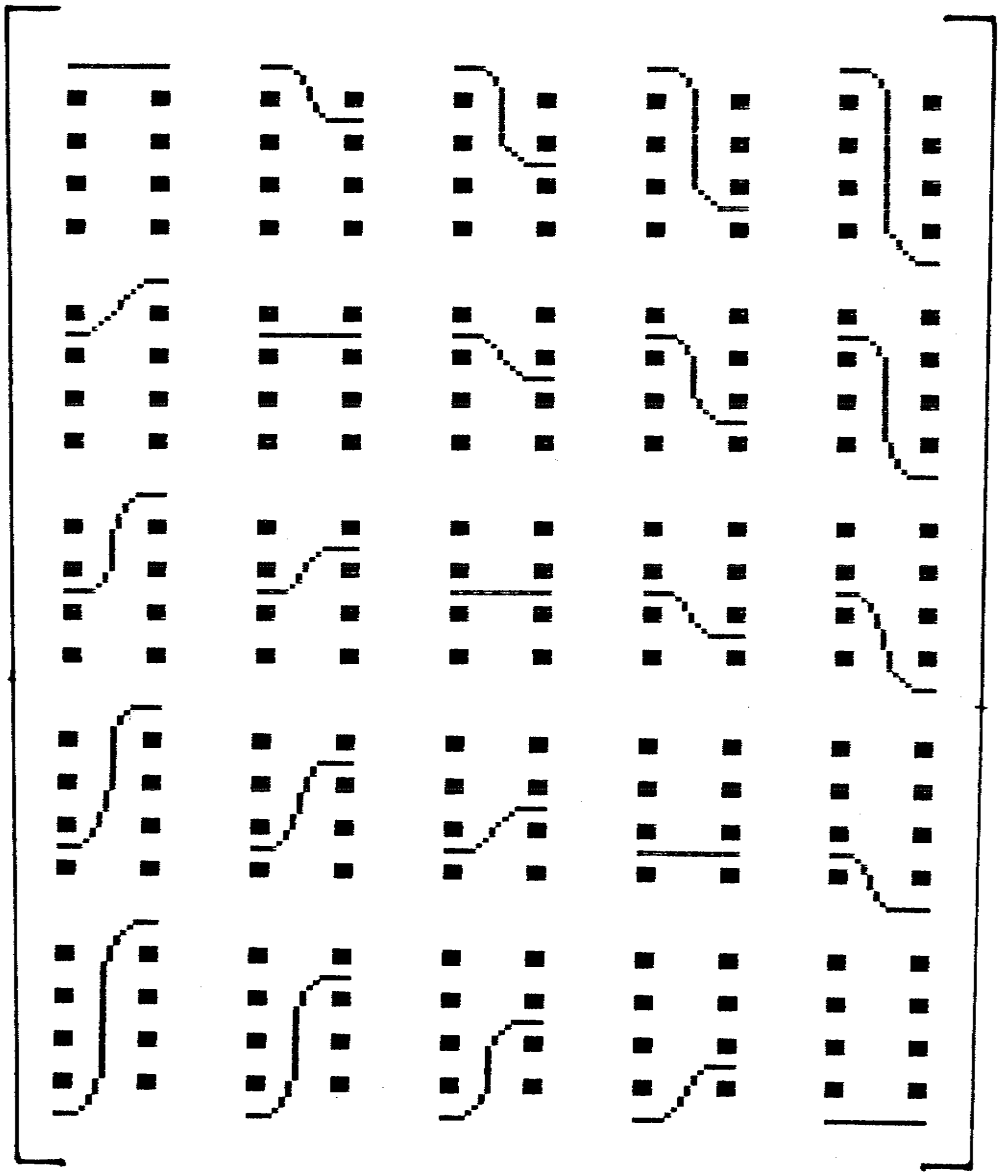


FIGURE 4.3.5.2

<u>COROLLARY 4.3.5.3</u>. The mapping from the interlacement sequence R to the corresponding order set of graphic primitives S for the quaternary fabric type F[4] is obtained by computing the index i where

(4.3.5.4) $$i = 5(r_j + r_{j+1} + r_{j+2} + r_{j+3}) + r_{j+4} + r_{j+5} + r_{j+6}$$

$$j = 1,5,9,\ldots,n-7.$$

The number of elements in the sequence R must be divisible by 4. If it is not, the final element(s) is dropped from R.

<u>Proof</u>. The proof follows <u>mutatis mutandis</u> from Corollary (4.3.1.7).

<u>COROLLARY 4.3.5.5</u>. Fourteen distinct labels are required to identify warp ends occurring in a coercive interaction in the quaternary fabric F[4]. These labels can be indexed by computing

(4.3.5.6) $$i = 8r_j + 4r_{j+1} + 2_{j+2} + r_{j+3} \qquad j = 1,5,9,\ldots,n-3$$

as an index to the ordered set of label tiles Z in Figure (4.3.5.7). If i belongs to {0,15}, no re-labelling is required.

$$\begin{bmatrix} + & \times & = & = & \blacksquare & \blacksquare & \| & \blacksquare & \| & \| & \| & \blacksquare & \blacksquare \\ \blacksquare & \times & \| & \times & \blacksquare & \blacksquare & \blacksquare & \times & + & \blacksquare & \| & \blacksquare & \blacksquare \\ \blacksquare & \| & \blacksquare & \| & \| & \blacksquare & \blacksquare & \blacksquare & = & \blacksquare & \times & \times & \blacksquare \\ \times & \| & \blacksquare & + & + & \| & \blacksquare & \blacksquare & \times & = & \times & + & \times \end{bmatrix}$$

FIGURE4.3.5.7

<u>Proof</u>. i ∈ {0,1,...,15}.

If i ∈ {0,15}, then the F[4] representation of the interaction is not coercive and no re-labelling is required.

If i = 1, then the bottom layer intersection moves to the top layer. The relative positions of the remaining three intersections, with respect to each other, are indeterminate.

If i = 2, then the third layer intersection moves to the top layer. The relative positions of the remaining three intersections, with respect to each other, are indeterminate.

If i = 3, then the intersections in the top two fabric layers exchange with the bottom and second bottom layers. The intersections within each pair of intersections are indeterminate with respect to each other.

If i = 4, then the second layer intersection moves to the top layer. The relative positions of the remaining three intersections, with respect to each other, are indeterminate.

If i = 5, then the second and fourth layer intersections are constrained in the top two layers and the first and third layer intersections are constrained in the bottom two layers. The relative positions of intersections within these two pairs of intersections are indeterminate.

If i = 6, then the second and third layer intersections are constrained in the top two layers and the first and fourth layer intersections are constrained in the bottom two layers. The relative positions of intersections within these two pairs of intersections are indeterminate.

If i = 7, then the top layer intersection moves to the bottom layer. The relative positions of the remaining three intersections are indeterminate with respect to each other.

If i = 8, then the top layer intersection is constrained in the top layer. The relative positions of the remaining three intersections are indeterminate with respect to each other.

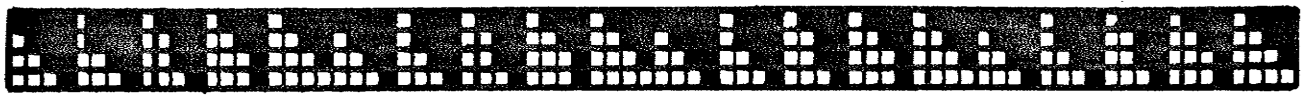
If i = 9, then the first and fourth layer intersections are constrained in the top two layers and the second and third layer intersections are constrained in the bottom two layers. The relative positions of intersections within each pair of intersections are indeterminate.

If i = 10, then the first and third layer intersections are constrained in the top two layers and the second and fourth layer intersections are constrained in the bottom two layers. The relative positions of intersections within each pair of intersections are indeterminate.

If i = 11, then the second layer intersection moves to the bottom layer. The relative positions of the remaining three intersections are indeterminate with respect to each other.

If i = 12, then the first and second layer intersections are constrained in the top two layers and the third and fourth layer intersections are constrained in the bottom two layers. The relative positions of intersections within each pair of intersections are indeterminate.
If i = 13, then the third layer intersection moves to the bottom layer. The relative positions of the remaining three intersections are indeterminate with respect to each other.
If i = 14, then the bottom layer intersection is constrained in the bottom layer. The relative positions of the remaining three intersections are indeterminate with respect to each other.

An example of a structural cross-section for the quaternary fabric type F[4] is given in Figure (4.3.5.8).



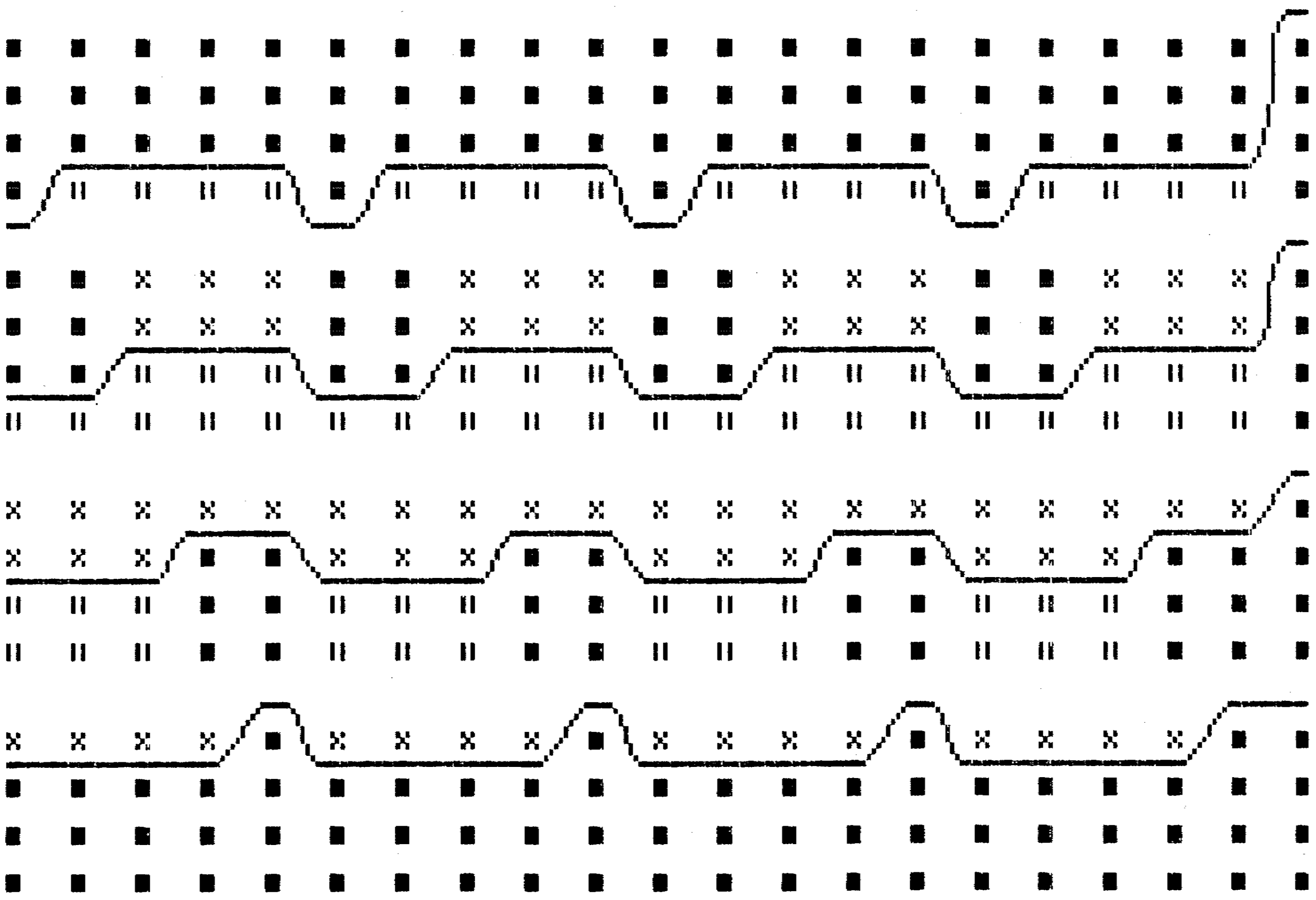


FIGURE 4.3.5.8

INTERLACEMENT ARRAY INTERPRETED AS AN F[4] CROSS-SECTION

DISJOINT LAYERS

4.3.6 OBSERVATIONS AND SUMMARY

Several observations can be made about the number of graphic tiles required to represent a given structure as well as about the nature of the mapping algorithm. These observations are summarized as follows:

1. The number of graphic tiles required to represent a fundamental structure (that is, one in which equal proportions of yarns lie in each fabric layer) can be determined by evaluating

(4.3.6.1) $n = (L + 1)^2$, where L is the number of layers.

2. The index value i for all fundamental structures can be computed from the interlacement sequence, where

(4.3.6.2) $$i = ((L + 1) \sum_{j=1}^{L} r_j) + \sum_{j=1}^{L} r_{L+j}.$$

3. The number of graphic tiles required to represent an F[2:1:k] fabric structure is given by

(4.3.6.3) $$n = 9 \times 2^{k-1}.$$

4. The index value i for F[2:1:k] fabric structures can be computed from the interlacement sequence, where

$$i = ((3k)^2\ (r_1 + r_2)) + \sum_{j=1}^{k} (3j)\, r_{j+2} + \sum_{j=1}^{2} r_{j+k+2}\,. \qquad (4.3.6.4)$$

5. The index value i for the re-labelling tiles to indicate coercive and exchange interactions can be computed from the interlacement sequence simply by interpreting the corresponding sequence as an L place binary number and evaluating the decimal equivalent.

CHAPTER 5
FLAT SECTIONAL REPRESENTATIONS

CONTENTS

5.1 INTRODUCTION

Definition 5.1.1. Cartesian woven structures are formed by the interlacement of two sets of parallel strands which lie orthogonal to one another, with the resulting intersections of these two strands lying in a plane. All conventionally woven fabrics are of this type.

Definition 5.1.2. Single layer cartesian woven structures correspond to irreducible binary interlacement arrays, where all of the intersections form a single fabric layer.

Definition 5.1.3. Double layer cartesian woven structures are defined to be reducible or partially reducible binary interlacement arrays corresponding to two fabric layers, with or without stitching points, which have been uniformly interleaved. These binary interlacement arrays are therefore two dimensional projections of three dimensional layered structures.

Definition 5.1.4. Non-cartesian woven structures are defined as those structures whose warp strands are no longer parallel, but are allowed to cross over each other. Cross woven fabrics, such as leno, gauze [60, p. 211-247] and sprang [12] are of this type.

Definition 5.1.5. A flat sectional representation is defined as a top view of the fabric corresponding to a given interlacement data structure.

In the case of cartesian structures, flat sections provide an alternative graphical representation which gives a clear illustration of the interaction between the warp and weft strands. The representations are useful for fabric design and are particularly well suited to the illustration of manuscripts. In the case of non-cartesian structures however, flat sections constitute the clearest and most effective graphical representation of the corresponding interlacement data. The data entry environment for such non-cartesian design data must therefore be appropriate for this type of graphical display.

This chapter will outline the process of developing an appropriate set of graphic output primitives for the flat sectional representation of single and double layer cartesian structures, as well as non-cartesian structures. The mapping algorithm from the data structure to the graphical display will also be discussed. In addition, a discussion of the design of a graphics editor for the data entry and display environment required for non-cartesian structures will be included.

5.2 <u>REPRESENTATION OF SINGLE LAYER CARTESIAN STRUCTURE</u>

A flat sectional representation of a single layer cartesian structure depicts the visual appearance of the interlacing warp and weft strands in a fabric when viewed from the top. It is achieved by the mapping of the corresponding data file to a set of appropriate output primitives. The design of these particular graphic primitives is constrained by the following factors:

1. Each intersection can be depicted independently of all other intersections. This implies that <u>end-centered graphic primitives,</u> which do not overlap when drawn, can be used and that

<u>**THEOREM 5.2.1**</u>. For a single layer cartesian structure representation, two graphic primitives are sufficient.

<u>Proof</u>. The ordered set S of these elements is:

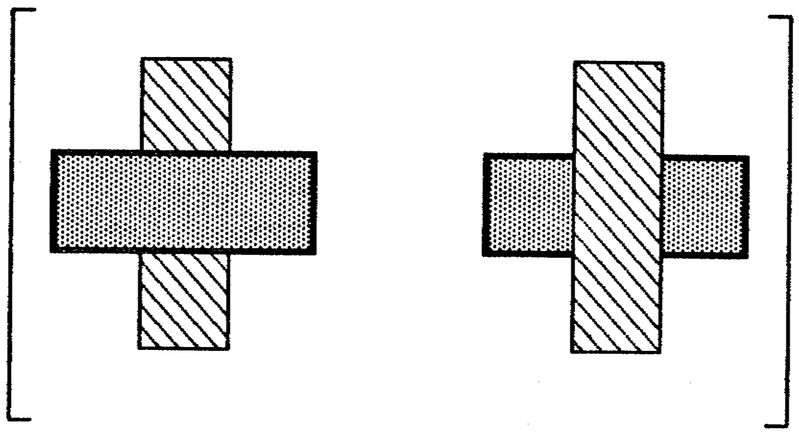


FIGURE5.2.2

<u>COROLLARY 5.2.3</u>. The mapping from an interlacement sequence R to the corresponding ordered set of graphic primitives S for a single layer cartesian structure representation is obtained by computing the index i where

(5.2.4) $$i = r_j + 1 \qquad j = 1,2,\ldots,n.$$

<u>Proof</u>. When $r_j = 0$ then $i = 1$. This corresponds to a weft over warp intersection and the appropriate primitive is indexed. Similarly, when $r_j = 1$ then $i = 2$. This corresponds to a warp over weft intersection and the appropriate primitive is indexed.

2. The warp and weft strands, as well as the empty spaces

between them are drawn. Therefore, delineating the outlines of the strands is not sufficient, since they can be easily confused with the inter-yarn spaces. Some interior pattern is required to define the strands.

3. For maximum differentiation between the warp and weft strands, a significantly different pattern is required for each set of strands.

4. The pattern used for the warp strands must be periodic on the height of the primitive and the pattern used for the weft strands must be periodic on the width of the primitive. This is required so that there will be no discontinuities in matching these graphic tiles when a multi-element representation is constructed.

Figure (5.2.5) is a point diagram representation of an interlacement array and Figure (5.2.6) is a single layer cartesian structure representation of this same data. The visual correlation between these two graphical images is high.

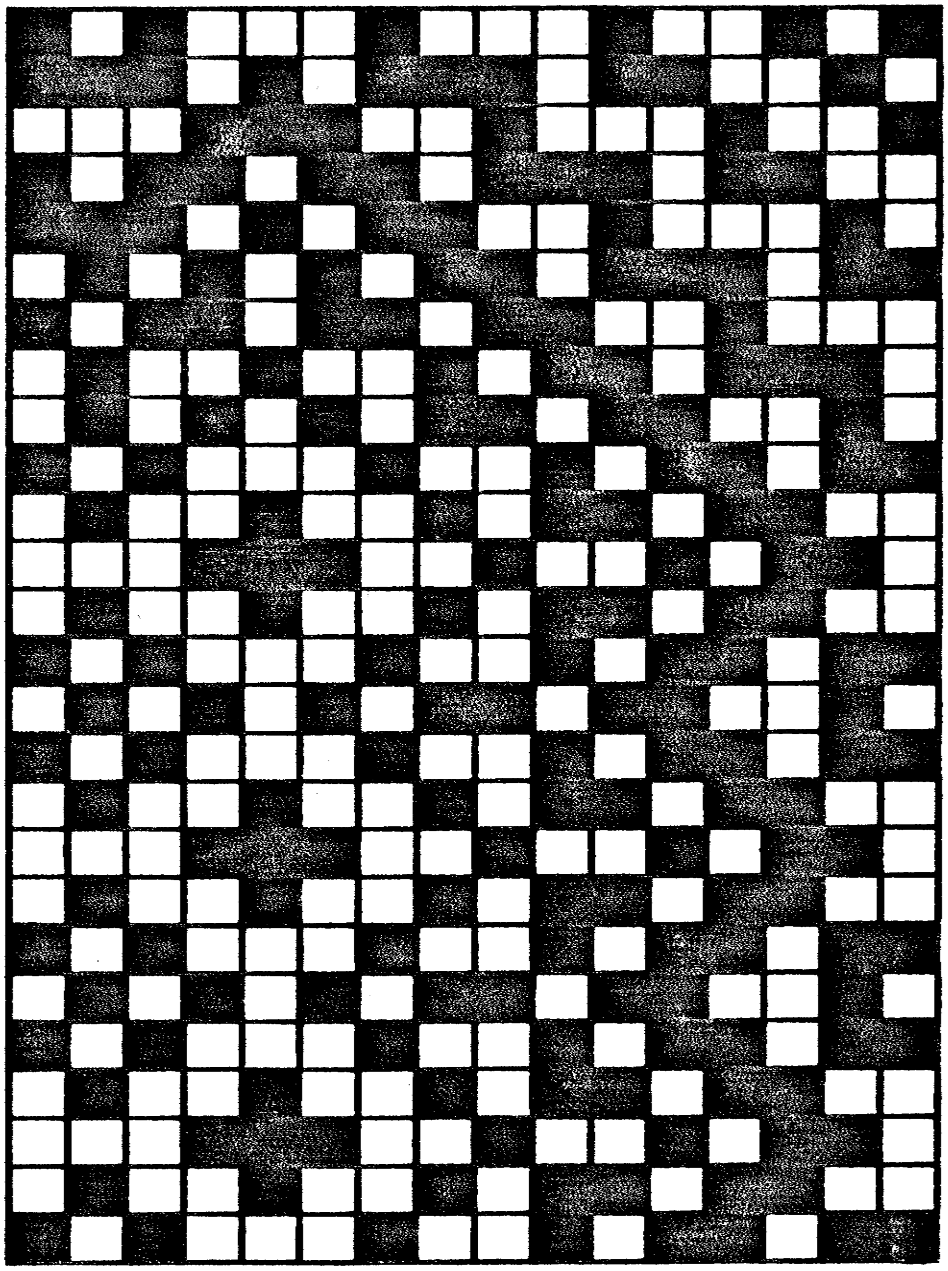


FIGURE 5.2.5

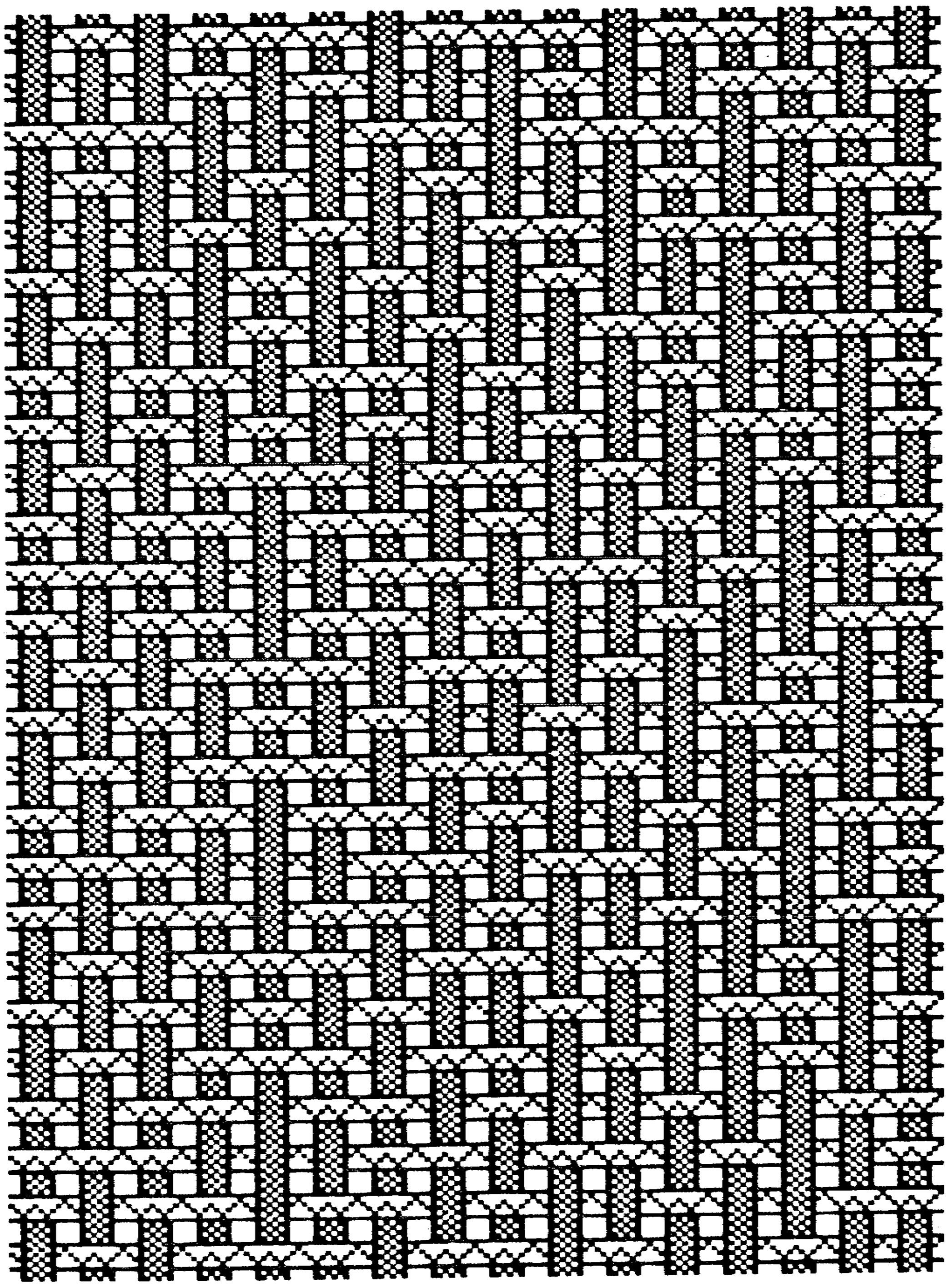


FIGURE 5.2.6

5.3 REPRESENTATION OF DOUBLE LAYER CARTESIAN STRUCTURES

A double layer cartesian structure representation corresponds to the interpretation of an interlacement array as a fundamental F[2] fabric, with the bottom layer fabric visible between the strands of the top layer fabric, when the whole is viewed from the top. Once again, the intersections can be interpreted independently of each other, with the parity of the row and column indices determining in which layer a given intersection lies. With no loss of generality, we can specify that odd numbered columns correspond to top layer warp strands and even numbered columns correspond to bottom layer warp strands. Similarly, odd numbered rows can be thought to correspond to top layer weft strands, while even numbered rows correspond to bottom layer weft strands.

As in the case of the single layer flat sectional representation, there are a number of criteria for choosing a particular set of graphic output primitives or tiles, and the corresponding mapping algorithm. These criteria are as follows:

1. As before, some differentiation must be made between the warp and weft strands of the two layers and the inter-yarn gaps. Some interior pattern must therefore appear on the strands.

2. For maximum differentiation between the warp and weft strands, the pattern for the warp strands must differ significantly from the pattern for the weft strands, for each fabric layer. Four patterns are therefore required in total.

3. The warp strand patterns must be periodic on the height of the tile and the weft strand patterns must be periodic on the width of the tile.

4. The tiles must be large enough to accommodate the need for visual distinction between the patterns but must also be small enough so that a reasonable number of intersections can be represented.

<u>**THEOREM 5.3.1**</u>. For a double layer cartesian structure representation, eight graphic primitives are sufficient.

<u>Proof</u>. The ordered set S of these elements is:

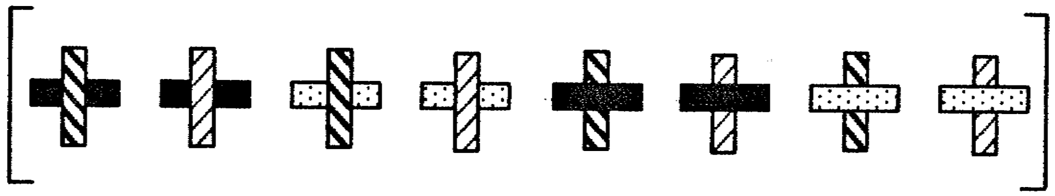


FIGURE5.3.2

<u>COROLLARY 5.3.3</u>. The mapping from an interlacement sequence R to the corresponding ordered set of graphic primitives S for a double layer cartesian structure is obtained by computing the index i where

(5.3.4) $$i = 1 + (4r_{k,j} + 2(k - 2 \times \lceil k/2 \rceil) + (j - 2 \times \lceil j/2 \rceil))$$

$$k = 1,2,\ldots,m$$
$$j = 1,2,\ldots,n.$$

<u>Proof</u>. Table (5.3.5) indicates the correspondence between the possible intersection types and row and column parities, the computed value for the index i, and the interlacement which is represented. □

Table 5.3.5

PARITY		$r_{k,j}$	INTERLACEMENT
k	j		
even	even	0	bottom layer weft over bottom layer warp
even	odd	0	top layer weft over bottom layer warp
odd	even	0	bottom layer weft over top layer warp
odd	odd	0	top layer weft over top layer warp
even	even	1	bottom layer warp over bottom layer weft
even	odd	1	bottom layer warp over top layer weft
odd	even	1	top layer warp over bottom layer weft
odd	odd	1	top layer warp over top layer weft

Figure (5.3.6) illustrates a point diagram representation of a given interlacement array and Figure (5.3.7) corresponds to a double layer cartesian structure representation of the same interlacement array. It is much more obvious in Figure (5.3.7), that the data corresponds to a double layer structure than it is in Figure (5.3.6).

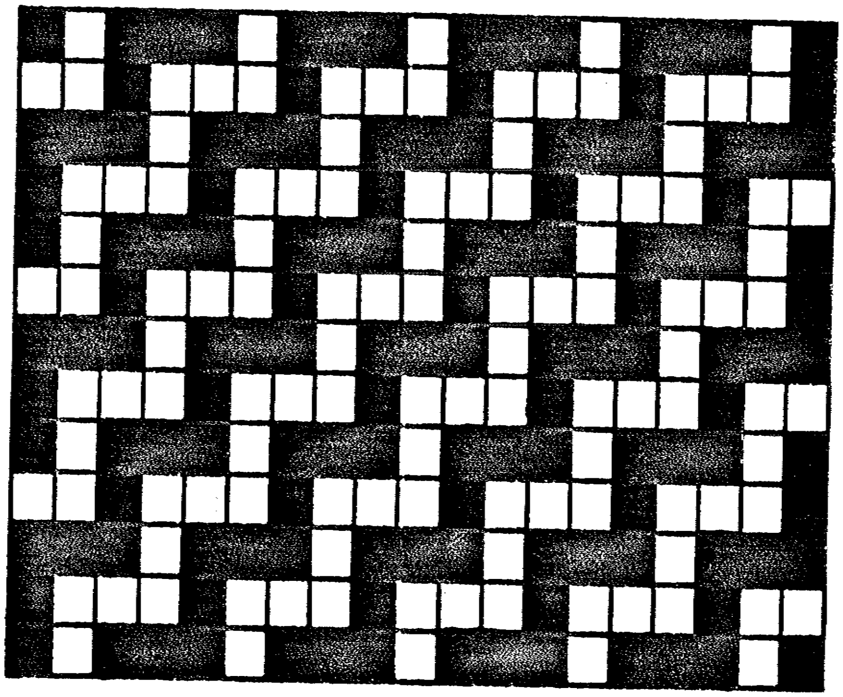


FIGURE 5.3.6

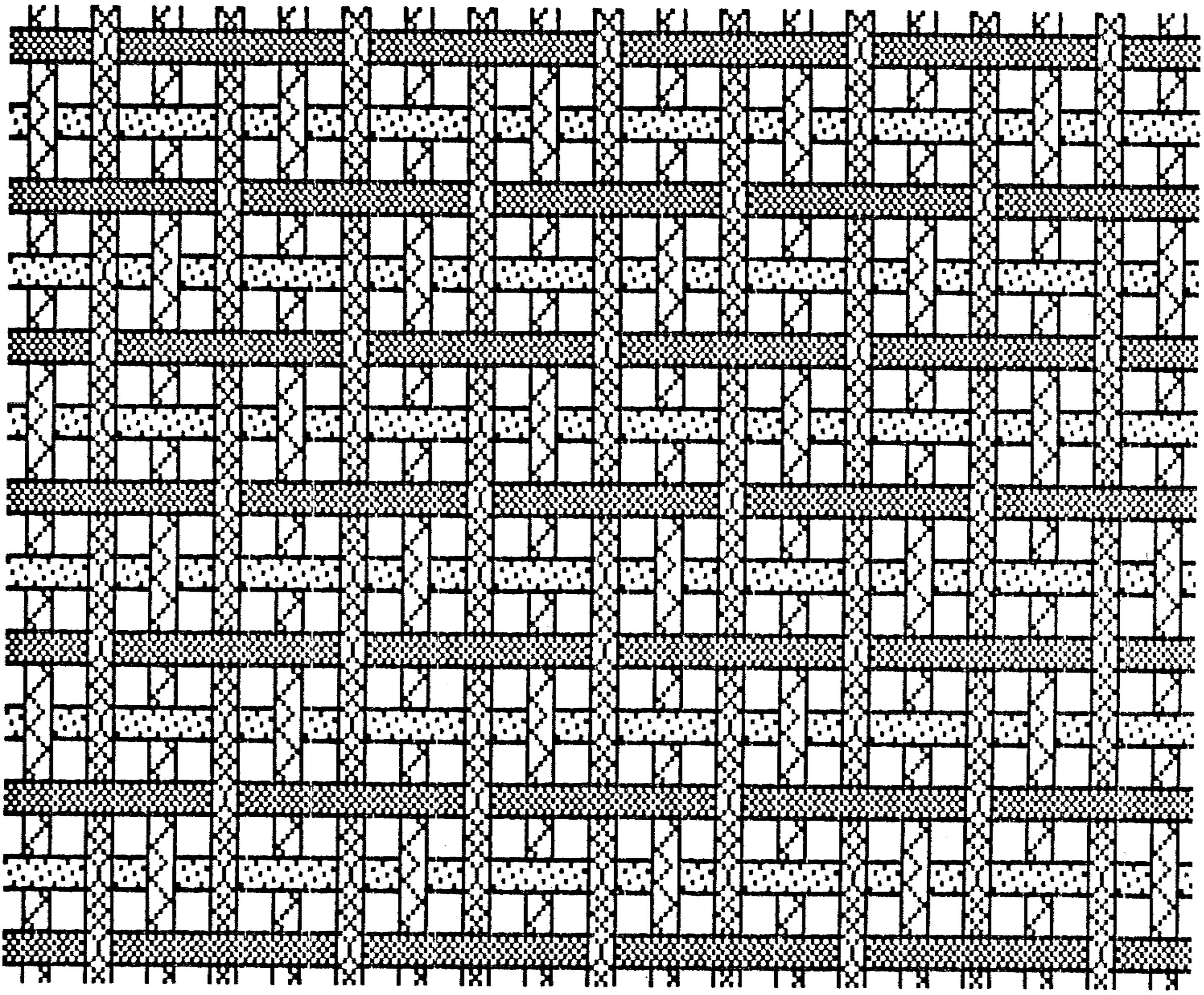


FIGURE 5.3.7

5.4 <u>REPRESENTATION OF NON-CARTESIAN STRUCTURES</u>

5.4.1 <u>CROSS-WOVEN AND GENERALIZED NON-CARTESIAN FORMS</u>

An interesting class of woven structures consists of those fabrics produced by crossed weaving. In cross woven structures such as leno and gauze [60, p.211-247], the paths of the lengthwise strands are no longer parallel, in that they are allowed to cross over each other between the intersections with the crosswise elements. This technique produces a fabric exhibiting a particularly stable structure while still maintaining an open quality, and is used extensively in drapes and decorative clothing fabrics, for industrial uses such as screens and sieves, [22, p.207-210] and for stabilizing the selvedges of shuttleless loom fabric [71].

In cross-woven structures, the twisting between warp strands occurs in pairs of strands, with this pairing remaining fixed for the length of the fabric. Within each pair of strands either one, or both of the yarns is allowed to deviate from the straight path typified by conventionally woven structures. The distinction between gauze and leno weaves is, in fact based on whether or not one of the sets of strands is held fixed. "A 'gauze' effect is developed by causing one series of warp threads, termed 'doup' threads, to form more or less zig-zag or wavy lines, whilst another series of warp threads, termed 'regular' or 'standard' threads, remain comparatively straight . . . .A 'leno' effect, however, is developed by causing both 'standard' and 'doup' warp threads to bend equally" [60, p.211-247].

Leno structures themselves are also of two types.

> In . . . [the first type], doup threads make only a partial or half turn around their respective standard threads: that is, they pass from one side to the other side of those threads, and then return to the same side, on different picks of weft, but do not completely twist around them. Sometimes, however, leno fabrics are produced in which doup threads are caused to completely encircle their standard threads, and thereby produce a full crossing or twist with them . . . [60, p. 253]

There are three interactions possible within a pair of warp strands in any cross-woven structure, namely:

1. two adjacent warp strands can remain parallel to each other with no twisting taking place;
2. the left-hand strand can twist over the right-hand strand;
3. the right-hand strand can twist over the left-hand strand.

Fabric structures such as sprang [12] and braiding [14, p.62] represent a generalization of the principles exhibited by the cross-woven forms in that, while warp strand interactions still occur in pairs, the pairing is no longer fixed throughout the length of the fabric. A warp strand can now twist with <u>any</u> currently adjacent warp. Clearly, a fourth type of interaction is now possible, in which

4. The left-hand strand can remain straight and the right-hand strand can interact with the adjacent strand to its right.

Non-cartesian woven structures actually consist of two distinct types of design rows, namely <u>interlacement</u> and <u>twist</u> rows. Interlacement rows are ones in which the warp strands intersect with a weft strand which lies perpendicular to them. The structure of the rows can be described using the conventional binary representation, where a "1" corresponds to a warp strand lying on top of a weft strand and a "0" corresponds to a weft strand lying on top of a warp strand.

Twist rows, on the other hand, deal strictly with interactions between adjacent pairs of warp strands and, since only four types of interaction are possible for each pair of warp strands, an ordered pair of binary digits can be used to represent each warp pair in a twist row.

Clearly, twist and interlacement rows are mutually exclusive. That is, a twist row contains no warp/weft intersections while an interlacement row exhibits no warp twisting. The corresponding data structure must therefore include a binary twist vector in which, for example, a "0" represents a twist row and a "1" represents an interlacement row.

Superficially then, the data structure for non-cartesian fabrics is nearly identical to that for conventionally woven cartesian structures. However, whereas each element of a binary interlacement array can be interpreted completely independently of all the other entries in the matrix, the data structure for non-cartesian structures is context sensitive.

Firstly, if the twist vector entry for a given row j is "1", then each matrix element in the j^{th} row is interpreted individually, as one of the two possible warp/weft intersections. Otherwise, if the twist vector entry for the j^{th} row is "0", then the matrix elements for the j^{th} row are interpreted as two place binary numbers in decimal form. Further, the elements of a twist row are paired 1 and 2, 3 and 4, . . . , 2i - 1 and 2i until or unless a pair corresponding to the fourth type of interaction is encountered. At this point, the parity of the pairing changes to 2i and 2i + 1, and continues thus to the end of the row or until changed again. Elements in a twist row are therefore also dependent for their interpretation upon all of the previous entries in that row.

5.4.2 DATA STRUCTURE AND GRAPHICAL DISPLAY

The graphical display of non-cartesian woven structures involves a mapping of the binary matrix data structure to two sets of ordered graphic output primitives or tiles, with the value of the twist vector at a given row determining which set of tiles is to be indexed.

From Theorem (5.2.1), two graphic output primitives are required to represent the possible intersections in an interlacement row R, as contained in the ordered pair of tiles:

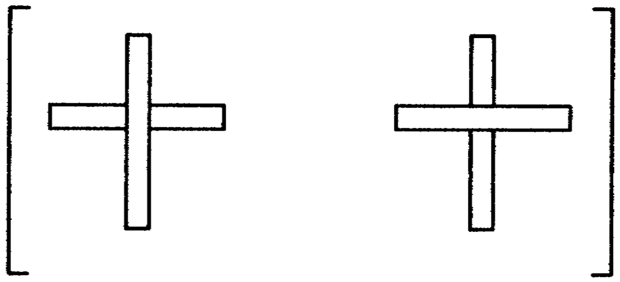


FIGURE5.4.2.1

An index i into this set of tiles can be computed using the formula (Corollary (5.2.3)):

$$i = r_j + 1 \qquad\qquad j = 1,2,\ldots,n.$$

For example, the binary sequence 1 0 0 1 corresponds to the following interlacement representation:

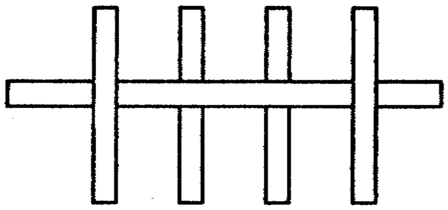


FIGURE5.4.2.2

As discussed in Section (5.2), considerable visual distinction between warp strands, weft strands, and inter-yarn spaces can be achieved by drawing outlines of the strands and adding some pattern within the strand boundaries, particularly if a significantly different interior pattern is used for the warp and the weft. The choice of pattern is highly constrained as to its period relative to the tile size, because there is a requirement that no discontinuities appear along the tile edges when the graphic display is generated. In displaying non-cartesian woven structures, the warp strands in twist rows must be represented by approximations to curved forms, thus rendering it impractical to use patterned strands. For this reason, warp and weft strands appearing in non-cartesian woven structures are represented by solid regions, which are broken at the intersections where appropriate.

<u>**THEOREM 5.4.2.3**</u>. Four graphic output primitives are required to

represent the possible interactions which can occur between warp strands in a twist row.

Proof. The ordered set S of these elements is: .

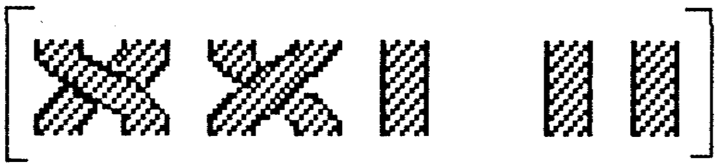


FIGURE5.4.2.4

CORollary 5.4.2.5. The mapping from the binary sequence R to the corresponding set of graphic primitives S for twist row interactions is obtained by computing the index i where

(5.4.2.6) $i = (2\, r_j + r_j) + 1$ $j = 1,2, \ldots ,n-1.$

The number of elements in the sequence R must be divisible by two. If it is not, the final element is dropped from R.

Proof. The values of i belong to the set of decimal integers {1,2,3,4} which can be used to identify uniquely each of the four graphic primitives.
□

It should be noted that the third graphic tile is half as wide as the other three tiles. This is because the third interaction corresponds to leaving the first of a pair of warp strands straight and considering the interaction between the second strand of the pair and its neighbour to the right. This amounts to a change of parity of the index j of the elements r_j in the indexing formula and applies from the middle of the current pair of strands to the end of the sequence, or until another of this type of interaction occurs. For example, the binary sequence 0 1 1 1 0 0 1 0 0 1 1 is mapped to the appropriate twist row graphical representation as follows:

$$\begin{array}{rl} \textbf{SEQUENCE:} & \underbrace{0\,1}\ \ \underbrace{1\,1}\ \ \underbrace{0\,0}\ \ \underbrace{1\,0}\ \ 0\ \ 1\ \ 1 \\ \textbf{I:} & \ \ 2\ \ \ \ \ 4\ \ \ \ \ 1\ \ \ \ \ 3\underbrace{\quad}_{1}\underbrace{\quad}_{4} \end{array}$$

FIGURE 5.4.2.7

__THEOREM 5.4.2.8__. The set S of graphic output primitives is a spanning set which can be used to represent all possible twist row interactions.

__Proof__. Consider all possible binary sequences of length four and the graphic output primitives to which they correspond. These results are given in Table (5.4.2.9). □

TABLE 5.4.2.9

BINARY SEQUENCE	GRAPHIC TILE NUMBERS
0 0 0 0	1 1
0 0 0 1	1 2
0 0 1 0	1 3 + (1 or 2)
0 0 1 1	1 4
0 1 0 0	2 1
0 1 0 1	2 2
0 1 1 0	2 3 + (1 or 2)
0 1 1 1	2 4
1 0 0 0	3 1 + (1 or 2)
1 0 0 1	3 1 + (4 or 3 + (1 or 2))
1 0 1 0	3 2 + (1 or 2)
1 0 1 1	3 2 + (4 or 3 + (1 or 2))
1 1 0 0	4 1
1 1 0 1	4 2
1 1 1 0	4 3 + (1 or 2)
1 1 1 1	4 4

When it is observed that "graphic tile 3" plus "graphic tile 3" is the same as "graphic tile 4" and that "graphic tile 3" plus "graphic tile 4" is equivalent to "graphic tile 4" plus "graphic tile 3", it is clear that all possible interactions between four warp strands can be represented by combinations of elements of S. By induction, the interaction between any number of warp strands can be represented using this set of graphic output primitives.

5.4.3 DESIGNING A GRAPHICS EDITOR FOR THE DISPLAY AND EDITING OF NON-CARTESIAN WOVEN REPRESENTATIONS

In the case of conventionally woven cartesian structures, the binary matrix data structure provides a visually meaningful representation of the interaction between warp and weft strands. Elements can therefore be entered directly into the data structure. At each stage the user receives appropriate graphical feedback as to what effect the new element has on the overall textile structure. This is not however the case with non-cartesian woven fabrics, where the binary matrix data structure does not provide an immediate graphical representation of the fabric but must be interpreted in a more complex, context-sensitive manner. This necessitates an alternative form of data input for non-cartesian woven structures.

The most appropriate and meaningful method of data entry is to look at a graphics screen and draw the appropriate tiles, using some simple, consistent interaction sequence [17, p.55-56], such as key presses. The tile which is chosen is interpreted by the software and the data file is updated.

Before data entry can actually take place, it is necessary that the screen and database be initialized. This initialization can either be to a previously created and stored structure, in which case data entry corresponds to editing this structure, or to some initial or foundation structure. In the second case, it is convenient to initialize the structure matrix and twist vector to zero. This corresponds to a fabric where all of the rows are twist rows in which the left-hand warp strand of each pair of warps crosses over the right-hand strand.

Obviously, if the twist vector entry for one of these rows is changed from 0 to 1, the entire structure row corresponding to this entry must be re-interpreted as an interlacement row and re-drawn. In this case, the entirely zero interlacement row corresponds to a weft yarn lying on top of all the warp yarns. This of course is not now a single cohesive fabric but is instead reducible (see Section {4.1}), with this particular weft yarn lifting completely free from the rest of the fabric.

This initialized display or "foundation structure" provides the equivalent of a grid to visually guide cursor movements and tile placement. The cursor is moved around the screen in unit movements, using the standard directional key pad. A unit move corresponds to one row in height and one column in width. In interlacement rows, a cross-wise move takes the cursor from one intersection to the next while, in twist rows, a single cross-wise move takes the cursor from one warp strand to an adjacent strand. Since the twist interactions occur primarily in pairs, the <u>usual</u> twist row horizontal move will be two units, although single unit movements are acceptable.

There are six possible tiles which can be drawn as part of a non-cartesian structure representation, of which two correspond to interlacement interactions and appear only in interlacement rows. The remaining four tiles correspond to twist interactions and only appear in twist rows. Data entry in this context therefore requires the use of either six keys plus range checking or a single key interpreted in context. The second solution requires less memorizing on the part of the user.

If the cursor is positioned in an interlacement row j, as indicated by some symbol such as "+" displayed in the j^{th} position in the twist vector, then a single key can be used to toggle from one possible intersection representation to the other. The key presses thus form a cyclic group of order two.

If the cursor is positioned on a twist row k, as indicated by a symbol such as "x" in the k^{th} position of the twist vector, the same key can be used to cycle through the four possible types of graphic tiles. These key presses now form a cyclic group of order four. Each of the twist row interactions depends on the value of two adjacent data elements for their immediate interpretation, with certain pairs of elements having broader implications. However, it is possible to advance the cursor in single unit moves from one warp strand to an adjacent one, with the data file also being addressed in successive and overlapping pairs of elements. Thus, it is possible that the cyclic group of key presses will be applied to a pair of data elements that were not interpreted as a pair when the graphic display was originally generated.

<u>THEOREM 5.4.3.1</u>. The set of twist row interactions form a cyclic group of order four, regardless of differences in the parity of data pairing between the original graphic display and the current data entry phase.

<u>Proof</u>. Consider the set T of all possible binary sequences of length four, and interpret T as two adjacent twist interactions, as in Theorem (5.4.2.8). Consider the action of the cyclic group A on T, where the elements of A are additions of one to the sequence $\{t_{i,1}, t_{i,2}\}$ or $\{t_{i,3}, t_{i,4}\}$, for i = 1,2, . . . ,16, and with addition taking place over the Galois Field of order two. The group A induces a permutation of the elements of T, as follows:

$A\ T$: (1 2 3 4) (5 6 7 8) (9 10 11 12) (13 14 15 16)
$A^2 T$: (1 3) (2 4) (5 7) (6 8) (9 11) (10 12) (13 15) (14 16)
$A^3 T$: (1 4 3 2) (5 8 7 6) (9 12 11 10) (13 16 15 14)
$A^4 T$: (1)

Consider the group B of additions as before, but with the additions being applied to the sequence $\{t_{i,2}, t_{i,3}\}$, for i = 1,2, . . . ,16. The permutations of T induced by B are:

$B\ T$: (1 3 5 7) (2 4 6 8) (9 11 13 15) (10 12 14 16)
$B^2 T$: (1 5) (2 6) (3 7) (4 8) (9 13) (10 14) (11 15) (12 16)

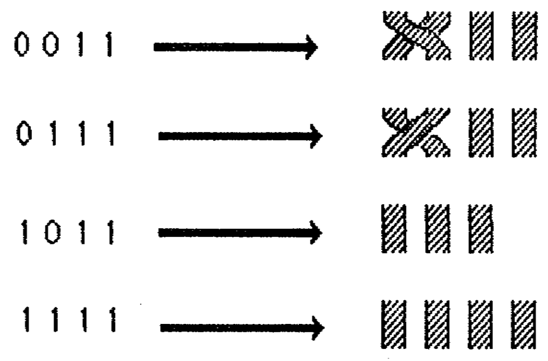
B^3T : (1 7 5 3) (2 8 6 4) (9 15 13 11) 10 16 14 12)

B^4T : (1)

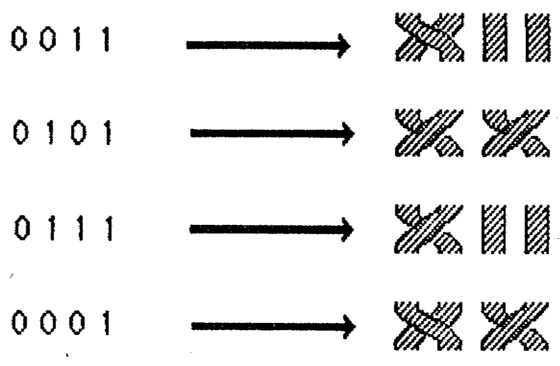
Thus, the group B is also of order four and the set of graphic output primitives will be correctly accessed regardless of where the cursor is positioned in a twist row. □

<u>EXAMPLE 5.4.3.2</u>

Positioning the cursor between the first two strands, the possible sequences and corresponding graphic images are:



Positioning the cursor between the middle two strands, the possible sequences and corresponding graphic images are:



Changes to a particular pair of data elements will necessitate the re-drawing of the corresponding tile, or tiles, and possibly all the tiles to the right, if the parity of the original display and design input are not the same. Tiles to the left will not be affected. From an implementation point of view however a convenient solution is simply to refresh the entire row to reflect changes to the corresponding region of the data file.

In addition to updating the structure display and the corresponding database, the twist vector must be addressable as well. Once again a single key, the same one as before, can be used to toggle between the two types of entries in this vector. The structure display and twist vector are treated as separate input areas, with the user specifying which of these areas is to be accessed. Cursor positioning and movement is now restricted to the specified region and the data entry key is interpreted in the appropriate context.

CHAPTER 6

AN INTERACTIVE TEXTILE DESIGN SYSTEM

CONTENTS

6.1 INTRODUCTION

The mixing of weaving technology and computer techniques is not a new phenomenon. Indeed, the used of punched cards to control a Jacquard loom predates the electronic computer by more than two centuries and is often thought to be the conceptual forerunner of many of the principles of modern computing [64]. Especially worthy of note is the fact that looms have even been used by the electronics industry to produce woven circuits [51, p.175].

The inverse relationship between computing and weaving has naturally appeared more recently. A paper presented at the 1966 A.C.M. National Meeting described an interactive computer graphics system for designing a Jacquard fabric [54]. Subsequent work by this group produced a Jacquard loom, unveiled April 6, 1968 at the San Antonio HemisFair, which was controlled by an IBM 360 mainframe computer [53]. More recently, a system has been described [61] whereby a mini-computer has been used in conjunction with an automatic pattern card punching machine for the direct control of a Jacquard loom. These developments have however required expensive, large and often special purpose computer hardware [22]. Today, textile designers can take advantage of the increasing availability of inexpensive microcomputers and medium resolution graphical display devices to create an environment well tailored to meet their needs.

The purpose of this chapter is to outline the functional structure and interactive features required of a microcomputer-based interactive textile design system in order to fulfill the needs of the designer of woven fabrics. This subject will be divided into a discussion of the over-all module hierarchy, the underlying application data structure, the graphic display program, and the applications programs, including the necessary computational algorithms.

As a testbed for the principles enunciated in this chapter, a restricted version of the theoretical system described was implemented on an Apple II+ microcomputer. The specific details of this implementation will be described in the appropriate sections.

6.2 MODULE HIERARCHY.

The components of this interactive textile design system can be divided broadly into three categories, namely the application data structure, the graphic display program and the applications programs. The applications can themselves be sub-divided into two major classifications. The first type of application is one whose primary function is to induce some transformation on the application data structure. Design input is an obvious example of this type of program. Also in this category are the design manipulation or editing functions and the data storage, retrieval and deletion functions. The second type of application includes those programs which make use of the application data structure as input to computational algorithms. These algorithms are either structural in nature, in which case the resulting output is data which is itself added to the data file, or are used to provide information in the form of graphic display.

The nature of woven textile design information is such that it is highly visually oriented. For this reason, the graphic display program is the central node in this system, through which every other function is connected.

Conceptually the module hierarchy of this textile design system is illustrated in Figure (6.2.1). It indicates the major components and their inter-relationships. All subsequent discussion will assume that transitions between program modules or between states within a program are either event-driven [17, p.57], specifically arising from user keyboard

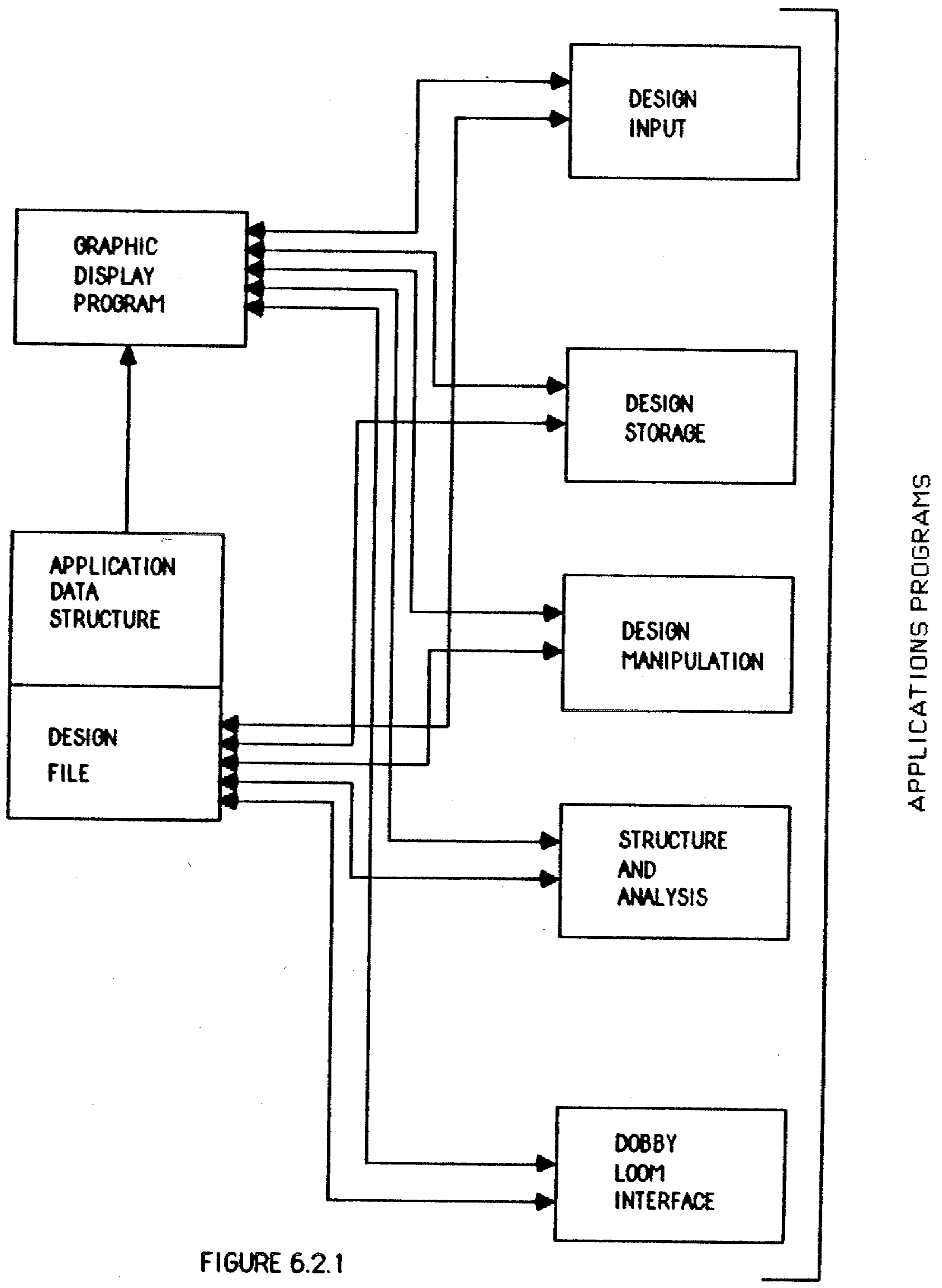


FIGURE 6.2.1

INTERACTIVE TEXTILE DESIGN SYSTEM
MODULE HIERARCHY

input, or occur as the natural termination of an application program. Attention will be concentrated throughout on the following design principles as enunciated by Foley and Van Dam [17, p.55-56]:

1. Provide simple, consistent interaction sequences.

2. Do not overload the user with too many different options and styles for communicating with the program.

3. Prompt the novice user at each stage of the interaction (but allow the more experienced user to bypass prompts).

4. Give appropriate feedback to the user.

5. Allow the user graceful recovery from mistakes.

This latter point is particularly important when creating a working environment for non-technical design personnel.

In implementing this system, the outlined modular structure has been used. However, due to the limited amount of available memory (48K of RAM), these modules have been further segmented into sets of small numbers of related tasks. Each set of operations is contained in its own mainline segment which is read into memory from disk when required. Associated with each of these segments is a menu of program functions as, for example, seen in Figure (6.2.2). All menus first appear with a white bar over the bottom item. The retype and backspace keys are used to move the white bar either up or down, over the menu items, with cyclic

DISPLAY MENU

DISPLAY THE DESIGN IN MEMORY

ANALYZE THE DESIGN IN MEMORY

DISPLAY WEFT DRAWDOWN

DEFINE WINDOW

DISPLAY DESIGN GRID AND ERASE MEMORY

DISPLAY THREADING GRID, ERASE MEMORY

EXIT TO COLOR DISPLAY MENU

EXIT TO DESIGN ENTRY MENU

EXIT TO PRINTER MENU

EXIT TO DESIGNER COMMAND MENU

FIGURE 6.2.2

MENU

wrap-around. The carriage return key is used to select a menu item.

The graphic display program of the theoretical system is handled by the DISPLAY MENU, COLOR DISPLAY MENU, COMPLETE DISPLAY MENU, COLOR AND WEAVE MENU, PRINTER MENU, STRUCTURE MENU and STRUCTURE SUB-MENU.

Menus are also used for many of the application programs. Design input is handled in the DESIGN ENTRY MENU; design storage or retrieval is performed by the FILER MENU, the PICTURE FILER MENU and the ARCHIVIST MENU; design manipulation uses the DESIGN EDIT MENU. The DOBBY CONTROL SYSTEM, which is the controlling software for a sixteen shaft dobby loom, is considered separately in an appropriate series of related menus.

The computational algorithms which synthesize a binary interlacement array from a given threading, tie-up and shed sequence matrix, or factor a given binary interlacement array into its corresponding threading, tie-up and shed sequence matrices are each contained in a separate segment. These programs each perform only a single task however, and are not therefore associated with menus.

Figure (6.2.3) illustrates how the various programs relate to each other. The specific operations performed in each of them will be discussed in conjunction with the relevant theoretical system.

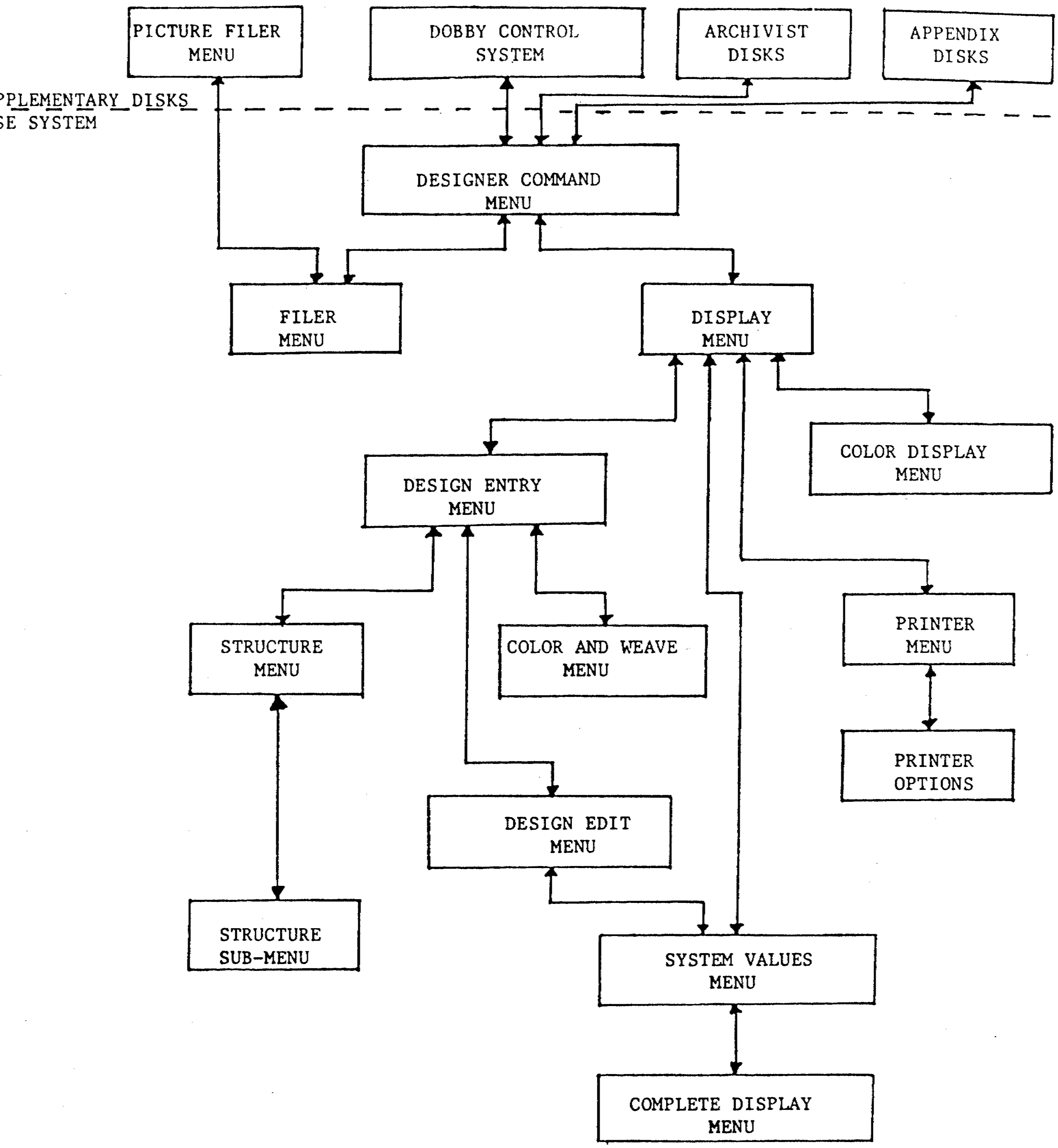

PICTURE FILER MENU
DOBBY CONTROL SYSTEM
ARCHIVIST DISKS
APPENDIX DISKS
PPLEMENTARY DISKS
SE SYSTEM
DESIGNER COMMAND MENU
FILER MENU
DISPLAY MENU
COLOR DISPLAY MENU
DESIGN ENTRY MENU
STRUCTURE MENU
COLOR AND WEAVE MENU
PRINTER MENU
PRINTER OPTIONS
DESIGN EDIT MENU
STRUCTURE SUB-MENU
SYSTEM VALUES MENU
COMPLETE DISPLAY MENU

FIGURE 6.2.3

6.3 APPLICATION DATA STRUCTURE.

The primary data structure which is created, stored and used as input to the applications programs is the binary interlacement array, which corresponds to a numeric representation of the intersections between the warp and weft yarns of a rectangular segment of woven fabric. With no loss of generality, values of one in this array can be used to represent warp over weft intersections, while values of zero represent weft over warp intersections. Traditionally, textile workers have depicted binary interlacement arrays diagrammatically, as a matrix of black and white squares, where a black square corresponds to a one and a white square corresponds to a zero. This representation, known as a point diagram, can commonly contain up to one million elements [22, p.1], although it is more usual to contain of the order of one thousand elements.

It was shown in Chapter 3 that, if the binary interlacement array is stored, then the threading, tie-up and shed sequence data are actually superfluous. Since the factorization of a binary interlacement array into these components is essentially unique, the threading, tie-up and shed sequence matrices which require the smallest number of shafts and treadles can always be derived from a given interlacement array. This means however that every time this component data is required, the factorization algorithm must be performed, which considerably increases the time required to access the data.

In addition, there are occasionally applications where it is important to specify precisely which warp strands are threaded on which shafts and which treadles control particular combinations of shafts. The factorization process, it will be recalled, leaves the threading and tie-up matrices invariant up to permutation of their rows and the shed sequence and tie-up matrices invariant up to permutation of their columns. There is therefore no assurance that a given threading matrix, having been used to create a binary interlacement array, will not be changed when regenerated by a factoring algorithm. For this reason, it is prudent to store the binary matrices corresponding to the threading, tie-up and shed sequence data as well as the binary interlacement array. The dimensions of these four matrices are also required.

A colour vector of encoded warp strand colours and a colour vector of encoded weft strand colours must also be stored. These two vectors, in conjunction with the binary interlacement array, allow the synthesis of a multi-valued matrix corresponding to the visible coloured pattern of the woven fabric.

Finally, some data storage positions must be allocated for flags and other controlling variables. These are, of course implementation specific and will be discussed as they appear in the following sections.

Since the primary application data structure is binary, a single bit of RAM can conceivably be used to store each of the elements. This compressed form of storage was in fact achieved in the test implementation, where the application data structure was stored in the 8K region of memory corresponding to the second page high resolution graphics screen. Since all display took place on the first page high resolution graphics screen, this region of memory was not used for graphical display, program code, or variable and string storage. Instead the program architecture was arranged so that when any module was used, this region of memory was not compromised. As such, the application data structure was maintained in memory as a resident global data file to be accessed and freely modified by all program modules.

High resolution graphics images on the Apple II+ correspond to bit maps stored in the corresponding 8K region of memory. This meant that a single element could be stored in this area simply by drawing a single unit dot at the specified screen co-ordinates. Retrieval of the data therefore required computational algorithms that would isolate a single bit or detect a visual bit. This latter more interesting approach relied on a graphics command and the high resolution collision counter. This type of storage had the additional unique advantage that the actual data file could be visually examined in a meaningful way by displaying the page two high resolution graphics screen. This arrangement in fact, simulated a process monitor which was subsequently very useful in developing computational algorithms with improved efficiency.

6.4 GRAPHIC DISPLAY PROGRAM

6.4.1 PRIMARY CONSIDERATIONS

Woven textile design data is visually extremely meaningful, even in cases where the structure rather than the visible pattern is documented. The binary interlacement array defines the inter-relationship between the set of warp strands and the set of weft strands intersecting it. Given that all of the warp strands are coloured black and all of the weft strands are coloured white, the traditional point diagram variation of the binary interlacement array also describes the visible pattern of the structure. Even if the strands are not coloured in this way, the interlacement data corresponds to the surface appearance of the fabric.

Textile designers are, for the most part, well adapted to designing woven structures and interpreting interlacement data, when it is recorded in this format. Thus it is obvious that some form of graphical display must form an integral part of an interactive textile design system. Indeed, the graphical representation of the data is sufficiently important that it is likely the quality of this display which will be the major factor in determining the utility of such a design system.

As mentioned in Section [6.1], binary interlacement arrays can be quite large. Since the use of a CRT display device introduces some inherent limitations in the number of identifiable pixels available, some balance must be reached between making the individual design elements large enough to see, yet small enough to accommodate a large design on the screen. Two techniques can be used to to help achieve this required balance.

The first method is to have a variable resolution display that is defined by the user. Thus, a small amount of data can be displayed at a large scale, and a large interlacement array can be displayed with the individual elements, of necessity, being very small. It should be noted that, because the data is discrete, there is no actual loss of detail, even in a high density representation. The relationships between individual elements will however become difficult to detect or separate visually. While this may be acceptable for some purposes, some applications, such as design manipulation, will demand a clear delineation of individual data points or design elements.

A second approach is to use a windowing technique, where only a segment of the data is selected to be displayed at any given time. This permits an easily visible representation of a subset of the data. Clearly this method is deficient in that it is difficult to examine relationships between pattern areas and to visualize the displayed segment as it fits into the design as a whole. Also, great care must be taken at the boundaries of window segments to ensure that anomalies or discontinuities with neighbouring structures are not created.

It is in fact likely that some combination of these two approaches is optimal, with the variable density method being used to obtain a visual representation of a structure in its entirety and the windowing technique being used where some modification to the individual data elements is required.

A second form of graphical display which is extremely important is the hard copy print. Paper copies are useful when a graphic representation of design data is needed in a location where it may be difficult to site a monitor, such as in the weaving shed. They are also required for manuscript preparation, brochures and written communications. Dot matrix printers with graphics capability provide a low-cost effective means of producing hard copy renderings of the required graphics images.

6.4.2 DISPLAY ENVIRONMENT FOR DATA INPUT

One of the first major concerns in creating an interactive textile design system is that of actually entering the design data. Numerous solutions to this problem have been devised for the automatic acquisition of data, including the use of optical scanning equipment [52] and light pens [53]. However, what is of prime importance in this context is that the data be acceptable in some form which is visually meaningful to the user. For example the 16 by 16 element interlacement array in Figure (6.4.2.1) could be represented in a compressed notation by considering each row of the matrix to be a 16 place bit string and interpreting it as a decimal number. The design would then be specified by the numbers 17510, 39389, 6545, 30566, 25670, 55709, 4505, 26486, 26180, 56729, 37145, 26231, 18020, 40409, 39185 and 30311. This compressed notation is however, much less meaningful in terms of the pattern which it represents. Clearly, some form of graphical input is indicated.

In the point diagram, the graphical form of the binary interlacement array, the matrix co-ordinates define the single warp and weft strands which intersect at a given location. Design elements must be placed completely accurately with respect to these co-ordinates. The high degree of precision required is best achieved by drawing an appropriate grid on the screen, in which squares can be coloured. This has the added advantage of being a form of design development with which textile designers are completely familiar. An x-y position indicator or locator [17, p.24] is simulated by a cursor drawn on the grid and moved up, down, left or right

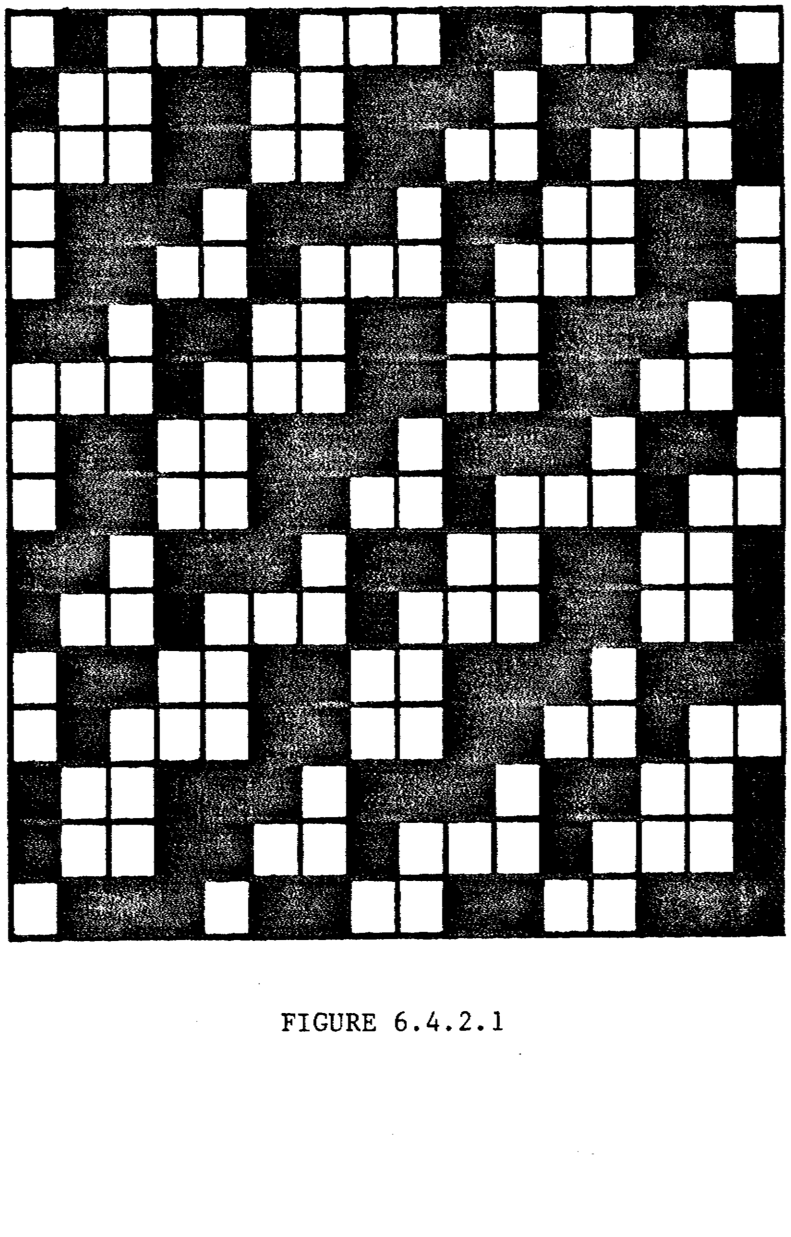


FIGURE 6.4.2.1

under the control of four keys from the keyboard, usually I, J, K, and M on the APPLE II series [17, p.200]. Cursor movements are obviously made in integer numbers of squares.

A further input unit is used as a pick to indicate selection or de-selection of a given grid cell. As the point diagram representation is created and modified, the corresponding application data file is appropriately updated on a continuous basis. In the case where coloured interlacement array data is to be accepted and the underlying data file is discrete but not binary, selection of a grid cell is accompanied by a numeric encoding of the colour drawn in the square.

The primary design data is contained in the binary interlacement array and input to this array is obviously required. It is not uncommon however for the textile design process to begin with the specification of a threading, tie-up and shed sequence matrix, from which the corresponding interlacement array is computed. Design input to each of these three matrices is also therefore required. Additionally, in the case of coloured interlacement arrays, the colour vectors which specify the colour of each warp and weft strand in the fabric representation must be addressable.

In the implemented system, the design entry environment is established in the DISPLAY MENU. One of the available options is to "Display Design Grid and Erase Memory", where the "Design Grid" refers to a rectangular array (Figure {6.4.2.2}) whose cells are mapped to elements

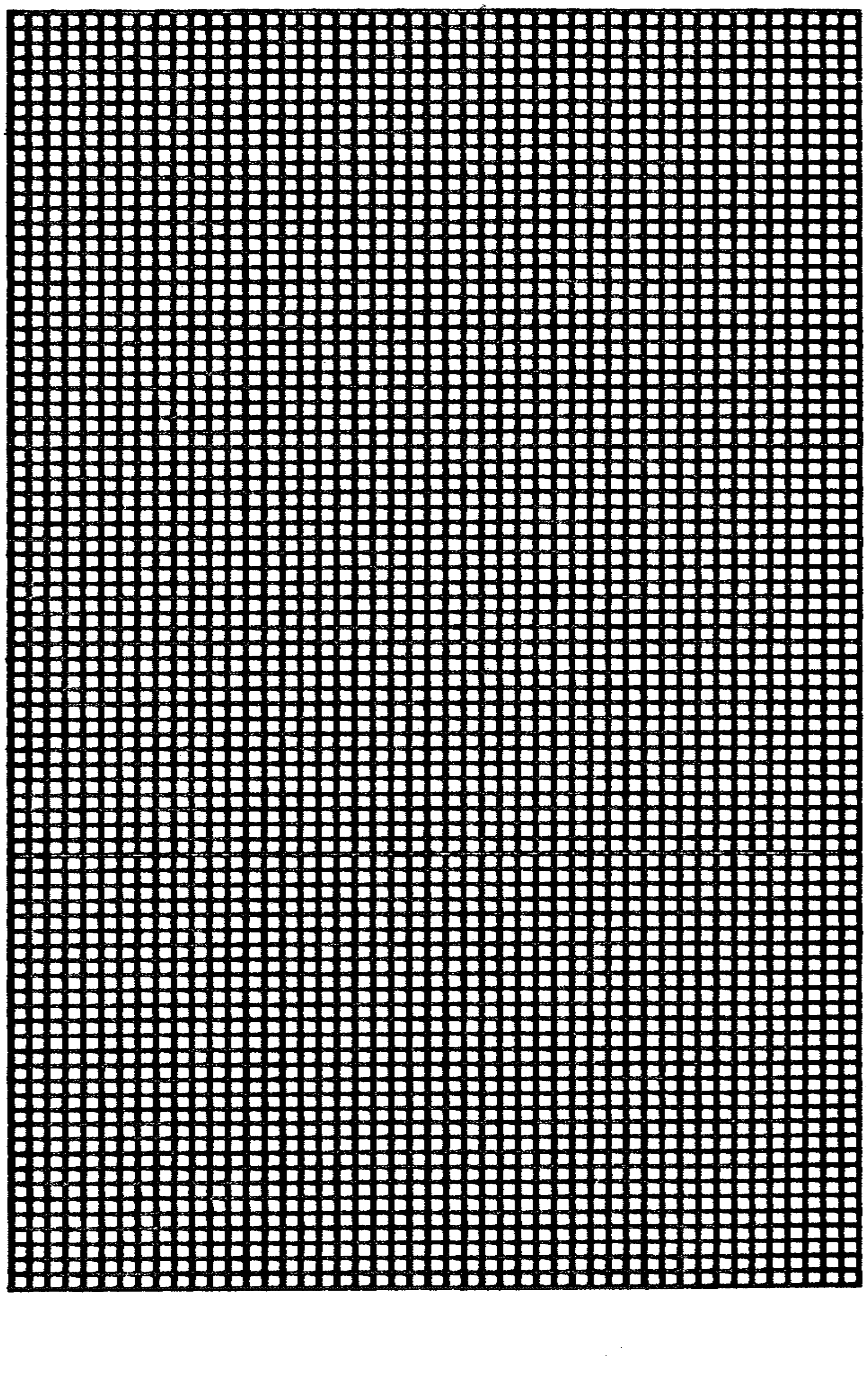


FIGURE 6.4.2.2 DESIGN GRID

of the binary interlacement array. When this option is invoked, a grid with empty squares is drawn and the entire data storage area is initialized, which corresponds to turning off all bits in the region of memory corresponding to the page two high resolution graphics screen.

The grid size is variable. The larger the grid size, the fewer the number of cells displayed and vice versa. Two keys are available to change the grid size, with cyclic wrap-around at the ends of the range, and a third key, carriage return, is used to accept the grid as drawn.

An alternative data entry environment for binary interlacement array data is established by displaying a data file which has already been created and which will be drawn as a grid with the appropriate squares filled in (Figure [6.4.2.3]).

Another option which is available is to "Display Threading Grid and Erase Memory", where the "Threading Grid" provides input areas A,B and C for threading, tie-up and shed sequence data respectively (Figure 6.4.2.4). Once again, the grid is drawn with completely blank squares and the data file is initialized to zero. The grid size is variable, as is the number of rows in the threading and tie-up matrices, and the number of columns in the tie-up and shed sequence matrices.

Selecting the option "Display Weft Drawdown" will also result in the display of a grid with threading, tie-up and shed sequence regions, but

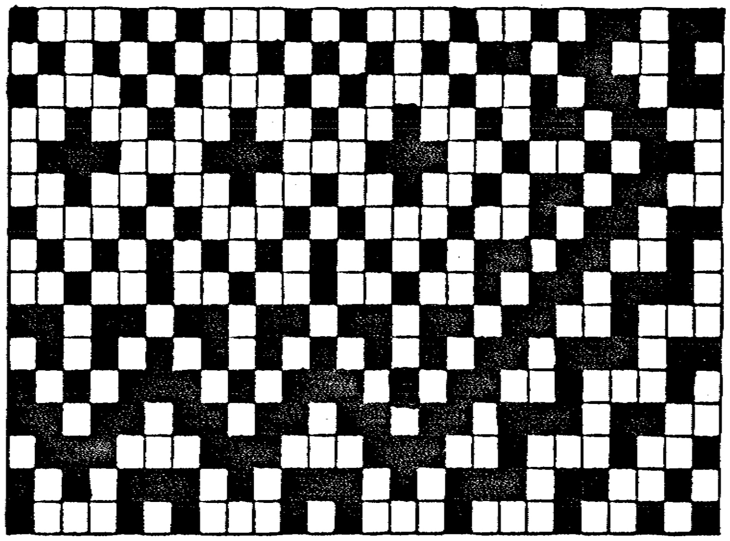


FIGURE 6.4.2.3

DESIGN

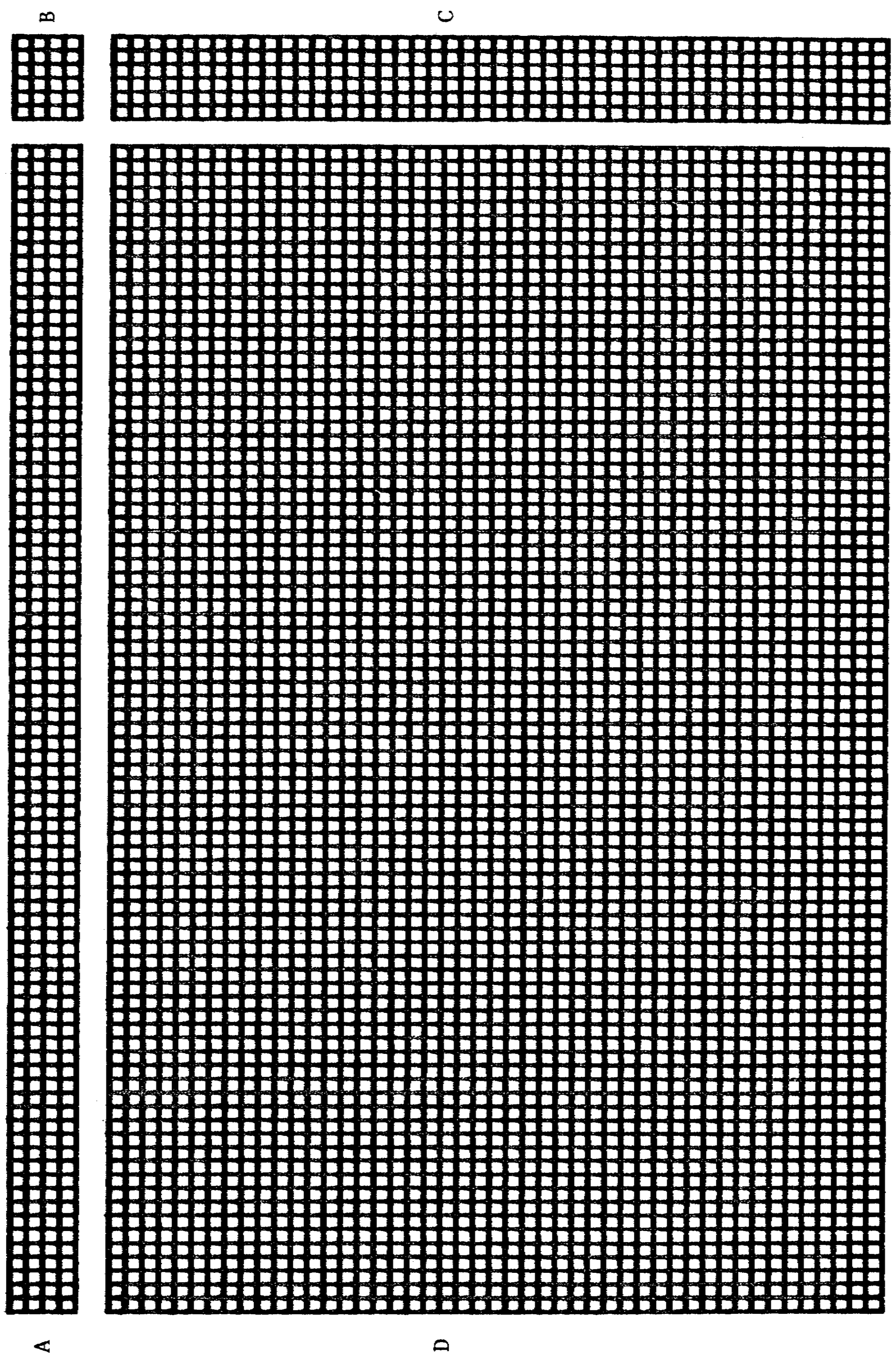


FIGURE 6.4.2.4 THREADING GRID

with data file entries appropriately represented in the display (Figure [6.4.2.5]). The term "Weft Drawdown" simply means that the complement of the conventional representation is shown. That is, warp over weft intersections correspond to blank squares and weft over warp intersections correspond to coloured squares. A facility for displaying the "Reverse Drawdown" allows the other representation to be displayed if required.

Also in the DISPLAY MENU is the option to "Define Window", which enables the user to specify a rectangular subset of the design area for display. This feature allows a large design area to be visualized at a small scale, while still permitting design input over a magnified portion of the region.

Once the design entry environment has been created, a different program module called the DESIGN ENTRY MENU is invoked which deals with the locator, cell selection and data file updating. If the design grid or design in memory have been displayed then "Drawdown Entry" can be selected: "Drawdown" is a synonym for point diagram. Four keys are used to move the cursor within the boundaries of the grid.

If a window is in effect and an attempt is made to move the cursor outside the grid, then the option is available of moving the window over a different region of the design. At any time, a key can be pressed which causes the application data file on the page two graphics screen to be

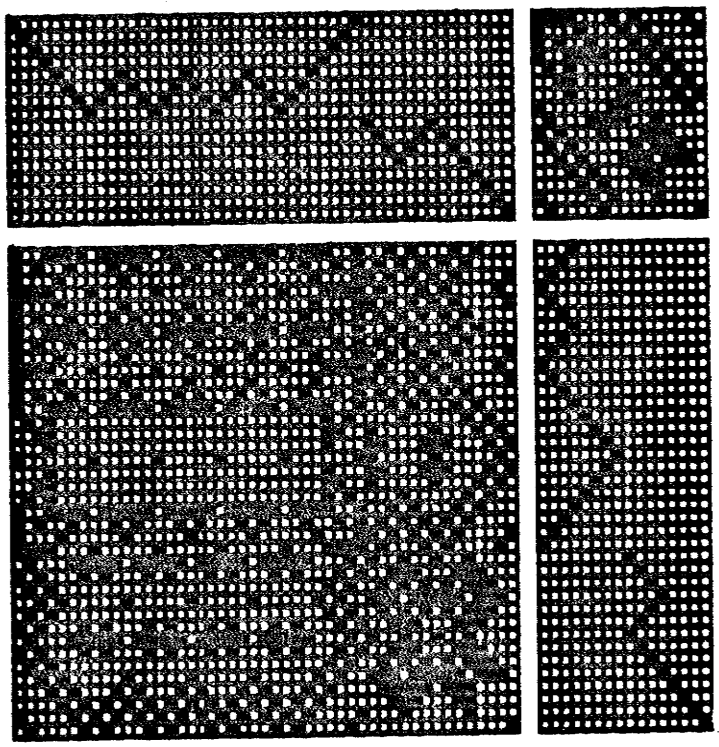


FIGURE 6.4.2.5

WEFT DRAWDOWN

shown. The boundaries of the defined window are outlined in the data area. This allows the design entry window grid to be visualized in the context of the entire design.

Having displayed a drawdown grid or weft drawdown, it is now possible to enter "Threading; Tieup; Treadling Entry" data. (Treadling is a synonym for shed sequence.) Each of the three data input areas is associated with a single letter code. Pressing one of these code letters, as prompted, places the cursor and defines the boundaries appropriately. As before, attempts to move outside of the input area will result in the option of moving the window.

A limited facility for displaying coloured interlacement arrays has been implemented. However, input to the corresponding colour vectors is used only for a local display and is not stored. This feature is discussed more fully in Section (6.4.3).

6.4.3 DATA DISPLAY

A major function of an interactive textile design system is to provide a graphical display of the interlacement data. This display may be used to facilitate data modification, as discussed in Section (6.4.2), or it may provide a graphical model of the corresponding fabric for examination with a view to assessing its suitability for a particular application.

Graphical display of the threading, tie-up and shed sequence matrices is another important requirement of a textile design system. A skilled designer or textile technologist can use this information to assess the particular loom requirements. In addition, these three factors define precisely how a loom should be threaded, tied-up and treadled in order to produce the fabric corresponding to the binary interlacement array. As such, this displayed data can be considered as a set of instructions for the setting-up and operation of a loom.

As noted in Section (6.4.2), displaying this data as a matrix of coloured and empty cells on a grid provides the necessary precision when individual elements are to be addressed. This is true in the case of data modification and also in the case where the graphical display is meant to provide the exact detail regarding a particular strand, such as precisely on which shaft a certain warp strand is threaded.

On the other hand, sometimes what is required is not a precise data map, but rather a general over-all impression of the structure. In this instance, grid lines detract from, rather than add to, the clarity of the image. An alternative form of graphical display is therefore desirable where the interlacement data corresponds to contiguous black and white squares. This form of representation has of course the added advantage that a greater amount of data can be accommodated on the screen at a given time. When a grid is used, a considerable number of pixels are required just for the grid lines. In addition, the cells which are used to represent the data elements must be of a size sufficient to distinguish them from grid line intersections.

In the test system, two forms of data display have been implemented. The first form is the one previously described, with a variable grid and windowing. The second form is called a "Complete Display" and consists of black and white squares drawn contiguously at a very fine scale, within the four defined data areas. This representation is visually very similar to the appearance of the actual data file.

In the data display schemes discussed thus far the application data structure and the design file are completely congruent. There is a one-to-one mapping from the elements of the data file to the design elements represented on the screen. The displayed elements are solely a function of the data points to which they correspond.

This is not however the case in the graphical display of a coloured interlacement array. In this instance, the binary interlacement array is mapped through the warp and weft colour vectors to compute the exact colour which should appear in each of the cells of the displayed array. The design file is no longer congruent to the application data structure but arises from the interaction between the interlacement data and the colour vectors.

If the user is permitted to create coloured interlacement arrays directly by filling in coloured squares of the design area, then there is an obvious need for continuous colour analysis and verification. Each time that a coloured element is drawn in the displayed matrix, the system must determine whether this entry is consistent with the existing coloured interlacement structure. At the termination of such a data entry session, the application data structure contains all the determinate interlacement data. Since the storage area is initialized to zero prior to any data entry, all intersections which are indeterminate are automatically stored as corresponding to the weft over warp type. Alternatively, "Colour and Weave Effects" [23, p.150] can be created by using colour vectors for the warp and weft strands together with some initialization colour, likely white, appearing in all positions not specified by the user. In the test system, a sub-set of this coloured interlacement array display facility was implemented. The technique begins with the binary interlacement array interpreted as a coloured interlacement array, where the warp strands are all coloured black and the weft strands are all coloured white. The displayed colour vectors correspond to this interpretation. Alterations to the visible pattern are effected by changing any or all of the warp strand colours to white or weft strand colours to black, with appropriate updating of the software being made to the design file.

A classical example of this type of effect is that of a fabric design known as "houndstooth". As shown in Figure (6.4.3.1), a twill structure is combined with regular alternate/two colour striping of the warp and weft strands to produce a small regular motif.

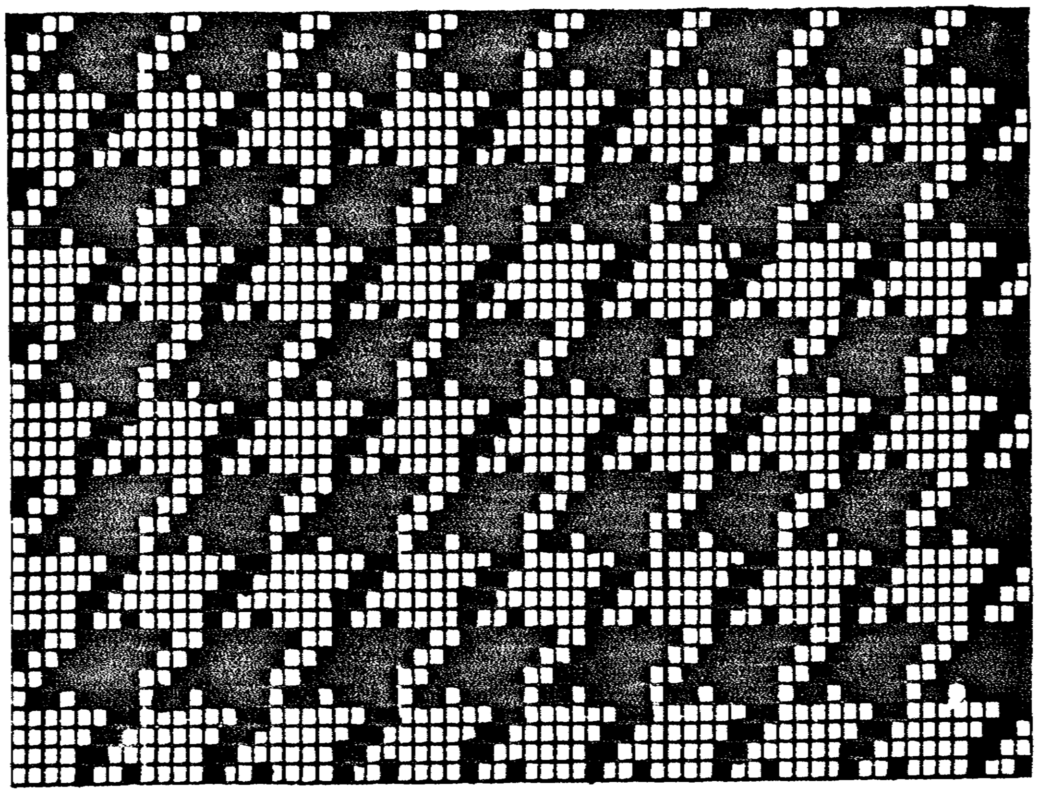


FIGURE 6.4.3.1

"HOUNDSTOOTH"

6.4.4 DATA MAPPED TO GRAPHIC OUTPUT PRIMITIVES

As previously discussed, woven textile design data is very visually meaningful when represented graphically as a point diagram. In some applications however, considerable insight into the structural relationships between various warp and weft strands can be gained through alternative graphical representations. In these cases, the application data structure is no longer congruent to the design file. Rather, each data element, or sequence of elements, is mapped to a graphic output primitive which characterizes the relationships between the mapped data elements, as well as with the remaining data. Three such mappings are considered, namely: i) cross-sectional representations ii) flat sectional representations and iii) profile substitution of design blocks and counter-blocks (that is, the replacement of each black square of a design by a design block or matrix and the replacement of each white square of a design by a different matrix, namely the counter-block) [8, p. 266, 287].

As discussed in Chapter 4, a diagram showing cross-sections through the fabric corresponding to a given interlacement array can clarify the interactions between adjacent data points. These sections, cut through either the warp or the weft strands, can reveal whether floats of yarn on the upper or lower surface are excessively long. They can also indicate intersection sequences which will result in a multi-layered fabric rather than a single-layered cloth.

Flat sectional drawings are of value in illustrating the visible structure of particular interlacement arrays. The relative lengths of yarn floats in the warp or weft direction, as well as patterns which arise from the relationship between these floats are highlighted by this type of representation. In the case of multi-layered fabrics this type of representation is less visually meaningful, because the interlacement arrays for the separate layers are completely merged and any patterning which is present in either of the layers is extremely difficult to detect visually. The flat sectional representation of these structures corresponds to an "exploded view" of the fabric where the fabric layers are shown loosely woven and offset so that the lower layers appear in gaps between the yarns of the upper layers.

These first two examples have a major characteristic in common in that in both cases the interlacement data is mapped to a set of pre-defined graphic output primitives according to an established algorithm. The graphic primitives can be considered as tiles, which are used to tessellate the screen, with the yarns drawn within the boundaries of each tile. The shapes depicted are primarily line drawings and the density of pixels turned on within a given tile is relatively low. For this reason, the sets of output tiles are chain length encoded [62]. Each of the graphic primitives consists of the sequence of plotting vectors required to produce the appropriate image.

In creating the graphic output primitives for either of these two applications, there are a number of factors which must be considered. The first such factor is the precise placement of these shapes on the screen so that each tile matches all possible adjacent tiles. This requires that each image be drawn with reference to a common origin, and that a standard tile format be used. Each tile is of constant width and height since the sequence in which they are plotted is not known _a priori_.

The size of the graphic tiles must also be carefully considered. The tiles must be large enough to obtain a good representation while being small enough to allow an adequate number of them to be drawn on the screen. The cross-sectional tiles must use enough pixels in representing the cut strands to differentiate between the various symbols that indicate coercion. The continuous strands must be separated from the cut strands by at least one pixel, and there must be a sufficient number of pixels across the tile so that the continuous strands can approximate the required curvature. If a multi-layered fabric is represented there must be an odd number of pixels between the cut strands to permit a continuous strand to be drawn through the mid-point between them.

The flat sectional tiles must be large enough to ensure that the intersection of the warp and weft strands can be represented unambiguously. This requires the outlines of the strands and some form of internal shading be drawn so as to differentiate them from each other and from the background.

In the test system, a separate shape table of graphic output primitives was created for each type of representation. The shapes were all drawn within a rectangular boundary using a Shape Table Editor. Figure (6.4.4.1) gives an example of a graphic output primitive created as part of the F[2:1:1] shape table. The size of each graphic tile was constant within a particular set of shapes but varied from one shape table to another. Separate drawing routines were used for each of the different representations to accommodate this difference, as well as to incorporate the correct mapping algorithm.

The third example of interlacement data mapped to an alternative form of graphical representation is that of profile substitution of design blocks and counter-blocks. In designing block weaves such as damask and doubleweave, it is often convenient to interpret the interlacement data as a description of the gross structure of a fabric rather than the actual intersections between the warp and weft yarns. In such cases the graphical data now defines only the <u>profile</u> of the design in terms of the relative size and relationship of the gross design elements. Implicit in this shorthand description is that the complete detailed design can be generated by the substitution of some interlacement array, or <u>block,</u> for every black square in the profile matrix, with a different interlacement array, or <u>counter-block,</u> for every white square. Figure (6.4.4.2) illustrates a binary interlacement array interpreted as a design profile, with appropriate substitution of satin blocks and sateen counter-blocks.

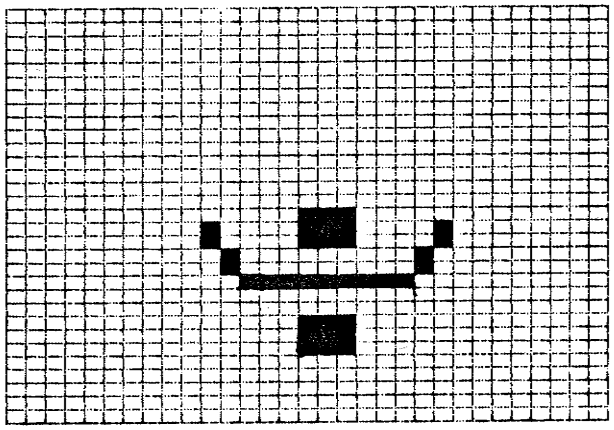
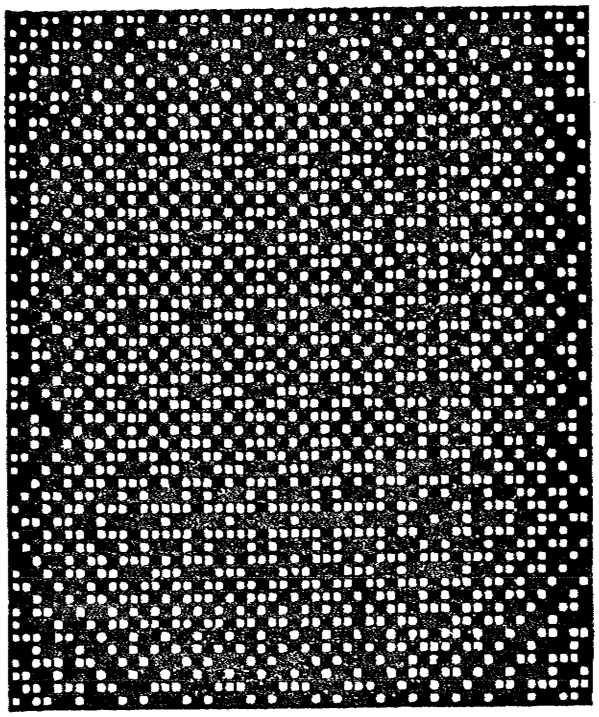


FIGURE 6.4.4.1



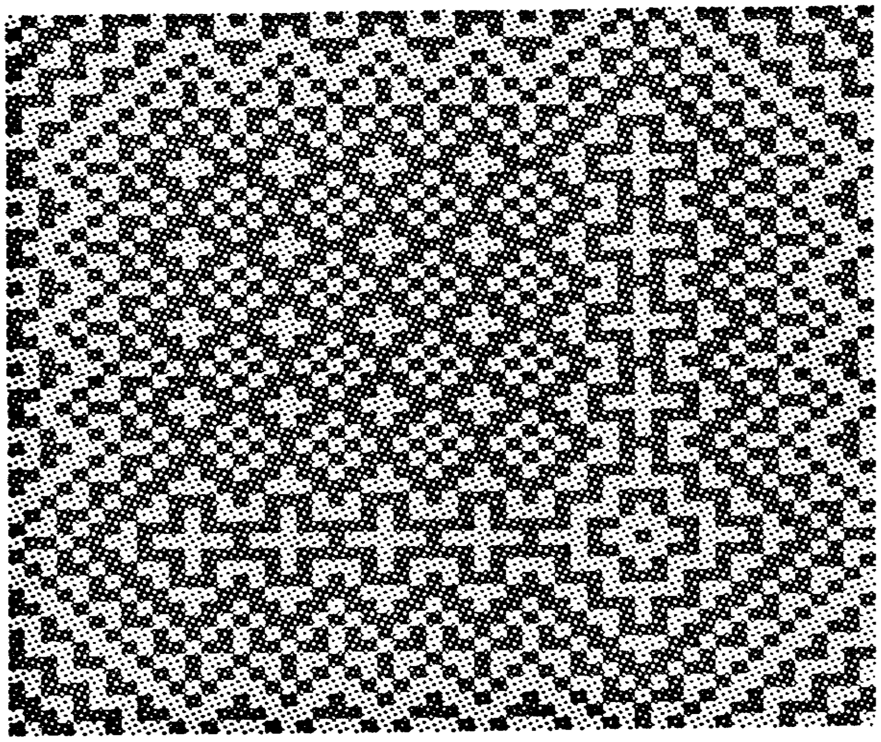


FIGURE 6.4.4.2

This facility is extremely valuable in that the textile designer need only create a macro-design and specify or develop an appropriate block and counter-block. The tedious, time-consuming and error prone task of mapping the profile design to the interlacement array is handled automatically in software. Much time can be saved by this division of the development process into two stages, where only the first stage, being the creation of the actual pattern and blocks, requires the designer's direct attention. The second stage, that of providing the design with the necessary structural integrity and fine pattern detail, is handled separately and independently by the computer.

The graphical display of the detailed interlacement array requires that the appropriate matrix of pixels be addressed for every element of the design. Since the substitution is not pre-defined and the user may, in fact, wish to define an entirely new block and counter-block, the system cannot be developed with a mapping to a completely pre-defined set of graphic output primitives for this application. Either the graphic primitives must be chain encoded during the actual execution of this program segment, or the block and counter-blocks must be individually drawn, pixel by pixel.

The test system uses both of these solutions. Graphic displays of designs with block substitution are drawn at a number of different scales ranging from one to ten pixels per intersection. In any display which uses more than one pixel per intersection, the appropriate interlacement matrix, either the block or counter-block, is plotted in the required position. In the case of a display involving one pixel per intersection, the block and counter-block data is used as input to a program segment which creates a shape table containing the two desired blocks. The data file is then mapped to the design file through this shape table, in a similar manner to that used for generating the cross-sectional and flat sectional representations.

6.5 APPLICATIONS PROGRAMS

One of the major requirements of an interactive textile design system is that there be a facility for the appropriate graphic display of the design data. In fulfilling this requirement, three applications programs, namely Design Input, Data Storage and Design Manipulation, are crucial in supporting the graphical display system,. Two additional applications programs have also been developed (Figure [6.2.1]). They provide implementations of computational algorithms for the analysis of structural properties associated with particular binary interlacement arrays, and a physical interface to a dobby loom, respectively.

6.5.1 DESIGN INPUT

The first of these application program segments, Design Input, provides a support environment for the display program because it creates the application data file. This program is in turn supported by the graphic display system which establishes the proper environment for user data input, as already discussed in Section (6.4.2).

6.5.2 <u>DATA STORAGE</u>

6.5.2.1 <u>DESIGN STORAGE</u>

The second major application is that of data storage. Having created an application data file in RAM and then graphically displayed this data, mapped it to a design file, or used it as input to one of the applications programs, we need some long-term non-volatile storage of the data file, such as on a floppy or hard disk. In this way, a personal library of textile designs can be kept. In addition to storage of the data itself there is of course a need to store a text file or catalogue of the names associated with each of the designs. This allows data files to be accessed in the storage medium by name and loaded into RAM, or for obsolete files to be deleted and their storage space released.

In the system as implemented, application data files are stored on 5.5 inch floppy disks. Each stored design consists of a binary file containing the contents of the entire 8K region of memory corresponding to the second high resolution graphics screen plus matrix dimensions and system variables. A text file is created that serves as a design catalogue operating separately from, and in addition to, the disk operating system catalogue.

6.5.2.2 ARCHIVES AND LIBRARY STORAGE

A second organized system of long-term data storage is also required for archives of classical patterns and structures such as those discussed in Chapter 2. Due to the extremely large volumes of data involved in these files, a search algorithm for designs with particular characteristics or size is also required.

In developing a twill array from an initial binary interlacement sequence, the rule to get from one row to the next is a simple shift through one position, with cyclic wrap-around. Thus, the number of possible twills with a given repeat size is defined completely by the number of inequivalent first rows. This number also specifies the number of different twills that can be woven on a given number of shafts. Consequently, in creating a twill database only the inequivalent first row sequences need be stored.

In the implemented system each interlacement sequence has been stored as a bit string in two bytes of storage, with sequences shorter than sixteen bits being padded with zeros. The sequences themselves are arranged in a canonical form such that the lowest order bit is always non-zero. It thus becomes a simple matter to find the first sequence of a given order by performing a count of the number of leading non-zero bits in each byte pair up to the required number. Having found this beginning point, an individual record can be readily isolated by counting forward in units of two bytes.

The database of twillins, color alternate twills and color alternate twillins is comprised of large binary files stored in an exactly similar format to the twills, where for example, the file designator of color alternate twillins indicates that complementation is required in the matrix generation rule. In the case of twillins and color alternate twillins, because the order of these structures has been restricted to sixteen shafts in the database, and because there is only one shift value which produces an isonemal array for any given sequence length, it has not been necessary to store the number of places through which a given sequence must be shifted in generating all subsequent rows.

The database of compound twillins with single, double or no complementation has been stored in three separate files. To simplify the generation algorithm, the first and second rows for each structure have been stored as bit strings, and again the file designator indicates the presence or absence of complementation in the generating rules.

Any of these archived structures can be used to generate a point diagram quickly and easily, simply by using the chosen interlacement array as a tie-up matrix and invoking the point diagram generation algorithm. Obviously an appropriate threading and shed sequence matrix need also be supplied, and this can be drawn from the archives as well.

Each of the inequivalent binary sequences used to generate the twills can be used to define all of the possible threadings and treadlings with a given number of breaks [44]. Each element of the sequence can be interpreted as a directed line segment of a specified length, so that the entire sequence defines the points and straight runs in a particular threading or treadling.

In the implemented system, the run length must be specified by the user. This value is then checked by the program to determine that it is a divisor of the number of shafts or treadles, whichever is appropriate. By way of an example to illustrate this approach, the design in Figure (6.5.2.2.1) was developed by first selecting a 16 x 16 compound twillin as a tie-up matrix. Next a threading was selected from the database. The particular one chosen had a sequence of 0 0 1 0 1 1 0 1 1 1 1 1 and a run length of four. Finally, a treadling was selected corresponding to the sequence 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 1, also with a run length of four.

A library of binary interlacement arrays commonly used as blocks and counter-blocks in profile substitution is another long-term storage requirement. This removes the need of defining these particular matrices every time such a mapped data display is required.

In the test system, each of the blocks and counter-blocks stored in the library corresponds to a text file of ones and zeros, with a corresponding catalogue of file names. These files can be loaded from disk

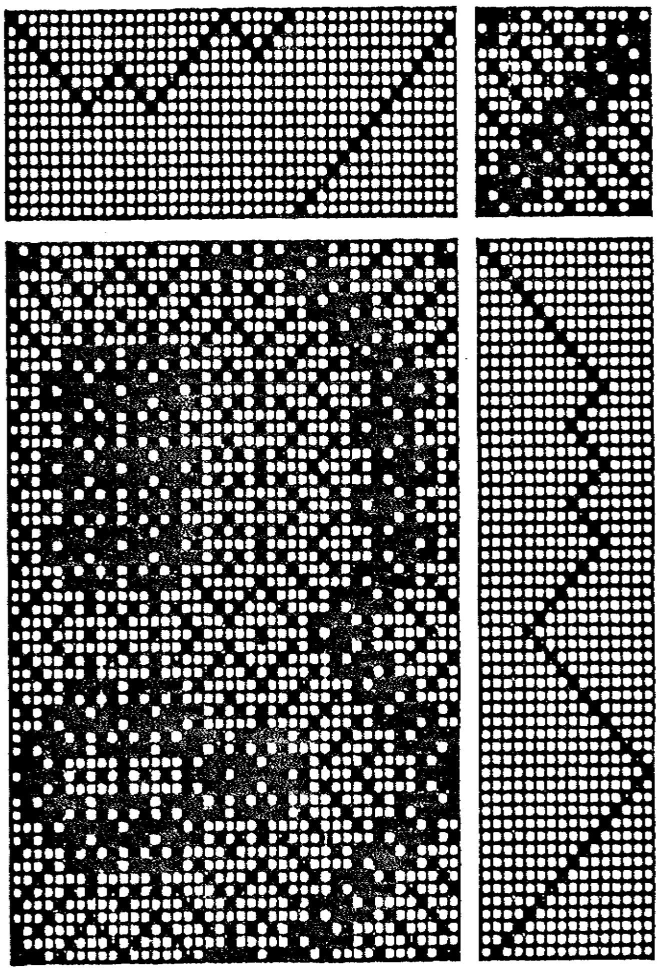


FIGURE 6.5.2.2.1

by the user, in which case each character is mapped to a numeric value of one or zero to be stored in a single byte in RAM. Alternatively the file can be deleted, so that new blocks and counter-blocks can be entered as character strings.

6.5.3 DESIGN MANIPULATION.

Frequently after creating an interlacement pattern, a designer wishes to know how the design will change if a particular transformation is applied to it. In the case of a hand drawn point diagram, the answer to this question is often obtained only after tedious and time-consuming re-drawing of the entire pattern. An interactive textile design and display system can therefore prove invaluable in facilitating modifications to an existing design.

Conceptually, the design manipulation and editing operations are performed by an application program acting on the application data structure, with the operations involved falling into three categories, namely:

1. Alteration of the state of data elements corresponding to design pixels whose positions are unrelated. This corresponds to a change in the type of intersection, either from warp over weft to weft over warp or vice versa, and is in fact simply a part of the design input process.
2. Alteration of the sequence of data elements corresponding to design pixels arranged in either rows or columns. Inserting a row or column, deleting a row or column or indeed, deleting the entire data structure are legitimate operations in this category.
3. Taking a subset of the application data structure elements and

tessellating the entire data file with these elements in such a way that the tile corresponding to each of these sets in the display file corresponds to a non-overlapping contiguous set of pixels.

It should be noted that cyclic rotation through a specified number of rows or columns, horizontal and vertical reflection, as well as transposition of the design should be regarded as operations included under category (2) above. These operations are extremely important in that they enable the examination and manipulation of the various symmetries of the pattern. The option to use the design as a pattern tile to tessellate the design space is also very important since most fabrics are constructed from a pattern with a finite repeat.

In the test system as implemented the category (1) operations are performed as part of the "Design Entry Menu", whereas the design manipulation features in category (2) are contained in the "Design Edit Menu". Due to the relatively slow processing speed of the microcomputer on which this system is resident, the cyclic rotation operations which require re-mapping of the entire application data file with updating to the design file have not been implemented.

Transposition of the design also requires complete re-mapping of the data file. However, this feature has been implemented so that the warp and weft strands can be interchanged. This is important because it means that the standard algorithm for mapping the data file to a cross-sectional representation can be applied to a cut through either the weft yarns or the warp yarns of the fabric.

The remaining options allow specific rows or columns of the design to be deleted, as well as the insertion of a row or column at a specified point. The option to restore the original unedited design is also included, in keeping with the design principle of allowing the user to recover gracefully from errors.

Tessellation of the screen with the pattern corresponding to the data file, as outlined in category (3) above, has been implemented in the "Color Display Menu". This is actually a part of the graphic display system but, because of the way that the data is stored in RAM, this program segment can actually be used to modify the application data structure as well as the design file. When the design is displayed in black and white using the finest detail, each pixel represents precisely one data element. By moving the entire 8K of memory corresponding to this bit map from the page one high resolution graphics screen to page two, the bit pattern is now in the correct format for the data file.

6.5.4 STRUCTURE AND ANALYSIS

At some point in the development of an interlacement pattern there comes the realization that the design is not merely a graphical display of black and white squares but a physical fabric with attendant structural properties and inherent integrity. Thus, any textile design system must incorporate some facility for interpreting and analyzing these structural properties.

The structural analysis process can be thought of as having two phases. The first phase is to interpret the complete design as a set of instructions for the set-up and operation of a loom, and the second phase is to examine the structural integrity of the fabric created according to these instructions. There is really no preferred order of execution of these two stages. In fact, as with all the other design steps, some amount of interplay is to be expected. The flexibility and ease with which a computer based system allows this interplay is of course one of its significant advantages.

All of the algorithms discussed in Chapter 3 for factoring a binary interlacement array into its corresponding threading, tie-up and shed sequence matrices require the rigorous comparison of individual data elements. This operation, which is routine and laborious, is ideally suited to the computer environment. Once analyzed of course, the resultant threading and tie-up must be compared with the available resources of the loom to be used. Should the number of shafts or treadles exceed the number available, some design modification will be required.

The inverse of this process is also required in an interactive design system. That is, this system must also be capable of dealing with the situation where a set of weaving instructions (i.e. threading, tie-up and shed sequence matrices) is developed and a representation of the resultant fabric design is required.

Finally, it is useful for a design system to contain an implementation of the reducibility algorithms described in Chapter 4. This form of analysis will determine <u>a priori</u> whether a certain binary interlacement array, when woven, will produce a single or multi-layered fabric. The combined need for data handling, numeric computation and graphical display which this process demands make it highly suited to computer processing.

6.5.5 <u>DOBBY LOOM INTERFACE</u>

Having performed a dobby analysis on a given binary interlacement array, the textile designer has determined the loom threading as well as the pegging plan [68, p. 98] required to weave the corresponding fabric. This pegging plan can be used as a set of instructions for manually setting the chain of pegged lags. Alternatively, by using a loom which interfaces directly to a microcomputer [eg. AVL Compu-dobby and Macomber's Weaver's Delight] the pegging plan data can be transmitted directly to the loom without the use of mechanical lags.

The test system has been implemented to interface directly to the AVL Compu-dobby loom, in which the mechanical dobby head is replaced by a bank of sixteen solenoids connected by a ribbon cable to an interface card within the Apple computer. Each row of the pegging plan is interpreted as the decimal equivalent of two eight bit binary integers. These two integers are stored in two bytes of memory and passed to the solenoid box where the corresponding pattern of solenoids is activated. The dobby mechanism is equipped with a light emitting diode and the sweep arm which actuates the shafts incorporates a mirror on it which continuously reflects the emitted light back to an optical sensor. In this way, the position of the sweep arm is detected by the hardware which is thereby able to determine when the solenoids should be re-set for the next pick.

An inherent limitation of mechanical dobby looms is that every row of a design repeat requires a lag in proper sequence and little advantage can been taken of pattern regularity. Special purpose attachments have been constructed to enable highly structured block weaves to be handled more easily, [29, p.22], [11, p.313-316]. These have achieved varying degrees of success. The computer interfaced dobby loom, with its library of block structures and automatic substitution algorithms, is ideally suited to overcoming this limitation.

Just as the profile matrix and substitution blocks provide a short-hand version of specifying an interlacement array, so a profile pegging plan can be used to abbreviate the length of the required pegging plan. Each element of the profile pegging plan is mapped through the appropriate block or counter-block to give the complete sequence for a given row. This saves enormously in the amount of time and space required in creating block designs, as well as eliminating an obvious source of designer error.

In the test system, this facility is combined with the profile substitution portion of the archivist system to provide full access to the established library of blocks and counter-blocks. Each time that a new pattern row is read, as many pegging sequences as there are rows in the substitution block are generated and sent in sequence to the solenoids.

The addition of supplementary ground weaves, as used, for example, in weaving Overshot designs [8, p. 174 - 263] require that one of two possible 1/1 plain picks be inserted between successive pattern picks. In a similar manner this can be implemented on the test system merely specifying the pattern rows and the order in which the plain picks are to be interwoven. The actual plain weave sequences are then computed automatically and unambiguously from the threading.

Implementation of the weaving sequence is tracked visually by a white bar which is drawn across the page two graphics screen at the row which is currently being woven. As successive design rows are read, this white bar moves, either forward or backward as specified by the user. An alternative display on page one shows each pattern row as it is read, along with a representation of the virtual lag and a notation of the current design row.

REFERENCES

1. Aho, Alfred V., Hopcroft, John E. and Ullman, Jeffrey D., Data Structures and Algorithms, (Reading, Mass.: Addison-Wesley Publishing Company, 1983)
2. Anderson, Clarita, Gordon, Judith and Towner, Naomi Whiting, Weave Structures Used in North American Coverlets, (Olney, Maryland: Clarita Anderson, 1979)
3. Barlow, Alfred, The History and Principles of Weaving by Hand and by Power, (London: Sampson Low, Marston, Searle & Rivington, 1878), p. vii
4. Bell, T.F., Jacquard Weaving and Designing, (London: Longmans, Green, and Co., 1895)
5. Black, Mary E., New Key to Weaving, (New York: Macmillan Publishing Co. Inc., 1957), p. 471
6. Bradbury, Fred, Jacquard Mechanism and Harness Mounting, (Belfast: F. Bradbury, 1912)
7. Burnham, Dorothy K., Warp and Weft: A Textile Terminology, (Toronto: Royal Ontario Museum, 1980)
8. Burnham, Harold B., and Burnham, Dorothy K., Keep Me Warm One Night: Early Handweaving in Eastern Canada, (Toronto: University of Toronto Press, 1972)
9. Burnside, W., Theory of Groups of Finite Order, (Cambridge: Cambridge University Press, 1897)

10. Clapham, C.R.J., When a Fabric Hangs Together, Bull. London Math. Soc. 12 (1980), 161-164

11. Collingwood, Peter, The Techniques of Rug Weaving, (London: Faber and Faber, 1968)

12. Collingwood, Peter, The Techniques of Sprang: Plaiting on Stretched Threads, (New York: Watson-Guptill Publications, 1974)

13. Cyrus-Zetterstrom, Ulla, Manual of Swedish Handweaving, translated by Alice Blomquist, (Newton Centre, Mass.: Charles T. Branford Co., 1977)

14. Emery, Irene, The Primary Structure of Fabrics, (Washington, D.C.: The Textile Museum, 1980)

15. Enns, T.C., An Efficient Algorithm Determining when a Fabric Hangs Together, Geometriae Dedicata 15 (1984), 259-260

16. Fabek, Diane, Long-Eyed Heddles and Rising Shed Looms, Interweave 4, 4 (1979), p. 28

17. Foley, J.D. and Van Dam, A., Fundamentals of Interactive Computer Graphics, (Philippines: Addison-Wesley Publishing Company, 1982)

18. France, Joseph, The Weaver's Complete Guide or The Webb Analyzed, (Rhode Island: Joseph France, 1814)

19. Freisner, Betty, A Note - Double-Twill on a Jack Loom Using 2 Sets of Ground Harnesses Equipped with Special Long-Eyed Heddles, The Complex Weavers Newsletter, June, 1980

20. Frey, Berta, Designing and Drafting for Handweavers, (New York: Collier Books, 1958), p. 10

21. Galvin, Nellie L. (editor), A German Weaver's Pattern Book 1784 - 1810, (Cuyahoga Falls, Ohio: Nellie L. Galvin, 1961)

22. Grosicki, Z.J., Watson's Advanced Textile Design: Compound Woven Structures, 4th ed., (London: Butterworth & Co. (Publishers) Ltd., 1977)

23. Grosicki, Z., Watson's Textile Design and Colour: Elementary Weaves and Figured Fabrics, 7th ed., (London: Newnes-Butterworths, 1975)

24. Grünbaum, Branko and Shephard, Geoffrey C., Satins and Twills: An Introduction to the Geometry of Fabrics, Mathematics Magazine 53 (1980), 139-161

25. Hauptmann, Bruno, Gewebetechnik, (Leipzig: Fachbuchverlag GmbH, 1952), p. 138

26. Hilts, Patricia, An Eighteenth Century German Court Weaver: Johann Michael Frickinger, Shuttle, Spindle and Dyepot 44 (1980), 16

27. Hilts, Patricia, Seventeenth and Eighteenth Century German Twills, Second Annual Conference on Complex Weave Structures, Mineral Point, Wisconsin, July, 1984

28. Holy Bible, 2 Kings, 23, 7

29. Hooper, Luther, Hand-Loom Weaving: Plain & Ornamental, (London: Pitman Publishing Limited, 1979 - first published in 1910)

30. Hoskins, J.A., Binary Interlacement Arrays and Structural Cross-Sections, Cong. Num. 40 (1983), 63-76

31. Hoskins, Janet A., Computerized Fabric Analysis, Shuttle, Spindle and Dyepot, 13 1 (1982), 26-27

32. Hoskins, J.A., Factoring Binary Matrices: A Weaver's Approach, Lecture Notes in Math. 952, Springer-Verlag (1982), 300-326

33. Hoskins, Janet A., Multi-layered Cloths: A Structured Approach, Ars Textrina 1 (1983), 137-158

34. Hoskins, J.A. and Hoskins, W.D., Algorithms for the Design and Analysis of Woven Textiles, Proceedings of the 1983 ACM Conference on Personal and Small Computers, 153-160

35. Hoskins, Janet A. and Hoskins W.D., The Solution of Certain Matrix Equations Arising from the Structural Analysis of Woven Fabrics, Ars Comb. 11 (1981), 51-59

36. Hoskins, J.A. and Hoskins, W.D., A Faster Algorithm for Factoring Binary Matrices, Ars Comb. 16b (1983), 341-350

37. Hoskins, J.A., Hoskins, W.D., Street, Anne Penfold and Stanton, R.G., Some Elementary Isonemal Binary Matrices, Ars Comb. 13 (1982), 3-38

38. Hoskins, J.A. and King, M.W., An Interactive Database for Woven Textile Design, Proceedings of the Textile Institute Annual Conference 1984, Hong Kong

39. Hoskins, J.A. and King, M.W., Interactive Design of Woven Textiles, Proceedings of the International Computer Color Graphics Conference, Tallahassee, Florida, 1983

40. Hoskins, Janet A., Praeger, Cheryl E. and Street, Anne Penfold, Balanced twills with bounded float length, Cong. Num. 40 (1983), 77-89

41. Hoskins, Janet A., Praeger, Cheryl E. and Street, Anne Penfold, Twills with bounded float length, Bull. Austral. Math. Soc. 28 (1983), 225-281

42. Hoskins, Janet A., Stanton, R.G. and Street, Anne Penfold, The Compound Twillins: Reflection at an Element, Ars Comb. (1984), 177-190

43. Hoskins, J.A., Stanton, R.G. and Street, Anne Penfold, Enumerating the Compound Twillins, Cong. Num 38 (1983), 3-22

44. Hoskins, W.D. and Street, Anne Penfold, Twills on a given number of harnesses, J. Austral. Math. Soc. (Series A) 33 (1982), 1-15

45. Hoskins, W.D. and Thomas, R.S.D., Conditions for Isonemal Arrays on a Cartesian Grid, J.L.A.A. 57 (1984), 87-103

46. Jacob Angstadt: His Weavers Patron Book, Replica of an 18th Century Manuscript Book owned and reproduced by Ruth N. Holroyd, (Hartford, Ct.: Ruth N. Holroyd, 1976)

47. King, M.W., Guidoin, R.G., Gunasekera, K.R. and Gosselin, C., Designing Vascular Prostheses for the Future, Med. Progr. Technol. 9 (1983), 217-226

48. Kocay, W., private communication

49. Koob, Katherine, Keller, Barbara and Howard, Ruth, The Use of Long-Eyed Heddles, The Complex Weavers Newsletter, No. 3, April, 1980

50. Lourie, Janice R., Loom-Constrained Designs: An Algebraic Solution, Proceedings of the ACM National Conference (1969), 185-192

51. Lourie, Janice R., Textile Graphics/Computer Aided, (New York: Fairchild Publications, Inc., 1973)

52. Lourie, Janice R., The Textile Designer of the Future, Handweaver and Craftsman, Winter 1966, 8

53. Lourie, Janice R. and Bonin, Alice M., Computer-Controlled Textile Designing and Weaving, Proceedings - IFIPS (1968), 884

54. Lourie, Janice R., Lorenzo, John J. and Bomberault, Abel, On-Line Textile Designing, Proceedings of the ACM National Meeting (1966), 537

55. Malloy, Bro. Kim, Two Jacquard Coverlet Weavers,Second Annual Conference on Complex Weave Structures, Mineral Point, Wisconsin, July, 1984

56. McDonnell Douglas advertisement, IEEE Spectrum,, 21 7 (1984), 16 F/G

57. Montgomery, Pauline, Indiana Coverlet Weavers and Their Coverlets, (Indianapolis: Hoosier Heritage Press, 1974)

58. Murphy, John, A Treatise on the Art of Weaving with Calculations and Tables for the Use of Manufacturers, (Glasgow: Blackie & Son, 1836)

59. Newton, A. and Sarkar, B.P., An Analysis of Compound Weaves, J. Text. Inst., 10 (1979), 427-438

60. Nisbet, H., Grammar of Textile Design, 3rd ed., (London: Benn, 1927)

61. Nishikawa, Shigeru, Automatic Patterning Technique on Jacquard Weaving Process, Bulletin of Research Institute for Polymers and Textiles, No. 105, (1974), 5

62. Pavlidis, Theo, Algorithms for Graphics and Image Processing, (Rockville, MD: Computer Science Press, 1982), p. 6

63. Pedersen, Jean J., "Some Isonemal Fabrics on Polyhedral Surfaces", The Geometric Vein: The Coxeter Festschrift, edited by Chandler Davis, Branko Grünbaum and F.A. Sherk, New York: Springer-Verlag New York Inc., 1981)

64. Reid-Green, Keith S., A Short History of Computing, BYTE 3, 7 (1978), 84

65. Robinson, A.T.C. and Marks, R., Woven Cloth Construction, (Manchester: The Textile Institute, 1973)

66. Shakespeare, William, King Henry VI - Second Part, Act 3, Scene 3

67. Shakespeare, William, Merry Wives of Windsor, Act 5, Scene 1

68. Straub, Marianne, Hand Weaving and Cloth Design, (London: Pelham Books, 1977)

69. Sutton, Ann, The Structure of Weaving, (London: Hutchinson & Co. (Publishers) Ltd., 1982), p. 13

70. Sutton, Ann, Collingwood, Peter and St. Aubyn Hubbard, Geraldine, The Craft of the Weaver, (Asheville, N.C.: Lark Books, 1983), p. 88

71. Vincent, J.J., Shuttleless Looms, (Manchester: The Textile Institute, 1980), p. 13-20

72. Wilson, Sadye Tune and Kennedy, Doris Finch, Of Coverlets: The Legacies, The Weavers, (Nashville: Tunstede, 1983)

73. Woods, H.J., The Geometrical Basis of Pattern Design, Textile Institute of Manchester Journal 26 (1935), Sect. 2, Transactions, 197-210, 293-308, 341-357

74. Worst, Edward F., Weaving with Foot-Power Looms, (New York: Dover Publications Inc., 1974), p. 204

75. Zielinski, Stanislaw A., Encyclopedia of Hand-Weaving, (New York: Funk & Wagnalls, 1959)