AN ABSTRACT OF THE THESIS OF

Frederick Wesley Weingarten, Jr. for the Ph. D. in Mathematics
(Name) (Degree) (Major)

Date thesis is presented April 11, 1966

Title ON TRIANGULATIONS OF FIXED POINTS IN THE PLANE

Abstract approved Redacted for privacy
(Major professor)

A theory of straight line triangulations of points in the plane is developed. A basic transformation is presented, and it is shown that any triangulation may be transformed into any other triangulation which has the same boundary by a finite sequence of the basic transformations.

The proof of the transformation theorem involves the presentation of an algorithm. Programs for processing triangulations of points are developed and coded in FORTRAN IV and in flow chart form.

ON TRIANGULATIONS OF FIXED POINTS IN THE PLANE

by

FREDERICK WESLEY WEINGARTEN, JR.

A THESIS

submitted to

OREGON STATE UNIVERSITY

in partial fulfillment of
the requirements for the
degree of

DOCTOR OF PHILOSOPHY

June 1966

APPROVED:

Redacted for privacy

Professor of Mathematics

In Charge of Major

Redacted for privacy

Head of Department of Mathematics

Redacted for privacy

Dean of Graduate School

Date thesis is presented April 11, 1966

Typed by Marion F. Palmateer

ACKNOWLEDGEMENT

The author would like to express his appreciation to Dr. H. Goheen and Dr. C. L. Lawson for their valuable assistance with this research. He would also like to acknowledge the opportunity provided by the Jet Propulsion Laboratory, Pasadena, California, to use their facilities for some of the work described in these pages.

TABLE OF CONTENTS

LIST OF FIGURES

ON TRIANGULATIONS OF FIXED POINTS IN THE PLANE

INTRODUCTION

This paper considers the properties of straight line triangulations of points whose position is fixed in the plane. This problem has received little attention in the literature, however the need for such a study has recently become apparent. In fact, the work presented here has arisen from a problem in applied mathematics, that of multivariate interpolation of a function given data points randomly scattered in the plane.

To find local interpolating surfaces it is necessary to divide the plane into regions which all have common geometric properties so that a general interpolating function can be applied to each region. The reasonable choice of region is a triangle, since any non-collinear set of points in the plane may be triangulated. That is, the points may be connected together to form a network of triangles. In most cases we consider the network of triangles to be contained in the polygon which forms the convex hull of all the points.

The specific problem which has been addressed in this work is that of studying the computer generation and modification of the triangulations. There is a theoretical section in which a theory of triangulations is developed, and there is an algorithms section in which this theory is applied in the form of algorithms for computer operations on triangulations.

The property of triangulations of points which motivates much of the theory is that of non-uniqueness of the networks of triangles. Even in the simple case of four points which form, for instance, a rectangle there are two distinct triangulations, as is shown in Figure I-1.

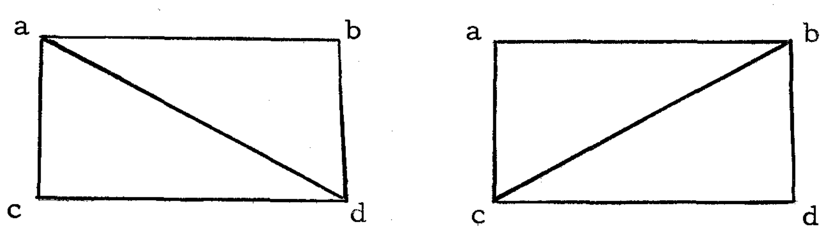


Figure I-1. The Two Triangulations of a Rectangle.

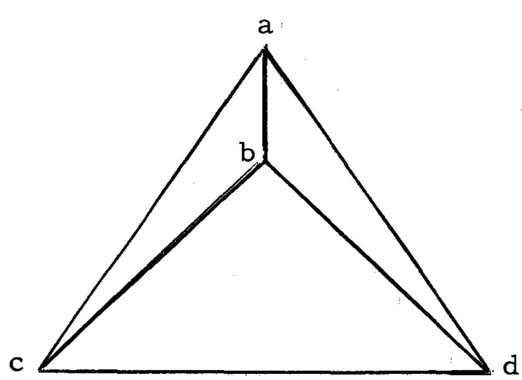


Figure I-2. A Unique Triangulation of Four Points.

However, Figure I-2 demonstrates that four points may be placed in such a way that there is only one possible triangulation.

That the set of points in Figure I-2 can only be triangulated in the manner shown follows from the corollary to Theorem II presented in section II of this paper. This corollary gives a test for uniqueness which can easily be applied to any triangulation.

A question which arises naturally is this. Given a particular, non-unique triangulation of a set of points, is there some way of producing the other triangulations from this initial one? This question is especially important in the interpolation problem, where the initial triangulation may be quite unsatisfactory for use with the approximation functions. But the question is also interesting in the abstract. Obviously, the problem of finding all possible triangulations of a given set of points is computable in the classical sense, since all subsets of the non-diagonal elements of PxP could be generated and examined to determine if they were legitimate triangulations. But this involves a great deal of computation. The number of non-empty subsets of the set of non-diagonal elements of PxP is $2^{n^2-n}-1$, and while it is true that cases could be eliminated, the amount of work involved in any simple-minded algorithm for studying the subsets in the case of a reasonably large number of points is still much too great. It would be best if an algorithm could be developed which could generate all possible triangulations by starting at one and transforming it into a new one.

This question of generation of triangles is partially answered by this work. A simple transform is developed in section II-C and

Theorem II shows that the set of triangulations having a common boundary is connected by this transform. That is, one triangulation may be transformed into any other triangulation by a finite sequence of the simple transforms.

The theoretical section sets up a general definition of a triangulation which places little restriction on the shape of the boundary. Thus, Theorem II has a generality which is not fully used in the algorithms section, but which is interesting and potentially important mathematically. The condition of a convex boundary mentioned above could be dispensed with.

The algorithms section of this paper presents some basic algorithms necessary for the study of triangulations in the plane. Memory is structured as a set of lists which allow the transformations to be made simply and which also allow any information about the triangulations to be accessed with a relatively small amount of computation. Programs are written to triangulate the region defined by the convex hull of the point set by generating successive convex hulls and then by filling in between the generated hulls. This procedure forms a complete triangulation. Additional subroutines are provided for modifying the information in the list structures in various ways.

The algorithms are written in FORTRAN IV, and flow charts are also included. FORTRAN IV was chosen as a convenient algorithmic language, since it would be easily modified to be implemented on most machines. However, no guarantee is made that they will run, as written, on any particular system. The algorithms in section III are intended to set forth general procedures and do not represent existing programs.

Some of the routines have, however, been coded and run on an IBM 7094-7040 direct-coupled system at Jet Propulsion Laboratory, Pasadena, California. Output from the direct-coupled system was put on magnetic tape in a format usable by a Stromberg-Carlson 4020 cathode ray tube display system which produced the results in graphical form on photographic film.

Some discussion of the historical references to the main problem attacked in this paper will be made before proceeding to the main body of the research. References to the problem of the properties of fixed line triangulations are virtually non-existant. However, a brief survey of work in tangential fields may help to provide an orientation for the work presented here.

The generation of a triangulation has been considered elsewhere. For instance, in a private communication to Dr. C. L. Lawson at Jet Propulsion Laboratory, B. Dimsdale discusses the generation of successive convex hulls in setting up an initial triangulation. Other investigators have, at some time, needed and used similar processes. But, nothing appears to have been published; possibly due to the apparent triviality of that particular problem. The method used by the author is that of B. Dimsdale, but the proof that it will always work is original.

In itself, the problem of efficiently generating a convex hull of a set of points requires some work. Butler, [4], has developed an algorithm for this process in 3-space. The author uses a different procedure which seems to be more direct and simpler in concept, although the important parameter of difference, run time on a computer, has not been examined. In this paper, the hull is generated by, first, finding the left-hand-most point in the set. The algorithm then proceeds around the hull determining if the line from the previous point to the next point under consideration has the maximum angle to the x-axis of all points remaining. This fact is determined by using the signed area of the triangle formed by: (1) the previous point known to be the hull, (2) the point presently determined to be the best after a partial search, and (3) the next unexamined point in the set. If the orientation of (1, 2, 3) is counter-clockwise, the point (3) is taken to be the new best-to-date.

Study of planar triangulations has proceeded in graph theoretical directions and is represented by the papers of Tutte, [12], and

Brown, [3], among others. These studies are combinatorial in nature and consider triangular graphs which are not necessarily formed by straight lines. This type of analysis makes the problem of counting triangulations amenable to the techniques of combinatorial analysis, but it has the drawback for our purpose that it is topological rather than metric. As was shown in Figures I-1 and I-2, the question of how many straight edge triangulations can be generated has no unique answer without specifying information about the arrangement of the points.

In addition to the graph theoretical work which has been done, some statistical research has been done on the properties of convex hulls of randomly scattered points in the plane, ([2], [9], [10] and [11]). References [1], [5], and [6] contain work carried out in a similar geometric framework as in this paper, but they do not refer to this problem of triangulations directly.

Computers have been used for graphical studies by several researchers, including Lee [7], who solved a shortest path problem by developing a set of algorithms which were then implemented on a digital computer. This work of Lee's is, thus, analogous to the work presented here.

There are two main contributions in this paper. First, an analytical framework and a major theorem in the study of straight line triangulations of fixed points is presented. Since this field has not been previously developed in the literature, it was necessary to invent a theory of triangulations along with the presentation of a major theorem. Section II contains the analytical portion of this work. Second, there is developed a technique for the study of these triangulations on a computer.

II. THEORY OF TRIANGULATIONS

A. Basic Definitions

Let $\underline{P}$ be any finite set of points in the plane. We define a function of an ordered triplet of points in $\underline{P}$.

<u>Definition 1</u>: Let $(\underline{a}, \underline{b}, \underline{c})$ be the ordered triplet of distict points $\underline{a}$, $\underline{b}$, and $\underline{c}$ in $\underline{P}$.

(1) $$\underline{\Delta} = \begin{vmatrix} \underline{X}_{\underline{a}} & \underline{Y}_{\underline{a}} & 1 \\ \underline{X}_{\underline{b}} & \underline{Y}_{\underline{b}} & 1 \\ \underline{X}_{\underline{c}} & \underline{Y}_{\underline{c}} & 1 \end{vmatrix} .$$

$\underline{\Delta}$ is the determinant of the matrix formed by columns composed of the X-coordinates of $\underline{a}$, $\underline{b}$, and $\underline{c}$, the Y-coordinates of $\underline{a}$, $\underline{b}$, and $\underline{c}$, and all ones, respectively. It will be used to determine the orientation of the triangle formed by the points. We define

(2) $$\begin{aligned} \text{sign}\,(\underline{a}, \underline{b}, \underline{c}) &= \frac{\underline{\Delta}}{|\underline{\Delta}|}\,, \quad \text{if } \underline{\Delta} \neq 0, \\ \text{sign}\,(\underline{a}, \underline{b}, \underline{c}) &= 0\,, \quad \text{if } \underline{\Delta} = 0. \end{aligned}$$

Let $(\underline{a}, \underline{b})$ and $(\underline{c}, \underline{d})$ be two line segments in the plane, one with endpoints $\underline{a}$ and $\underline{b}$ and the other with endpoints $\underline{c}$ and $\underline{d}$, $\underline{a}$, $\underline{b}$, $\underline{c}$, and $\underline{d}$ being points of $\underline{P}$. Then $(\underline{a}, \underline{b})$ and $(\underline{c}, \underline{d})$ intersect if both of the following two conditions hold

(3)

a) $\text{sign}\,(\underline{a}, \underline{b}, \underline{c}) = -\text{sign}\,(\underline{a}, \underline{b}, \underline{d}),$

b) $\text{sign}\,(\underline{c}, \underline{d}, \underline{a}) = -\text{sign}\,(\underline{c}, \underline{d}, \underline{b}).$

A simple geometric argument shows that these conditions are consistent with the more common notion of intersection of two non-parallel line segments.

Thus, we will use, later, the notion of a <u>point</u> of intersection. That is, $\exists$ a point $\underline{X}$ not in $\underline{P}$, such that $\underline{X} \in (\underline{a}, \underline{b})$ and $X \in (\underline{c}, \underline{d})$ when $(\underline{a}, \underline{b})$ and $(\underline{c}, \underline{d})$ intersect.

Let $\underline{\tilde{M}}(\underline{P})$ be any subset of PxP such that

(i) $(\underline{a}, \underline{b}) \in \underline{\tilde{M}}(\underline{P}) \Rightarrow (\underline{b}, \underline{a}) \in \underline{\tilde{M}}(\underline{P}),$

(ii) $(\underline{a}, \underline{a}) \notin \underline{\tilde{M}}(\underline{P}) \quad \forall \; \underline{a} \in \underline{P},$

(iii) $\{(\underline{a}, \underline{b}), (\underline{c}, \underline{d})\} \subset \underline{\tilde{M}}(\underline{P})$ and $(\underline{a}, \underline{b}) \neq (\underline{c}, \underline{d}) \Rightarrow (\underline{a}, \underline{b})$ does not intersect $(\underline{c}, \underline{d}),$

(iv) $\forall \; \underline{a} \in \underline{P} \; \exists \, \underline{b} \ni : (\underline{a}, \underline{b}) \in \underline{\tilde{M}}(\underline{P}),$

(v) $\forall \; \underline{d} \in \underline{P}, \quad (\underline{a}, \underline{b}) \in \underline{\tilde{M}}(\underline{P}) \Rightarrow \underline{d} \notin (\underline{a}, \underline{b}) \quad .$

<u>Definition 2</u>: Let $\underline{\tilde{M}}(\underline{P})$ satisfy properties (i) through (v) above, and let $\underline{M}(\underline{P})$ be the canonical set of equivalence classes of $\underline{\tilde{M}}(\underline{P})$ with the equivalence relation

(EL) $(\underline{a}, \underline{b}) \equiv (\underline{c}, \underline{d})$ iff $(\underline{a}, \underline{b}) = (\underline{c}, \underline{d})$ or $(\underline{a}, \underline{b}) = (\underline{d}, \underline{c}).$

Then $\underline{M}(\underline{P})$ is called a lineation of $\underline{P}$.

Definition 3: Let $\underline{a}, \underline{b}, \underline{c}$, and $\underline{d}$ be distinct points in $\underline{P}$. Then $\underline{d}$ is internal to $(\underline{a}, \underline{b}, \underline{c})$ iff

(4) $$\text{sign}(\underline{a}, \underline{b}, \underline{d}) = \text{sign}(\underline{a}, \underline{d}, \underline{c}) = \text{sign}(\underline{d}, \underline{b}, \underline{c})$$

We can state the following lemmas concerning the sign function and internal points.

Lemma A:

(5) $$\text{sign}(\underline{a}, \underline{b}, \underline{c}) = \text{sign}(\underline{b}, \underline{c}, \underline{a}) = \text{sign}(\underline{c}, \underline{a}, \underline{b}) = -\text{sign}(\underline{b}, \underline{a}, \underline{c})$$
$$= -\text{sign}(\underline{c}, \underline{b}, \underline{a}) = -\text{sign}(\underline{a}, \underline{c}, \underline{b})$$

Lemma B: Let $\underline{a}, \underline{b}, \underline{c}$, and $\underline{d}$ be distinct points in $\underline{P}$. Then $\underline{d}$ is internal to $(\underline{a}, \underline{b}, \underline{c})$ iff $\underline{d}$ is internal to any permutation of $(\underline{a}, \underline{b}, \underline{c})$.

Definition 4: Let $\underline{M}(\underline{P})$ be a lineation of $\underline{P}$. Then a triangle generated by $\underline{M}(\underline{P})$ is an element $(\underline{a}, \underline{b}, \underline{c})$ in PxPxP $\ni$:

(i) $\underline{a}, \underline{b}, \underline{c}$ are distinct points,

(ii) $\{(\underline{a}, \underline{b}), (\underline{b}, \underline{c}), (\underline{a}, \underline{c})\} \subset \underline{M}(\underline{P})$,

(iii) $\underline{d}$ is internal to $(\underline{a}, \underline{b}, \underline{c}) \Rightarrow \underline{d} \notin \underline{P}$.

The sides $(\underline{a}, \underline{b})$, $(\underline{b}, \underline{c})$, and $(\underline{a}, \underline{c})$ of $(\underline{a}, \underline{b}, \underline{c})$ will be called the lines associated with $(\underline{a}, \underline{b}, \underline{c})$ in the discussion.

The set of all triangles generated by a lineation M(P) has an obvious equivalence relation on it.

(ET) $\quad$ $(d, e, f) \equiv (a, b, c)$ iff (d, e, f) is any permutation of the points (a, b, c).

With equivalence relation (ET) we have the following definition of a triangulation.

Definition 5: Let M(P) be a lineation of P and let $\tilde{T}(M, P)$ be the set of all triangles generated by M(P). Then the canonical set, T(M, P), of equivalence classes of $\tilde{T}(M, P)$ with respect to equivalence relation (ET) is called the triangulation of P generated by M(P).

Definition 6: Let $(a, b) \in M(P)$ and let $n(a, b, M)$ be the number of triangles which have (a, b) as an associated line. Then (a, b) is called a border line of M(P) iff $n(a, b, M) \leq 1$, and (a, b) is called an internal line of M(P) iff $n(a, b, M) = 2$. The set of all border lines in M(P) is called the border of (M(P), B(M, P); and the set of all internal lines is called the interior of M(P), I(M, P).

Definition 7: Let M1 and M2 be two lineations of P. Then M1 is said to be congruent to M2 (denoted by $\cong$) iff

(i) $\quad$ $B(M1, P) = B(M2, P)$

(ii) $\quad$ $(a, b) \in B(M1, P) \Rightarrow n(a, b, M1) = n(a, b, M2)$

Statement (ii) rules out the pathological case which is exemplified in Figure II-1.

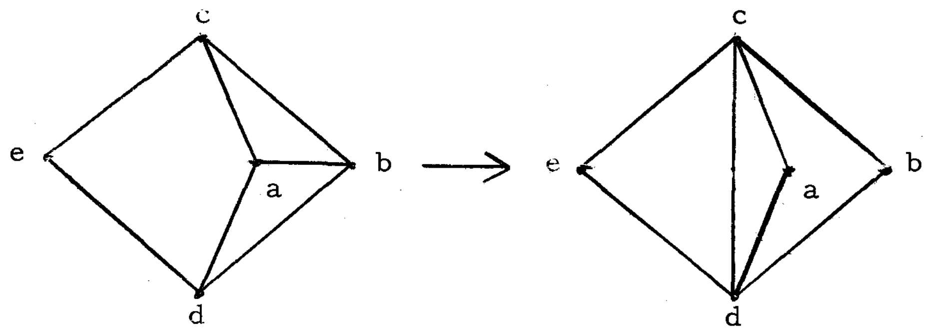


Figure II-1. Two Triangulations Which Fulfill Condition (i) but Violate Condition (ii).

Definition 8: Let $\{\underline{a}, \underline{b}, \underline{c}, \underline{d}\} \subset \underline{P}$ be a set of four distinct points in $\underline{P}$. Then the ordered quadruplet, $(\underline{a}, \underline{b}, \underline{c}, \underline{d})$ is said to be strictly convex iff

(6) $$\text{sign}(\underline{a}, \underline{b}, \underline{c}) = \text{sign}(\underline{b}, \underline{c}, \underline{d}) = \text{sign}(\underline{c}, \underline{d}, \underline{a}) \neq 0$$

This definition could be generalized to any ordered n-tuple of points.

B. Preliminary Lemmas

Some basic lemmas which will be used later in the development are presented here.

Lemma C: Let $\underline{T}(\underline{M}, \underline{P})$ be a triangulation of a point set $\underline{P}$ generated by a lineation $\underline{M}(\underline{P})$. Let $(\underline{a}, \underline{b}, \underline{c}) \in \underline{T}(\underline{M}, \underline{P})$ and $(\underline{a}, \underline{b}, \underline{d}) \in \underline{T}(\underline{M}, \underline{P})$, where $\underline{c} \neq \underline{d}$. Then

(7) $$\text{sign}\,(\underline{a}, \underline{b}, \underline{c}) = -\text{sign}\,(\underline{a}, \underline{b}, \underline{d}) \neq 0$$

Proof: The case sign $(\underline{a}, \underline{b}, \underline{c}) = 0$ is dispensed with immediately, since that would imply that $\underline{a}$, $\underline{b}$, and $\underline{c}$ were collinear; and, since they are distinct, one would necessarily lie in the open line segment determined by the other two, an impossibility as shown by condition (v) of Definition 2 above. We will then assume

(8) $$\text{sign}\,(\underline{a}, \underline{b}, \underline{c}) = \text{sign}\,(\underline{a}, \underline{b}, \underline{d}) \neq 0.$$

We know, by Definition 4, that

(i) $\underline{c}$ is not internal to $(\underline{a}, \underline{b}, \underline{d})$, and

(ii) $\underline{d}$ is not internal to $(\underline{a}, \underline{b}, \underline{c})$.

By equation (4), (8) implies that either

(a) $$\text{sign}\,(\underline{b}, \underline{d}, \underline{c}) = -\,\text{sign}\,(\underline{b}, \underline{d}, \underline{a})\ ,\ \text{or}$$

(9)

(b) $$\text{sign}\,(\underline{a}, \underline{c}, \underline{d}) = -\text{sign}\,(\underline{a}, \underline{b}, \underline{d})\ .$$

And, similarly, either

(a) $$\text{sign}\,(\underline{a}, \underline{b}, \underline{c}) = -\,\text{sign}\,(\underline{d}, \underline{b}, \underline{c}),\ \text{or}$$

(10)

(b) $$\text{sign}\,(\underline{a}, \underline{b}, \underline{c}) = -\,\text{sign}\,(\underline{a}, \underline{d}, \underline{c})\ .$$

Now the pair of equations, (9a) and (10a), as well as the pair (9b) and (10b), implies that

$$\text{sign}(\underline{a}, \underline{b}, \underline{c}) = -\,\text{sign}(\underline{a}, \underline{b}, \underline{d}).$$

But this contradicts equation (8), and thus either (9a) and (10b) hold, or (9b) and (10a) hold. But simple algebra shows that (9a) and (10b) imply that $(\underline{a}, \underline{c})$ intersects $(\underline{b}, \underline{d})$, and (9b) and (10a) imply that $(\underline{a}, \underline{d})$ intersects $(\underline{b}, \underline{c})$, contradicting the hypothesis.

Corollary C. 1: Since Lemma C implies that any line in a triangulation may be associated with at most two triangles, we see that, if $(\underline{a}, \underline{b}) \in \underline{M}(\underline{P})$, either $(\underline{a}, \underline{b}) \in \underline{B}(\underline{M}, \underline{P})$, or $(\underline{a}, \underline{b}) \in \underline{I}(\underline{M}, \underline{P})$. That is, $\underline{B}(\underline{M}, \underline{P})$ and $\underline{I}(\underline{M}, \underline{P})$ form a partition of $\underline{M}(\underline{P})$. It is to be noted that Lemma C furnishes a justification for Definition 6.

Corollary C. 2: $\forall\ (\underline{a}, \underline{b}) \in \underline{M}(\underline{P}), \exists\ \underline{c}$ and $\underline{d} \ni : \{(\underline{a}, \underline{b}, \underline{c}), (\underline{a}, \underline{b}, \underline{d})\} \subset \underline{T}(\underline{M}, \underline{P})$, and

(11) $$\text{sign}(\underline{a}, \underline{b}, \underline{c}) = -\text{sign}(\underline{a}, \underline{b}, \underline{d}).$$

Lemma D: Let $(\underline{a}, \underline{b}) \in \underline{M1}(\underline{P})$ and $(\underline{c}, \underline{d}) \in \underline{M2}(\underline{P})$, where $\underline{M1}(\underline{P}) \cong \underline{M2}(\underline{P})$. Then, $(\underline{a}, \underline{b})$ intersects $(\underline{c}, \underline{d}) \Rightarrow (\underline{c}, \underline{d}) \in \underline{I}(\underline{M2}, \underline{P})$.

Proof: Suppose $(\underline{c}, \underline{d}) \notin \underline{I}(\underline{M2}, \underline{P})$. Then, by Corollary C. 1 and the assumptions of the Lemma, $(\underline{c}, \underline{d}) \in \underline{B}(\underline{M2}, \underline{P})$. Now, $\underline{M1} \cong \underline{M2}$ $\Rightarrow (\underline{c}, \underline{d}) \in \underline{B}(\underline{M1}, \underline{P})$ and, thus, $(\underline{c}, \underline{d}) \in \underline{M1}(\underline{P})$. This is impossible, since $(\underline{c}, \underline{d})$ intersects $(\underline{a}, \underline{b})$.

C. Congruent Triangulations and the $\mathcal{T}$- transform

Let $(\underline{a}, \underline{b}) \in \underline{I}(\underline{M}, \underline{P})$. We can express $\underline{M}(\underline{P})$ as

(12) $$\underline{M}(\underline{P}) = \overline{\underline{M}}(\underline{P}) \cup \{(\underline{a}, \underline{b})\}, \text{ where } \overline{\underline{M}}(\underline{P}) \cap \{(\underline{a}, \underline{b})\} = \emptyset .$$

Now, by Lemma C, $(\underline{a}, \underline{b}) \in \underline{I}(\underline{M}, \underline{P}) \Rightarrow \{(\underline{a}, \underline{b}, \underline{c}), (\underline{a}, \underline{b}, \underline{d})\} \subset \underline{T}(\underline{M}, \underline{P})$ such that $\text{sign}(\underline{a}, \underline{b}, \underline{c}) = -\text{sign}(\underline{a}, \underline{b}, \underline{d}) \neq 0$.

Let

(13) $$\underline{\mathcal{T}} (\underline{a}, \underline{b}, \underline{M}, \underline{P}) = \overline{\underline{M}}(\underline{P}) \quad (\underline{c}, \underline{d}).$$

Definition 9: $\underline{\mathcal{T}}(\underline{a}, \underline{b}, \underline{M}, \underline{P})$ is called the $\mathcal{T}$- transform of M(P) with respect to (a, b).

We would like to have necessary and sufficient conditions for $\underline{\mathcal{T}}(\underline{a}, \underline{b}, \underline{M}, \underline{P}) \cong \underline{M}(\underline{P})$, since it is conceivable that the set after the transform may not even be a lineation. Figure II-2, below, gives an example of such a case.

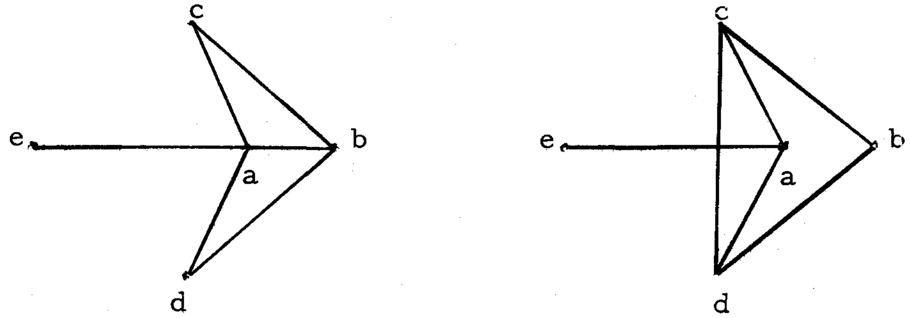


Figure II-2. A Transformation Which Does Not Produce a Lineation.

Figure II-1 suggests the following necessary and sufficient condition for the congruence of $\underline{M}(\underline{P})$ and $\underline{T}(\underline{a}, \underline{b}, \underline{M}, \underline{P})$.

Theorem 1: Let $(\underline{a}, \underline{b}) \in \underline{I}(\underline{M}, \underline{P})$. Then $\underline{T}(\underline{a}, \underline{b}, \underline{M}, \underline{P}) = \underline{\tilde{M}}(\underline{P})$ is a lineation, and $\underline{\tilde{M}}(\underline{P}) \cong \underline{M}(\underline{P})$ iff $(\underline{a}, \underline{c}, \underline{b}, \underline{d})$ is strictly convex.

Proof: i) Let $(\underline{a}, \underline{c}, \underline{b}, \underline{d})$ be not strictly convex. We then have inequality somewhere in equations (6). We may suppose

$$\text{sign}(\underline{d}, \underline{a}, \underline{c}) \neq \text{sign}(\underline{b}, \underline{d}, \underline{a}).$$

If either sign is equal to zero, condition (v) of Definition 2 is violated, and $\underline{\tilde{M}}(\underline{P})$ is not a lineation. Then let

$$\text{sign}(\underline{d}, \underline{a}, \underline{c}) = -\text{sign}(\underline{b}, \underline{d}, \underline{a}).$$

The line $(\underline{b}, \underline{d})$ then is not an associated line of triangle $(\underline{a}, \underline{d}, \underline{b})$. But the triplet $(\underline{c}, \underline{d}, \underline{b})$ will not be in $\underline{T}(\underline{\tilde{M}}, \underline{P})$ since simple algebra shows that a is internal to $(\underline{c}, \underline{d}, \underline{b})$. However, any other triangle in $\underline{T}(\underline{\tilde{M}}, \underline{P})$ which is associated with $(\underline{b}, \underline{d})$ must be in $\underline{T}(\underline{M}, \underline{P})$. Then the number of triangles associated with $(\underline{b}, \underline{d})$ is decreased by one. This is not possible if $\underline{\tilde{M}}(\underline{P}) \cong \underline{M}(\underline{P})$. Thus, either $\underline{\tilde{M}}(\underline{P})$ is not a lineation, or $\underline{\tilde{M}}(\underline{P}) \not\cong \underline{M}(\underline{P})$.

ii) Let $(\underline{a}, \underline{c}, \underline{b}, \underline{d})$ be strictly convex. Then by Definition 8 we have

(14) $$\text{sign}(\underline{a}, \underline{c}, \underline{b}) = \text{sign}(\underline{c}, \underline{b}, \underline{d}) = \text{sign}(\underline{b}, \underline{d}, \underline{a}) = \text{sign}(\underline{d}, \underline{a}, \underline{c}).$$

From equation (14) we see that

(15) $$\operatorname{sign}(\underline{d}, \underline{a}, \underline{c}) = -\operatorname{sign}(\underline{d}, \underline{b}, \underline{c}).$$

and, by Lemma C,

(16) $$\operatorname{sign}(\underline{a}, \underline{b}, \underline{c}) = -\operatorname{sign}(\underline{a}, \underline{b}, \underline{d}).$$

But equations (15) and (16) imply that $(\underline{a}, \underline{b})$ and $(\underline{c}, \underline{d})$ intersect, and so $(\underline{c}, \underline{d})$ is an internal line by Lemma D.

The only lines in $\underline{M}(\underline{P})$ which would have different associated triangles in $\underline{T}(\underline{\tilde{M}}, \underline{P})$ than in $\underline{T}(\underline{M}, \underline{P})$ are $(\underline{a}, \underline{d})$, $(\underline{d}, \underline{b})$, $(\underline{b}, \underline{c})$, and $(\underline{c}, \underline{a})$. Since $\operatorname{sign}(\underline{a}, \underline{d}, \underline{b}) = \operatorname{sign}(\underline{a}, \underline{d}, \underline{c})$ and so on for the other lines, each triangle in $\underline{T}(\underline{M}, \underline{P})$ associated with the line is replaced by a triangle in $\underline{T}(\underline{\tilde{M}}, \underline{P})$ with the same orientation. Thus, if $\underline{M}(\underline{P})$ is a lineation, $\underline{\tilde{M}}(\underline{P}) \cong \underline{M}(\underline{P})$. It remains, then, to show that $\underline{\tilde{M}}(\underline{P})$ is a lineation. The only postulate in Definition 2 which could be violated by the T-transform acting on a lineation would be (iii), that is, the intersection property. Let $(\underline{c}, \underline{d})$ intersect some $(\underline{e}, \underline{f}) \in \underline{\tilde{M}}(\underline{P})$ at a point $\underline{x}$. Simple algebra shows that $\underline{x}$ is internal to either $(\underline{a}, \underline{b}, \underline{c})$ or $(\underline{a}, \underline{b}, \underline{d})$. But $\underline{e}$ and $\underline{f}$ must be external to those two triangles, and, thus, $(\underline{e}, \underline{f})$ must intersect some edge of the triangle to which $\underline{x}$ is internal. Then $\underline{M}(\underline{P})$ could not be a lineation, for the intersection property would be violated. So $\underline{\tilde{M}}(\underline{P})$ must be a lineation, by contradiction.

D. The Star Lemmas

Definition 10: Let $\underline{a}$ be a point in $\underline{P}$ and $\underline{M}(\underline{P})$ be a lineation of $\underline{P}$. Then the star of a with respect to M(P) is

$$(14) \qquad \underline{S}(\underline{a}, \underline{M}) = \{(\underline{a}, \underline{b}) \;\forall\; \underline{b} \ni : (\underline{a}, \underline{b}) \in \underline{M}(\underline{P})\} .$$

Definition 11: Let $(\underline{a}, \underline{b})$ and $(\underline{a}, \underline{c})$ be two lines in $\underline{M}(\underline{P})$. A third line, $(\underline{a}, \underline{d})$ is said to be between $(\underline{a}, \underline{b})$ and $(\underline{a}, \underline{c})$ iff

(a) $\qquad \mathrm{sign}(\underline{a}, \underline{b}, \underline{d}) = \mathrm{sign}(\underline{a}, \underline{b}, \underline{c})$

(15) and

(b) $\qquad \mathrm{sign}(\underline{a}, \underline{d}, \underline{c}) = \mathrm{sign}(\underline{a}, \underline{b}, \underline{c})$.

The lines $(\underline{a}, \underline{b})$ and $(\underline{a}, \underline{c})$ are said to be adjacent in $\underline{M}(\underline{P})$ iff $(\underline{a}, \underline{d})$ is between $(\underline{a}, \underline{b})$ and $(\underline{a}, \underline{c}) \Rightarrow (\underline{a}, \underline{d}) \notin \underline{S}(\underline{a}, \underline{M})$.

Lemma 1: Let $(\underline{a}, \underline{b}, \underline{c}) \in \underline{T}(\underline{M}, \underline{P})$ and $(\underline{a}, \underline{d})$ be between $(\underline{a}, \underline{b})$ and $(\underline{a}, \underline{c})$. Then $(\underline{a}, \underline{d})$ intersects $(\underline{b}, \underline{c})$, and, thus, $(\underline{a}, \underline{d})$ is not in $\underline{M}(\underline{P})$.

Proof: The proof follows from simple manipulations of (15) and equation (4) applied to the fact that $\underline{d}$ is not internal to $(\underline{a}, \underline{b}, \underline{c})$.

Lemma 2: Let $(\underline{a}, \underline{b})$ and $(\underline{a}, \underline{c})$ be two internal adjacent lines in $\underline{S}(\underline{a}, \underline{M})$. Then $(\underline{a}, \underline{b}, \underline{c}) \in \underline{T}(\underline{M}, \underline{P})$.

Proof: Since $(\underline{a}, \underline{b})$ is an internal line, $\exists\; (\underline{a}, \underline{d}) \in \underline{M}(\underline{P}) \ni$:

$(a, b, d) \in T(M, P)$ and $\ni$:

(a) $\qquad \operatorname{sign}(a, b, d) = \operatorname{sign}(a, b, c)$.

Now, either

(16) (b) $\qquad \operatorname{sign}(a, c, d) = \operatorname{sign}(a, c, b)$,

or

(c) $\qquad \operatorname{sign}(a, c, d) = \operatorname{sign}(a, b, c)$.

Equation (16b) implies that (a, d) is between (a, b) and (a, c). But those two lines are adjacent. Similarly, equation (16c) imples that (a, c) is between (a, b) and (a, d). Thus, by Lemma 1, $(a, c) \notin M(P)$ which contradicts the hypothesis.

<u>Definition 12:</u> Let $\{(a, b), (a, c)\} \subset S(a, M)$, and $\exists\, (a, d) \ni$: $\operatorname{sign}(a, b, d) \neq 0 \neq \operatorname{sign}(a, c, d)$. Then (a, d) is <u>outside</u> (a, b) and (a, c) iff it is not between (a, b) and (a, c).

<u>Definition 13:</u> We speak of the <u>interval $[(a, b), (a, c)]$</u>, where $\{(a, b), (a, c)\} \subset S(a, M)$. <u>The $\left\{\begin{matrix}\text{outside}\\ \text{inside}\end{matrix}\right\}$ of $[(a, b), (a, c)]$ is empty in $M(P)$</u> iff (a, d) is $\left\{\begin{matrix}\text{outside}\\ \text{between}\end{matrix}\right\}$ (a, b) and $(a, c) \Rightarrow (a, d) \notin M(P)$.

Definition 14: The $\left\{\begin{matrix}\text{inside}\\ \text{outside}\end{matrix}\right\}$ of $[(a, b), (a, c)]$ is <u>filled in $M(P)$</u> iff $\exists$ a sequence of $n \geq 0$ elements (a, d_1), (a, d_2), . . . (a, d_n), $\ni$:

(i) $\qquad (a, d_i) \in S(a, M)\ \forall\, i, = 1, \ldots, M$,

(ii) $\qquad (a, d_i)$ is $\left\{\begin{matrix}\text{between}\\ \text{outside}\end{matrix}\right\}$ (a, b) and $(a, c)\ \forall\, i = 1, \ldots, n$, and

(iii) if $\underline{d}_0 = \underline{b}$ and $\underline{d}_{\underline{n}+1} = \underline{c}$, then $(\underline{a}, \underline{d}_{\underline{i}}, \underline{d}_{\underline{i}+1}) \in \underline{T}(\underline{M}, \underline{P})$

$\forall\ \underline{i} = 1, \ldots, \underline{n}$.

Lemma 3: $(\underline{a}, \underline{d}_{\underline{i}})$ is internal, $\forall\ \underline{i} = 1, \ldots, \underline{n}$

Lemma 4: The $\left\{\begin{matrix}\text{outside}\\ \text{inside}\end{matrix}\right\}$ of $[(\underline{a}, \underline{b}), (\underline{a}, \underline{c})]$ filled in $\underline{M}(\underline{P})$ and $\underline{M}(\underline{P}) \cong \underline{\tilde{M}}(\underline{P})$ imply that the $\left\{\begin{matrix}\text{outside}\\ \text{inside}\end{matrix}\right\}$ $[(\underline{a}, \underline{b}), (\underline{a}, \underline{c})]$ is contained in the filled inside or outside of an interval in $\underline{\tilde{M}}(\underline{P})$.

Proof: If there are any boundary points in $\underline{S}(\underline{a}, \underline{M})$, then let $(\underline{a}, \underline{b}^*)$ be selected as follows. If $(\underline{a}, \underline{b})$ is a boundary line, then let $\underline{b}^* = \underline{b}$, if not, there is a $\underline{b}'$ such that $(\underline{a}, \underline{b}, \underline{b}') \in \underline{S}(\underline{a}, \underline{M})$ and sign $(\underline{a}, \underline{b}, \underline{b}') =$ -sign$(\underline{a}, \underline{b}, \underline{d}_1)$ where $\underline{d}_1$ is as defined in Definition 14. If $\underline{b}'$ is a boundary line, let $\underline{b}^* = \underline{b}'$. If not a $\underline{b}''$ exists, etc. Since $\underline{S}(\underline{a}, \underline{M})$ is finite we must exhaust the lines or find the first boundary line. We call this process stepping away from (a, d_1). Stepping toward $(\underline{a}, \underline{d}_1)$ is the same, except that we let sign$(\underline{a}, \underline{b}, \underline{b}') =$ sign$(\underline{a}, \underline{b}, \underline{d}_1)$.

When $(\underline{a}, \underline{b}^*)$ has been found we step back around $\underline{S}(\underline{a}, \underline{\tilde{M}})$, starting at $(\underline{a}, \underline{b}^*)$ and choose points which fill the inside or outside of an interval which contains $(\underline{a}, \underline{b}), (\underline{a}, \underline{c})$. The case that there is no boundary point in $\underline{S}(\underline{a}, \underline{M})$ is treated by simply selecting the entire star, $\underline{S}(\underline{a}, \underline{M})$ as the filled region. This is represented by the interval $[(\underline{a}, \underline{b}^*), (\underline{a}, \underline{b}^*)]$, and we may say that the outside of $[(\underline{a}, \underline{b}^*), (\underline{a}, \underline{b}^*)]$

contains $[(a, b), (a, c)]$ and is filled in $S(a, M)$. Some line (a, b^*) must exist in $S(a, M)$ by condition (iv) of Definition 2.

Lemma E: Let (a, b) be an internal line $\ni : (a, b) \in M(P)$ and $(a, b) \notin \tilde{M}(P)$, where $M(P) \cong \tilde{M}(P)$. Then $\exists\ (c, d) \ni : (c, d)$ intersects (a, b).

Proof: $(a, b) \in I(M, P)$ implies that $\exists\ d_1$ and $d_2 \ni : (a, b, d_1) \in T(M, P)$ and $(a, b, d_2) \in T(M, P)$. Now, if (a, b) is $\left\{\begin{matrix}\text{inside}\\ \text{outside}\end{matrix}\right\}$ $[(a, d_1), (a, d_2)]$, the $\left\{\begin{matrix}\text{inside}\\ \text{outside}\end{matrix}\right\}$ of $[(a, d_1), (a, d_2)]$ is filled in $M(P)$. Thus, the $\left\{\begin{matrix}\text{inside}\\ \text{outside}\end{matrix}\right\}$ of $[(a, d_1), (a, d_2)]$ is contained in the inside or outside of an interval $[(a, d_1^*), (a, d_2^*)]$ which is filled in $\tilde{M}(P)$. Since $(a, b) \in \tilde{M}(P)$ it must be between two adjacent internal lines in $S(a, M)$, and, thus, intersects a line in $\tilde{M}(P)$ by Lemma 1.

Lemma F: Let $(a, b, c) \in T(M, P)$ and $(d, e) \in \tilde{M}(P) \cong M(P)$. Let (d, e) intersect (a, b) and $d \neq c \neq e$. Then (d, e) intersects (a, c) or (b, c).

Proof: Let, without loss of generality,

(17) $$\operatorname{sign}(a, b, d) = \operatorname{sign}(a, b, c)$$

and

(18) $$\operatorname{sign}(d, e, c) = \operatorname{sign}(d, e, a) = -\operatorname{sign}(d, e, b).$$

Now, equation (17) implies that (b, d) is not inside $[(b, c), (b, a)]$.

Thus, since $(\underline{a}, \underline{b})$ is between $(\underline{e}, \underline{b})$ and $(\underline{d}, \underline{b})$, $(\underline{c}, \underline{b})$ is between $(\underline{b}, \underline{e})$ and $(\underline{b}, \underline{d})$. Since, by equation (18), $\underline{c}$ is not internal to $(\underline{d}, \underline{b}, \underline{a})$, $(\underline{b}, \underline{c})$ intersects $(\underline{d}, \underline{e})$.

Lemma G: Let $(\underline{a}, \underline{b}) \in \underline{M}(\underline{P})$, but not in $\underline{\tilde{M}}(\underline{P})$, where $\underline{M}(\underline{P}) \cong \underline{\tilde{M}}(\underline{P})$. Let $(\underline{c}, \underline{d}) \in \underline{\tilde{M}}(\underline{P})$ be such that it intersects $(\underline{a}, \underline{b})$ at a point $\underline{x}$ such that no other point of intersection of $(\underline{a}, \underline{b})$ with a line in $\underline{\tilde{M}}(\underline{P})$ is in the the open line segment with endpoints $\underline{c}$ and $\underline{x}$. Then $(\underline{a}, \underline{b}, \underline{c}) \in \underline{T}(\underline{M}, \underline{P})$.

Proof: This is a direct corollary of Lemma F, since $(\underline{a}, \underline{b}, \underline{c}^*)$ must exist, and by Lemma F, if $\underline{c} \neq \underline{c}^*$ then $(\underline{c}, \underline{d})$ must intersect $(\underline{a}, \underline{c}^*)$ or $(\underline{c}^*, \underline{b})$. Since this reasoning holds true on both sides of $\underline{x}$, and, thus, is independent of the orientation of $(\underline{c}, \underline{d})$, one intersection must be in the open line segment with endpoints $\underline{c}$ and $\underline{x}$. Then the Lemma is proved by the contradiction.

It may be, of course, that there is no line $(\underline{c}, \underline{d})$ with the required property.

E. Basic Theorem of Triangulations

Procedure: Let $\underline{M}(\underline{P})$ and $\underline{M}^*(\underline{P})$ be two distinct, congruent lineations of $\underline{P}$. Let $\underline{N}(\underline{M}, \underline{M}^*)$ be the number of intersections of elements of $\underline{M}(\underline{P})$ with elements of $\underline{M}^*(\underline{P})$. We will define a procedure which makes $\underline{M}(\underline{P})$ into a new lineation, $\underline{\tilde{M}}(\underline{P})$, $\ni$: $\underline{N}(\underline{M}, \underline{\tilde{M}}) \leqq N(M, \overline{M}) - 1$, by applying τ- transform.

$N(M, \overline{M}) \neq 0$ by Lemma E. Then we let $(a, b) \in \overline{M}(P)$ and $(a, b) \notin M(P)$. Let (c, d) be the line in $M(P)$ which intersects (a, b) closest to the point a. See Figure II-3.

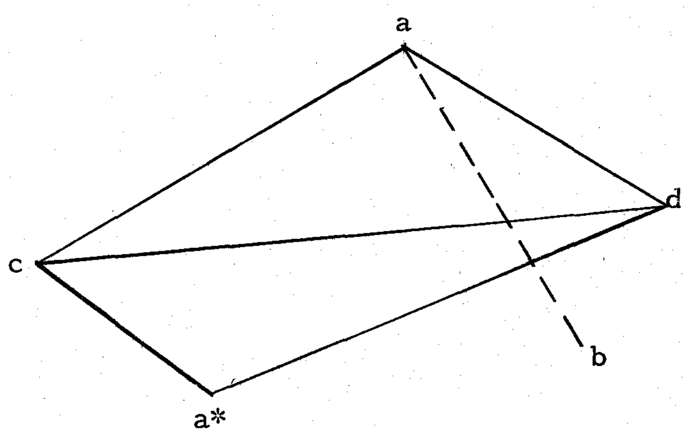


Figure II-3. Procedure if There is a Convex Quadrilateral.

By Lemma G, $(a, c, d) \in T(M, P)$, and by Lemma D, (c, d) is internal. Thus by Lemma C $\exists$ $a^* \in P \ni$; $\text{sign}(a^*, c, d) = -\text{sign}(a, c, d)$, and $(a^*, c, d) \in T(M, P)$.

We now apply the following two tests. If (a, c, a^*, d) is strictly convex, and there is no line in $\overline{M}(P)$ which intersects (a^*, d) and does not intersect (c, d), then follow Procedure A. If (a, c, a^*, d) is not strictly convex then follow Procedure B. If (a, c, a^*, d) is strictly convex, but there is a line in $\overline{M}(P)$ which intersects (a^*, d) and not (c, d) then follow Procedure C.

A. The quadrilateral is strictly convex. In this case let $\tilde{M}(P) = \mathcal{T}(c, d, M, P)$. By Theorem 1, $\tilde{M}(P)$ is a lineation which is congruent to $M(P)$. Since (c, d) is replaced by a line segment which does not intersect ($\underline{a}$, $\underline{b}$), we have reduced the total number of intersections by at least one.

B. The quadrilateral is not strictly convex. Then the situation is as represented in Figure II-4.

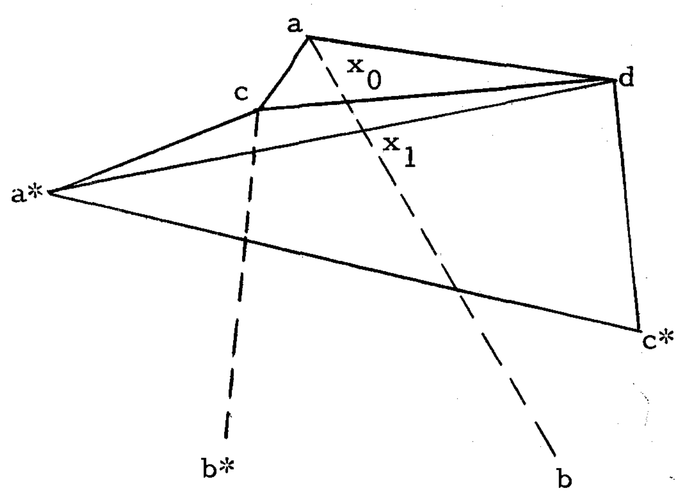


Figure II-4. Procedure When the Quadrilateral is Not Strictly Convex.

Lex $\underline{x}_0$ be the point of intersection of the line ($\underline{c}$, $\underline{d}$) with the line ($\underline{a}$, $\underline{b}$), and let $\underline{x}_1$ be the point of intersection of the line ($\underline{a}^*$, $\underline{d}$) with the line ($\underline{a}$, $\underline{b}$); $\underline{x}_1 \in$ ($\underline{a}$, $\underline{b}$) by Lemma F. The line ($\underline{a}^*$, $\underline{d}$) is internal, and so $\exists\ \underline{c}^* \ni$; ($\underline{a}^*$, $\underline{c}^*$, $\underline{d}$) $\in$ $\underline{T}(\underline{M}, \underline{P})$, and we have the quadrilateral ($\underline{c}$, $\underline{d}$, $\underline{c}^*$, $\underline{a}^*$). By Lemma 4 $\exists$ at least one line in $\underline{\tilde{M}}(\underline{P})$ which is in the outside of [($\underline{a}^*$, $\underline{c}$), ($\underline{c}$, $\underline{a}$)]. Since it cannot be inside [($\underline{a}$, $\underline{c}$), ($\underline{c}$, $\underline{d}$)] (for it would then intersect ($\underline{a}$, $\underline{b}$)), the line must be inside ($\underline{c}$, $\underline{d}$), ($\underline{c}$, $\underline{a}^*$) and, thus intersects ($\underline{a}^*$, $\underline{c}^*$); and, by Lemma F, it intersects either ($\underline{a}^*$, $\underline{c}^*$) or ($\underline{c}^*$, $\underline{d}$).

C. The quadrilateral is strictly convex, but there is at least one line in $\underline{M}(\underline{P})$,$(\underline{e}, \underline{f})$,which intersects $(\underline{a}^*, \underline{d})$ but not $(\underline{c}, \underline{d})$. (and thus either has a vertex at $\underline{c}$ or intersects $(\underline{c}, \underline{a}^*)$ It is easily seen that, though the T-transform may be applied, it would not necessarily result in a net decrease in the total number of intersections. Referring back to Figure II-3, we see that the line $(\underline{a}^*, \underline{d})$ is internal and so $\exists\ \underline{c}^* \neq \underline{c}\ \ni: (\underline{a}^*, \underline{c}^*, \underline{d}) \ni \underline{T}(\underline{M}, \underline{P})$.

Procedures C and B now become identical. We try, again, to apply the T-transform, this time on $(\underline{a}^*, \underline{c}, \underline{d}, \underline{c}^*)$, which would reduce the total intersections by eliminating the intersection of $(\underline{a}^*, \underline{d})$ with either $(\underline{c}, \underline{b}^*)$ (from Procedure B) or $(\underline{e}, \underline{f})$ (from Procedure C). If the transform cannot be made, that is, if it fails the tests described above, let $\underline{x}_2$ be the intersection point of $(\underline{a}^*, \underline{c}^*)$ with $(\underline{a}, \underline{b})$, and repeat the previous argument. The process must eventually reach a quadrilateral which is strictly convex and which has no line in M(P) which intersects side $(\underline{a}_{\underline{i}+1}, \underline{b}_{\underline{i}})$ and not $(\underline{a}_{\underline{i}+1}, \underline{d}_{\underline{i}})$ in terms of the labels in Figure II-5, below. Figure II-5 diagrams the $\underline{i}$' th step in the procedure. Each time a quadrilateral is found for which the transform may not be made, a new point $\underline{x}_{\underline{i}+1}$ is found between $\underline{x}_{\underline{i}}$ and $\underline{b}$.

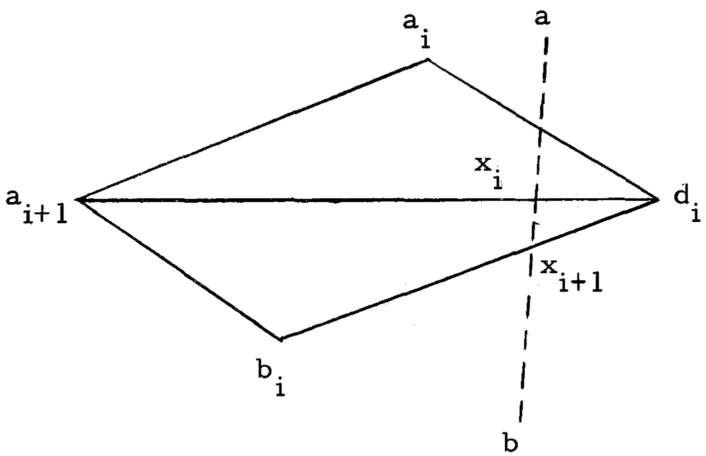


Figure II-5. General Step in Procedure.

Summary of General Procedure: Referring to Figure II-5, we say:

(a) $\underline{b}_i$ exists by the fact that $(\underline{a}_{i+1}, \underline{d}_i)$ is internal;

(b) $(\underline{a}, \underline{b})$ intersects $(\underline{b}_i, \underline{d}_i)$ or $(\underline{a}_{i+1}, \underline{b}_i)$ by Lemma F;

(c) If $(\underline{a}, \underline{b})$ intersects $(\underline{b}_i, \underline{d}_i)$, let $\underline{b}_i$ become the new $\underline{a}_{i+1}$, if not let $\underline{b}_i$ become the new $\underline{d}_i$;

(d) If the quadrilateral $(\underline{a}_i, \underline{a}_{i+1}, \underline{b}_i, \underline{d}_i)$ cannot be $\mathcal{T}$-transformed, step through the procedure again.

Theorem II: Given $\underline{M}(\underline{P}) \neq \tilde{\underline{M}}(\underline{P})$ and $\underline{M}(\underline{P}) \tilde{=} \tilde{\underline{M}}(\underline{P})$. Then $\exists$ a sequence of $\mathcal{T}$-transforms which map $\underline{M}(\underline{P})$ into $\tilde{\underline{M}}(\underline{P})$.

Proof: The proof to the theorem follows directly from the procedure described above. Since any two distinct, congruent triangulations, $\underline{M1}$ and $\underline{M2}$, have a non-zero number of intersecting internal lines, $\underline{n}$, the procedure shows that a $\mathcal{T}$-transform may always be applied to M1 in order to reduce the number of intersections by at least one. Thus, a finite number of T-transforms, each successively reducing the number of intersections by at least one, will eventually reduce the number of intersections to zero, at which time the triangulation M1 will have been transformed into triangulation M2.

Corollary II: A triangulation such that there cannot be a T-transform applied is a unique triangulation.

We need one more theorem which will be used in generating the successive convex hulls of a set of points.

Theorem III: Given two non-intersecting line segments $(\underline{a}, \underline{b})$ and $(\underline{c}, \underline{d})$, and the connecting line $(\underline{a}, \underline{c})$, then, either d is not internal to $(\underline{a}, \underline{b}, \underline{c})$ or $\underline{b}$ is not internal to $(\underline{a}, \underline{d}, \underline{c})$.

Proof: Assume that d is internal to $(\underline{a}, \underline{b}, \underline{c})$ and that $\underline{b}$ is internal to $(\underline{a}, \underline{d}, \underline{c})$.

The first condition implies that

$$\text{sign}\,(\underline{a}, \underline{b}, \underline{d}) = \text{sign}(\underline{a}, \underline{b}, \underline{c}), \tag{19}$$

and the second implies

$$\text{sign}(\underline{a}, \underline{d}, \underline{c}) = \text{sign}(\underline{a}, \underline{d}, \underline{b}). \tag{20}$$

But equations (19) and (20), together, imply that

$$\text{sign}(\underline{a}, \underline{d}, \underline{c}) = -\text{sign}(\underline{a}, \underline{b}, \underline{c}) \tag{21}$$

which violates the condition that $\underline{d}$ is internal to $(\underline{a}, \underline{b}, \underline{c})$.

III ALGORITHMS

A. Introduction

The purpose of the routines presented below is to provide some basic operations on triangulations of points in a plane. A complete program is not presented since it would not be generally useful in the computer study of triangulations. Instead a set of subroutines which provide the researcher with most of the basic operations he will need are developed below. The routines, in general, support one or both of the following two tasks. These tasks are: (1) the generation of an initial triangulation and (2) the modification of any particular triangulation by application of the $\mathcal{T}$-transform.

The initial triangulation is developed with a convex outer hull, since it is obvious that any non-convex region can be extended to a convex region by the addition of more lines. Thus the study of non-convex triangulations can be carried out in the framework of a convex triangulation.

The modification routines assume that the user has determined by some algorithm of his own whether the $\mathcal{T}$- transform should be applied to a particular line. He then can call the appropriate subroutines to carry out the change in terms of the internal representation in the machine.

The programs are coded in a version of FORTRAN IV which appears on the IBM 7094. Although this feature takes away somewhat from the generality of the routines, translation to another dialect of FORTRAN is a relatively simple matter.

Two main sets of information are kept in list form and operated on by the subroutines. They are, first, a list of all lines generated in which each line appears only once, and, second, a list of all points in the star of each point. That there are as many star lists as there are points. Data is inserted into the routines via three variables. N is the total number of points, and X(I), Y(I) are subscripted variables giving the X and Y coordinates of each point, I.

The following subroutine map shows the relation of each of the routines presented to one another. The operation of each routine is discussed in more detail in the succeeding section.

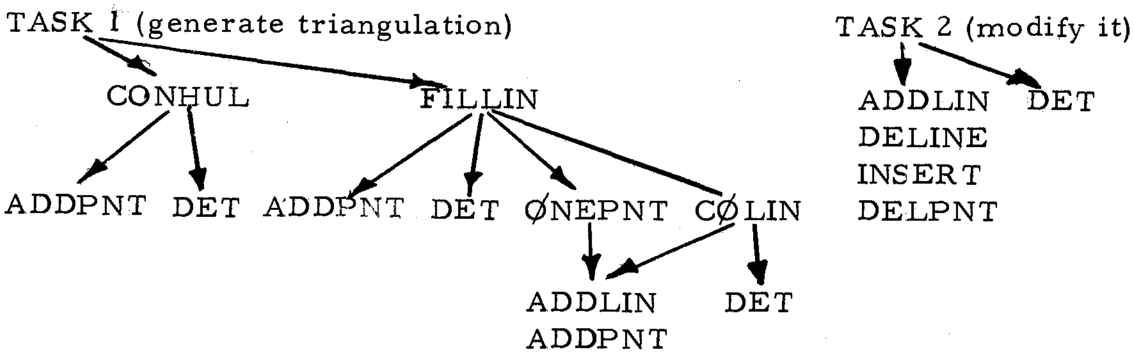


Figure III-1. Subroutine Map.

B. Description of the Routines

CØNHUL

CØNHUL generates successive convex hulls of a set of points. After the first convex hull is generated, the points used in the hull are eliminated from consideration, and the convex hull of the remaining set of points is generated. This process is repeated until one of the following three conditions are met.

a) There are no more points to consider;

b) there is one point left to consider;

c) there are several points to consider, but they are all collinear, thus forming a degenerate hull.

FILLIN

FILLIN takes the hulls generated by CØNHUL and fills them in to form a triangulation of the set of points. At first, depending upon the condition on which CØNHUL terminates, FILLIN calls ØNEPNT or CØLIN for cases (b) and (c) respectively. Case (a) is taken care of in FILLIN.

ØNEPNT

ØNEPNT connects the innermost point to all points in the innermost hull, thus forming a triangulation of the inner region.

CØLIN

CØLIN connects the inner collinear set of points to the innermost hull in such a way as to form a triangulation of the inner region.

ADDPNT, DELPNT

ADDPNT, DELPNT add and delete, respectively, points from the star of another point. The action of ADDPNT can be modified by a calling parameter which specifies where, in the star, the new point is to be connected, and, in one case, that no new point is to be connected, but that the star is to be connected together in a special way. The stars of a point is, as defined in Section II, the collection of all points which are connected to that point by lines.

ADDLIN, DELINE

ADDLIN and DELINE add and delete, respectively, lines from the line list.

DET

DET is a simple determinant function which calculates twice the signed area of the triangle determined by three ordered points. Thus DET furnishes us with a determination of the orientation of the three points. That is, if DET is positive, the three points are counterclockwise in order. IF DET is zero, they are collinear, and if DET is negative they are clockwise in order.

INSERT

INSERT, like ADDPNT, puts a point in the star of another point, but it requires two placement parameters instead of only one, as required by ADDPNT. Since we may be ignorant of the order of two points in a star and only know that they are adjacent, it may be useful to have a routine which will insert a point in the star of another point between two others. INSERT performs this function.

C. Memory Organization

The data area is viewed as one large singly subscripted array, called LIST(I). All list operations are then in terms of the index parameter of LIST. A cell is considered to be the basic unit of operation and is two or four words long, depending on the particular list. Initially the empty cell area is structured as four word cells, and as two word cells are required they are created out of the four word cells.

The last word in all cells contains the forward pointing connector, i. e. the index of the next point in the list. If POINT were the index of a cell (four word), then

NEXT=LIST(PØINT+3)

would assign to NEXT the index of the next cell. Since we will allow no non-positive indices, zero will be used to indicate the end of a list if there is one.

The first 5+N indices (where N is the number of points to be considered) of LIST are reserved for preassigned pointers to specific lists. See the Map of List Array Pointers (Figure III-7). Locations 6+N through 5+2N form a special one word cell list which is used and destroyed during the convex hull generation.

The function and structure of each list is described below.

Four Word Empty Cell List

This list is a list of all unused four word cells in memory. The first three words in the cell are zero and the last word points to the next cell. The structure is

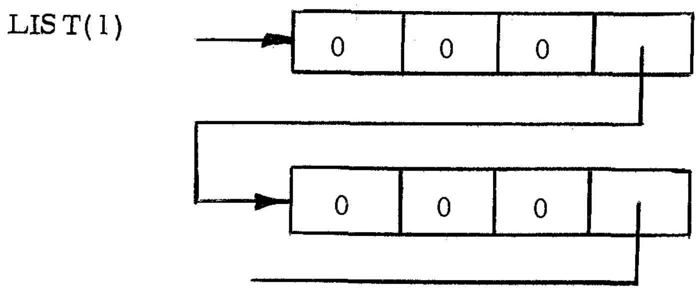


Figure III-2. Diagram of the Four Word Empty Cell List.

Two Word Empty Cell List

This list contains all unused two word cells. If the program runs out of two word cells, two more are generated, two at a time, from an empty four word cell. It is assumed, of course, that there are no memory size restrictions. Initially the two word list is empty. The structure of the list is diagrammed below. The first element in the cell contains the index of the empty cell (which is actually the address of the first word in the cell). The second word in the cell contains the index of the next cell in the empty list, or in the case of the last cell in the list, the termination code.

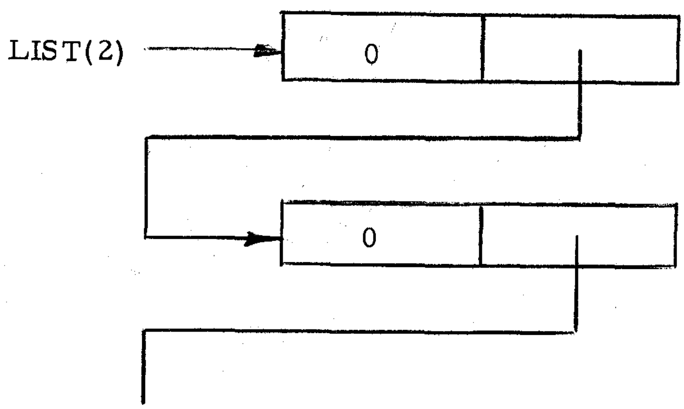


Figure III-3. Diagram of the Two Word Empty Cell List.

Line List

This list contains each line that has been generated by the programs. It uses four word cells. The first word in the cell contains the number of a point on the line, and the second word contains the number of another point on the line. The order of the points is immaterial, but there are no duplications. That is, if (K1, K2) is in the list, then (K2, K1) is not in the list. Word three is unused, but the word is open for any further programming use, as e. g. the index of the previous cell in the line list. The fourth word contains the index of the next entry in the line list. This is called the forward pointer. The line list is diagrammed below.

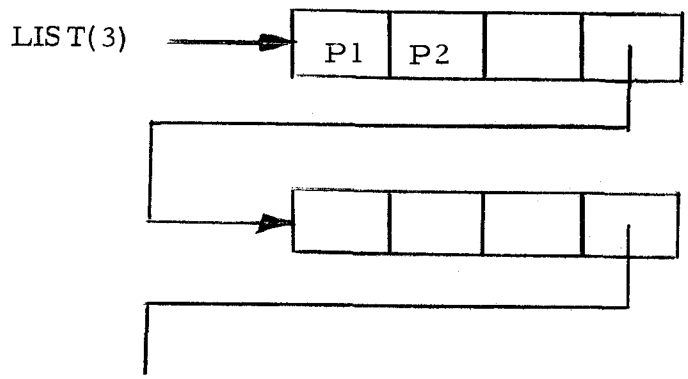


Figure III-4. Diagram of the Line List.

Hull List

The hull list is really a list of point lists. Each convex hull in the initial triangulation is represented by a point list, where the first word in a cell is the number of that particular point in the hull, and the second word contains the pointer to the next entry in the point list which contains the next point in a clockwise direction around the hull. The ordering of the points is accomplished by ordering them with respect to their X-coordinates, and an arbitrary assignment in case of a tie. Since there is no way of predicting exactly how many hulls there will be in advance, it is necessary to have a list of the hulls which is called the hull list. The first word in the cell of the hull list points to the first cell of a new hull, if there is one and otherwise contains the termination code. The second word in a cell points to the next cell in the hull list. The hulls are stored from the innermost hull to the outermost hull, and the entire hull list is destroyed during the FILLIN operation. Figure III-5 shows the organization of the hull list.

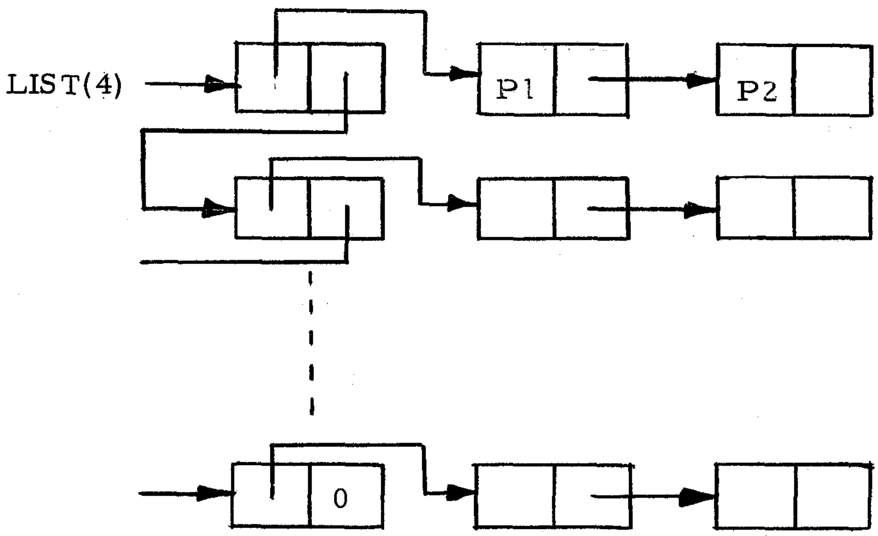


Figure III-5. Diagram of the Hull List.

Star(I)

Star(I) contains all points in the star of point I as defined in section II. That is, Star(I) is the set of all points connected to point I by a line in the triangulation. Additional information is retained by the Star(I) in the routines; that information is the order in which the points are arranged around point I. Forward, in the ordering convention, is counter-clockwise around the point I. The list Star(I) is doubly threaded. That is, each cell in the list contains the index of the previous cell in the list as well as the index of the next cell. In addition, the cell points to the address of the entry which represents in the line list that element in the star. The information is stored in the following format. The first word contains the name of the point in the star. The second word contains the index of the cell which represents in the line list the line in the star. The third word contains the index of the previous cell, that is, the next point in the star in the clockwise direction. The fourth word in the cell, as usual, points to the next cell in the list which represents the next point in the counter-clockwise direction. This arrangement is to make it convenient to find neighboring points to a line in order to determine the quadrilateral associated with the line. This operation would be necessary in the process of modifying the triangulation. The structure of the Star(I) is diagrammed in Figure III-6.

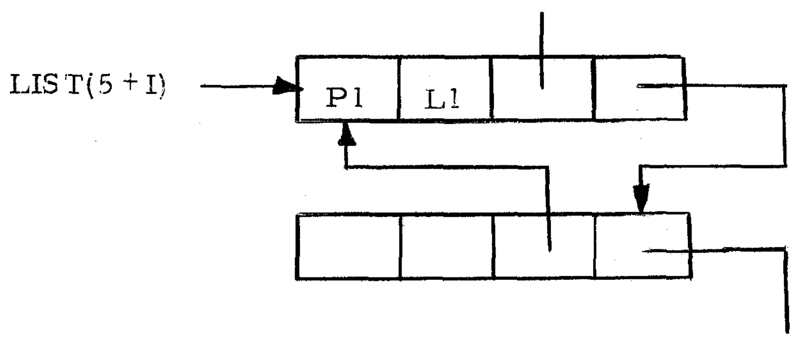


Figure III-6. Diagram of Star(I).

It should be noted that the star list is cyclically ordered at the end of FILLIN, and, hence, has no need for the termination code after that routine.

Unused List

This is a one word list set up during initialization and destroyed by the CØNHUL routine. In the case of this list, its own index is a simple function of the number of the point it represents. A cell with index 5+N+I represents point I where N is the total number of points in the set under consideration. The contents of word 5+N+I is the index of the next point in the set initially. That point would be 6+N+I. As points are added to hulls, they are eliminated from the unused list.

D. Examples

A few examples of how the information is stored in the list structure are appropriate. These will be only illustrative, since, by the very nature of list processing, it is difficult to predict exactly which locations will be used for which purpose.

Suppose that we have the set of points shown in Figure III-8(a). The first operation is to generate the successive convex hulls of the set of points. This is shown in Figure III-8(b). Figure III-9 shows how the inner hull might appear in memory.

INDEX	USE	SIZE	INITIAL
1	4 word empty cell list pointer	4	6+2N
2	2 word empty cell list pointer	2	0
3	line list pointer	4	0
4	hull list pointer	4	0
5	unused point list pointer	1	6+N
6	pointer to STAR(1)	4	0
.	.	.	.
.	.	. .	.
.	.	.	.
5+i	pointer to STAR(i)	4	0
.	.	.	.
.	.	.	.
.	.	.	.
5+N	pointer to STAR(N)	4	0
6+N	point 1		7+N
.	.		.
.	.		.
.	.		.
5+N+i	point i		6+N+i
.	.		.
.	.		.
.	.		.
5+2N	point N		0
6+2N	empty list		
	.		
	.		
	.		
	.		

Figure III-7. Map of List Array Pointers.

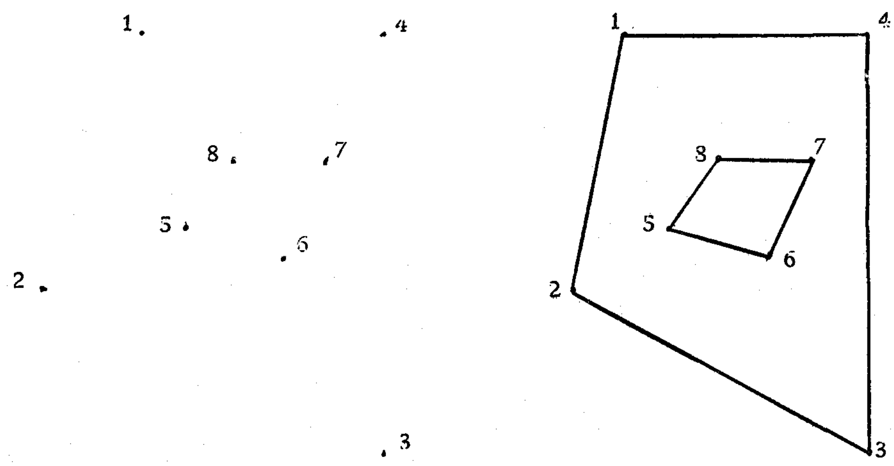


Figure III-8. A Set of Eight Points and Their Convex Hulls.

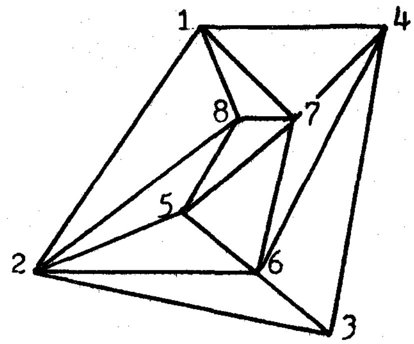
Next, FILLIN takes control and fills in the region between the hulls. This is shown in Figure III-10(a). FILLIN operates in the following manner. First the indicator NINTPT is examined to see the contents of the innermost hull. If the hull is empty, the first point in the hull is connected to all other points in the hull except for those directly adjacent to it, which are already joined with it by a line. If there is one point, or a collinear set in the innermost hull, the routines ØNEPNT and CØLIN are called, respectively, to handle the triangulation. ØNEPNT connects every point in the next hull to the inner point, and CØLIN acts similar to FILLIN when inserting lines.

FILLIN, itself, when filling in between two hulls first connects

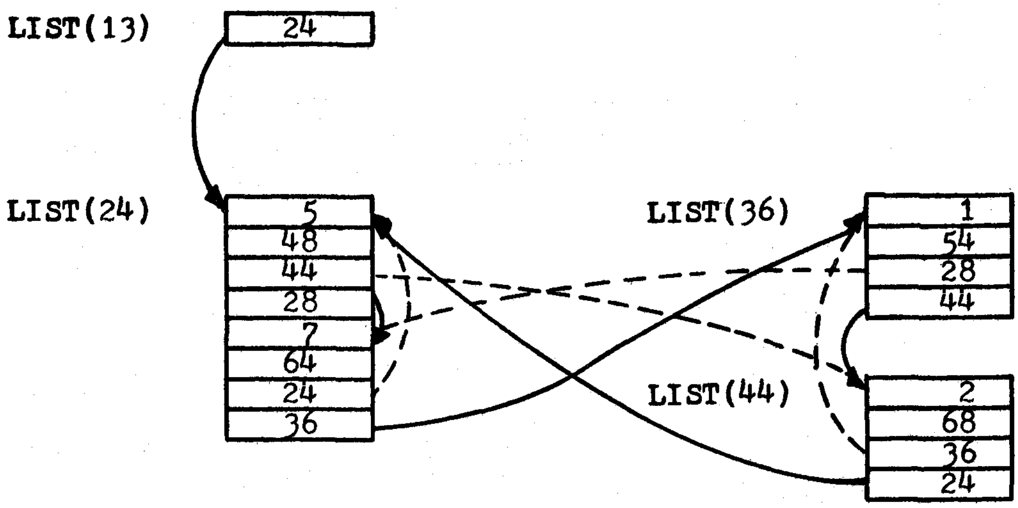
Address	Contents	Remarks
LIST(4)	22	Points to first cell in hull list
LIST(22)	24	Points to first point in inner hull
	26	Points to next cell in hull list
LIST(24)	5	First point in inner hull
	32	Points to next point in inner hull
LIST(26)	34	Points to first point in outer hull
	0	Termination code for hull list
LIST(32)	8	Next point in inner hull
	40	Points to next point in inner hull
	2	First point in outer hull
	42	Points to next point in outer hull
LIST(40)	7	
	48	
	1	
	50	
LIST(48)	6	
	0	Termination code for inner hull
	4	
	52	
	3	
	0	Termination code for outer hull

Storage Schematic of Convex Hulls

Figure III-9



(a) Triangles Generated by FILLIN



(Solid lines with arrows represent forward pointers, and dotted lines with arrows represent backward pointers.)

(b) Storage Schematic for Star(8)

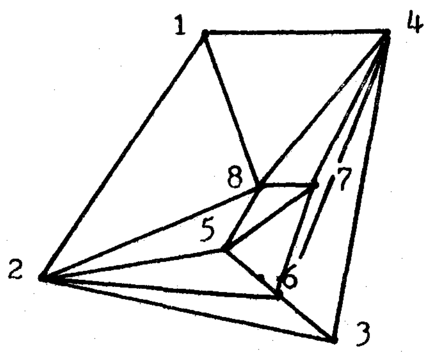
Figure III-10

the first points in each hull. The routine checks to see if the outside endpoint of the last line generated can be connected to the next point in the inner hull. If it can be connected so, it is. If not, Theorem III tells us that the next outer point in the hull may be connected to the innermost endpoint of the last generated line. The routine proceeds in this way around the hull.

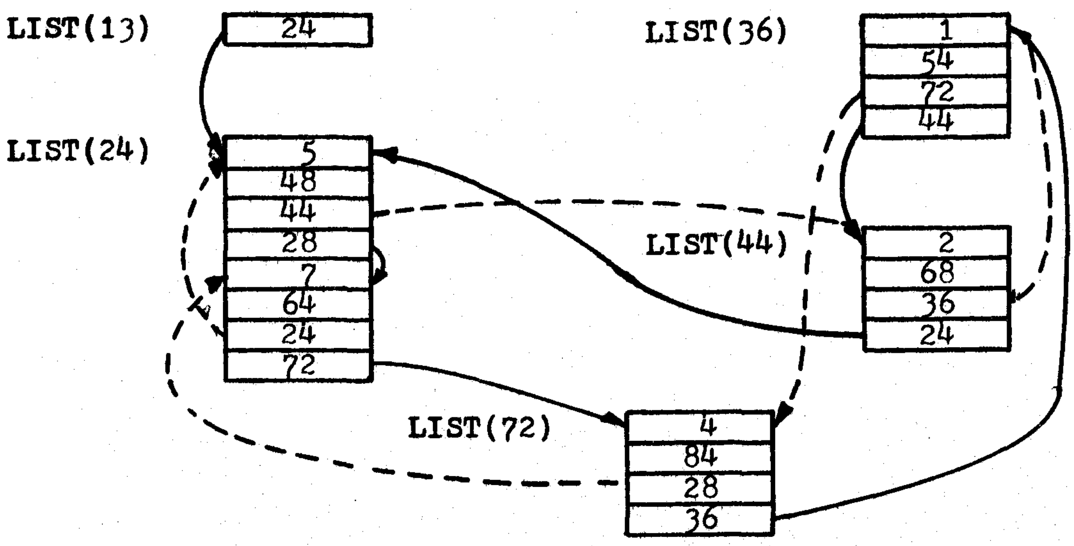
The star of each point is finished and its appearance for point 8 in memory is shown in Figure III-10(b). Notice that the list is "end-around". That is, there is no end to the list, for the last cell points forward to the first cell, and the first cell points back to the last cell.

To show how the subroutines work, we demonstrate some operations on the star of point 8. Say, for instance, that it is desired to make the $\mathcal{T}$-transform on line (1, 7). Then line (8, 4) is added to Star(8). After using either INSERT, if we don't know which order points 1 and 7 are in, or ADDPNT if we know that point 4 must follow point 7 in Star(8), the storage of Star(8) looks like the example shown in Figure III-11.

If, instead of making the transform on (1, 7), we wish to make the $\mathcal{T}$-transform on (8, 5), we must delete the line (8, 5) from Star(8). DELPNT is used for this purpose, and the result is shown in Figure III-12.



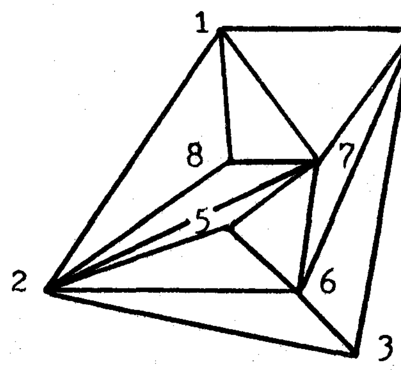
(a) Triangles After $\mathcal{T}$-Transform on (1,7)



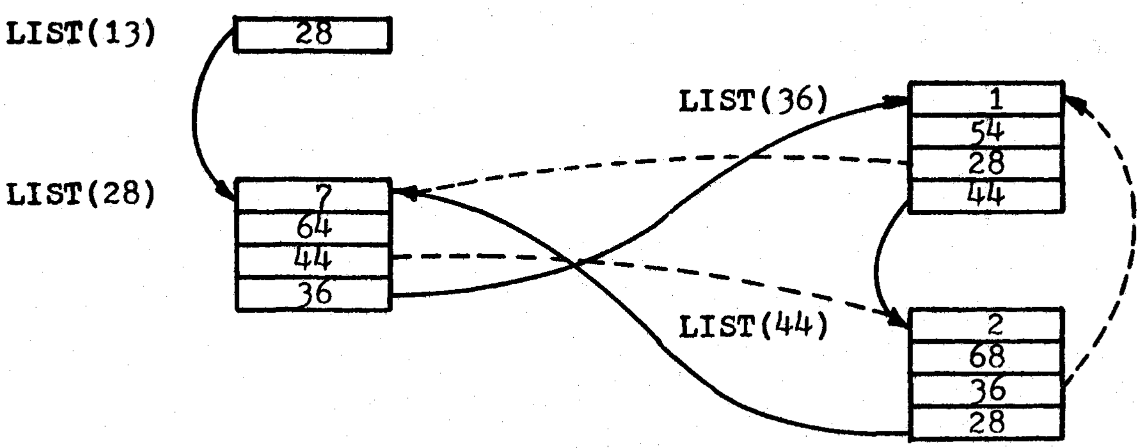
(Solid lines with arrows represent forward pointers, and dotted lines with arrows represent backward pointers.)

(b) Star(8) After $\mathcal{T}$-Transform on (1,7)

Figure III-11



(a) Triangles After $\mathcal{T}$-Transform on (8,5)



(b) Star(8) After $\mathcal{T}$-Transform on (8,5)

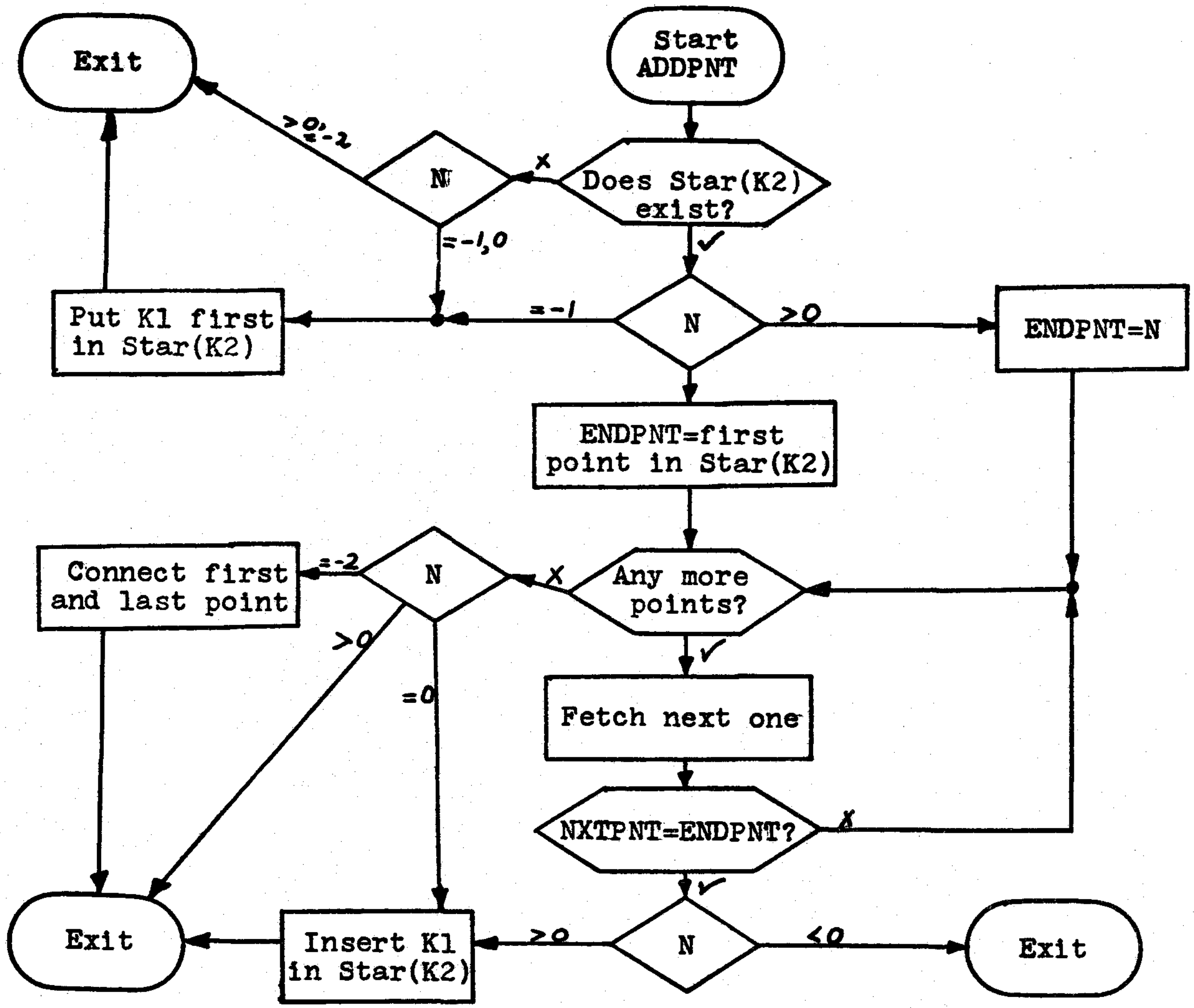
Figure III-12

E. Algorithms

Each algorithm in the succeeding pages is presented first in FORTRAN IV and second as a brief flow chart. Comments in the text of the programs should help clarify their operation.

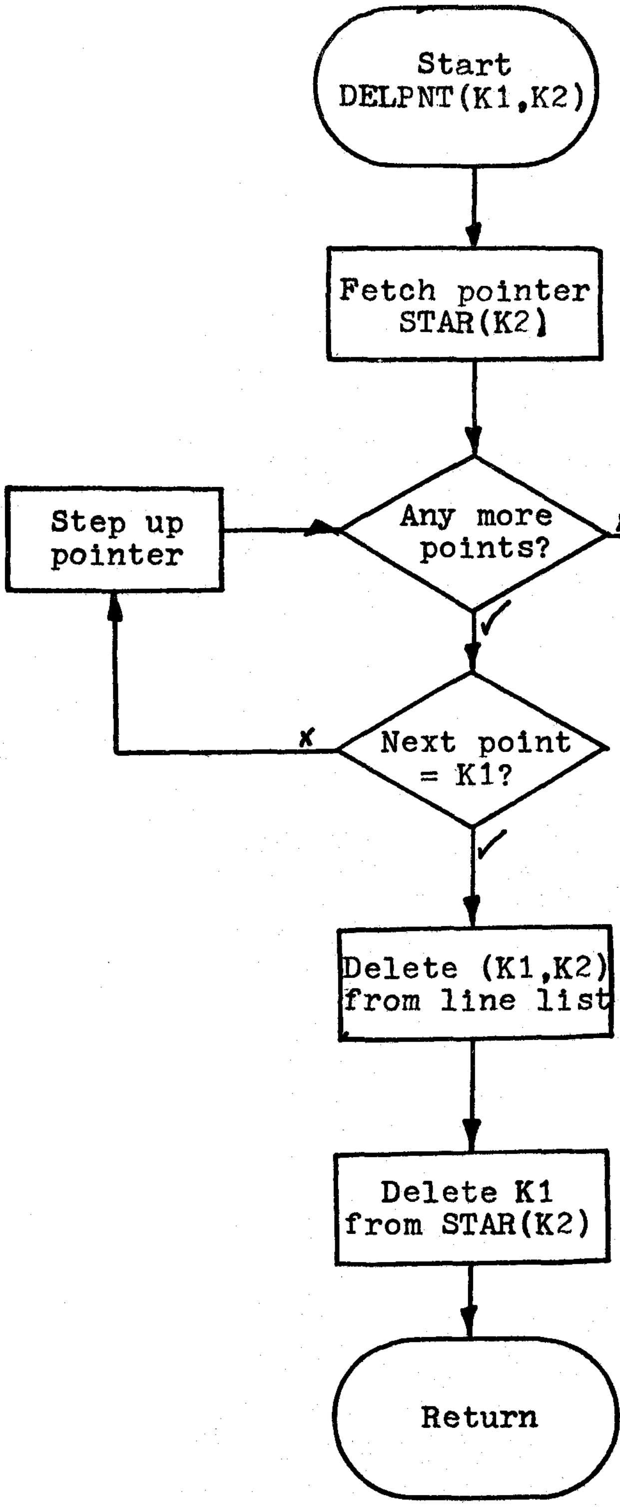
```
      SUBRØUTINE ADDPNT(K1, K2, N, L)
      CØMMØN LIST(1)
      INTEGER PØINT, ENDPNT, TEMP
C
C           THIS RØUTINE ADDS PØINT K1 TØ THE STAR(K2).  ITS
C     ØPERATIØN IS MØDIFIED BY THE PARAMETER N AS
C     FØLLØWS
C           N=-2 CØNNECT FIRST AND LAST PØINTS IN STAR(K2)
C           N=-1 K1 BECØMES THE FIRST PØINT IN STAR(K2)
C           N= 0 K1 BECØMES THE LAST PØINT IN STAR(K2)
C           N> 0 K1 FØLLØWS PØINT N IN STAR(K2)
C
C                 SEE IF STAR EXISTS AND GØ TØ 10 IF SØ
      PØINT=LIST(K2+5)
      IF (PØINT. EQ. 0) GØ TØ 10
C                 TRANSFER ØN N
      IF(N. EQ. -1) GØ TØ 15
      IF(N. GT.O) GØ TØ 20
C                 SET END POINT
      ENDPNT=LIST(PØINT)
C                 SEE IF AT LAST PØINT AND GET NEXT IF NØT
   30 IF(LIST(PØINT). NE. ENDPNT) GØ TØ 35
      IF(LIST(PØINT+3). EQ. 0) GØ TØ 40
      PØINT=LIST(PØINT+3)
      GØ TØ 30
   35 IF(N. GT. O) GØ TØ 50
  100 RETURN
C                 NØ STAR EXISTS
   10 IF(N. EQ. -2) GØ TØ 100
C                 THEN MAKE ONE
C
      IF(N. GT. O) GØ TØ 100
   15 TEMP=LIST(1)
      LIST(1)=LIST(TEMP+3)
      LIST(TEMP+3)=PØINT
      LIST(K2+5)=TEMP
      IF(PØINT. NE. O) LIST(PØINT+2)=TEMP
      LIST(TEMP)=K1
      LIST(TEMP+1)=L
      LIST(TEMP+2)=O
      GØ TØ 100
C
   20 ENDPNT=N
      GØ TØ 30
```

```
C
   40 IF(N. GT. O) GØ TØ 100
      IF(N. EQ. O) GØ TØ 50
C                         CONNECT FIRST AND LAST
      TEMP=LIST(K2+5)
      LIST(PØINT+3)=TEMP
      LIST(TEMP+2)=PØINT
      GØ TØ 100
C                         INSERT K1 IN STAR(K2)
   50 TEMP=LIST(1)
      LIST(1)=LIST(TEMP+3)
      NEXT=LIST(PØINT+3)
      LIST(TEMP+3)=NEXT
      LIST(PØINT+3)=TEMP
      LIST(TEMP+2)=PØINT
      IF(NEXT. NE. O) LIST(NEXT+2)=TEMP
      LIST(TEMP)=K1
      LIST(TEMP+1)=L
      GØ TØ 100
```



Flow Diagram for ADDPNT
Figure III-13

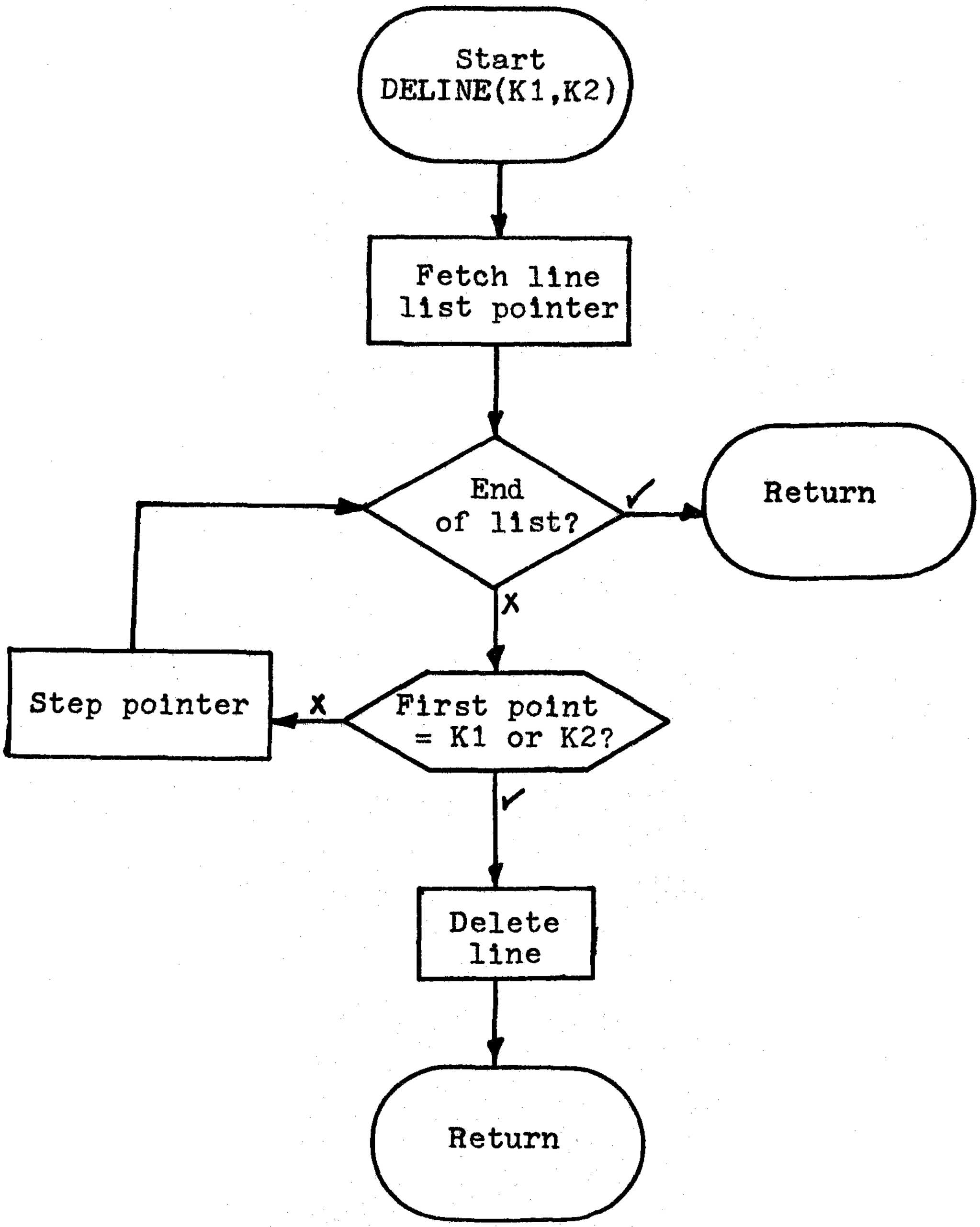
```
      SUBRØUTINE DELPNT(K1,K2)
C
      CØMMØN LIST(1)
      INTEGER PØINT,SBACK,SFØRW
C
C             THIS RØUTINE DELETES PØINT K1 FRØM STAR(K2)
      PØINT=K2+2
   10 IF(LIST(PØINT+3).EQ.O) GØ TØ 100
      PØINT=LIST(PØINT+3)
      IF(LIST(PØINT).NE.K1) GØ TØ 50
C                       DELETE FRØM STAR
   50 SBACK=LIST(PØINT+2)
      SFØRW=LIST(PØINT+3)
      IF(SBACK.NE.O) LIST(SBACK+3)=SFØRW
      IF(SFØRW.NE.O) LIST(SFØRW+2)=SBACK
      LIST(PØINT)=0
      LIST(PØINT+1)=0
      LIST(PØINT+2)=0
      LIST(PØINT+3)=LIST(1)
      LIST(1)=PØINT
  100 RETURN
```



Flow chart for DELPNT

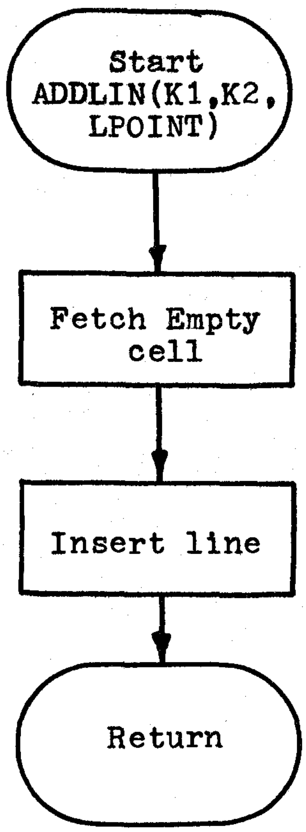
Figure III-14

```
      SUBRØUTINE DELINE(K1, K2)
C
      CØMMØN LIST(1)
      INTEGER PØINT
C           THIS RØUTINE DELETES THE LINE (K1, K2) FRØM THE
C     LIST
C
      PØINT=0
   10 PØINT=LIST(PØINT+3)
      IF(PØINT. EQ. O) GØ TØ 100
      IF(LIST(PØINT). EQ. K1) GØ TØ 20
      IF(LIST(PØINT). NE. K2) GØ TØ 10
      IF(LIST(PØINT+1). NE. K1) GØ TØ 10
      GØ TØ 30
   20 IF(LIST(PØINT+1). NE. K2) GØ TØ 10
   30 LBACK=LIST(PØINT+2)
      LFØRW=LIST(PØINT+3)
      IF(LBACK. NE. O) LIST(LBACK+3)=LFØRW
      IF(LFØRW. NE. O) LIST(LFØRW+2)=LBACK
      LIST(PØINT)=0
      LIST(PØINT+1)=0
      LIST(PØINT+2)=0
      LIST(PØINT+3)=LIST(1)
      LIST(1)=PØINT
  100 RETURN
```



Flow chart for DELINE
Figure III-15

```
      SUBRØUTINE ADDLIN(K1, K2, L)
C
C
      CØMMØN LIST(1)
C           THIS RØUTINE ADDS THE LINE (K1, K2) TØ THE LIST,
C           TRANSMITS THE PØINTER BACK THRØUGH L.
C
      L=LIST(1)
      LIST(1)=LIST(L+3)
      LNEXT=LIST(3)
      IF(LNEXT.NE.O) LIST(LNEXT+2)=L
      LIST(L+3)=LNEXT
      LIST(L+2)=0
      LIST(L+1)=K2
      LIST(L)=K1
      LIST(3)=L
  100 RETURN
```

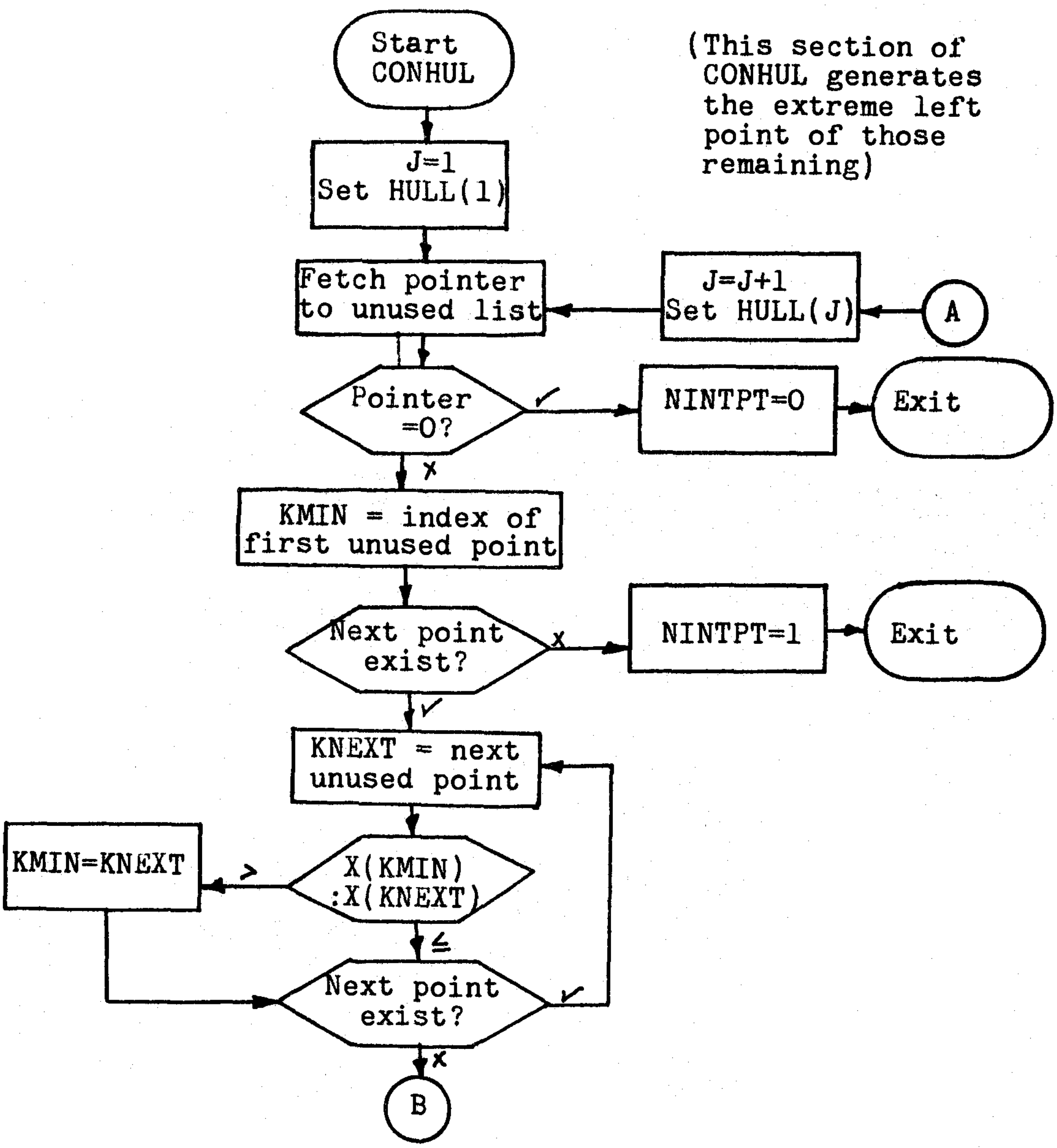


Flow chart for ADDLIN
Figure III-16

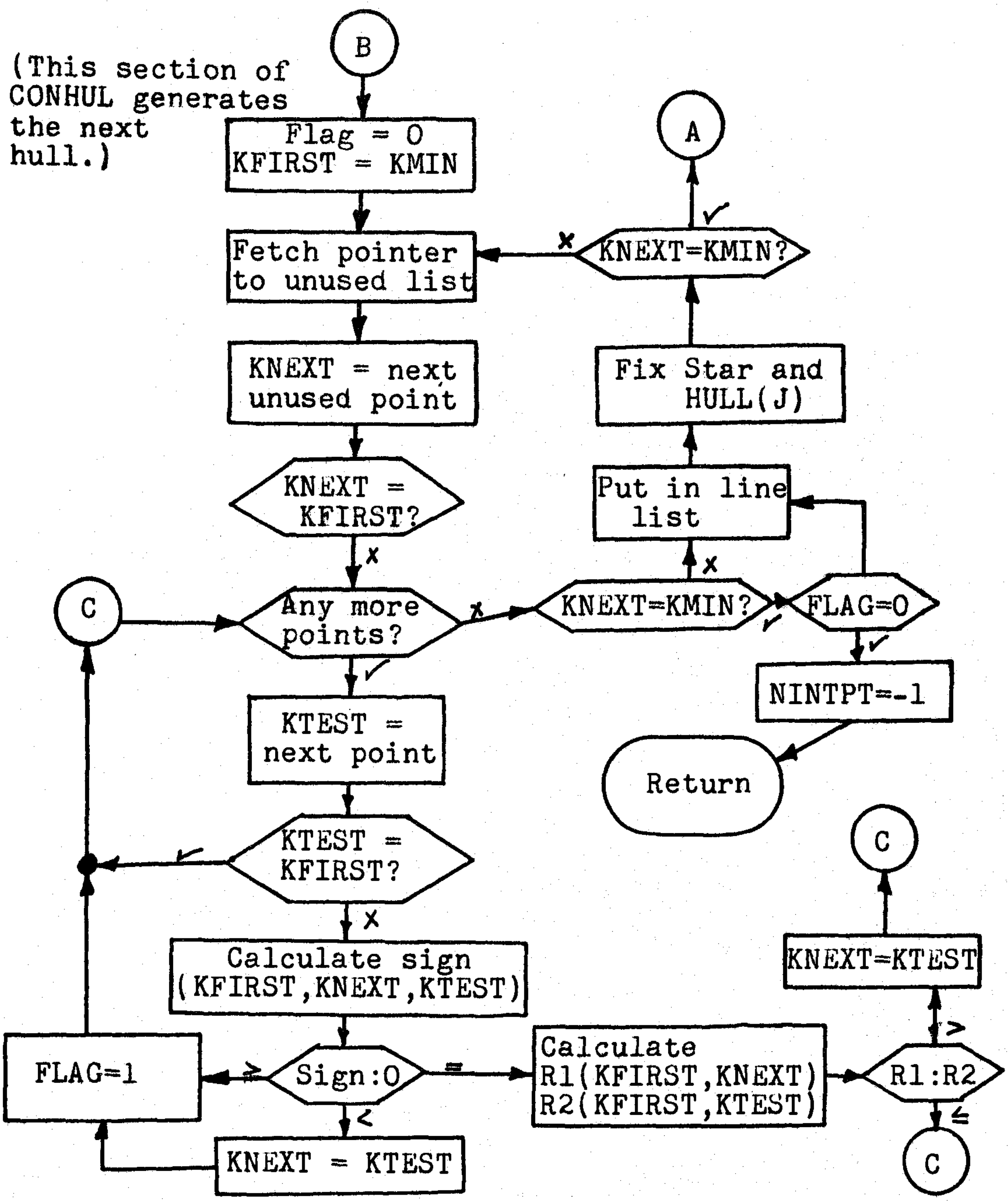
```
      SUBROUTINE CØNHUL(X, Y, N, NINTPT)
C
      CØMMØN LIST(1)
      INTEGER PØINT, TEMP, FLAG, PREVPT
      DIMENSION X(1), Y(1)
C           THIS RØUTINE GENERATES THE SUCCESSIVE CØNVEX
C           HULLS ØF AN ARRAY ØF PØINTS AND SETS THEM UP
C           IN THE LIST STRUCTURE.
C           X, Y ARE THE CØØRDINATES AND N SPECIFIES THE
C           NUMBER ØF PØINTS TØ BE CØNSIDERED
C           ØNE CARD HAS BEEN OMITTED : THAT CARD
C           SPECIFIES THE VALUE ØF SMALL: THIS PARAMETER
C           IS USED AS A TØLERANCE LEVEL WHEN CALCULATING
C           THE AREA ØF THETRIANGLES. WHEN THE DET IS
C           LESS THAN SMALL IT IS CØNSIDERED TØ BE EQUAL
C           TØ ZERØ
C
C                  THIS SECTIØN OF CØNHUL FINDS THE
C                  EXTREME LEFTHAND PØINT ØF THØSE
C                  PØINTS STILL UNUSED
C
      DØ 1000 J=1, N
C                  INITIALIZE HULL(J)
      IF(LIST(2). NE. O) GØ TØ 105
      TEMP=LIST(1)
      LIST(1)=LIST(TEMP+3)
      LIST(2)=TEMP
      LIST(TEMP+1)=TEMP+2
      LIST(TEMP+3)=0
  105 LIST(TEMP+1)=LIST(4)
      LIST(TEMP)=0
      LIST(4)=TEMP
C                  FETCH PØINTER TØ UNUSED LIST
      PØINT=LIST(5)
      IF(PØINT. EQ. O) GØ TØ 2010
      KMIN=PØINT-5-N
      PØINT=LIST(PØINT)
      IF(PØINT. EQ. O) GØ TØ 2020
C                  LET KNEXT EQUAL NEXT UNUSED NUMBER
  100 KNEXT=PØINT-5-N
      IF(X(KMIN). GT. X(KNEXT)) KMIN=KNEXT
      PØINT=LIST(PØINT)
      IF(PØINT. NE. O) GØ TØ 100
C
```

```
C                      THIS SECTIØN ØF CØNHUL GENERATES THE
C                      THE NEXT CØNVEX HULL IN THE ARRAY
C
  200 FLAG=0
C                      FLAG WILL INDICATE IF ALL PØINTS
C                      CØLINEAR
      KPREV=5
      KFIRST=KMIN
  300 PØINT=LIST(5)
  301 KNEXT=PØINT-5-N
      PREVPT=PØINT
      PØINT=LIST(PØINT)
      IF(KNEXT. EQ. KFIRST) GØ TØ 302
  350 IF(PØINT. EQ. O) GØ TØ 500
      KTEST=PØINT-5-N
      PREVPT=PØINT
      PØINT=LIST(PØINT)
      IF(KFIRST. EQ. KTEST) GØ TØ 350
C                      FIND SIGNED AREA
      AREA=DET(KFIRST, KNEXT, KTEST, X, Y)
      IF(ABS(AREA). LT. SMALL) GØ TØ 400
      IF(AREA. LT. O) GØ TØ 370
  375 FLAG=1
      GØ TØ 350
C
C                      KNEXT AND KTEST ARE CØLINEAR WITH
C                      KFIRST
  400 R1=(X(KFIRST)-X(KNEXT))**2+(Y(KFIRST)-Y(KNEXT))**2
      R2=(X(KFIRST)=X(KTEST))**2+(Y(KFIRST)-Y(KTEST))**2
      IF(R1. GT. R2) KNEXT=KTEST
      GØ TØ 350
C
C                      NØ MØRE PØINTS
  500 IF(KNEXT. NE. KFIRST) GØ TØ 550
      IF(FLAG. EQ. O) GØ TØ 2030
  550 CALL ADDLIN(KFIRST, KNEXT, LPØINT)
      CALL ADDPNT(KFIRST, KNEXT, O, LPØINT)
      CALL ADDPNT(KNEXT, KFIRST, -1, LPØINT)
C                      DELETE KNEXT FRØM UNUSED LIST
      LIST(KPREV)=PØINT
C                      PUT PØINT IN HULL(J)
      POINT=LIST(4)
      IF(LIST(2). NE. O) GØ TØ 555
      TEMP=LIST(1)
      LIST(2)=TEMP
```

```
      LIST(1)=LIST(TEMP+3)
      LIST(TEMP+1)=TEMP+2
      LIST(TEMP+3)=0
  555 TEMP=LIST(2)
      LIST(2)=LIST(TEMP+1)
      LIST(TEMP+1)=LIST(PØINT)
      LIST(PØINT)=TEMP
      LIST(TEMP)=KNEXT
      IF(KNEXT.NE.KMIN) GØ TØ 300
 1000 CØNTINUE
C                       HØUSEKEEPING ROUTINES
C
 2010 NINTPT=0
      GØ TØ 2000
 2020 NINTPT=1
      GØ TØ 2000
 2030 NINTPT=-1
 2000 RETURN
C
  370 KNEXT=KTEST
      KPREV=PREVPT
      GØ TØ 375
  302 KPREV=PREVPT
      GØ TØ 301
```



Flow chart for CONHUL - I
Figure III-17

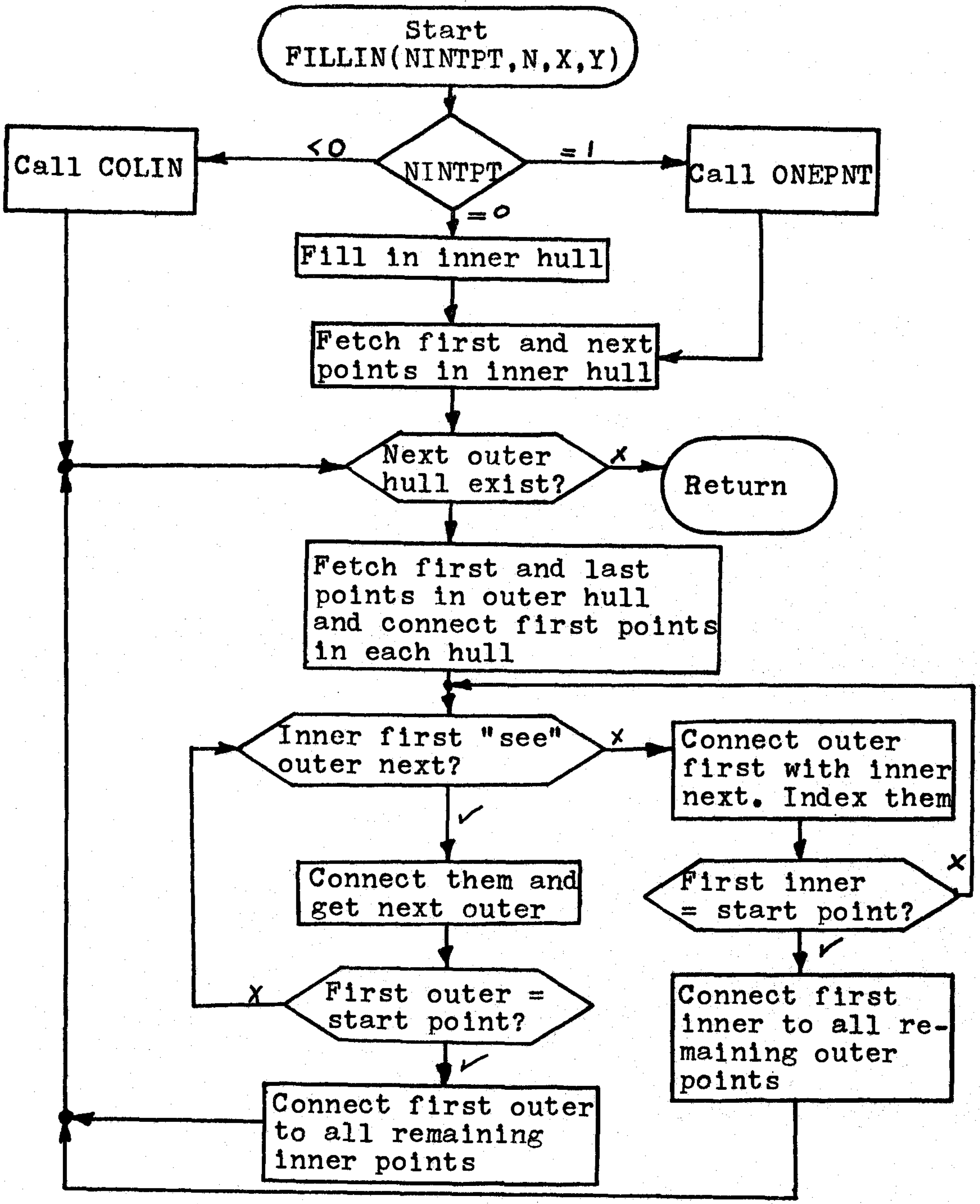


Flow chart for CONHUL - II
Figure III-18

```
      SUBRØUTINE FILLIN(NINTPT, N, X, Y)
C
      CØMMØN LIST(1)
      INTEGER HULLI, HULLØ, FIRSTI, FIRSTØ, STARTI, STARTØ,
      TEMP
C
C           THIS RØUTINE FILLS IN THE SUCCESSIVE CØNVEX
C           HULLS GENERATED BY CØNHUL:
C
      IF(NINTPT. EQ. 1) GØ TØ 10
      IF(NINTPT. LT. O) GØ TØ 15
C                   NINTPT INDICATES THE STATUS OF THE
C                   INNERMØST REGIØN ØF THE HULLS IF THE
C                   INSIDE IS EMPTY IT IS TRIANGULATED
C                   AS FØLLØWS
      HULLI=LIST(4)
      HULLI=LIST(HULLI)
      KFIRST=LIST(HULLI)
C                   CØNNECT THE STARS ØF EACH PØINT AS
C                   IT IS PRØCESSED
      CALL ADDPNT(D, KFIRST, -2, D)
      HULLI=LIST(HULLI+1)
      KPREV=LIST(HULLI)
      CALL ADDPNT(D, KPREV, -2, D)
      HULLI=LIST(HULLI+1)
      TEMP=LIST(KFIRST+5)
      TEMP=LIST(TEMP)
   20 KNEXT=LIST(HULLI)
      CALL ADDPNT(D, KNEXT, -2, D)
      IF(LIST(HULLI+1). EQ. O) GØ TØ 50
      HULLI=LIST(HULLI+1)
      CALL ADDLIN(KFIRST, KNEXT, L)
      CALL ADDPNT(KFIRST, KNEXT, KPREV, L)
      CALL ADDPNT(KNEXT, KFIRST, TEMP, L)
      KPREV=KNEXT
      GØ TØ 20
C                   READY TØ GØ WITH FILLING IN THE
C                   SUCCESSIVE HULLS
C
   50 HULLI=LIST(4)
      IF(LIST(HULLI+1). EQ. O) GØ TØ 1000
      HULLØ=LIST(HULLI+1)
      LIST(4)=HULLØ
      HULLI=LIST(HULLI)
      FIRSTI=LIST(HULLI)
```

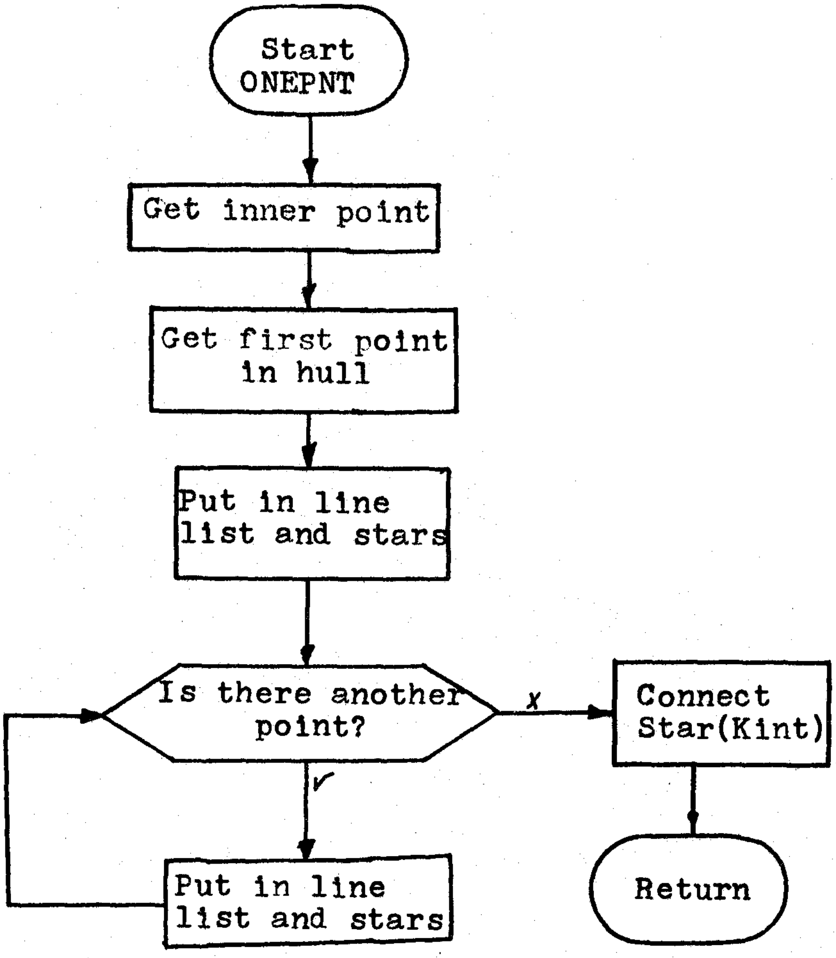
```
      STARTI=FIRSTI
      HULLI=LIST(HULLI+1)
      NEXTI=LIST(HULLI)
      HULLI=LIST(HULLI+1)
      HULLØ=LIST(HULLØ)
      FIRSTØ=LIST(HULLØ)
      STARTØ=FIRSTØ
      HULLØ=LIST(HULLØ+1)
      NEXTØ=LIST(HULLØ)
      HULLØ=LIST(HULLØ+1)
C                      THE SEARCH HAS BEEN INITIALIZED. THAT
C                      IS THE FIRST TWØ PØINTS IN THE INNER
C                      AND ØUTER HULLS RESPECTIVELY HAVE
C                      BEEN FØUND AND LABELED FIRST AND NEXT,
C                      RESPECTIVELY NØW CØNNECT THE FIRST
C                      PØINTS IN EACH HULL
C
      CALL ADDLIN(FIRSTI, FIRSTØ, L)
      CALL ADDPNT(FIRSTØ, FIRSTI, NEXTI, L)
      CALL ADDPNT(D, FIRSTØ, -2, D)
      TEMP=LIST(FIRSTØ+5)
      TEMP=LIST(TEMP)
      CALL ADDPNT(FIRSTI, FIRSTØ, TEMP, L)
C                      DOES FIRSTI SEE NEXTØ
  100 D=DET(FIRSTI, NEXTI, NEXTØ, X, Y)
      IF(D. LE. SMALL) GØ TØ 500
C                      AS IN CØNHUL SMALL HAS BEEN LEFT TØ
C                      THE USER TØ INSERT AS ANØTHER CARD TØ
C                      DEFINE A TØLERANCE FØR ZERØ
C                      FIRSTI SEES NEXTØ SØ CØNNECT THEM
      CALL ADDLIN(FIRSTI, NEXTØ, L)
      CALL ADDPNT(NEXTØ, FIRSTI, NEXTI, L)
      CALL ADDPNT(D, NEXTO, -2, D)
      CALL ADDPNT(FIRSTI, NEXTØ, FIRSTØ, L)
C                      SEE IF WE HAVE NEARLY CØMPLETED CYCLE
      IF(NEXTØ. EQ. STARTØ) GØ TØ 600
C                      IF NØT
      FIRSTØ=NEXTØ
      IF(HULLØ. EQ. O) GØ TØ 110
      NEXTØ=LIST(HULLØ)
      HULLØ=LIST(HULLØ+1)
      GO TO 100
  110 NEXTØ=STARTØ
      GØ TØ 100
C
```

```
C                    IF FIRSTI DØES NOT SEE NEXTØ THEN
C                    CØNNECT FIRSTØ WITH NEXTI
  500 CALL ADDLIN(FIRSTØ, NEXTI, L)
      CALL ADDPNT(NEXTI, FIRSTØ, FIRSTI, L)
      TEMP=LIST(NEXTI+5)
      TEMP=LIST(TEMP+2)
      TEMP=LIST(TEMP)
      CALL ADDPNT(FIRSTØ, NEXTI, TEMP, L)
C                    SEE IF NEARLY DØNE
      IF(NEXTI. EQ. STARTI) GØ TØ 700
C                    IF NØT
      FIRSTI=NEXTI
      IF(HULLI. EQ. O) GØ TØ 550
      NEXTI=LIST(HULLI)
      HULLI=LIST(HULLI+1)
      GØ TØ 100
  550 NEXTI=STARTI
      GØ TØ 100
C                    IF ØUTER RING IS FINISHED
C                    SEE IF CØMPLETELY DØNE
  600 IF(NEXTI. EQ. STARTI) GØ TØ 50
  610 CALL ADDLIN(STARTØ, NEXTI, L)
      CALL ADDPNT(NEXTI, STARTØ, FIRSTI, L)
C
      TEMP=LIST(NEXTI+5)
      TEMP=LIST(TEMP+2)
      TEMP=LIST(TEMP)
      CALL ADDPNT(STARTØ, NEXTI, TEMP, L)
      FIRSTI=NEXTI
      IF(HULLI. EQ. O) GØ TØ 50
      NEXTI=LIST(HULLI)
      HULLI=LIST(HULLI+1)
      GØ TØ 610
C                    IF INNER RING IS FINISHED
C                    SEE IF COMPLETELY DØNE FIRST
  700 IF(NEXTØ. EQ. STARTØ) GØ TØ 50
  710 CALL ADDLIN(STARTI, NEXTØ, L)
      CALL ADDPNT(NEXTØ, STARTI, STARTØ, L)
      CALL ADDPNT(STARTI, NEXTØ, FIRSTØ, L)
      FIRSTØ=NEXTØ
      IF(HULLØ. EQ. O) GØ TØ 50
      NEXTØ=LIST(HULLØ)
      HULLØ=LIST(HULLØ+1)
      GØ TØ 710
C                    GARGAGE CØLLECT
   10 CALL ØNEPNT(N)
      GØ TØ 50
   15 CALL CØLIN
      GØ TØ 50
 1000 RETURN
```



Flow chart for FILLIN
Figure III-19

```
      SUBRØUTINE ØNEPNT(N)
C
      CØMMØN LIST(1)
      INTEGER PØINT
C           THIS RØUTINE CØNNECTS THE INNER PØINT TØ ALL
C           PØINTS IN THE SURRØUNDING HULL
C
C                     GET INNER PØINT
      PØINT=LIST(5)
      LIST(5)=0
      KINT=PØINT-5-N
C                     GET FIRST PØINT IN HULL
      PØINT=LIST(4)
      PØINT=LIST(PØINT)
      KFIRST=LIST(PØINT)
C                     PUT IN THE LINE
      CALL ADDLIN(KFIRST, KINT, L)
      CALL ADDPNT(D, KFIRST, -2, D)
      CALL ADDPNT(KFIRST, KINT, -1, L)
      TEMP=LIST(KFIRST+5)
      TEMP=LIST(TEMP)
      CALL ADDPNT(KINT, KFIRST, TEMP, L)
      KPREV=KFIRST
C                     GET NEXT PØINT IN HULL
  100 IF(LIST(PØINT+1). EQ. O) GØ TØ 1000
      PØINT=LIST(PØINT+1)
      KNEXT=LIST(PØINT)
      CALL ADDLIN(KINT, KNEXT, L)
      CALL ADDPNT(D, KNEXT, -2, D)
      CALL ADDPNT(KNEXT, KINT, -1, L)
      CALL ADDPNT(KINT, KNEXT, KPREV, L)
      KPREV=KNEXT
      GØ TØ 100
C                     EXIT RØUTINE
 1000 CALL ADDPNT(D, KINT, -2, D)
      RETURN
```

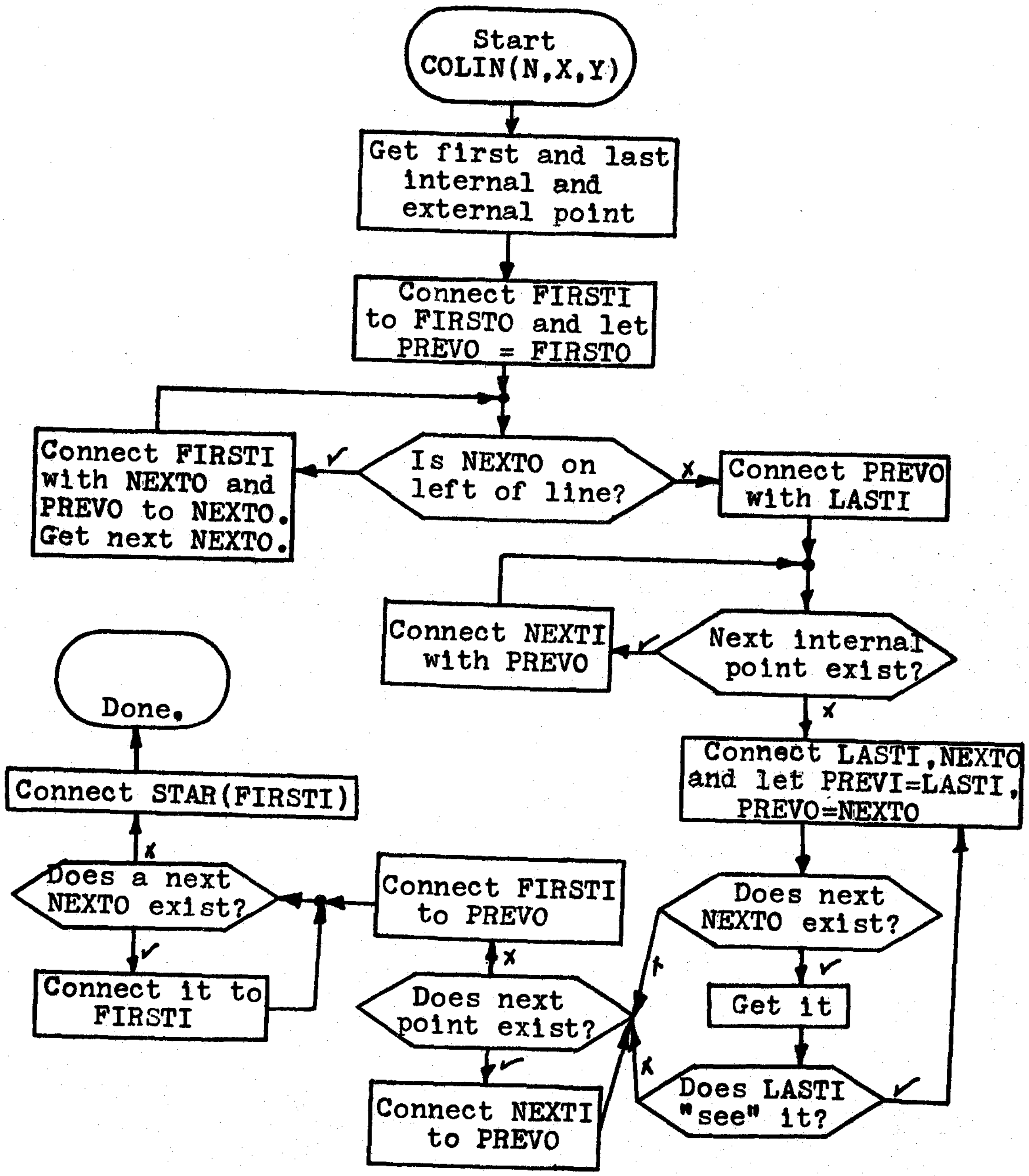


Flow chart for ONEPNT
Figure III-20

```
      SUBRØUTINE CØLIN
C
      CØMMØN LIST(1)
      INTEGER PØINT, TEMP, FIRSTI, FIRSTØ, HULLI, HULLØ,
      PREVI, PREVØ
C
C            THIS RØUTINE TRIANGULATES THE INTERIØR ØF THE
C            INNERMØST HULL WHEN IT CØNTAINS A SERIES ØF
C            CØLLINEAR PØINTS
C
C                       GET FIRST PØINT IN INNER LINE
      POINT=LIST(5)
      LIST(5)=0
      FIRSTI=PØINT-5-N
      PØINT=LIST(4)
      HULLI=LIST(PØINT)
      LASTI=LIST(HULLI)
      HULLI=LIST(HULLI+1)
      PØINT=LIST(PØINT+1)
      HULLØ=LIST(PØINT)
      FIRSTØ=LIST(HULLØ)
      HULLØ=LIST(HULLØ+1)
      NEXTØ=LIST(HULLØ)
      HULLØ=LIST(HULLØ+1)
C                       INSERT FIRSTI  FIRSTØ IN LIST
      CALL ADDLIN(FIRSTI, FIRSTØ, L)
      CALL ADDPNT(FIRSTØ, FIRSTI, O, L)
      CALL ADDPNT(D, FIRSTØ, -2, D)
      TEMP=LIST(FIRSTØ+5)
      TEMP=LIST(TEMP)
      CALL ADDPNT(FIRSTI, FIRSTØ, TEMP, L)
      PREVØ=FIRSTØ
   50 D=DET(FIRSTI, LASTI, NEXTØ, X, Y)
      IF(DET. LE. SMALL) GØ TØ 100
C                       AGAIN, SMALL IS TØ BE DEFINED AT TIME
C                       ØF USE
      CALL ADDLIN(FIRSTI, NEXTØ, L)
      CALL ADDPNT(FIRSTI, NEXTØ, PREVØ, L)
      CALL ADDPNT(NEXTØ, FIRSTI, -1, L)
      CALL ADDPNT(D, NEXTØ, -2, D)
      PREVØ=NEXTØ
      NEXTØ=LIST(HULLØ)
      HULLØ=LIST(HULLØ+1)
      GØ TØ 50
  100 CALL ADDLIN(LASTI, PREVØ, L)
```

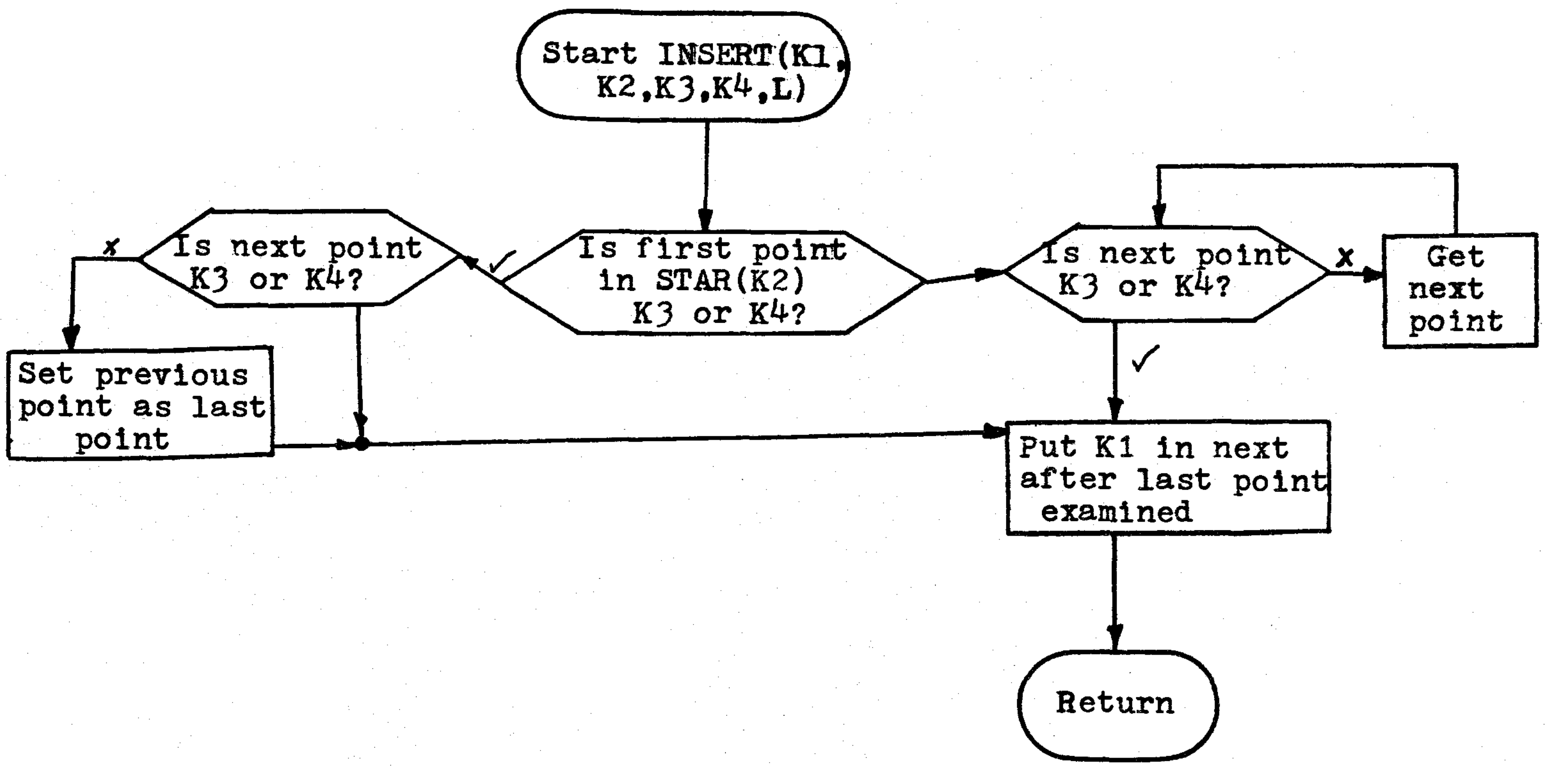
```
      CALL ADDPNT(LASTI, PREVØ, FIRSTI, L)
      CALL ADDPNT(PREVØ, LASTI, -1, L)
      PREVI=LASTI
C                       CØNNECT ALL INTERNAL PØINTS TØ PREVØ
  150 IF(HULLI. EQ. O) GØ TØ 200
      NEXTI=LIST(HULLI)
      HULLI=LIST(HULLI+1)
      CALL ADDLIN(NEXTI, PREVØ, L)
      CALL ADDPNT(NEXTI, PREVØ, FIRSTI, L)
      CALL ADDPNT(PREVØ, NEXTI, PREVI, L)
      CALL ADDPNT(D, NEXTI, -2, D)
      PREVI=NEXTI
      GØ TØ 150
C                       CØNNECT LASTI TO ALL PØINTS IT CAN SEE
  200 CALL ADDLIN(FIRSTI, PREVI, L)
      CALL ADDPNT(FIRSTI, PREVI, PREVØ, L)
      CALL ADDPNT(NEXTI, PREVI, -1, L)
  250 CALL ADDLIN(LASTI, NEXTØ, L)
      CALL ADDPNT(NEXTØ, LASTI, -1, L)
      CALL ADDPNT(LASTI, NEXTØ, PREVØ, L)
      CALL ADDPNT(D, NEXTØ, -2, D)
      PREVØ=NEXTØ
      IF(HULLØ. EQ. O) GØ TØ 300
      NEXTØ=LIST(HULLØ)
      HULLØ=LIST(HULLØ+1)
      D=DET(LASTI, FIRSTI, NEXTØ, X, Y)
      IF(D. GT. SMALL) GØ TØ 250
C                       CONNECT ALL INTERNAL PØINTS TØ THE
C                       LAST PØINT CØNNECTED TØ LASTI
  300 HULLI=LIST(4)
      LIST(4)=LIST(HULLI+1)
      HULLI=LIST(HULLI)
      HULLI=LIST(HULLI+1)
      CALL ADDPNT(D, LASTI, -2, D)
      PREVI=LASTI
  350 IF(HULLI. EQ. O) GØ TØ 400
      NEXTI=LIST(HULLI)
      HULLI=LIST(HULLI+1)
      CALL ADDLIN(NEXTI, NEXTØ, L)
      CALL ADDPNT(NEXTI, NEXTØ, PREVI, L)
      TEMP=LIST(NEXTI+5)
      TEMP=LIST(TEMP+2)
      TEMP=LIST(TEMP)
      CALL ADDPNT(NEXTØ, NEXTI, TEMP, L)
      PREVI=NEXTI
      GØ TØ 350
```

```
400 CALL ADDLIN(PREVI, FIRSTI, L)
    CALL ADDPNT(PREVI, FIRSTI, -1, L)
450 CALL ADDLIN(NEXTØ, FIRSTI, L)
    CALL ADDPNT(NEXTO, FIRSTI, -1, L)
    CALL ADDPNT(FIRSTI, NEXTØ, PREVI, L)
    IF(HULLI. EQ. O) GØ TØ 500
    PREVI=NEXTI
    NEXTI=LIST(HULLI)
    HULLI=LIST(HULLI+1)
    GØ TØ 450
500 CALL ADDPNT(D, FIRSTI, -2, D)
    RETURN
```



Flow chart for COLIN
Figure III-21

```
      SUBRØUTINE INSERT(K1, K2, K3, K4, L)
C
      CØMMØN LIST(1)
      INTEGER PØINT, TEMP
C           THIS RØUTINE INSERTS PØINT K1 IN THE STAR(K2)
C           BETWEEN K3 AND K4.  FØR SPEED VERY LITTLE
C           ERROR CØRRECTING IS DØNE.  IT IS ASSUMED ALL
C           IS PRØPER.
C
      PØINT=LIST(K2+5)
      KFIRST=LIST(PØINT)
      IF(KFIRST. EQ. K3) GØ TØ 100
      IF(KFIRST. EQ. K4) GØ TØ 100
  300 PØINT=LIST(PØINT+3)
      IF(LIST(PØINT). EQ. K3) GØ TØ 400
      IF(LIST(PØINT). EQ. K4) GØ TØ 400
      IF(LIST(POINT. EQ. KFIRST) GØ TØ 1000
      GØ TØ 300
  400 KFØRW=LIST(PØINT+3)
      TEMP=LIST(1)
      LIST(1)=LIST(TEMP+3)
      LIST(KFØRW+2)=TEMP
      LIST(TEMP+2)=PØINT
      LIST(PØINT+3)=TEMP
      LIST(TEMP+3)= KFØRW
      LIST(TEMP+1)=L
      LIST(TEMP)=K1
 1000 RETURN
  100 TEMP=LIST(PØINT+3)
      IF(LIST(TEMP). EQ. K4) GØ TØ 400
      IF(LIST(TEMP). EQ. K3) GØ TØ 400
      PØINT=LIST(PØINT+2)
      GØ TØ 400
```



Flow chart for INSERT
Figure III-22

```
      FUNCTIØN DET(I, J, K, X, Y)
C           THIS RØUTINE FINDS THE DETERMINANT ØF THE
C           AREA ØF THE TRIANGLE I, J, K WITH THE CØØRDI-
C           NATE PØINTS GIVEN BY X AND Y ARRAYS
      DIMENSIØN X(1), Y(1)
C
      ODET=(X(J)*Y(K)-X(K)*Y(J))-(X(I)*Y(K)-X(K)*Y(I))
      1        +(X(I)*Y(J)-X(J)*Y(I))
C
      RETURN
```

BIBLIOGRAPHY

1. Baston, V. G. D. Some properties of polyhedra in Euclidean n-space. New York, Pergamon Press, 1965. 212 p. (International Series of Monographs in Pure and Applied Mathematics, vol. 71.)

2. Baxter, Glen. A combinatorial lemma for complex numbers. Annals of Mathematical Statistics 32:901-904. 1961.

3. Brown, W. G. Enumeration of triangulations of the disk. Proceedings of the London Mathematical Society 3:746-768. 1964.

4. Butler, Karen J. An algorithm for determining the convex hull of n points in 3-space. Master's thesis. Corvallis, Oregon State University, 1963. 121 numb. leaves.

5. Eggleston, H. G. Problems in Euclidean space: applications of convexity. New York, Pergamon Press, 1965. 165 p. (International Series of Monographs on Pure and Applied Mathematics, vol. 5.)

6. Klein, Felix. Elementary mathematics from an advanced viewpoint - geometry. Dover, 1939. 214 p.

7. Lee, C. Y. An algorithm for path connection and its applications. I. R. E. Transactions on Electric Computers 9:346-357. 1961.

8. Luttman, F. W. and T. J. Rivlin. Some numerical experiments in the theory of polynomial interpolation. I. B. M. Journal of Research and Development 9:187-191. 1965.

9. Renyi, A. and R. Sulanke. Über die konvexe Hulle von n zufällig gewählten Punkten, I. Zeitschrift fuer Wahrscheinlichkeitstheorie and Verwandte Gebiete 2:75-84. 1963. (Abstracted in Mathematical Reviews June, 1964. p. 1185).

10. Renyi, A. and R. Sulanke. Über die konvexe Hulle von n zufällig gewählten Punkten, II. Zeitschrift fuer Wahrscheinlichkeitstheorie und Verwandte Gebiete 3:138-147. 1964. (Abstracted in Mathematical Reviews June, 1965. p. 1200).

11. Spitzer, F. and H. Widom. The circumference of a convex polygon. Proceedings of the American Mathematical Society 12:506-509. 1961.

12. Tutte, W. T. A consus of planar triangulations. Canadian Journal of Mathematics 14:21-38. 1964.