12

NSWCDD/TR-93/345

AN ALGORITHM TO FIND THE INTERSECTION OF TWO CONVEX POLYGONS

BY ARMIDO R. DIDONATO
STRATEGIC AND SPACE SYSTEMS DEPARTMENT

SEPTEMBER 1993

Approved for public release; distribution is unlimited.

NSWC
Naval Surface Warfare Center
DAHLGREN DIVISION

NAVAL SURFACE WARFARE CENTER
DAHLGREN DIVISION
Dahlgren, Virginia 22448-5000

NSWCDD/TR-93/345

AN ALGORITHM TO FIND THE INTERSECTION OF TWO CONVEX POLYGONS

BY ARMIDO R. DIDONATO
STRATEGIC AND SPACE SYSTEMS DEPARTMENT

SEPTEMBER 1993

NAVAL SURFACE WARFARE CENTER
DAHLGREN DIVISION
Dahlgren, Virginia 22448-5000

FOREWORD

The work described in this report was performed in the Space and Surface Systems Division of the Strategic and Space Systems Department at the request of the Cruise Missile Weapon Systems Division (L10) of the Strike Systems Department. A description of the analysis and software developed to find the intersection of two convex polygons is given.

The intersection of convex polygons was required by Sibille Tallant of L11 during her study to determine operating areas for strikes against multiple targets. This algorithm is used in a strike analysis tool that she developed called STANT. The author has benefited from numerous discussions with her on the application of the algorithm and is appreciative of her thorough review of this report. Mrs. Tallant also produced the drawings that appear in this document.

Approved by:

[illegible]
R. L. SCHMIDT, Head
Strategic and Space Systems Department

ABSTRACT

An algorithm is given that finds the intersection of two convex polygons. It is coded in Fortran for the IBM PC desktop computer. The program is robust and fast. It has been used successfully in targeting applications that require a rapid determination of the common intersection of more than 100 convex polygons each specified by more than 150 vertices.

CONTENTS

		Page
I.	INTRODUCTION	1
II.	SUBROUTINE XINT: MASTER ROUTINE (PART 1)	2
III.	FUNCTION ITR: ALGORITHM A	4
IV.	SUBROUTINE FST: ALGORITHM B (with SORT)	5
	SUBROUTINE SORT - CALLED BY FST	6
V.	SUBROUTINE SOLV: ALGORITHM C	6
VI.	SUBROUTINE XINT (PART 2)	7
	FUNCTION INT - CALLED BY XINT	8
	REFERENCES	9
	APPENDIX - EXAMPLES USING FINTX, BASED ON INTSEC	A-1
	DISTRIBUTION	(1)

I. INTRODUCTION

This report describes an algorithm, INTSEC, that determines the region of intersection, INTX, of two convex closed polygons. We shall refer to the entire PC Fortran code for INTSEC by the name FINTX. The code is robust; it will always find INTX, except in cases where the single precision arithmetic of the PC cannot resolve or distinguish between different points (a double precision version of the code that provides greater resolution is also available). FINTX is also fast. In a recent application, using an IBM compatible 486-66 DX2 desktop computer, FINTX found the common intersection of 172 polygons each with 180 vertices in 6.9 seconds. Examples using FINTX are given in the appendix.

INTSEC has important applications in computer graphics, computer chip design, and targeting studies. For example in targeting, INTSEC is useful in generating an operating area against several targets. Some papers studied on the intersection problem are given in references [1], [3], [4], [5], [6]. None of these papers gave sufficient detail of the actual implementation of their algorithms to evaluate their speed. For example, an algorithm may be carried out in a small number of steps but if each step is expensive time-wise, as in computing arctangents, its efficiency is reduced. No remarks were found concerning robustness of their algorithms.

The only requirement of the two given polygons, in addition to being closed and convex, is that they be positively oriented (PO). A simple polygon P is positively oriented (PO) if its vertices are ordered with the interior of P on the left as the boundary of P is traversed in the direction of increasing indices k of the vertices p_k, $k = 1, 2,$ N. If left is replaced by right then P is negatively oriented (NO). The two polygons are denoted by XY and UV, where XY is specified by its vertices z_i with coordinates (x_i, y_i), $i = 1, 2, ..., NX$. Similarly, UV is specified by its vertices w_j with coordinates (u_j, v_j), $j = 1, 2, ..., NU$. Since the polygons are closed $(x_1, y_1) = (x_{NX}, y_{NX})$ and $(u_1, v_1) = (u_{NU}, v_{NU})$. In our procedure we also include $(x_{NX+1}, y_{NX+1}) = (x_2, y_2)$ and $(u_{NU+1}, v_{NU+1}) = (u_2, v_2)$. In the remainder of this report this fact will not be referred to explicitly.

Note that INTX is also convex and is completely determined by its vertices. It can have at most KK of them with $KK \leq NX1 + NU1$, where

$$NX1 = NX - 1 \geq 3, \qquad NU1 = NU - 1 \geq 3.$$

The vertices will be ordered so that INTX is PO with the first vertex taken as the one that has the smallest ordinate. If more than one such vertex exists, then the one of that set that has the minimum abscissa is taken as the first vertex.

INTSEC is made up of three basic algorithms (A, B, C) and some auxiliary algorithms. Algorithm A establishes if a given vertex w of UV is either inside XY, outside XY, or on its boundary, ∂XY. It also determines if a given vertex z of XY is inside UV, outside UV, or on its boundary, ∂UV.

Algorithm B finds the first intersection of ∂XY and ∂UV that has not been found by A. Algorithm C finds all the remaining vertices of INTX not found by A. A vertex of INTX, which is not a vertex of either XY or UV, is found by C at the intersection of a line segment (edge) of XY and a line segment (edge) of UV, where a line segment (edge) is specified by the coordinates of its end points.

For easy reference, we list here the names of the routines associated with the algorithms mentioned above. They will be discussed in the order listed.

ROUTINE	ALGORITHM	SECTION
XINT (PART 1)	Master routine	Described in Section II
ITR	ALGORITHM A	Described in Section III
FST	ALGORITHM B	Described in Section IV
SORT	Called by FST	Described in Section IV
SOLV	ALGORITHM C	Described in Section V
XINT (PART 2)	Master routine	Described in Section VI
INT	Called by XINT	Described in Section VI

No proofs are given in this paper.

II. SUBROUTINE XINT, MASTER ROUTINE (PART 1)

The first part of the master or executive routine XINT is described here. Before proceeding, some notation is introduced. A given polygon P may be defined by the sequence of its vertices $\{p_k\}$, $k = 1,2,..., N$, where P is generated by taking the p_k in increasing order of the subscripts. The statements q is in P, q is contained in P, or $q \epsilon P$ means q is located inside P or on its boundary, ∂P. As noted earlier we refer to XY by its vertices $\{z_i\}$, $i = 1,2,..., NX$ or simply by $\{z\}$. Similarly, UV is defined by $\{w\}$, or $\{w_j\}$, $j=1,2,..., NU$. Note that XY has $NX1 = NX - 1$ distinct vertices and UV has $NU1 = NU - 1$ such points. No preliminary processing of XY or UV is necessary.

XINT begins by calling the Fortran function ITR(u, v, NX1, X, Y), which is based on a part of algorithm A, to find if the vertex $w = (u, v)$ of UV is contained in XY, where X and Y are arrays containing the x and y coordinates, respectively, of $\{z\}$. ITR(u, v, NX1, X, Y) can take one of three values that is stored in Lw. If $Lw = 1$, then w is in XY but it does not coincide with a vertex of XY. If $Lw = 0$, then w coincides with a vertex of XY. If $Lw = -1$, then w is not in XY. XINT determines Lw for each w in UV. If ITR finds NU1 vertices of UV in XY, then INTX has been found, namely INTX = UV. If this is not the case, XINT calls ITR(x, y, NU1, U, V) to find if the vertex $z = (x, y)$ of XY is contained in UV, where U and V contain the coordinates of the $\{w\}$. The result from ITR(x, y, NU1, U, V) is stored in Lz. Lz takes one of the three values (1, 0 , or -1) with meanings analogous to those for Lw. XINT determines Lz for each z in XY. If NX1 vertices of XY are found in UV by ITR, then INTX = XY. The abscissas and ordinates of the vertices belonging to INTX found by ITR are stored in the arrays WW and ZZ, respectively. In order to avoid duplications, if $Lz = 0$ then the coordinates of the vertex are not saved, since they have already been found by $Lw = 0$, $w = z$ and stored in WW and ZZ. At this point, we assume that WW and ZZ each contain K1 elements, with $0 \le K1 < NX1 + NU1$. We use WW(K) and ZZ(K) to denote the Kth element of these arrays.

XINT now calls FST, which is based on algorithm B. FST searches for the first intersection point c of ∂XY and ∂UV that is not a vertex of either of the polygons. Let z_1z_2 denote the directed line segment from end point z_1 to end point z_2. The search for c begins with directed edges z_1z_2 and w_1w_2. If their intersection produces c an exit occurs, otherwise the search continues by looking for a smallest j such that w_jw_{j+1}, $2 \le j \le NU1$, intersects z_1z_2. If no c is found, the procedure is repeated with $j = 1$ and with z_1z_2 replaced by z_iz_{i+1}, $i = 2$. The entire procedure is continued by incrementing j through its range for each $i \le NX1$ until c is found for the smallest values of j and i. If no c is found then INTX has zero area and an exit is made. If c exists for some smallest $i = n$ and $j = m$, this implies that there is an edge of XY, z_nz_{n+1}, and an edge of UV, w_mw_{m+1}, that intersect at c. Recall that c is not a vertex of XY or UV. The vertices of XY and UV are then reordered by the auxiliary routine SORT such that the sequences $\{z\}$ and $\{w\}$ are rotated as shown:

$$\{z_n, z_{n+1}, \ldots, z_{NX1}, z_1, \ldots, z_n\} \rightarrow \{z_1, z_2, \ldots, z_{NX}\}$$
$$\{w_m, w_{m+1}, \ldots, w_{NU1}, w_1, \ldots, w_m\} \rightarrow \{w_1, w_2, \ldots, w_{NU}\}.$$

More details on FST are given in Section IV.

After finding c XINT calls SOLV, which is based on algorithm C. SOLV finds the remaining intersection points that make up INTX by moving around ∂XY and ∂UV and systematically finding the missing points. The details of this search by SOLV are given in Section V. After all vertices of INTX have been found, the routine HULL, given in [2], is used to reorder the points as described in Section I and to store their coordinates in new output arrays W and Z. Then a final search is made in W and Z for any successive duplicate points; if any exist, only one set of such points is retained. The arrays W and Z hold the coordinates of the ordered vertices of INTX.

An example of how INTSEC operates is shown by Figure 1.

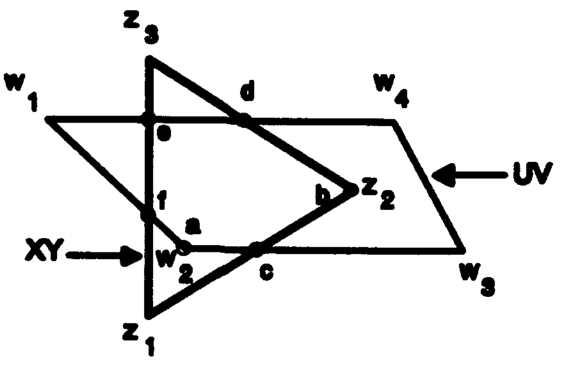


FIGURE 1. AN EXAMPLE, INTX FOUND BY INTSEC

Points a, b are found in that order by ITR. Then point c is found by FST. Note that since c is found by FST, the points of UV are reordered by SORT such that

$$w_2 \to w_1, \quad w_3 \to w_2, \quad w_4 \to w_3, \quad w_1 \to w_4, \quad w_2 \to w_5,$$

or as stated above

$$(w_2, w_3, w_4, w_1, w_2) \to (w_1, w_2, w_3, w_4, w_5).$$

Then SOLV is called to find the remaining points d, e, f of INTX. HULL is then used to reorder the points of INTX yielding INTX = {a, c, b, d, e, f, a}.

The remainder of XINT will be described in Section VI. The order with which the array elements of XY and UV are presented to SOLV for finding the intersection points of the edges of XY and UV that have not been obtained by ITR is discussed.

III. FUNCTION ITR: ALGORITHM A

Given a PO polygon P specified by its vertices $\{p_1, p_2, ..., p_j, ..., p_N\}$ and a point q, ITR determines if $q \in P$ with $q \neq p_j$ for each j, if $q = p_j$ for some j, or if q is outside P. From ITR a parameter Lq is assigned a value 1, 0, −1, accordingly. The evaluation of Lq by algorithm A depends strongly on the following:

Let Δ denote a triangle. Then $S(\Delta)$, or S for short, has the properties that $|S|$ is twice the area of Δ and can be given in terms of the Cartesian coordinates (ξ, η) of the vertices of Δ. Specifying Δ by

$$\{p_1, p_2, p_3, p_1\} \text{ with } p_j = [\xi(j), \eta(j)] \,,$$

$$S = [\xi(2) - \xi(1)]\,[\eta(3) - \eta(1)] + [\xi(3) - \xi(1)]\,[\eta(1) - \eta(2)]. \tag{1}$$

If Δ is PO then $S > 0$, and if Δ is NO then $S < 0$; $S = 0$ implies the vertices are colinear.

Again, we have for a PO polygon P and a point q, with N1 = N − 1:

a) Lq = 1. $q \in P$, but not at a vertex of P.
b) Lq = 0. $q = p_j$ for some $j = 1, 2, ..., N1$.
c) Lq = −1. $q \notin P$.

Let $q = (\bar{x}, \bar{y})$. Algorithm A, using ITR, begins by checking to see if $q = p_1$. If so, then Lq = 0 and an exit from ITR occurs. Otherwise, it keeps the line segment qp_1 fixed and proceeds counterclockwise around P looking at the sequence of triangles $\Delta_j = \{q, p_1, p_j, q\}$, $j = 2, 3, ..., N1$. Starting with j = 2, the quantity

$$S(\Delta_j) = [\xi(1) - \bar{x}]\,[\eta(j) - \bar{y}] + [\xi(j) - \bar{x}]\,[\bar{y} - \eta(1)] \tag{2}$$

is evaluated to determine the orientation of Δ_j. For simplicity in notation let $S(\Delta_j) = S_j$. If for each j $S_j > 0$, then q is not in P and Lq = −1. If there exists an integer k such that $2 \le k < N1$ and $S_k \le 0$, then an additional new triangle, $\overline{\Delta}$, is considered. It is defined by its sequence of vertices (q, p_{k-1}, p_k, q) and its orientation is determined by

$$S(\overline{\Delta}) = \overline{S}_{k-1} = [\xi(k-1) - \bar{x}]\,[\eta(k) - \bar{y}] + [\xi(k) - \bar{x}]\,[\bar{y} - \eta(k-1)]. \tag{3}$$

Using these concepts we summarise the results of the various possibilities.

If $q = p_1$, then $Lq = 0$.

If $S_j > 0$ for each j, then $Lq = -1$.

If $S_k \leq 0$ and $\overline{S}_{k-1} < 0$, then $Lq = -1$.

If $S_k < 0$ and $\overline{S}_{k-1} \geq 0$, then $Lq = 1$.

If $S_k = 0$ and $\overline{S}_{k-1} > 0$, then $Lq = 1$.

If $S_k = 0$ and $\overline{S}_{k-1} = 0$, then $Lq = 0$.

IV. SUBROUTINE FST: ALGORITHM B (with SORT)

Subroutine FST finds the first intersection c of an edge of XY with an edge of UV that is not at an end point of either edge. The procedure begins by looking for the first intersection of the ith edge $z_i z_{i+1}$, $i = 1$, of XY and at edges $w_j w_{j+1}$, for increasing j, j = 1, 2, ..., NU1 of UV. If one is not found, then i is incremented by 1 and the process is repeated. If for i = NX1 no intersection has been found, then INTX has area 0 and an exit is made from XINT. Thus, let $c = (\xi, \eta)$ denote the intersection point of the ith edge of XY with the jth edge of UV, where the end points of the ith edge have coordinates $[x(i), y(i)]$ and $[x(i+1), y(i+1)]$, and the end points of the jth edge have coordinates $[u(j), v(j)]$ and $[u(j+1), v(j+1)]$. Then the equations to be satisfied are

$$Dx\ \eta - Dy\ \xi = B$$

$$Du\ \eta - Dv\ \xi = C,$$

where

$$Dx = x(i+1) - x(i) \qquad Dy = y(i+1) - y(i)$$

$$Du = u(j+1) - u(j) \qquad Dv = v(j+1) - v(j)$$

$$B = y(i)\ x(i+1) - x(i)\ y(i+1) \qquad C = v(j)\ u(j+1) - u(j)\ v(j+1)$$

$$\xi = \frac{C\ Dx - B\ Du}{DEL} \qquad \eta = \frac{C\ Dy - B\ Dv}{DEL}$$

$$DEL = Dy\ Du - Dx\ Dv. \tag{4}$$

Now, let

$$T \equiv |Dx\,Du| + |Dy\,Dv|.$$

If $|DEL| \leq T E$, $E = \epsilon/4 = 1.25*10^{-7}$, then the two edges under consideration are numerically parallel and cannot yield c. Hence, assume that $|DEL| > T E$. We check to see if (ξ, η) is contained in the rectangle $R(\epsilon)$ specified by the inequalities

$$Xmn - \epsilon\ |Xmn| \leq \xi \leq Xmx + \epsilon\ |Xmx|$$
$$Ymn - \epsilon\ |Ymn| \leq \eta \leq Ymx + \epsilon\ |Ymx|\ ,$$

where

$$Xmn \equiv \max[\min(x(i), x(i+1)), \min(u(j), u(j+1))]$$
$$Xmx \equiv \min[\max(x(i), x(i+1)), \max(u(j), u(j+1))]$$
$$Ymn \equiv \max[\min(y(i), y(i+1)), \min(v(j), v(j+1))]$$
$$Ymx \equiv \min[\max(y(i), y(i+1)), \max(v(j), v(j+1))].$$

The value of ϵ is chosen as a small multiple of the smallest positive single precision number in the IBM PC for which $1+\epsilon > 1$. Figure 2 shows R(0).

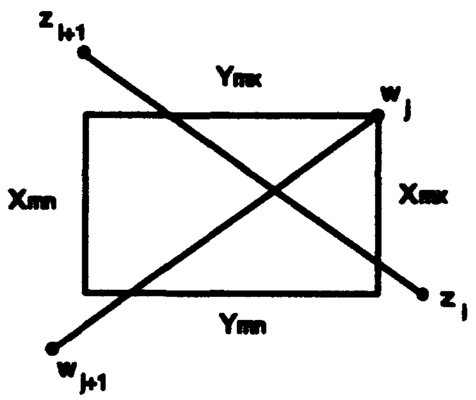


FIGURE 2. RECTANGLE R(0)

SUBROUTINE SORT – CALLED BY XNIT

If (ξ, η) is in $R(\epsilon)$ then it is accepted as the first intersection point, provided it does not coincide with an end point of either edge which can occur in spite of the fact that such points have already been found by ITR. Assuming c has been found, the polygons are reordered by the SORT routine. This routine is most easily described by simply listing its few lines of Fortran code, which is done below for the XY polygon. Suppose $z_k z_{k+1}$ denotes the XY edge of the intersection. SORT stores x(k) in x(1) as the first element of the sequence, x(k+1) in x(2), etc., as described earlier. Similarly, the y(i) are reordered in the same way. Hence if X and Y are the arrays to be reordered, then SORT requires as input NX1, X, Y, and k. The algorithm is given by

```
      SUBROUTINE SORT(NX1, X, Y, k)
      (DIMENSION STATEMENT FOR X, Y, X1, Y1)
      DO 5  M = 1, k - 1
        X1(M) = X(M)          ! X1 AND Y1 ARE TEMPORARY STORAGE ARRAYS
5       Y1(M) = Y(M)          ! WITH THE SAME DIMENSIONS AS X AND Y.
      L = 0
      DO 10  M = k, NX1
      L = L+1
        X(L) = X(M)
10      Y(L) = Y(M)
      N2 = NX1 - k + 1
      DO 15  M = 1, k - 1
        X(N2 + M) = X1(M)
15      Y(N2 + M) = Y1(M)
      X(NX1 + 1) = X(1)
      Y(NX1 + 1) = Y(1)
      X(NX1 + 2) = X(2)
      Y(NX1 + 2) = Y(2)
      END
```

V. SUBROUTINE SOLV: ALGORITHM C

After using FST, XINT calls SOLV in order to find the remaining intersection points of INTX. SOLV takes as input the coordinates of the end points of a directed edge of XY and the end points of a directed edge of UV and determines if the two lines cross. The coordinates of the intersection point are given as output and a parameter MO is assigned one of the output values: 1, 2, 3. If MO = 1, then the lines do not cross. If MO = 2, then the lines cross inside their end points and the intersection point is accepted as a new point of INTX. If MO = 3, then the two edges overlap and the crossing point coincides with at least one of the four end points that has already been found by ITR. The procedure to determine the crossing point is the same as described for FST and the same equations hold.

Of course once MO returns a value, XINT must determine how to proceed and not miss any remaining intersection points. If MO = 3, then we have found a previous intersection point, so we treat it as a crossing point, as if MO = 2, but do not store it in our INTX arrays WW and ZZ. Hence, it is sufficient to have a way of proceeding in the two different situations MO = 1 and MO = 2. The procedures are described in Section VI with the use of the function INT.

VI. SUBROUTINE XINT (PART 2)

In this section we describe how the remaining points of INTX are determined by using, in addition to MO, two parameters DEL and IDEL. DEL, as given by (4), is used whenever MO = 2. Its sign determines whether, for the next cycle, the j index associated with UV should be incremented or whether the i index associated with XY should be incremented. Also, when MO = 2 IDEL is set to DEL/|DEL|. It is used when MO = 1 and its sign determines whether j or i should be incremented for the next cycle.

We begin here with the assumption that the first intersection point c has been found by FST and the XY and UV arrays have been reordered by SORT as described above. SOLV is called with input coordinates (x_i, y_i) of vertex z_i and (u_j, v_j) of w_j starting with $i = j = 1$. Its outputs are MO = 2, the coordinates of the intersection point c, and DEL. We assume for the first intersection point c that DEL > 0. Thus the angle with vertex at c, measured in a counterclockwise direction from the line segment cw_2 to the line segment cz_2, is positive. This means that at least a part of z_1z_2 will belong to INTX, so we increment the index j. Thus in the next cycle SOLV will be called to decide if z_1z_2 and w_2w_3 have an intersection point. However before the call is made, INT is called to see if starting with z_2 a successive set of points $Z = \{z_2, z_3, ..., z_m\}$ have already been found by ITR (see below for a description of the function INT). If this is not the case then SOLV considers z_1z_2 and w_2w_3 for the next crossing. Otherwise the set Z is not empty for some m such that $2 \leq m \leq$ NX1 and i is set to m. Then SOLV considers z_mz_{m+1} and w_2w_3 for the next intersection. The parameter IDEL is set to one.

If on the other hand DEL < 0, then the roles of the XY and UV edges are interchanged and IDEL $= -1$.

The role of IDEL comes into play if the output from SOLV at some stage gives MO $= 1$, which implies that the XY and UV edges under consideration do not cross. In this case, if IDEL $= 1$ then j is incremented by one, and if IDEL $= -1$ then i is incremented by one. No consideration is given to points previously found by ITR as was done with the Z sequence when MO $= 2$.

If MO $= 3$, then SOLV has obtained a value for DEL that is numerically zero. Thus the two edges under consideration are parallel and overlap; if they do not overlap then MO $= 1$ and XINT proceeds as described in the preceding paragraph. In case of overlap, the end points of the overlap belonging to INTX have already been obtained by ITR. The indices are advanced as described for MO $= 2$.

FUNCTION INT – CALLED BY XINT

The function INT(x, y, k) is used to determine if x is one of the first k elements of the WW array and if y is one of the first k elements of the array ZZ. If both conditions are true then INT $\neq 0$; otherwise INT $= 0$. INT is called by XINT with k $=$ K1, the number of vertices of INTX found by ITR, (see page 3), i.e., it examines x and y against the elements of WW and ZZ, which are the arrays containing the coordinates of the intersection points obtained by ITR.

REFERENCES

1. Chin, F., Sampson, J., Wang, C. A., *Unifying Approach for a Class of Problems in the Computational Geometry of Polygons,* Visual Compt. (1), #2, Oct 1985, pp. 124-132.
2. Morris, A., *NSWC Library of Mathematics Subroutines,* Report NSWCDD/TR-92/425, Naval Surface Warfare Center, Dahlgren, VA. 22448-5000, Jan 1993.
3. Patnaik, L.M., Sheney, R.S., Drishnan, D., *Set Theoretic Operations on Polygons Using the Scan-Grid Approach,* Computer Aided Design (18), #5, Jun 1986, pp. 275-279.
4. Shamos, M. I., Hoey, D., *Geometric Intersection Problems,* Seventeeenth Annual IEEE Symposium on Foundations of Computer Science, Oct 1976, pp. 208-215.
5. Toussaint, G. T., *Simple Linear Algorithm for Intersecting Convex Polygons,* Visual Compt. (1), #2, Oct 1985, pp. 1188-1223.
6. Widmayer, P., Wu, Y.F., Schlag, M.D.F., Wong, C.K., *On Some Union and Intersection Problems for Polygons with Fixed Orientations,* Computing (36), #3, 1986, pp 183-197.

APPENDIX

EXAMPLES USING FINTX, BASED ON INTSEC

EXAMPLES USING FINTX, BASED ON INTSEC

Here we give some examples of convex polygons XY and UV for which the intersection INTX is determined using the Lahey Fortran code, FINTX, based on the INTSEC algorithm. A figure is given for each example showing the geometry of XY and UV. The format of the examples is explained below.

I,X,Y specify the ith vertex and its (x,y) coordinates. In Example 2, the first line refers to $x(1) = 8.0$, $y(1) = 0.0$; J,U,V specify the jth vertex and its (u,v) coordinates. In Example 2, the sixth line refers to $u(2) = 4.0$, $v(2) = 2.0$.

The next group of data is the result of using ITR. It specifies the vertices of UV contained in XY followed by the vertices of XY contained in UV. In Example 2, the nineth line indicates that [u(2), v(2)] or w_2 is contained in XY and is stored as the first element in WW and ZZ. Note LW = 1. Line 10 indicates that the first vertex of XY, z_1, is contained in UV and its coordinates are stored in the second element of the arrays WW and ZZ. Note LZ = 1.

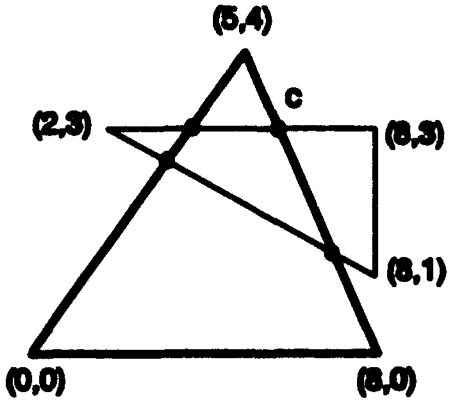
FST refers to the algorithm that determines the first intersection point of INTX not found by ITR. In Example 2, P and Q refer to the x and y coordinates of this intersection point, which is noted on the figure by c, i.e., $P = x = 6.50$, $Q = y = 2.0$.

Next MO, I, J, P, Q, DEL are given. They refer to input to and output from SOLV. On line 14 of Example 2, the input is x(3), y(3), u(2), v(2), the output is MO = 2, the coordinates of the intersection point $x = P = 6.00$ and $y = Q = 0.0$, and DEL = 32. Hence for Example 2, SOLV yields the result that the line segment z_3z_4 of XY and w_2w_3 of UV have an intersection point $x = 6.00$, $y = 0.0$. The fact that DEL > 0 means that J will be increased by one for the next cycle that can be seen in line 15. But I is also incremented by one, since z_4 is an element found by ITR at $x(4) = x(1) = 8.00$, $y(4) = y(1) = 0.0$.

Finally, INTX is given in terms of the coordinates of its vertices that are stored in the arrays W and Z. For Example 2, there are four distinct points (vertices of INTX), the first of which is stored in $W(1) = 6.00$, $Z(1) = 0.0$, as shown on line 17.

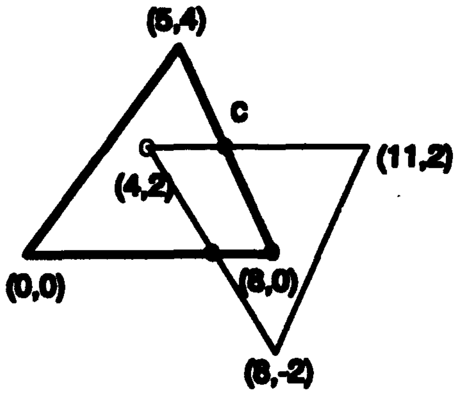
EXAMPLE 1

```
I,X,Y  1   8.000000   0.000000
I,X,Y  2   5.000000   4.000000
I,X,Y  3   0.000000   0.000000
I,X,Y  4   8.000000   0.000000

J,U,V  1   8.000000   3.000000
J,U,V  2   2.000000.  3.000000
J,U,V  3   8.000000   1.000000
J,U,V  4   8.000000   3.000000

FST,P,Q   5.750000   3.000000

MO,I,J,P,Q,DEL  2  1  1   5.750000   3.000000 -24.0
MO,I,J,P,Q,DEL  2  2  1   3.750000   3.000000  24.0
MO,I,J,P,Q,DEL  2  2  2   3.235294   2.588235 -34.0
MO,I,J,P,Q,DEL  1  3  2
MO,I,J,P,Q,DEL  2  4  2   7.000000   1.333333  18.0
MO,I,J,P,Q,DEL  1  4  3
MO,I,J,P,Q,DEL  2  4  4   5.750000   3.000000 -24.0

JJ,W,Z  1   7.000000   1.333333
JJ,W,Z  2   5.750000   3.000000
JJ,W,Z  3   3.750000   3.000000
JJ,W,Z  4   3.235294   2.588235
```



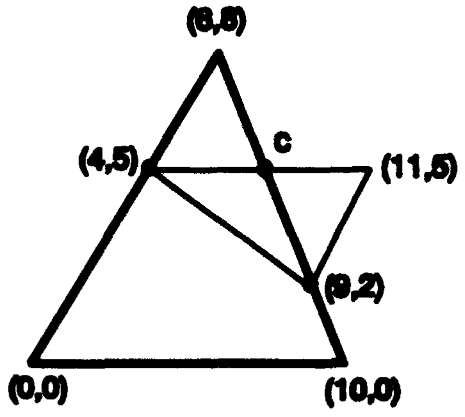
EXAMPLE 2

```
I,X,Y  1   8.000000   0.000000
I,X,Y  2   5.000000   4.000000
I,X,Y  3   0.000000   0.000000
I,X,Y  4   8.000000   0.000000

J,U,V  1  11.000000   2.000000
J,U,·  2   4.000000   2.000000
J,U,V  3   8.000000  -2.000000
J,U,V  4  11.000000   2.000000

J,K,WW,ZZ,LW   2  1   4.000000   2.000000  1
I,K,WW,ZZ,LZ   1  2   8.000000   0.000000  1

FST,P,Q   6.500000   2.000000

MO,I,J,P,Q,DEL  2  1  1   6.500000   2.000000 -28.0
MO,I,J,P,Q,DEL  1  2  2
MO,I,J,P,Q,DEL  2  3  2   6.000000   0.000000  32.0
MO,I,J,P,Q,DEL  1  4  3
MO,I,J,P,Q,DEL  2  4  4   6.500000   2.000000 -28.0

JJ,W,Z  1   6.000000   0.000000
JJ,W,Z  2   8.000000   0.000000
JJ,W,Z  3   6.500000   2.000000
JJ,W,Z  4   4.000000   2.000000
```



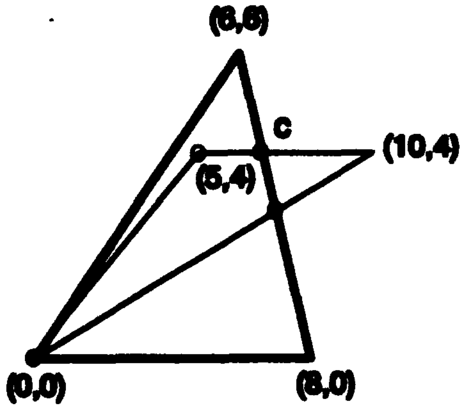
EXAMPLE 3

```
I,X,Y   1  10.000000   0.000000
I,X,Y   2   6.000000   8.000000
I,X,Y   3   0.000000   0.000000
I,X,Y   4  10.000000   0.000000

J,U,V   1  11.000000   5.000000
J,U,V   2   4.000000   5.000000
J,U,V   3   9.000000   2.000000
J,U,V   4  11.000000   5.000000

J,K,WW,ZZ,LW   2   1   4.000000   5.000000  1

J,K,WW,ZZ,LW   3   2   9.000000   2.000000  1

FST,P,Q   7.500000   5.000000

MO,I,J,P,Q,DEL   2   1   1   7.500000   5.000000 -56.0
MO,I,J,P,Q,DEL   1   2   3
MO,I,J,P,Q,DEL   1   3   3
MO,I,J,P,Q,DEL   2   4   3   9.000000   2.000000  28.0
MO,I,J,P,Q,DEL   2   4   4   7.500000   5.000000 -56.0

JJ,W,Z   1   9.000000   2.000000
JJ,W,Z   2   7.500000   5.000000
JJ,W,Z   3   4.000000   5.000000
```



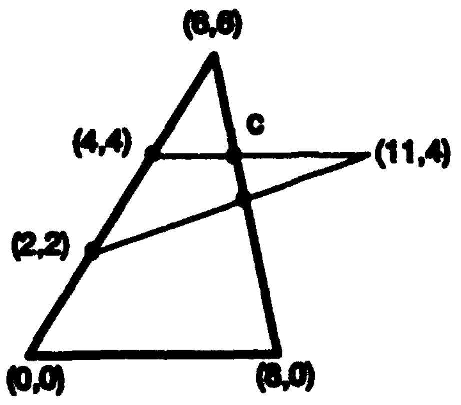
EXAMPLE 4

```
I,X,Y   1   8.000000   0.000000
I,X,Y   2   6.000000   6.000000
I,X,Y   3   0.000000   0.000000
I,X,Y   4   8.000000   0.000000

J,U,V   1  10.000000   4.000000
J,U,V   2   5.000000   4.000000
J,U,V   3   0.000000   0.000000
J,U,V   4  10.000000   4.000000

J,K,WW,ZZ,LW   2   1   5.000000   4.000000  1

J,K,WW,ZZ,LW   3   2   0.000000   0.000000  0

FST,P,Q   6.666667   4.000000

MO,I,J,P,Q,DEL   2   1   1   6.666667   4.000000 -30.0
MO,I,J,P,Q,DEL   2   2   3   0.000000   0.000000 -36.0
MO,I,J,P,Q,DEL   2   3   3   0.000000   0.000000 -32.0
MO,I,J,P,Q,DEL   2   4   3   7.058824   2.823529  68.0
MO,I,J,P,Q,DEL   2   4   4   6.666667   4.000000 -30.0
JJ,W,Z   1   0.000000   0.000000
JJ,W,Z   2   7.058824   2.823529
JJ,W,Z   3   6.666667   4.000000
JJ,W,Z   4   5.000000   4.000000
```



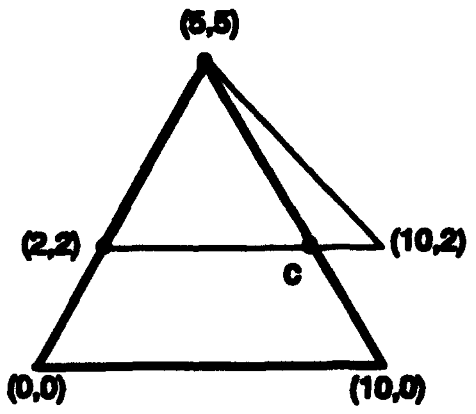
EXAMPLE 5

```
I,X,Y  1   8.000000   0.000000
I,X,Y  2   6.000000   6.000000
I,X,Y  3   0.000000   0.000000
I,X,Y  4   8.000000   0.000000

J,U,V  1  11.000000   4.000000
J,U,V  2   4.000000   4.000000
J,U,V  3   2.000000   2.000000
J,U,V  4  11.000000   4.000000

J,K,WW,ZZ,LW  2  1  4.000000  4.000000  1
J,K,WW,ZZ,LW  3  2  2.000000  2.000000  1

FST,P,Q   6.666667   4.000000

MO,I,J,P,Q,DEL  2  1  1  6.666667  4.000000 -42.0
MO,I,J,P,Q,DEL  2  2  3  2.000000  2.000000 -42.0
MO,I,J,P,Q,DEL  1  3  3
MO,I,J,P,Q,DEL  2  4  3  6.965517  3.103448  58.0
MO,I,J,P,Q,DEL  2  4  4  6.666667  4.000000 -42.0

JJ,W,Z  1   2.000000   2.000000
JJ,W,Z  2   6.965517   3.103448
JJ,W,Z  3   6.666667   4.000000
JJ,W,Z  4   4.000000   4.000000
```



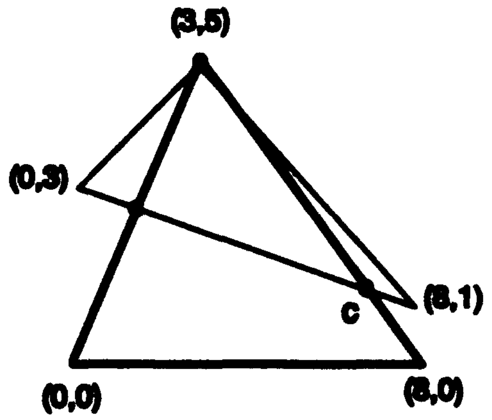
EXAMPLE 6

```
I,X,Y  1  10.000000   0.000000
I,X,Y  2   5.000000   5.000000
I,X,Y  3   0.000000   0.000000
I,X,Y  4  10.000000   0.000000

J,U,V  1  10.000000   2.000000
J,U,V  2   5.000000   5.000000
J,U,V  3   2.000000   2.000000
J,U,V  4  10.000000   2.000000

J,K,WW,ZZ,LW  2  1  5.000000  5.000000  0
J,K,WW,ZZ,LW  3  2  2.000000  2.000000  1

FST,P,Q   8.000000   2.000000

MO,I,J,P,Q,DEL  2  1  1  8.000000  2.000000  40.0
MO,I,J,P,Q,DEL  2  2  2  5.000000  5.000000  40.0
MO,I,J,P,Q,DEL  3  2  3  5.000000  5.000000  0.0
MO,I,J,P,Q,DEL  1  3  4
MO,I,J,P,Q,DEL  2  4  4  8.000000  2.000000  40.0

JJ,W,Z  1   2.000000   2.000000
JJ,W,Z  2   8.000000   2.000000
JJ,W,Z  3   5.000000   5.000000
```



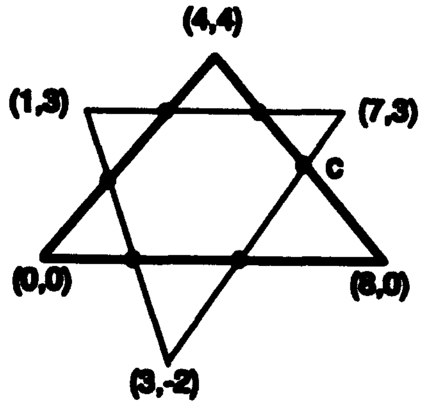
EXAMPLE 7

```
I,X,Y  1  8.000000  0.000000
I,X,Y  2  3.000000  5.000000
I,X,Y  3  0.000000  0.000000
I,X,Y  4  8.000000  0.000000

J,U,V  1  0.000000  3.000000
J,U,V  2  8.000000  1.000000
J,U,V  3  3.000000  5.000000
J,U,V  4  0.000000  3.000000

J,K,WW,ZZ,LW  3  1  3.000000  5.000000  0

FST,P,Q  6.666667  1.333333

MO,I,J,P,Q,DEL  2  1  1  6.666667  1.333333  30.0
MO,I,J,P,Q,DEL  2  2  2  3.000000  5.000000  37.0
MO,I,J,P,Q,DEL  2  2  3  3.000000  5.000000   9.0
MO,I,J,P,Q,DEL  2  2  4  1.565217  2.608696 -46.0
MO,I,J,P,Q,DEL  1  3  4
MO,I,J,P,Q,DEL  2  4  4  6.666667  1.333333  30.0

JJ,W,Z  1  6.666667  1.333333
JJ,W,Z  2  3.000000  5.000000
JJ,W,Z  3  1.565217  2.608696
```



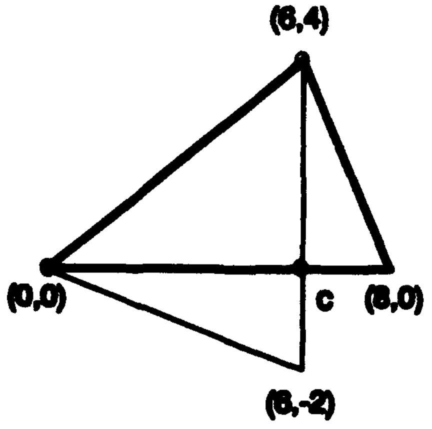
EXAMPLE 8

```
I,X,Y  1  8.000000  0.000000
I,X,Y  2  4.000000  4.000000
I,X,Y  3  0.000000  0.000000
I,X,Y  4  8.000000  0.000000

J,U,V  1  3.000000 -2.000000
J,U,V  2  7.000000  3.000000
J,U,V  3  1.000000  3.000000
J,U,V  4  3.000000 -2.000000

FST,P,Q  6.111111  1.888889

MO,I,J,P,Q,DEL  2  1  1  6.111111  1.888889  36.0
MO,I,J,P,Q,DEL  2  1  2  5.000000  3.000000 -24.0
MO,I,J,P,Q,DEL  2  2  2  3.000000  3.000000  24.0
MO,I,J,P,Q,DEL  2  2  3  1.571429  1.571429 -28.0
MO,I,J,P,Q,DEL  2  3  3  2.200000  0.000000  40.0
MO,I,J,P,Q,DEL  2  3  4  4.600000  0.000000 -40.0
MO,I,J,P,Q,DEL  2  4  4  6.111111  1.888889  36.0

JJ,W,Z  1  2.200000  0.000000
JJ,W,Z  2  4.600000  0.000000
JJ,W,Z  3  6.111111  1.888889
JJ,W,Z  4  5.000000  3.000000
JJ,W,Z  5  3.000000  3.000000
JJ,W,Z  6  1.571429  1.571429
```



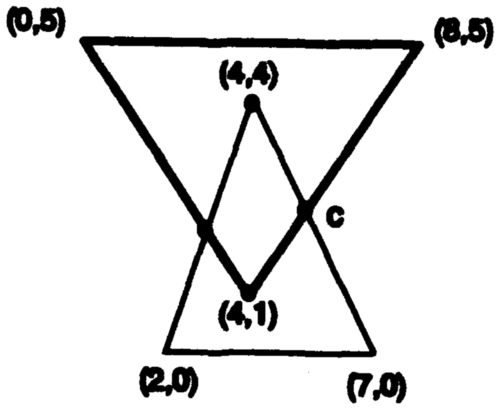
EXAMPLE 9

```
I,X,Y  1  0.000000  0.000000
I,X,Y  2  8.000000  0.000000
I,X,Y  3  6.000000  4.000000
I,X,Y  4  0.000000  0.000000

J,U,V  1  6.000000 -2.000000
J,U,V  2  6.000000  4.000000
J,U,V  3  0.000000  0.000000
J,U,V  4  6.000000 -2.000000

J,K,WW,ZZ,LW  2  1  6.000000  4.000000 0
J,K,WW,ZZ,LW  3  2  0.000000  0.000000 0

FST,P,Q  6.000000  0.000000

MO,I,J,P,Q,DEL  2  1  1  6.000000  0.000000 -48.0
MO,I,J,P,Q,DEL  1  2  3
MO,I,J,P,Q,DEL  2  3  3  0.000000  0.000000 -36.0
MO,I,J,P,Q,DEL  2  4  3  0.000000  0.000000  16.0
MO,I,J,P,Q,DEL  2  4  4  6.000000  0.000000 -48.0

JJ,W,Z  1  0.000000  0.000000
JJ,W,Z  2  6.000000  0.000000
JJ,W,Z  3  6.000000  4.000000
```



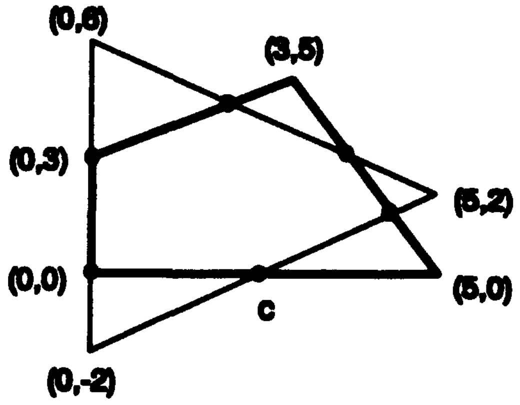
EXAMPLE 10

```
I,X,Y  1  4.000000  1.000000
I,X,Y  2  8.000000  5.000000
I,X,Y  3  0.000000  5.000000
I,X,Y  4  4.000000  1.000000

J,U,V  1  7.000000  0.000000
J,U,V  2  4.000000  4.000000
J,U,V  3  2.000000  0.000000
J,U,V  4  7.000000  0.000000

J,K,WW,ZZ,LW  2  1  4.000000  4.000000 1

I,K,WW,ZZ,LZ  1  2  4.000000  1.000000  1

FST,P,Q  5.285714  2.285714

MO,I,J,P,Q,DEL  2  1  1  5.285714  2.285714 -28.0
MO,I,J,P,Q,DEL  1  2  2
MO,I,J,P,Q,DEL  2  3  2  3.000000  2.000000  24.0
MO,I,J,P,Q,DEL  1  4  3
MO,I,J,P,Q,DEL  2  4  4  5.285714  2.285714 -28.0

JJ,W,Z  1  4.000000  1.000000
JJ,W,Z  2  5.285714  2.285714
JJ,W,Z  3  4.000000  4.000000
JJ,W,Z  4  3.000000  2.000000
```



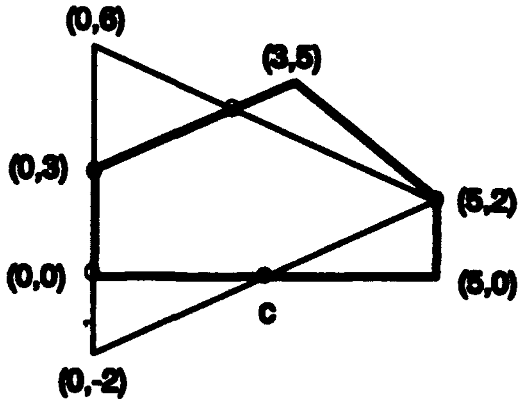
EXAMPLE 11

```
I,X,Y   1   0.000000   0.000000
I,X,Y   2   5.000000   0.000000
I,X,Y   3   3.000000   5.000000
I,X,Y   4   0.000000   3.000000
I,X,Y   5   0.000000   0.000000

J,U,V   1   0.000000  -2.000000
J,U,V   2   5.000000   2.000000
J,U,V   3   0.000000   6.000000
J,U,V   4   0.000000  -2.000000

I,K,WW,ZZ,LZ   1   1   0.000000   0.000000  1
I,K,WW,ZZ,LZ   4   2   0.000000   3.000000  1

FST,P,Q   2.500000   0.000000

MO,I,J,P,Q,DEL   2   1   1   2.500000   0.000000 -20.0
MO,I,J,P,Q,DEL   2   2   1   4.393939   1.515152  33.0
MO,I,J,P,Q,DEL   2   2   2   3.823529   2.941176 -17.0
MO,I,J,P,Q,DEL   2   3   2   2.045455   4.363636  22.0
MO,I,J,P,Q,DEL   2   5   3   0.000000   0.000000  40.0
MO,I,J,P,Q,DEL   2   5   4   2.500000   0.000000 -20.0

JJ,W,Z   1   0.000000   0.000000
JJ,W,Z   2   2.500000   0.000000
JJ,W,Z   3   4.393939   1.515152
JJ,W,Z   4   3.823529   2.941176
JJ,W,Z   5   2.045455   4.363636
JJ,W,Z   6   0.000000   3.000000
```



EXAMPLE 12

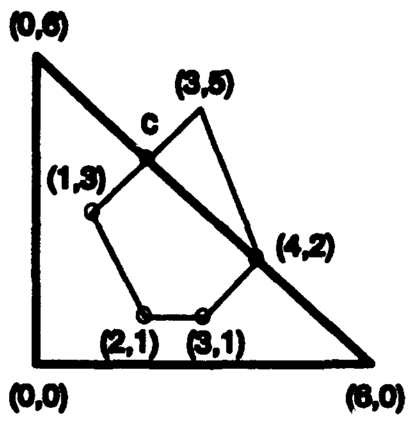
```
I,X,Y   1   0.000000   0.000000
I,X,Y   2   5.000000   0.000000
I,X,Y   3   5.000000   2.000000
I,X,Y   4   3.000000   5.000000
I,X,Y   5   0.000000   3.000000
I,X,Y   6   0.000000   0.000000

J,U,V   1   0.000000  -2.000000
J,U,V   2   5.000000   2.000000
J,U,V   3   0.000000   6.000000
J,U,V   4   0.000000  -2.000000

J,K,WW,ZZ,LW   2   1   5.000000   2.000000  0

I,K,WW,ZZ,LZ   1   2   0.000000   0.000000  1
I,K,WW,ZZ,LZ   5   3   0.000000   3.000000  1

FST,P,Q   2.500000   0.000000

MO,I,J,P,Q,DEL   2   1   1   2.500000   0.000000 -20.0
MO,I,J,P,Q,DEL   2   2   2   5.000000   2.000000 -10.0
MO,I,J,P,Q,DEL   2   3   2   5.000000   2.000000  -7.0
```



```
MO,I,J,P,Q,DEL  2  4  2  2.045455  4.363636  22.0
MO,I,J,P,Q,DEL  2  6  3  0.000000  0.000000  40.0
MO,I,J,P,Q,DEL  2  6  4  2.500000  0.000000 -20.0

JJ,W,Z  1  0.000000  0.000000
JJ,W,Z  2  2.500000  0.000000
JJ,W,Z  3  5.000000  2.000000
JJ,W,Z  4  2.045455  4.363636
JJ,W,Z  5  0.000000  3.000000
```

EXAMPLE 13

```
I,X,Y  1  0.000000  0.000000
I,X,Y  2  6.000000  0.000000
I,X,Y  3  0.000000  6.000000
I,X,Y  4  0.000000  0.000000

J,U,V  1  1.000000  3.000000
J,U,V  2  2.000000  1.000000
J,U,V  3  3.000000  1.000000
J,U,V  4  4.000000  2.000000
J,U,V  5  3.000000  5.000000
J,U,V  6  1.000000  3.000000

J,K,WW,ZZ,LW  1  1  1.000000  3.000000  1
J,K,WW,ZZ,LW  2  2  2.000000  1.000000  1
J,K,WW,ZZ,LW  3  3  3.000000  1.000000  1
J,K,WW,ZZ,LW  4  4  4.000000  2.000000  1

FST,P,Q  2.000000  4.000000

MO,I,J,P,Q,DEL  2  1  1  2.000000  4.000000 -24.0
MO,I,J,P,Q,DEL  1  2  5
MO,I,J,P,Q,DEL  1  3  5
MO,I,J,P,Q,DEL  2  4  5  4.000000  2.000000  12.0
MO,I,J,P,Q,DEL  2  4  6  2.000000  4.000000 -24.0

JJ,W,Z  1  2.000000  1.000000
JJ,W,Z  2  3.000000  1.000000
JJ,W,Z  3  4.000000  2.000000
JJ,W,Z  4  2.000000  4.000000
JJ,W,Z  5  1.000000  3.000000
```



DISTRIBUTION

	COPIES
DOD ACTIVITIES (CONUS)	
ATTN CMP/PMA 280	1
CMP/PMA 281	1
CMP/PMA 282	1
NAVAL AIR SYSTEMS COMMAND CRUISE MISSILE PROJECT NAVAL AIR SYSTEMS COMMAND HEADQUARTERS WASHINGTON DC 20361-1014	
ATTN SPAWAR 31 COMMANDER SPACE AND NAVAL WARFARE SYSTEMS COMMAND WASHINGTON DC 20363-5100	1
ATTN N81 CHIEF OF NAVAL OPERATIONS DEPARTMENT OF THE NAVY WASHINGTON DC 20350-2000	1
ATTN J8 JOINT CHIEFS OF STAFF THE PENTAGON WASHINGTON DC 20318-0001	1
DEFENSE TECHNICAL INFORMATION CENTER CAMERON STATION ALEXANDRIA VA 22304-6145	12
ATTN PROVOST NAVAL POST GRADUATE SCHOOL MONTEREY CA 93943-5000	1
ATTN GIFT AND EXCHANGE DIV LIBRARY OF CONGRESS WASHINGTON DC 20540	4
INTERNAL	
E 231	3
E 232	2
K10 J. SLOOP	1
K104 A. DIDONATO	5
K12 W. ORMSBY	1
K13 G. TALLANT	1
LO4 S. PARKER	1
L1OMP S. WOOD	1
L11 S. TALLANT	2
N74 (GIDEP)	1

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1993	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE An Algorithm to Find the Intersection of Two Convex Polygons	5. FUNDING NUMBERS
6. AUTHOR(S) Armido R. DiDonato	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Surface Warfare Center Dahlgren Division (Code K104) Dahlgren, VA 22448-5000	8. PERFORMING ORGANIZATION REPORT NUMBER NSWCDD/TR-93/345
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An algorithm is given that finds the intersection of two convex polygons. It is coded in Fortran for the IBM PC desktop computer. The program is robust and fast. It has been used successfully in targeting applications that require a rapid determination of the common intersection of more than 100 convex polygons, each specified by more than 150 vertices.

14. SUBJECT TERMS Closed, Convex Polygons Algorithm, INTSEC	15. NUMBER OF PAGES 27
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19 SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and its title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

BLOCK 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If Known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2
NTIS	- Leave blank

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract Include a brief ***(Maximum 200 words)*** factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code ***(NTIS only)***

Block 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of this page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited