22
4-19-77
250 HT/S

UCID-17430

Lawrence Livermore Laboratory

AREA OF INTERSECTION OF ARBITRARY POLYGONS

Stephen I. Warshaw

March 1977

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

AREA OF INTERSECTION OF ARBITRARY POLYGONS*

Stephen I. Warshaw

ABSTRACT

The geometric aspects of calculating the area or volume of the region of overlap between two intersecting arbitrary polygons are considered in detail, for the purpose of adapting them to a workable algorithm suitable for a digital computer. Such an algorithm is described, and certain floating point arithmetic errors that occur that are detrimental to the operation of the algorithm are also discussed. Methods for circumventing these difficulties are presented.

NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

MASTER

* This work was performed under the auspices of the United States Energy Research and Development Administration under contract No. W-7405-Eng-48.

Table of Contents

Abstract.	i
Preface and Acknowledgements.	iii
I. Introduction.	1
II. Geometric Considerations	
A. Convex Polygons	5
B. Intersection of Two Convex Polygons.	8
C. Area of the Intersection Polygon	12
D. Nonconvex Polygons	14
E. Crossed Polygons	17
III. Overlap Area Calculation Pathologies	
A. Vertex-Side, Vertex-Vertex and Side-Side Coincidences	19
B. Resolution of the Coincidence Problem.	22
C. Almost Convex Polygons	25
D. Redundant Polygon Vertices	26
E. Kershaw's Rotation Test.	27
IV. Overlap Area or Volume Algorithm	
A. Executive Routine	29
B. Overlap Area or Volume Calculation	31
C. Alternatives	32
V. Appendices	
A. Area or Volume of Revolution of a Plane Polygon.	34
B. Numerical Example of a Vertex-Line Pathology.	35
References.	39
Distribution.	40

Preface and Acknowledgements

The material contained herein was assembled "on the back burner", as it were, over the better part of a few months. During and prior to that time, the author had many interesting discussions regarding problems of mapping with those who program for Lagrangian and Eulerian hydrodynamics codes at LLL; taken collectively these discussions seemed to indicate the need for a document such as this one, particularly by those whose interests lie in carrying mapping concepts over to vector or parallel type computers.

The author is especially indebted to David Kershaw for a stimulating discussion in which he provided him with a rapid and convenient method of testing his material on the computer, and for impressing upon him the need for taking floating point arithmetic problems seriously. Were it not for this, the project this paper represents might have taken longer than it did. All references herein to "D. Kershaw, private communication" are to material touched on in that discussion.

The author is also indebted to Jerry Owens for many discussions on feasible methods of adapting the contents of this paper to vector type machines. Thanks are due John Woodruff for catching an apparent inconsistency in the application of the concave polygon methodology. This resulted not only in improvements to the algorithm, but this manuscript as well.

Area of Intersection of Arbitrary Polygons

Stephen I. Warshaw

I. Introduction

The calculation of the area of the region common to two plane, overlapping polygons is a simple matter. Much of this simplicity is lost when the calculation is done by a digital computer, particularly when speed and economy of storage are serious considerations. In this paper, we outline the salient features of this calculation, and present an approach to an algorithm to perform it on a scalar type machine.

There are three reasons for considering this apparently trivial calculation in detail. First, those who use computer simulation codes based on a partition of the plane into a finite mesh of polygonal cells often find it necessary to transfer information from one such mesh to another having cells of different shapes. At some point in this mapping process, the areas (or volumes of revolution) of the region common to overlapping cells must be calculated. If the number of cells involved is large, or this process must be repeated frequently, the area calculation can turn out to be the most expensive part of the process. It is therefore desirable to have in the calculation algorithm only those operations which seem essential for speed and efficiency. We present a set of such operations here, with the hope that if it is not an optimum choice, it may very well be close to one.

Second, an overlap area calculation when done on a computer inevitably contains programming logic that depends on the result of a comparison of two floating point numbers. That is, at certain places in the coding, a decision

as to which of several paths the calculation is to take will rest solely on whether one floating point number is larger than another. Normally, this causes no problems. But if the numbers happen to be very close to one another, the calculated difference may be of either sign. Such randomness is not the fault of the comparison itself, but is due to errors which each number has acquired in the calculations that have led up to them. These errors are not conveniently predictable, and therefore neither are the logical decisions that rest on this comparison. As a consequence, this randomness can and does create serious logical damage to the result, so that answers vastly different from the true area are obtained. (Few things can be more infuriating than an absolutely precise, but completely wrong answer obtained because a wrong logical path was taken.) Therefore, our second reason for considering these matters is to present methods for circumventing, or at least minimizing this logical damage in places where it really matters.

In satisfying these two reasons, contradictions may arise. If we succeed in speeding up a calculation by a good choice of operations, we may well slow it down with coding designed to eliminate the randomness discussed previously. On a vector type machine, it is not yet clear that these contradictions will be more easily resolved.

The third reason for this paper is to provide a reference source for overlap-type calculations. To the author's knowledge, overlap area calculations on computers have existed for a long time, but little, if any, documentation is conveniently available, and what there is does not explicitly address the

details of the algorithms used. We suspect, however, that what we present here has much in common with some of the older algorithms, and so it is hoped that those who already have or plan to use overlap area algorithms, particularly for vector-type machines, will find the contents of this manuscript of value. For this reason, also, we have avoided any explicit consideration of peculiarities characteristic of any given computer, except where it would serve for illustrative purposes.

In computer simulation with finite mesh codes, two types of polygonal overlap area calculations are generally distinguished. One type is where both of the overlapping polygons are arbitrary, as occurs in meshes of Lagrangian hydrodynamics calculations. The other type is where one of the polygons is always a rectangle, as, for example, derived from an Eulerian hydrodynamics mesh, where the mesh grid lines are maintained parallel to the Cartesian coordinate axes. We treat only the former type in this paper, since the latter type has already been treated by the author in a previously published memorandum.^[1] Readers who compare this paper with that memorandum will readily see why this distinction has been made: a rectangular (or other curvilinear orthogonal) grid offers a distinctively different, and perhaps easier approach to the overlap area calculation than that given here. While it could be suggested that the arbitrary polygon approach involve the elements of a rectangular grid, this may not be desirable if the rectangular alternative is faster, as the author thinks it is.

In what follows, we shall first give an outline of the geometry of the overlap area calculation, and then consider the problems associated with the floating point comparison decision process. Only those mathematical

expressions that are employed in the algorithm used by the author will be considered in detail. Thus, the geometrical section following is actually a detailed description of that algorithm.

II. Geometric Considerations

A. Convex Polygons

We begin by discussing intersections of convex polygons, for by doing so great simplifications in the overall calculation are achieved.

A convex polygon is one each of whose interior vertex angles is not greater than 180° . This description is consistent with the canonical definition of a convex region, whereby a region is said to be convex if all straight line segments drawn connecting any two of its points lie wholly within it or on its boundary. Thus, doughnuts and kidney beans are not convex solids, but eggs and grapefruit are. Since the boundary of the region is included in this definition, convex regions are also closed, simply connected, and bounded.

One may prove that the intersection of two convex regions is again a convex region. (Details and proofs of this and other matters related to what follows are given in reference [2]. We shall not present any proofs to support the mathematical statements made here, but rather, refer the reader to [2].)

Before proceeding further, we must establish the conventions and notation to be used throughout. We specify any polygon by listing in serial order the coordinates of its vertices. Thus, straight line segments beginning and ending on adjacent vertices in this list will define the periphery of the polygon. Furthermore, we specify the convention that if we traverse this boundary from one vertex to the next, the interior of the polygon lies to left. This means that as one proceeds through the ordered list of vertices

as given, the boundary of the polygon will be traversed in a counter-clock-wise sense.

By adhering to these conventions it becomes convenient and useful to regard the sides of any polygon as oriented straight line segments, or vectors.

Let a polygon be defined by N vertex points (x_i, y_i) , $i = 1, 2, \dots, N$.

Alternatively, these vertices can be represented by vectors $\vec{r}_i = \hat{i} x_i + \hat{j} y_i$ from some origin to the vertex points. Then the vector differences

$\vec{r}_{i+1} - \vec{r}_i$ will represent these oriented straight line segments, as shown in Figure 1.

Figure 1

Now, if a polygon is convex, the following relationship holds for all pairs of adjacent sides with common vertex $\vec{r}_i$, assuming the usual right-handed coordinate system conventions:

$$(\vec{r}_i - \vec{r}_{i-1}) \times (\vec{r}_{i+1} - \vec{r}_i) \geq 0 \quad (1)$$

That is, none of these cross products can be negative. If, when subjecting an unknown polygon to this test, a vertex point $\vec{r}_i$ common to two adjacent

sides is found for which (1) is not true, then this vertex shall be referred to as a concave vertex. Of this, more later.

One very useful and powerful property of convex polygons now deserves mention: every N-sided convex polygon is the intersection of N closed half planes.^[2] Since a closed half plane is that part of the plane lying to one side of a straight line and including the line, it is immediately evident that the sides of a convex polygon are edges of such half planes. To formulate an analytic representation of a half plane consistent with our conventions, let $\vec{r}_1$ and $\vec{r}_2$ define two points on a straight line. Then we shall say that a half plane is all those points $\vec{r}$ lying on and to the left of $(\vec{r}_2 - \vec{r}_1)$. This means that the following inequality is satisfied by $\vec{r}$:

$$(\vec{r}_2 - \vec{r}_1) \times (\vec{r} - \vec{r}_1) \geq 0 \quad (2)$$

Note that this formulation is oriented, because if $\vec{r}_1$ and $\vec{r}_2$ are switched, (2) will define the other half plane. As a consequence, the points that define a convex polygon are given by the following condition that $\vec{r}$ must satisfy relative to each of the vertex pairs $\vec{r}_i$ and $\vec{r}_{i+1}$:

$$(\vec{r}_{i+1} - \vec{r}_i) \times (\vec{r} - \vec{r}_i) \geq 0, \quad i = 1, 2, \dots, N. \quad (3)$$

The order of $\vec{r}_i$ and $\vec{r}_{i+1}$ is important, and must be as shown. A comparison of (1) with (3) shows that every vertex not lying on a side of a convex polygon must lie to the left of it.

When the equals sign in (2) holds, the equation of the line itself

results. Then we obtain

$$(\vec{r}_2 - \vec{r}_1) \times \vec{r} = \vec{r}_2 \times \vec{r}_1 \quad (4)$$

or

$$(x_2 - x_1)y - (y_2 - y_1)x = x_2y_1 - y_2x_1 \quad (5)$$

B. Intersection of Two Convex Polygons

When two convex polygons overlap, their intersection is also a convex polygon. We now address the problem of how to tell a computer to find the vertices of this intersection result.

An example of two intersecting convex polygons is shown in Figure 2, where the region of intersection is shown shaded. Inspection of the figure reveals that if any vertex of one polygon lies inside the other, that vertex is also a vertex of the intersection polygon. These are labeled A, B, E, F and H in the figure.

Figure 2

Further examination also shows that if any two adjacent vertices of one polygon lie inside the other, then the side between them also lies inside, and is therefore a side of the intersection polygon. In the figure, these are sides AB and EF. Note that they do not cross any side of the other polygon. These results are a direct consequence of the canonical definition of a convex region given before.

To analytically test if a vertex of one polygon lies inside the other, we ask if that vertex lies on all the half planes of the other polygon. Let $\vec{r}_i, \vec{r}_{i+1}, \vec{r}_{i+2}, \text{etc.}$ be the vertices for one polygon, and let $\vec{r}_j, \vec{r}_{j+1}, \vec{r}_{j+2}, \dots$ define the vertices of the other. (Here, we use the index labels i and j to distinguish the different polygons.) Equation (3) is applied to each $\vec{r}_i$ (of one polygon) for every adjacent vertex pair $\vec{r}_j, \vec{r}_{j+1}$, (of the other polygon). Any failure of an $\vec{r}_i$ to satisfy the relation

$$(\vec{r}_{j+1} - \vec{r}_j) \times (\vec{r}_i - \vec{r}_j) \geq 0 \quad (6)$$

for all j is sufficient to reject it as an intersection polygon vertex. That is, any vertex of one polygon lying outside the other is not part of the intersection polygon. (See, e.g. vertices K, L, M, N, P and Q in Figure 2). In scalar form, condition (6) is

$$(x_{j+1} - x_j)(y_i - y_j) \geq (y_{j+1} - y_j)(x_i - x_j) \quad (7)$$

Now, if we find that all the vertices of one polygon lie inside the other, then that polygon is inside, and we are done. The area or volume of

overlap is just that of the inner polygon.

Consider next the situation where some or all of the vertices of each polygon lie outside the other. Then a search must be made for those pairs of sides (one from each polygon) which cross each other. (Recall from the above that sides completely inside are exempt.)

The power of the half plane concept is such that we can use it to analytically detect whether two sides intersect without having first to calculate the intersection point. Since a side segment of a polygon lies on the edge of one of that polygon's half planes, we test to see if one, and only one vertex endpoint of one segment lies on the half plane of the other, and conversely. If this happens to be so, an intersection has been found. If not (the endpoints of a segment lie both on or both off) there is no intersection. In Figure 3, three possible arrangements are shown. The shading symbolizes the half planes. Only in the first situation is there an intersection.

Figure 3

Applying equation (3), the following two relations must be satisfied simultaneously in order for two side segments to intersect:

$$\text{sgn} [(\vec{r}_{j+1} - \vec{r}_j) \times (\vec{r}_i - \vec{r}_j)] \neq \text{sgn} [(\vec{r}_{j+1} - \vec{r}_j) \times (\vec{r}_{i+1} - \vec{r}_j)] \quad (8)$$

$$\text{sgn} [(\vec{r}_{i+1} - \vec{r}_i) \times (\vec{r}_j - \vec{r}_i)] \neq \text{sgn} [(\vec{r}_{i+1} - \vec{r}_i) \times (\vec{r}_{j+1} - \vec{r}_i)] \quad (9)$$

That the cross products involved have to have different signs is the same as saying that the endpoints of one line segment must each lie on opposite sides of the other line segment, and conversely.

On detection of an intersection, its coordinates are calculated as follows:

$$x = \frac{1}{\Delta} [(\vec{r}_{j+1} \times \vec{r}_j)(x_{i+1} - x_i) - (\vec{r}_{i+1} \times \vec{r}_i)(x_{j+1} - x_j)] \quad (10)$$

$$y = \frac{1}{\Delta} [(\vec{r}_{j+1} \times \vec{r}_j)(y_{i+1} - y_i) - (\vec{r}_{i+1} \times \vec{r}_i)(y_{j+1} - y_j)] \quad (11)$$

where

$$\Delta = (\vec{r}_{j+1} - \vec{r}_j) \times (\vec{r}_{i+1} - \vec{r}_i) \quad (12)$$

Note (10) and (11) can be combined to produce the following "vector equivalent":

$$\vec{r} = \frac{1}{\Delta} [(\vec{r}_{j+1} \times \vec{r}_j)(\vec{r}_{i+1} - \vec{r}_i) - (\vec{r}_{i+1} \times \vec{r}_i)(\vec{r}_{j+1} - \vec{r}_j)] \quad (13)$$

(This is not a true vector relation, since all the cross products here are symbols for their algebraic equivalents in terms of x and y.) One of the virtues of this particular approach is that the quantities used in (8) and (9) may be saved in the computer for the calculation in (10) and (11).

C. Area of the Intersection Polygon

From the vertices found by the previous tests we may calculate the area or volume of revolution of the intersection polygon. If one of the polygons lies inside the other, this area is easily determined. If not, then a new problem arises: the vertices of the intersection polygon have to be put in proper order. In keeping the previous calculations to a minimum, we have deliberately not kept track of the order each found intersection polygon vertex has relative to the others, and now must determine it.

The fact that the intersection polygon is known to be convex permits the proper ordering of its vertices to be found with little difficulty. For example, the centroid of a convex figure certainly lies within the figure, and if we draw radius vectors from the centroid to each of the intersection vertices, these vertices may then be properly ordered according to their polar angles (compass directions of those vectors).

A perhaps cheaper alternative is to consider directly the slopes of all lines drawn from one vertex to all the others. These lines radiate outward in a fan, and suggest the possibility of an ordering based on the cotangent or tangents of their polar angles. The essentials of this method

are as follows: choose a convenient vertex as the "base point", such as, for example, the one for which its y coordinate is the maximum. Draw radius vectors from this base point to every other vertex of the intersection polygon, as shown in Figure 4.

Figure 4

If we consider the inverse slopes of all these vectors, i.e. the cotangent ratios $\frac{\Delta x}{\Delta y}$ of each point relative to the base point, then the points can be ordered according to these ratios. Thus, the proper ordering for the intersection polygon points is a listing of the vertices in order of decreasing cotangent ratio, which, from Figure 4, is indeed counter-clockwise. That this method is valid rests solely on the fact that the intersection polygon is convex: these radius vectors always lie within or on the boundary of the polygon, regardless of choice of base point.

Formulas for the area or volume of revolution of a polygon in terms of its vertex coordinates are given in Appendix A.

D. Non-convex Polygons

The virtue of the convex polygon approach lies in the fact that it makes the calculation of the area of intersection of two arbitrary polygons almost trivial. The reason for this is that any non-convex polygon may be recast into a convex polygon and one or more completely flipped triangles. By various combinations of overlap of these convex elements, the calculation of the general case is reduced to a short sequence of computations involving only convex figures.

As an example of this restructuring, consider the nonconvex polygon shown in Figure 5.

Figure 5

If we start at vertex A and apply the test for convexity given by Eq. (1) to each successive vertex, we find that the vertex at E fails this test, and E is therefore a concave point. At this juncture we save the vertices of triangle DFE and alter the polygon by removing vertex E. Then we repeat the process over, starting one vertex back, i.e. at C on the

reformed polygon shown in Figure 6. Now D is found also to be a concave

Figure 6

point. In exactly the same manner, triangle CFD is saved, and the vertex at D removed with the result shown in Figure 7.

Figure 7

This second generation polygon is convex. It is with this polygon along with the two "concave triangles" found that the overlap calculations are done, since now all the polygons involved are convex.

Because of this restructuring the calculation of the area of overlap between two polygons which may each have concave vertices can be handled with Boolean logic. Nothing more is involved than the construction of the appropriate Venn diagram, using the "reduced" convex polygons and their adjacent triangular "concave elements" as elements.

Consider two arbitrary polygons, P and Q. Let them each be reformed as respectively convex envelopes P^+ and Q^+ , with corresponding concave elements P^- and Q^- . (That these sets of concave elements may each contain two or more triangles offers no difficulty, because the triangles within each set do not overlap one another.) Let

$$\left. \begin{aligned} S_0 &= \text{area of overlap between } P^+ \text{ and } Q^+ \\ S_1 &= \text{area of overlap between } P^+ \text{ and } Q^- \\ S_2 &= \text{area of overlap between } Q^+ \text{ and } P^- \\ S_3 &= \text{area of overlap between } P^- \text{ and } Q^- \end{aligned} \right\} \quad (14)$$

Then the area of intersection of the two polygons P and Q is

$$S = S_0 + S_3 - (S_1 + S_2) \quad (15)$$

Equation (15) may easily be verified by regarding the above polygons as point sets, where $P^+ = P \cup P^-$, $P \cap P^- = 0$, $P^+ \cap P^- = P^-$, with analogous relations for Q. If P^+ and Q^+ are made to overlap in a Venn diagram, it can readily be observed that S_0 is the actual area of $P^+ \cap Q^+$ in the diagram, S_1 is the area of $P^+ \cap Q^-$, and so on. By the appropriate graphical superpositions, Eq. (15) is then obtained for S as the area of $P \cap Q$.

As an example of the foregoing, consider the intersection of two concave five-sided polygons as shown in Figure 8.

Figure 8

The area of intersection, shown shaded, is calculated from Eq. (15), where:

- S_0 = area of overlap of quadrilateral FHJK with triangle ABE
- S_1 = area of overlap of quadrilateral FHJK with the two "concave triangles" BDC and BED
- S_2 = area of overlap of triangle ABE with the "concave triangle" GFH
- S_3 = area of overlap of "concave triangle" GFH with the "concave triangles" BDC and BED.

The reader may satisfy himself that this is so by drawing the actual diagrams of the four figures represented by the S_i .

E. Crossed Polygons

This discussion would be incomplete if we did not consider a situation that comes up frequently: that where two or more sides of a given polygon may indeed intersect. The simplest example of this is the "bowtied"

quadrilateral shown in Figure 9, where two opposite sides, AD and BC, intersect at E.

Figure 9

There is nothing about crossed polygons that the methodology developed up to this point cannot handle; the decomposition process will also work for them. Difficulty arises only if no provision is made to accept overlap area results that are negative, or no allowance is made for negative components of areas of crossed polygons, such as represented by either triangle ABE or EDC of Figure 9.

This problematical situation arises when computations are required of areas of overlap between crossed polygons that develop in twisted meshes in, e.g., Lagrangian hydrodynamics studies. Regardless of which algorithm they choose to pursue the matter, those who calculate these areas must be able to provide usable and consistent interpretations of negative area components. These interpretations are really outside the scope of this paper, and will not be considered further.

III. Overlap Area Calculation Pathologies

Problems will arise in the overlap area calculation if the limitations inherent in a computer for handling real numbers are ignored. In this part we shall discuss these problems in detail, and then describe Kershaw's method^[3] for testing overlap algorithms.

A. Vertex-Side, Vertex-Vertex and Side-Side Coincidences

Consider a case where two overlapping polygons are situated so that a vertex of one lies on a side of the other, as shown in Figure 10.

Figure 10

In the process of finding if point B is part of the intersection polygon, a test is made to see on which side of line EF point B lies. This is done by asking if the following condition holds:

$$(\vec{r}_F - \vec{r}_E) \times (\vec{r}_B - \vec{r}_E) \geq 0 \quad (16)$$

which is just Eq. (3) applied to the process. If (16) is not satisfied, then B is assumed to lie outside triangle GEF, and we would expect two intersections to be found near B: one occurring between line segments BC and

EF, and the other between AB and EF. Applying Eqs. (8) and (9) for detection of an intersection, we would expect for the AB-EF intersection that

$$\text{sgn}[(\vec{r}_B - \vec{r}_A) \times (\vec{r}_E - \vec{r}_A)] \neq \text{sgn}[(\vec{r}_B - \vec{r}_A) \times (\vec{r}_F - \vec{r}_A)] \quad (17)$$

$$\text{sgn}[(\vec{r}_F - \vec{r}_E) \times (\vec{r}_A - \vec{r}_E)] \neq \text{sgn}[(\vec{r}_F - \vec{r}_E) \times (\vec{r}_B - \vec{r}_E)] \quad (18)$$

and for the BC-EF intersection that

$$\text{sgn}[(\vec{r}_C - \vec{r}_B) \times (\vec{r}_E - \vec{r}_B)] \neq \text{sgn}[(\vec{r}_C - \vec{r}_B) \times (\vec{r}_F - \vec{r}_B)] \quad (19)$$

$$\text{sgn}[(\vec{r}_F - \vec{r}_E) \times (\vec{r}_B - \vec{r}_E)] \neq \text{sgn}[(\vec{r}_F - \vec{r}_E) \times (\vec{r}_C - \vec{r}_E)] \quad (20)$$

Notice that the cross product in (16) is repeated in the right side of (18) and the left side of (20). If (16) is not satisfied, this cross product is negative, and we would therefore expect that both the left side of (18) and the right side of (20) would have positive sign. Indeed, from inspection of Figure 10, we find this is so. The signs of the other cross products also appear equally unambiguous.

Theoretically, at least, there is no difficulty. Either B lies on the line EF, to the left of EF, or to the right of EF. For these first two cases (16) will be satisfied, and for the third case where it is not, the two intersections will be detected and calculated. There is therefore no way, in principle, that we can get into trouble.

However, if these calculations are done by computer, we can lose all three points! The assumption underlying all the above is that the cross product $(\vec{r}_F - \vec{r}_E) \times (\vec{r}_B - \vec{r}_E)$ was calculated so as to have identical values in equations (16), (18) and (20). If, and only if we set up the coding in the

algorithm so this assumption is guaranteed can we absolutely justify it. Otherwise, anything can happen.* In a finite word size machine, the associative and distributive laws for the arithmetic of floating point numbers do not hold.^[4] If, for reasons of speed and efficiency we should choose variously differing calculational paths to reach this cross product at each of the three places indicated by (16), (18) and (20), then we have given up the assurance that the results of (16), (18) and (20) will be consistent. Not only the values, but the signs of the cross product may vary.

In Appendix B, we give an actual example of a case where, after having decided that B was outside GEF, the algorithm then passed on to decide that neither AB nor BC intersected EF. The result was that B was effectively eliminated, and the algorithm calculated the overlap area between triangle GEF and the "logically mutilated" triangle ACD. The answer was precise, to be sure, but very, very wrong.

This pathology is characteristic of all situations where a vertex of one polygon lies anywhere on the boundary of the other. By "on" we mean at the machine limit, where the last few bits may determine the future course of calculations involving that vertex. Vertex-vertex or side-side coincidences are in principle no different from vertex-side coincidences, since in all cases the same test applies: that for the location of a point in respect of a line. Except for certain peculiarities to be discussed further on, there appears no need to consider here point-point comparisons as such.

* ... and it will.. (Murphy's Law)

B. Resolution of the Coincidence Problem

There are two ways to get around the difficulty posed by coincidences. One way is to arrange the coding so that once a particular cross product is calculated, the result is saved to be used again in other parts of the code that need it. A considerable amount of bookkeeping and storage may result, not to mention the possibility of losses in speed and efficiency. The other way is to judiciously apply tolerance checking, as will be described below. Depending on the applications intended for the overlap area algorithm, either approach may be preferred over the other.

In the tolerance approach, decisions are made based on how close the coincidence is. As it turns out, this is surprisingly easy and cheap to do, since the cross product formalism lends itself quite naturally to it.

In employing cross products to classify a vertex, the purpose has been to determine on which side of an oriented line a point lies. The viewpoint taken was always whether the point fell on or off the associated half plane, and all the difficulties mentioned in the previous section have to do with points "teetering on the edge". In Figure 11, consider an oriented line segment AB, where its associated half plane lies to its left as the line is traversed from A to B. Suppose a point, C, to lie very close to AB, but off the plane, as shown.

Figure 11

Let C be shifted perpendicular to AB toward the plane by a small amount so it rests at C'. This shift can be obtained by calculating the coordinates of C' from

$$x_{C'} = x_C - t(y_B - y_A) \quad (22)$$

$$y_{C'} = y_C + t(x_B - x_A) \quad (23)$$

where t is a small, positive proportionality factor measuring the amount and direction of the shift. (If t was negative, the shift would be in the opposite direction.)

Then, if application of Eq. (3) in determining where C' is relative to AB shows C' to lie on the half plane, we shall claim that the unshifted point C lies on the half plane within a factor of tolerance t.

There are two attractive features of this proposal. First, t is a proportion, equal to the ratio of the shift distance CC' to the length of AB. Thus, this tolerance test scales according to the size of the line segment, i.e. is not absolute. Second, its implementation is extremely simple. For resolution of the coincidence problem, the only place we need to apply it is where a decision is made on whether a vertex of a polygon lies inside the other, viz. Eq. (7). In this equation, $\vec{r}_i$ is the point in question. On applying the dummy shifts represented by (22) and (23) to its coordinates x_i and y_i , the following "tolerance-augmented" form of (7) is obtained:

$$\Delta x_j (y_i - y_j + t \Delta x_j) \geq \Delta y_j (x_i - x_j - t \Delta y_j) \quad (24)$$

where $\Delta x_j = x_{j+1} - x_j$, and $\Delta y_j = y_{j+1} - y_j$. The vector equivalent of (24) may easily be shown to be

$$(\vec{r}_{j+1} - \vec{r}_j) \times (\vec{r}_i - \vec{r}_j) \geq -t|\vec{r}_{j+1} - \vec{r}_j|^2 \quad (25)$$

which is just the tolerance-augmented form of (6). Since this dummy shift of $\vec{r}_i$ is toward the inside of the polygon two of whose vertices are $\vec{r}_j$ and $\vec{r}_{j+1}$, we shall claim that any failure of $\vec{r}_i$ to satisfy (24) or (25) for all j is sufficient to reject $\vec{r}_i$ as a vertex of the intersection polygon, within tolerance t . An example of this application is given also in Appendix B.

The reason why this is the crucial place to apply the tolerance test and not also during the search for side-side intersections as represented by (8) and (9) is this: once a vertex very slightly off a side has been identified within tolerance as an intersection polygon vertex, the subsequent calculation of the side-side intersections which occur at this vertex will not really matter. Both the two intersections (if found) and the vertex will be so close together that the area of the triangle they form will be of order t^2 of the main overlap result. If t is sufficiently small (say $\sim 10^{-13}$ or 10^{-14} for a 60-bit machine), the error made in omitting this small triangle may not be detectable.

Finally, we observe that an argument could indeed be made for also applying the tolerance approach to (8) and (9), but in the opposite sense, viz., to reject found side-side intersections that are very close or "on top of" a found vertex. Since the number of side-side combinations is much larger than the sum of the vertices of both polygons, this approach should be made gingerly if computational speed is of the essence. However, such multiply degenerate vertices of an intersection polygon do not substantially affect the accuracy of the main result, except for a special situation to be mentioned further on.

C. Almost Convex Polygons

Another place where the above tolerance check proves useful is when a polygon is initially tested for concave points. This application is likewise easy, and will be sketched briefly here.

If a polygon is very slightly concave at one of its vertices, then the area of the saved "concave triangle" (see Part II, Section D) may be so small as to not be worth involving in the subsequent overlap calculations. What can be done for this situation is to apply tolerance checking to Eq. (1), which tests for convexity. Thus, if we replace $\vec{r}_{i+1}$ in (1) with its dummy-shifted counterpart as prescribed in (22) and (23), we are effectively straightening out or making convex the potentially concave vertex at $\vec{r}_i$. (Consider, for example, vertices C, D and F in Figure 6, corresponding to $i-1$, i and $i+1$.) On carrying out the indicated replacement $[x_{i+1} - t(y_i - y_{i-1})$ for x_{i+1} and $y_{i+1} + t(x_i - x_{i-1})$ for $y_{i+1}]$ we obtain, in place of (1),

$$(\vec{r}_i - \vec{r}_{i-1}) \times (\vec{r}_{i+1} - \vec{r}_i) \geq -t|\vec{r}_i - \vec{r}_{i-1}|^2 \quad (26)$$

If this test for convexity at $\vec{r}_i$ is passed with the dummy $\vec{r}_{i+1}$, then we shall claim, in the spirit of the previous section, that $\vec{r}_i$ is not a concave point.

In this connection, suppose a vertex $\vec{r}_i$ satisfies (26) for t positive, but fails it when t is made negative. Then for all practical purposes $\vec{r}_i$ may be said to lie on the straight line joining $\vec{r}_{i+1}$ and $\vec{r}_{i-1}$, and thus may be omitted safely from the original polygon vertex array. If we decide to so preselect its vertices at the same time a polygon is undergoing tests for convexity, then we potentially gain in overall speed of computation.

D. Redundant Polygon Vertices

Suppose either of the overlapping polygons has two adjacent vertices that lie practically on top of one another. If we allow these points to be carried through the overlap process as described so far, then we run the risk of generating a series of "almost zero" cross product problems similar to those previously encountered. It therefore becomes desirable to remove this redundancy. As it turns out, this is automatically taken care of by the tests given on the previous page for screening collinear vertices. If we consider $\vec{r}_i$ and $\vec{r}_{i+1}$ of the previous page to be coincident, then by this test $\vec{r}_i$ will be removed. The coincident case is a particular case of the collinear one, as the only difference here is in the relative proximity of the two points. We note that this removal will also significantly lessen the number of operations required for the overlap calculation.

Next, consider the situation where some of the found points of the intersection polygon turn out to be likewise coincident. No difficulties will arise unless, during the ordering of the vertices, the "base point" used for the ordering* is a multiple point. Then the polar cotangents calculated from these points would be ratios of "almost zero" numbers, and, as with the almost zero cross products, can take on any value whatever. Since we still have yet to order the vertices, this base point redundancy has to be removed by a different scheme than any considered so far. One way to do so is as follows.

From the maximum and minimum coordinate values of the intersection polygon vertices we calculate the differences $\Delta x_D = x_{\max} - x_{\min}$ and $\Delta y_D = y_{\max} - y_{\min}$. These are the x and y diameters of the polygon. Let $\vec{r}_0$ represent the chosen base point, and denote $\vec{r}_m$ one of the possibly coincident polygon

* See Figure 4

vertex points that are suspected of redundancy. Then we ask if the following relationship is satisfied:

$$|\vec{r}_m - \vec{r}_o|^2 \leq t^2[(\Delta x_D)^2 + (\Delta y_D)^2] \quad (27)$$

where t is a relative tolerance factor. If (27) is satisfied, then $\vec{r}_m$ and $\vec{r}_o$ may be regarded as coincident, and, one or the other ignored. This test is an order of magnitude comparison between the separation of the two points and the diameter of the polygon, and as with the previous tolerance checking of cross products, likewise scales with the size of the polygon.

E. Kershaw's Rotation Test

A severe machine test of an overlap algorithm as suggested by Kershaw^[3] consists of deliberately setting up two polygons which contain vertex coincidence pathologies, and then calculating their area of overlap as the combined figures are rigidly rotated through various angles about some origin. Sooner or later, because of the truncation and round off errors inherent in calculation of the sines and cosines of the rotation angle, a situation will be produced that will test all the "bad" cases the tolerance checks were designed to circumvent.

For example, consider the intersection of two triangles shown in Figure 12. Wherever the other vertex points happen to lie, a vertex-line coincidence is forced by putting into the computer program a calculation of a vertex point of one polygon (A) to lie on a side (DE) of the other.

Figure 12

Thus, we ask the computer to calculate the coordinates of A from

$$\vec{r}_A = f \vec{r}_D + (1 - f) \vec{r}_E \quad (28)$$

where f is some proportion between 0 and 1. Then we rotate this entire collection of points through some angle $\pm \alpha$ by applying the transformation

$$x' = x \cos \alpha \pm y \sin \alpha \quad (29)$$

$$y' = y \cos \alpha \mp x \sin \alpha \quad (30)$$

to each of the points. This is likewise done by the computer. What will then happen is that as successive angles are used, the locations of the points relative to each other as seen by the machine will experience jitter because of the truncation and roundoff produced in (29) and (30). For some angles α , therefore, A will lie on one side of DE, for others it will lie on the other side, and for still others it will "vanish" in the overlap calculations.

This test was used to generate the material presented in Appendix B.

IV. Overlap Area or Volume Algorithm

The overlap area algorithm developed by the author based on the preceding outlines divides naturally into two parts. The first part is basically an executive routine, which examines and reprocesses the polygons for convexity, and then, if the need arises, directs the machine to carry out the various subsequent overlaps between convex and concave parts. The second part is the overlap calculation itself, which is designed to specifically handle convex polygons.

Only the broad aspects of the algorithm will be sketched here, since it was felt no useful purpose would be served by actual programming detail; those who intend to use it are better judges in this respect of what their own needs are. However, if requested, the author will provide the CHAT FORTRAN^[5] source deck containing the algorithm which he used.

In what follows, all numbers in parentheses are equation numbers of this paper, and every indentation represents a FORTRAN DO loop.

A. Executive Routine

1. Get and store coordinates of vertices of each polygon
2. Check each polygon for concave vertices: loop over contents of vertex lists for each polygon.
 - a) Apply test (26) with negative t to vertex. If test passes, skip to f).
 - b) Apply test (26) with positive t . If test passes, skip to d).
 - c) A concave vertex is found. Save "concave triangle" and keep count.
 - d) Delete vertex from original list.
 - e) Back up one vertex and go back to a).

- f) Calculate x and y differences of adjacent vertices and save.
- 3. Calculate area or volume of overlap S_0 between first and second polygons as outlined in B, following.
- 4. If no concave elements in either polygon, return to calling routine.
- 5. Save vertices of second polygon in scratch storage.
- 6. If applicable, loop over concave triangles found for second polygon.
 - a) Fill second polygon storage slots with vertices of triangle.
 - b) Calculate area or volume of overlap as in B.
 - c) Accumulate these overlap values in S_1 .
- 7. Return vertices of second polygon to their original slots.
- 8. If applicable, loop over concave triangles found for first polygon.
 - a) Fill first polygon storage slots with vertices of triangle.
 - b) Calculate area or volume of overlap as in B.
 - c) Accumulate these overlap values in S_2 .
- 9. Loop over concave triangles of first polygon.
 - a) Fill storage slots of first polygon with saved vertices of its concave triangles.
 - b) Loop over concave triangles of second polygon
 - (i) Fill storage slots of second polygon with saved vertices of its concave triangles
 - (ii) Calculate area or volume of overlap as in B.
 - (iii) Accumulate these overlap values in S_3 .

10. Apply (15).
11. Return to calling routine.

B. Overlap Area or Volume Calculation

1. Loop over vertices of first polygon.
 - a) Loop over vertices of second polygon (i.e., the sides)
 - (i) Apply test (24).
 - (ii) If test fails, go to 1.
 - (iii) If test passes, go to (a).
 - b) Save coordinates of vertex of first polygon.
 - c) Increment intersection polygon vertex count by 1.
2. Check vertex count. If equal to number of vertices of first polygon, calculate its area from (33) or volume from (36) and return to executive routine.
3. Repeat 1. and 2. with roles of polygons reversed.
4. Loop over vertices of first polygon (i.e., the sides).
 - a) Loop over vertices of second polygon (i.e., the sides).
 - (i) Apply tests (8) and (9).
 - (ii) If either test fails, go back to (a).
 - (iii) If both tests pass, increment vertex count, apply (10), (11) and (12), and save these intersection coordinates.
5. Check vertex count to be 3 or greater. If not, set area or volume to zero, and return to executive routine.
6. The saved coordinates are vertices of the intersection polygon. Find

- maximum and minimum x and y values, and set highest vertex to be the base point (Part II, Section C).
7. Remove redundancies from base point (Part III, Section D).
 8. Loop over saved coordinates. Calculate and save polar cotangents relative to the base point. (Part II, Section C).
 9. Order vertices of intersection polygon by polar cotangent value.
 10. Calculate for intersection polygon area from (33) or volume from (36).
 11. Return to executive routine.

C. Alternatives

We cannot, like Dr. Pangloss, claim the foregoing to represent the best of all possible worlds. There are different ways of choosing and arranging the operations in the algorithm; the contents of the previous section represent one such way. Others may suggest themselves. For example, we can offer three possibilities thought of, but not carried out.

One possibility arises from the observation that many specific cross products are repeatedly calculated, as we showed, for example, in Part III, Section A. It may well be worth the initial labor to rearrange the coding so that not only are these cross products calculated just once, but the resulting values (or signs) made easily accessible in some bookkeeping scheme that optimizes efficiency of the overall calculation.

A second scheme consists of arranging the coding so there is no need to carry out the ordering of vertices as outlined in Part II, Section C. Two facts suggest this possibility. One is that the two sets of

original polygon vertices are formulated in an ordered list of vertex coordinates. The other is that in calculating side-side intersections, only one main loop was needed over the vertices of one polygon. We did not need a similar main loop over the other. (See step 4. of B., above) One could therefore keep track of the ordering of the intersection polygon vertices as they are found. The bookkeeping required to do so would be by no means trivial, but perhaps worth the effort in terms of avoiding a relatively expensive ordering procedure.

In the third possibility, consider two convex polygons that are being considered for overlap. If all of the vertices of one polygon lie off even one half plane of the other, then neither polygon will intersect the other. This procedural check might be useful in a preliminary screening to see if an intersection is at all possible. However, it may be expensive.

".....Instead of the inside of a system being perfectly hidden, it is perfectly revealed -- a white box rather than a black box.....however, because of our limitations, no box is ever entirely revealed to us, even if we construct it ourselves. Even the simplest systems sometimes contain surprises for their builders." [8]

V. Appendices

A. Area or Volume of Revolution of a Plane Polygon

In Figure 1, suppose any point $\vec{r}_0$ be marked on the plane from which radius vectors $\vec{r}_i - \vec{r}_0$ are drawn to the vertices of the N-sided polygon. Then the area of the polygon is the sum of the areas of all triangles whose sides are two adjacent radius vectors and the common polygon side between them:

$$A = \frac{1}{2} \sum_{i=1}^N (\vec{r}_i - \vec{r}_0) \times (\vec{r}_{i+1} - \vec{r}_0) \quad (31)$$

where $\vec{r}_{N+1} \equiv \vec{r}_1$. Since this holds for any $\vec{r}_0$, (31) reduces to

$$A = \frac{1}{2} \sum_{i=1}^N (\vec{r}_i \times \vec{r}_{i+1}) \quad (32)$$

or

$$A = \frac{1}{2} \sum_{i=1}^N (x_i y_{i+1} - x_{i+1} y_i) \quad (33)$$

Suppose now this polygon is rotated about the x-axis, sweeping out a toroid whose cross-section is the polygon. The volume of this toroid is likewise the sum of the volumes of all the toroids of triangular section which were also swept out. From the theorem of Pappus and (31), we obtain

$$V = 2\pi \sum_{i=1}^N \frac{1}{3} (y_{i+1} + y_i + y_0) \cdot \frac{1}{2} (\vec{r}_i - \vec{r}_0) \times (\vec{r}_{i+1} - \vec{r}_0) \quad (34)$$

which, since $\vec{r}_0$ and y_0 are arbitrary, reduces to

$$V = \frac{\pi}{3} \sum_{i=1}^N (y_{i+1} + y_i) (\vec{r}_i \times \vec{r}_{i+1}) \quad (35)$$

or

$$V = \frac{\pi}{3} \sum_{i=1}^N (y_{i+1} + y_i)(x_i y_{i+1} - x_{i+1} y_i) \quad (36)$$

B. Numerical Example of a Vertex-Line Pathology

We illustrate the concepts in Part II by a numerical case encountered when Kershaw's test* was applied to a configuration similar to those shown in Figures 10 and 12. This calculation was done on the CDC 7600 computer.⁽⁶⁾ In that machine⁽⁷⁾, floating point numbers are represented by a string of 60 binary bits (20 octal places) where the leftmost 12 bits (4 places) contain sign and exponent values, and the rightmost 48 bits (16 places) contain the value of the number. The least significant bits are the ones at the right.

Figure 13 shows schematically the relevant vertex-line geometry. The labels i and j differentiate two polygons. Vertex $\vec{r}_i$ was calculated so as to lie on the side between $\vec{r}_j$ and $\vec{r}_{j+1}$.

Figure 13

This configuration was successively rotated through small angular increments, and each time the overlap area calculated. At several such times,

* Part III, Section E.

vertex-line logical damage occurred where the vertex at $\vec{r}_j$ seemingly "vanished" from consideration. For one such situation, the numerical values for the coordinates of the vertices in the computer were as shown in Table I, where both conventional and octal representations are given, corresponding to the easily decipherable FORTRAN symbols.

TABLE I

COORDINATES OF VERTICES		
X(J)	1.72784683	17206722502561613577
Y(J)	1.00724640	17204016656317104113
X(J+1)	-1.45876132	60572124351710036037
Y(J+1)	2.78604163	17215444720144013516
X(I-1)	0.56174949	17174374732050542670
Y(I-1)	1.02197725	17204055004643230631
X(I)	0.66564411	17175246364707571024
Y(I)	1.60017815	17206315124314612421
X(I+1)	-0.18671843	60622014634211070133
Y(I+1)	1.51166671	17206027711332451360

In Table II, on the next page, the resulting factors and cross products are shown corresponding to those in Eqs. (6) and (7) for vertex $\vec{r}_i$ relative to side $\vec{r}_{j+1} - \vec{r}_j$, as well as those for corresponding terms in Eqs. (8) and (9) which test the intersection of the two sides $\vec{r}_{i+1} - \vec{r}_i$ and $\vec{r}_{j+1} - \vec{r}_j$. In this table we also show the FORTRAN combination actually used to achieve the numbers given. In the interests of clarity, the results for the intersection of sides $\vec{r}_i - \vec{r}_{i-1}$ with $\vec{r}_{j+1} - \vec{r}_j$ are not given, as they were identical to the intersection result shown.

TABLE II

EQUATION AND FORTRAN SEQUENCE	CALCULATED VALUE	
L.H.S. OF EQ. (7) DX(J)=X(J+1)-X(J) P=DX(J)*(Y(I)-Y(J))	-1.88944113	60570342331302763622
R.H.S. OF EQ. (7) DY(J)=Y(J+1)-Y(J) Q=DY(J)*(X(I)-X(J))	-1.88944113	60570342331302763625
L.H.S. OF EQ. (6) CP=P-Q	-2.13E-14	6135177777777777777
L.H.S. OF EQ. (8) DA(J)=X(J)*Y(J+1)-X(J+1)*Y(J) CP=DX(J)*Y(I)-DY(J)*X(I)+DA(J)	0.	00000000000000000000
R.H.S. OF EQ. (8) CP=DX(J)*Y(I+1)-DY(J)*X(I+1)+DA(J)	1.79822972	17207142614413056604
L.H.S. OF EQ. (9) DA(I)=X(I)*Y(I+1)-X(I+1)*Y(I) CP=DX(I)*Y(J)-DY(I)*X(J)+DA(I)	}	(NOT CALCULATED BECAUSE TEST AT (8) FAILED.)
R.H.S. OF EQ. (9) CP=DX(I)*Y(J+1)-DY(I)*X(J+1)+DA(I)		

From Table II we see that not only did the tests at (6) and (7) fail, but so did the test at (8), since both sides of (8) were regarded as positive. (One might argue that an identically zero value be regarded as "neutral" and coding put in accordingly. The trouble with this is that the CDC 7600 has two 60-bit representations for a floating point zero^[7], each of which may be interpreted as "+0." and "-0.", and so nothing is gained thereby.) Note also that the left side of (7) differs from the right side in the last three binary bit locations (2 is 010 and 5 is 101).

TABLE III

EQUATION AND FORTRAN SEQUENCE	CALCULATED VALUE	
L.H.S. OF EQ. (24)		
TOL=1.E-14		
DX(J)=X(J+1)-X(J)	-1.88944113	60570342331302763641
P=DX(J)*(Y(I)-Y(J)+TOL*DX(J))		
R.H.S. OF EQ. (24)		
DY(J)=Y(J+1)-Y(J)	-1.88944113	60570342331302763620
Q=DY(J)*(X(I)-X(J)-TOL*DY(J))		
L.H.S. OF EQ. (25)		
CP=P-Q	1.21E-13	16454200000000000000

Finally, in Table III, we show the effects of applying tolerance checking to this situation. Instead of tests at (6) and (7), the corresponding tests at (25) and (24) were applied, with a tolerance factor of $t = 10^{-14}$. Now, the vertex at $\vec{r}_i$ was "calculated" to lie on the inside of the polygon, and $\vec{r}_i$ was made a vertex of the intersection polygon. The effects of this dummy shift are clearly evident in the last two places of the octal representation.

References

1. S. I. Warshaw, Areal Mapping of Arbitrary Plane Regions onto Rectangular Meshes, March 1977, UCID-17415 (formerly UCIR-726, August 1973).
2. R. V. Benson, Euclidean Geometry and Convexity, McGraw-Hill, 1966.
3. D. Kershaw, LLL, Private Communication.
4. B. Wendroff, First Principles of Numerical Analysis, Addison-Wesley, 1969.
5. J. T. Martin, R. G. Zwakenberg, S. V. Solbeck, LRLTRAN Language Used with the CHAT and STAR Compilers, LTSS Part III, Chapter 207, Edition 3, July 1973.
6. Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Energy Research & Development Administration to the exclusion of others that may be suitable.
7. CDC 7600 Hardware Reference Manual, 1972.
8. G. M. Weinberg, An Introduction to General Systems Thinking, Wiley, 1975.

:ps

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research & Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights.

NOTICE

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Energy Research & Development Administration to the exclusion of others that may be suitable.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy \$: Microfiche \$3.00

Page Range	Domestic Price	Page Range	Domestic Price
001-025	\$ 3.50	326-350	10.00
026-050	4.00	351-375	10.50
051-075	4.50	376-400	10.75
076-100	5.00	401-425	11.00
101-125	5.50	426-450	11.75
126-150	6.00	451-475	12.00
151-175	6.75	476-500	12.50
176-200	7.50	501-525	12.75
201-225	7.75	526-550	13.00
226-250	8.00	551-575	13.50
251-275	9.00	576-600	13.75
276-300	9.25	601-up	*
301-325	9.75		

*Add \$2.50 for each additional 100 page increment from 601 to 1,000 pages;
add \$4.50 for each additional 100 page increment over 1,000 pages.

Technical Information Department

LAWRENCE LIVERMORE LABORATORY

University of California | Livermore, California | 94550