SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

092001

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina USA 27709

SPLA CONTROL NUMBER: 222

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the SHARE Reference Manual, Section 6.

(1) Program Number (to be filled by SPLA) 360D-09.2.001

(2) Title of Program . . . . . SLAC ENHANCEMENTS TO IBM FORTRAN MOD II LIBRARY

(3) System Type(s) (Machine) . . . . . SYSTEM 360, SYSTEM 370

(4) Search Key(s) . . . . . ERROR MONITOR, ERROR HANDLING, TRACEBACK, ABEND HANDLING, INTERRUPTION HANDLING.

(5) Programming Systems/Languages . . . . FORTRAN, ASSEMBLER LANGUAGE

(6) Primary Subject Code 09.2 (04.2, 41.0, 02.0)

(7) Minimum System Requirements Ability to run Fortran Mod II Library

(8) New (N) or Revision (R) (if revision, show prior Program Number in Item 1) N

(9) Date of Submittal . . . . . 06 MARCH 1978

(10) Documentation (number of original pages submitted) 1

(11) Author's Name and Address . . . . . Dr. John R. Ehrman
SLAC Computing Services
Mail Bin 97
P.O. Box 4349
Stanford, CA 94305

(12) Direct Technical Inquiries to Name & Address (if different than Author)

(13) Submitter's Installation Membership Code . . . . . SLA

(14) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

SPLA 1001 Revised 11-76

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. Purpose
b. Programming Language used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

These enhancements to IBM's Fortran Mod II Library provide (1) improved diagnostics for ABEND handling by module IHOSTAE; (2) accurate and detailed traceback information, including the parameters passed to each routine; (3) expanded information for program interrupts giving the general and floating-point registers and an added diagnostic line; (4) more uniform application of standard linkage conventions in twelve of the mathematical service routines; and (5) various other minor improvements.

When installed, these modified routines allow users of Fortran to detect and diagnose their errors more easily and rapidly without needing as much expert assistance.

The distribution tape is a complete update and test package. The source for the Fortran Mod II Library must be obtained from IBM under a standard Program Product license.

(Please attach additional pages if necessary) . . . . . Total pages attached _____

An "Acknowledgement of Assistance" statement must be attached to this Submittal Form.

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program"

(15) Signature of Submitter and Date John R. Thomas / 06 March 1978

(15) Signature of Installation Addressee _____

SPLA 1001 (R) Revised 11-76

SLAC Modifications to IBM Fortran Mod II Library

Description of Distribution Tape

The SLAC mods to the IBM Fortran Mod II Library are contained in 5 files on a 9-track 1600 BPI tape, with standard labels, volid=SLAL22. All files except the first are Partitioned Data Sets unloaded by IEHMOVE. The first file is a card-image data set describing the tape, and gives complete installation instructions. The DCB parameters for the first file are RECFM=FB, BLKSIZE=3120. The files and their contents are:

File	Cards	Dsname	Contents
1	234	WYL.SF.JRE.LIB22DES	Installation description
2	---	WYL.SF.JRE.LIB22UTL	Utility update program IEBUPDTX
3	---	WYL.SF.JRE.LIB22UPD	Updates to IBM source
4	---	WYL.SF.JRE.LIB22LNK	Input cards for Linkage Editor
5	---	WYL.SF.JRE.LIB22TST	Test programs for new library

Including the standard end-of-tape indication, there are 16 tape marks on the distribution tape.

To read the first file, run a job like the following:

```
//jobname  JOB  account
// EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD SYSOUT=A,DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSUT1   DD UNIT=TAPE-1600,VOL=SER=SLAL22,DISP=(,KEEP),
//            LABEL=(1,SL),DSN=WYL.SF.JRE.LIB22DES,
//            DCB=(RECFM=FB,LRECL=80,DEN=3,BLKSIZE=3120)
```

Direct technical inquiries and suggestions to

John R. Ehrman (SLA)
SCS-SCIP (Mail Bin 97)
Stanford Linear Accelerator Center
P. O. Box 4349
Stanford, California 94305
(415) 854-3300 ext. 2631

INTERSECTION DETECTION IN THREE DIMENSIONS

A TOOL FOR COMPUTER AIDED

ENGINEERING DESIGN AND GRAPHIC DISPLAY

Paul G. Comba

December 1967

SHARE Program Library Agency
Triangle Universities Computation Center
P. O. Box 12076
Research Triangle Park, N. C. 27709

Direct all inquiries to:

Mr. P. G. Comba
IBM Cambridge Scientific Center
545 Technology Square
Cambridge, MA 02139

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The author expresses his gratitude to the following individuals:

Dr. J. Greenstadt and Dr. R. T. Mertz, whose suggestions were most helpful in reaching a formulation of the pseudo-characteristic function and selecting an optimizing procedure;

Mr. R. H. Bullen, Jr., who programmed most of the test cases and helped produce the final version of the program.

PREFACE

This manual is divided in three parts.

The first part, User Information, contains a general description of the purpose and organization of the ID/3D program, as well as detailed external specifications. A close study of Chapter III, Input/Output, is essential in order to use the program. It is also recommended that the example in section V.B. be examined in detail. The chapter on Operating Characteristics contains storage and timing information that may be useful when ID/3D is to be used in conjunction with other programs. The information in Chapter VI, Details of Mathematical Method, is of theoretical interest; it is not essential to the use of ID/3D.

The second part of the manual, Operating Instructions, explains how the ID/3D program, as well as two sample problems, can be obtained from the program tape, and how they are to be run.

The third part, Systems Material, contains the internal specifications of the program. This information is provided for those who wish to understand in detail and/or modify or extend the code.

USER INFORMATION

I. INTRODUCTION

A. GENERAL

ID/3D is an experimental program that was originally designed to provide a solution to a problem of fundamental importance in computer aided engineering design, i.e., the problem of detecting intersections of 3-dimensional objects. The techniques developed for this purpose were later found to be also applicable to a problem arising in the processing of images of 3-dimensional objects: the hidden line problem.

The following subsections of this introduction discuss in general terms the two problems in question, and the later sections of this document describe the organization and operation of ID/3D.

The experimental nature of the program is reflected in the fact that an attempt was made to incorporate those features that would make it possible to test the validity of the approach, and enable the user to experiment with various ways of using the program. On the other hand, a number of features were considered but omitted when it was felt that, however desirable they might be in a production program, their later incorporation would not present serious conceptual problems.

2. INTERSECTION DETECTION IN ENGINEERING DESIGN

One of the major and recurring problems faced by the designer of physical systems is the placement problem. The solution of this problem in its simplest form consists of a specification of the position of each object in the system such that all objects lie in a certain region of space and no two objects occupy the same space.

The restriction that two objects cannot occupy the same space at the same time is, of course, an expression of the impenetrability of physical bodies. As such it underlies all macroscopic physical systems. By contrast most of the other design constraints are more closely related to the purpose of a particular system, the environment where it must operate, and the production techniques that are available. For example, the design of a jet engine will involve considerations of mechanical stress, thermal stress, vibration resistance, accessibility of components, manufacturing limitations, etc.

Besides being aware of constraints, the systems designer aims at optimizing certain criteria that may be more or less explicitly defined and measurable. At this point, the simple formulation of the placement problem given above is no longer adequate. It often happens that while the major functional components of a system have a relatively fixed shape at an early design stage, many of the other "components" to be placed consist of pipes, cables and other connectors whose shapes are not predefined but can vary continuously in terms of many parameters. The design solution is then typically reached by an iterative process involving a certain

amount of trial and error and also involving countless changes. And for each proposed alternative one must verify that it does not violate any constraints.

The problem of verifying that the plan of a system satisfies the relevant spatial constraints varies in complexity depending on how one represents the objects in the system. If one uses a scale model, the no-overlap condition is trivial. The problem is not so easy with scale drawings, where three-dimensional objects with curved surfaces have to be represented on a two-dimensional sheet of paper. And it becomes increasingly difficult when one has to work with mixed graphic-symbolic representations like sketches, schematics and diagrams, or purely symbolic ones like equations, connection matrices, etc. In the development of a complex system all these methods of representation may be appropriate at different stages and for different purposes; the more symbolic methods, however, play a larger role in very complex systems, since they are the ones that lend themselves to mathematical treatment and computer processing.

If a system design project involves many individuals or groups, the problems of control and documentation (what space belongs to whom, and who has placed what where) assume major importance.

In summary, an automatic procedure for detecting intersections of three-dimensional objects becomes increasingly desirable as one moves

a) from simple to complex systems

b) from loosely packed to closely packed systems

c) from individual to group projects
d) from design problems admitting direct solutions to those requiring an iterative approach
e) from a physical to a graphic to a symbolic representation of the system design
f) from special purpose to general purpose systems design tools.

C. THE HIDDEN LINE PROBLEM

This problem occurs in the execution (whether manual or automatic) of line drawings of 3-dimensional objects. The lines appearing in such drawings may be either intersections between two surfaces bounding an object, or "edges", i.e., lines where the line of sight is tangent to a bounding surface. The edges of an object may be hidden from view by another object, and the intersections may be hidden by another object or by the object itself. The problem is to determine which portions of these lines are hidden, so they can be omitted from the drawing or shown as dotted lines. This is in general a difficult question. The ID/3D program provides a new approach to its solution.

INTERSECTION DETECTION IN THREE DIMENSIONS

USER INFORMATION

II. OVERALL DESCRIPTION

A. FACILITIES PROVIDED BY THE SYSTEM

ID/3D is a system of FORTRAN subroutines which provide facilities for executing the following procedures:

1. Defining 3-dimensional objects bounded by planes or quadric surfaces (e.g., a parallelepiped, a hemisphere, a truncated cone). Each object is defined in terms of a set of built-in objects, known as components, by specifying appropriate parameters that determine their position, scale and attitude (i.e., orientation in space).

2. Defining line segments in 3-space by specifying their end points.

3. Testing whether any two objects intersect, i.e., have any points in common, or whether a segment and an object intersect. It is also possible to execute a multiple test between one object (or segment) and all other objects.

4. Removing previously defined objects or segments.

5. Specifying a region of space where the intersection test is to be executed (see note [1]), a tolerance parameter

that controls the precision with which the test is carried out, and an output level parameter to selectively suppress the printing of information generated by the system.

Each object defined in the system can be either a single component or an intersection of components. The reasons for and implications of this method of definition are discussed in [2].

The input to ID/3D can be either through data cards, where each card contains a keyword as well as alphanumeric parameters, or via subroutine calls. The latter method is of course much more flexible, since it makes possible the removal and redefinition of objects or segments based on the results of previous intersection tests or other calculations.

The parameters mentioned in 5 above are set by the system at initialization time, and the user does not have to specify them unless he wants to change the initial settings.

The system does a certain amount of error checking on the input, but there are certain forms of invalid input (detailed in later sections) that will give rise to incorrect output.

B. THE MATHEMATICAL METHOD

The intersection test is based on the use of a pseudo-characteristic function by means of which the mathematical ex-

pressions that represent the several bounding surfaces of the objects to be tested are combined into a single function. This function has the property that, if the two objects have a non-empty intersection, it is negative in a region that closely approximates that intersection, and positive outside; if the two objects do not intersect, the pseudo-characteristic function is always positive. The test is thus reduced to deciding whether there is any points where the p-c function is negative. This is done by means of a descent method described by Fletcher and Powell [3]. For the purposes of this test it is usually not necessary to obtain the minimum of the p-c function because:

a) as soon as a negative value of the p-c function is obtained, the test can be terminated;

b) it is possible to compute a lower bound to the values of the function; if this lower bound is positive, the test can be terminated.

In the execution of the intersection test the method outlined above is preceded by a gross test that relies on the construction of a sphere containing each object, and a comparison of the distance between the centers of the spheres and the sum of their radii.

Although the p-c function method is inherently approximate, the user can control the degree of approximation by specifying the tolerence parameter. The method works reliably and efficiently to within tolerances of 10^{-4} to 10^{-5} times the linear dimension of the objects.

For a discussion of some limitations or the p-c function method see [4].

C. SYSTEM ORGANIZATION

Each box in Figure 1 represents one subroutine except the box marked "ID26" which represents 26 separate subroutines. The function of the several subroutines is briefly described below.

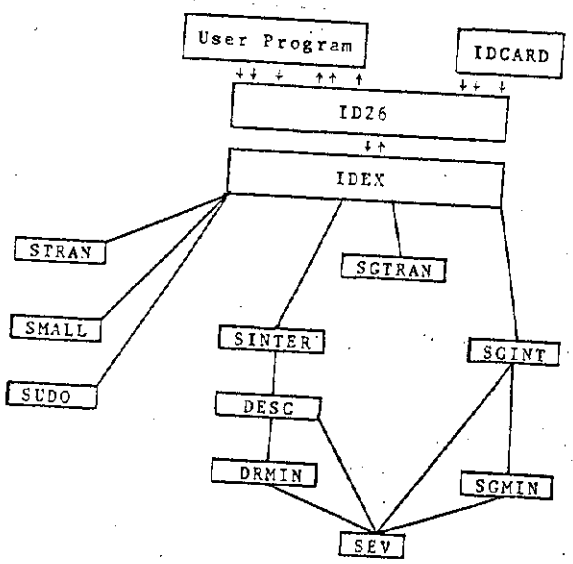

Figure 1. System Organization

IDCARD is used for reading card input. It must be called from another program (e.g. a dummy main program) but it does not communicate with it via parameters. IDCARD reads the input cards, interprets the keyword in each card, and

calls the appropriate ID26 routine. When the input to the ID/3D system comes directly from the user program, the subroutine IDCARD need not be used (or even loaded), and the user program calls the various ID26 subroutines.

The ID26 subroutines are listed individually in the next section. Their mnemonic names give an indication of their specific functions. Globally they serve the purpose of passing information between the user program and the subroutine IDEX.

IDEX is the central routine. It does most of the housekeeping functions, including initialization, setting of global parameters, error checking, and most of the output. It also updates the tables of objects names and the lists of bounding surfaces for the individual objects.

The other subroutines are called by IDEX for the execution of specific tasks and can be grouped into four classes.

1. OBJECT DEFINITION

The external parameters that specify the position, scale and orientation of an object are translated into internal parameters, suitable for rapid computation of the pseudo-characteristic function, by STRAN. The subroutines SMALL and SUDO compute the center and radius of a sphere enclosing the object.

2. INTERSECTION TEST BETWEEN OBJECTS

The subroutine SINTER executes a gross test by checking whether the spheres containing the objects overlap. If there is

no overlap, the test is done. If there is overlap, the p-c function method must be used. SINTER then sets up the parameters needed for the computation of the p-c function and calls DESC. DESC is the descent procedure by means of which the minimum of the p-c function is approached. This procedure involves the computation of the minimum in specified directions, which is done by DRMIN. The subroutine SEV evaluates the p-c function and its gradient.

3. SEGMENT DEFINITION

SGTRAN computes the internal parameter values of the segment being defined.

4. SEGMENT-OBJECT INTERSECTION

SGINT does the gross geometric checking and SGMIN computes the minimum of the p-c function along the segment.

III. INPUT/OUTPUT

A. GENERAL

When input is to be read from cards, each card must contain a keyword consisting of 3 or 4 alphanumeric characters. When input is transmitted via subroutine calls, one subroutine call corresponds to one card of input. The names of the subroutines consist of the letters ID followed by one of the keywords.

Table 1 contains a list and grouping of all keywords.

The form and meaning of the I/O will first be described in detail with regard to subroutine calls, then in summary form with regard to card input. The type of the I/O parameters (integer or real) is indicated by their initial letter according to the usual I-through-N convention.

B. PROGRAM CONTROL

IDINIT (T1)

A call to IDINIT must precede all other calls. This subroutine initializes the system and sets all the necessary parameters. In particular it defines the region for the intersection tests as a sphere with center at the origin and radius

INIT	initialization	program control
END	end	
OUT	output option	user parameters
REG	region definition	
TOL	tolerance	
HSP	half-space	object definition
SLA	slab	
SPH	sphere	
ELL	ellipsoid	
CYL	cylinder	
CON	cone	
PAR	paraboloid	
HYP	hyperboloid	
BOX	box	
SEG	segment definition	segment definition
CLR	clear all objects, segments	object and segment removal
REMO	remove object	
REMS	remove segment	
TOS	test object single	test request
TSS	test segment single	
TOM	test object multiple	
TSM	test segment multiple	
PR1		system parameters
PR2		
PR3		
PR4		

Table 1. Keywords

10., and the tolerance as .001. The parameter T1 specifies the output level as follows:

-1. No printed output.

0. Standard printed output. If the user wishes to have printed output, this is the option that should be used. The output will contain a record of the initial values and any subsequent changes in user parameters, of all definitions and removals of objects and segments, and of the results of all the intersection tests that are executed. All pertinent error messages will also be printed.

1.
2.
3. Extended printed output. These values of T1 result in progressively more detailed output relating to the internal operation of the ID/3D system. They are useful mainly for system testing and performance evaluation.

Values of T1 other than the above will be equated to one of the above. Regarding suppression of system messages see [8].

IDEND

A call to this subroutine may be used to force the printing of any pending output relating to an object definition (see section on Object Definition - Naming of Objects). With card input the keyword END signals the end of the input and should appear only in the last input card.

C. USER PARAMETERS

IDOUT (T1)

This subroutine is used for changing the value of the printed output parameter. The meaning of T1 is the same as in the case of the IDINIT subroutine.

IDREG (T1, T2, T3, T4)

This subroutine defines a spherical region within which the interference tests are executed. T1, T2, T3 specify the coordinates of the center of the sphere, and T4 specifies its radius. T4 should be positive, but this condition is not checked for. Only the intersections that occur within the region are detected by ID/3D.

IDTOL (T1)

This subroutine sets the value of the tolerance parameter. If the value of T1 is not positive, a call to IDTOL does not change the previous value of the tolerance parameter.

D. OBJECT DEFINITION - NAMING OF OBJECTS

As explained earlier, an object can be either a single component or the intersection of several components. To define an object requires one subroutine call for each component. The list of component subroutines follows:

IDHSP (J,N1,N2,T1,T2,T3,T4,T5,T6)
IDSLA (J,N1,N2,T1,T2,T3,T4,T5,T6,T7)
IDSPH (J,N1,N2,T1,T2,T3,T4)
IDELL (J,N1,N2,T1,T2,T3,T4,T5,T6,T7,T8,T9)
IDCYL (J,N1,N2,T1,T2,T3,T4,T5,T6,T7,T8,T9)
IDCON (J,N1,N2,T1,T2,T3,T4,T5,T6,T7,T8,T9)
IDPAR (J,N1,N2,T1,T2,T3,T4,T5,T6,T7,T8,T9)
IDHYP (J,N1,N2,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10)
IDBOX (J,N1,N2,T1,T2,T3,T4,T5,T6,T7,T8,T9)

The parameters N1 and N2 are used for naming objects. (These parameters are integers, but in the printed output they are printed under the format A4. The reason for this, and for the double name, is to maintain compatibility with card input, where names of up to 8 characters are allowed. Since BPS FORTRAN does not handle alphanumeric variables, a subroutine named KAL has been provided to facilitate the conversion between EBCDIC codes and integers. See [5]). If in a call to a component subroutine the values of N1 and N2 are either 0 or the numeric equivalent of four blanks (i.e. 1077952576), the call is said to reference a null name. Now to define an object the first call to a component subroutine must reference a non-null name, which becomes the name of the object, and subsequent calls (in the case of multi-component objects) must reference a null name. Different objects should have different names, but the system does not check for this condition.

The parameter J is a return parameter. The normal return is J = 4. Any of the conditions listed below will result in a return J = 5, and the call will have no effect on

the state of the system. In the printed output the specific error condition is stated.

a) Unnamed object. This occurs when a component subroutine call with null name follows a non-component subroutine call.

b) Too many objects in the system.

c) Too many subcomponents for a given object.

d) Too many subcomponents in the system.

The last three conditions relate to the sizes of the internal tables and are specified in the Section IV.A.

A return J = 6 indicates a system error; this should never occur if the system is operating normally.

Because of the way the system handles object definitions and the fact that several calls to component subroutines may be needed to define a single object, the end of an object definition is recognized only when a subsequent call either to a component subroutine with a non-null name or to a non-component subroutine is executed. Therefore, any output relating to the definition of an object is delayed until the execution of that subsequent call. If the program that calls the ID/3D subroutines produces output of its own, the different outputs will not appear in the desired order. To avoid this and force the printing of the ID/3D output as soon as an object definition is complete, the calling program should contain a call to IDEND right after the calls that define the

object. (The only exception to the above rule about delayed output is that a call to IDINIT results in immediate reinitialization of the system.)

E. OBJECT DEFINITION - GEOMETRY OF COMPONENTS

The real parameters T1 through T10 specify the position, scaling and attitude of the several components. These attributes are specified with respect to an implied absolute coordinate system. (No transformations of coordinates are possible in ID/3D. This of course is not a restriction on the geometric shapes that can be described, but only on the manner in which they are described.)

The components are illustrated in Figure 2.

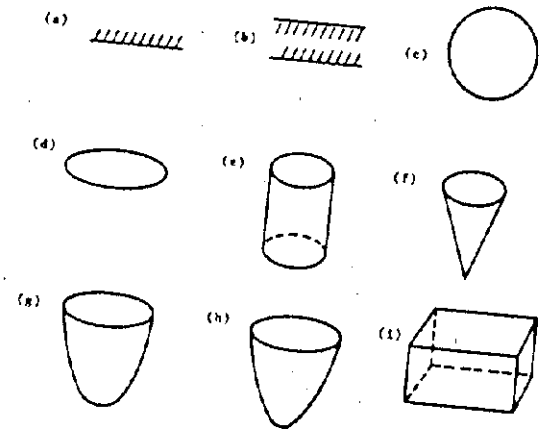

Figure 2. Geometric Components for Construction of Objects in the ID/3D System.

(a) half-space
(b) slab, i.e., region bounded by two parallel planes
(c) sphere
(d) ellipsoid
(e) right elliptic cylinder
(f) right elliptic cone
(g) segment of elliptic paraboloid
(h) segment of one sheet of elliptic hyperboloid
(i) rectangular parallelepiped

The following remark applies to all components: the parameters T1,T2,T3 always specify the X-,Y-,Z- coordinates of a point related to the component.

The following remarks apply to all components except the first three:

(a) These components can be rotated in space by specifying in T7, T8 and T9 three Eulerian angles [6], expressed in degrees; this rotation has as center the point (T1, T2, T3). A procedure to bypass the specifications of Eulerian angle is described in [7].

(b) The components are described below in their "standard attitude", i.e. before they are rotated.

(c) The terms "vertical" and "down" are used as abbreviations for "parallel to the Z-axis" and "toward the negative Z direction".

(d) The segments whose lengths are designed by T4,T5, T6 are in the X-,Y-,Z- direction respectively when the component is in the "standard attitude".

HSP

	<u>half-space</u>
T1,T2,T3	on bounding place
T4,T5,T6	normal vector pointing into solid

SLA

	<u>slab</u>, i.e. region between two parallel planes
T1,T2,T3	on mid-plane
T4,T5,T6	vector normal to planes
T7	half-thickness

SPH	sphere
T1,T2,T3	center
T4	radius
ELL	ellipsoid
T1,T2,T3	center
T4,T5,T6	half-axes
T7,T8,T9	Eulerian angles
CYL	cylinder, i.e. the intersection of an infinite cylinder with vertical axis and elliptical cross section, and a horizontal slab
T1,T2,T3	center
T4,T5	half-axes of base
T6	half-height
T7,T8,T9	Eulerian angles
CON	cone, i.e. intersection of an infinite cone with vertical axis and elliptical cross section, and a horizontal slab going through the origin and lying in the region $Z \geq 0$; the vertex of the cone is down
T1,T2,T3	vertex
T4,T5	half-axes of base
T6	height
T7,T8,T9	Eulerian angles

PAR	paraboloid, i.e. intersection of an infinite paraboloid with vertical axis and elliptical cross section, and a horizontal slab going through the origin and lying in the region $Z \geq 0$; the vertex of the paraboloid is down
T1,T2,T3	vertex
T4,T5	half-axes of base
T6	height
T7,T8,T9	Eulerian angles
HYP	hyperboloid, i.e. intersection of one sheet of a two-sheeted hyperboloid with vertical axis and elliptical cross section, and a horizontal slab lying in the region $Z \geq 0$; the vertex of this sheet is down with respect to the base; the center of the hyperboloid is down with respect to the vertex
T1,T2,T3	vertex
T4,T5	half-axes of base
T6	height
T7,T8,T9	Eulerian angles
T10	distance from center to vertex
BOX	box, i.e. rectangular parallelepiped
T1,T2,T3	center
T4,T5,T6	half-edges
T7,T8,T9	Eulerian angles

F. OBJECT DEFINITION - ERROR CONDITIONS

The following conditions must be observed in defining components:

re HSP,SLA: the three components T4,T5,T6 of a direction vector must not all be 0.

re all: a parameter that represents a length (thickness, radius, half-axis, height, distance) must be positive.

These conditions are not tested for by the system, and their violation will result in serious errors.

G. SEGMENT DEFINITION

IDSEG(J,N1,N2,T1,T2,T3,T4,T5,T6)

The parameters N1,N2 specify the name of the segment, which must be a non-null name. J is the return parameter. Normal return is J = 4. Error return is J = 5, which will occur if a null name is referenced, or if too many segments have been defined (see section on Program Limits). T1,T2,T3 are the coordinates of one end point of the segment and T4,T5,T6 the coordinates of the other end point.

H. OBJECT AND SEGMENT REMOVAL

IDCLR

A call to this subroutine cancels all previously entered object and segment definitions.

IDREMO

A call to this subroutine cancels the definition of the last object that has been defined.

IDREMS

A call to this subroutine cancels the definition of the last segment that has been defined.

If there are no objects or segments defined in the system, a call to one of the above subroutines results in no action. In the case of IDREMS and IDREMO a printed message is given.

I. TEST REQUESTS

IDTOS (J,N1,N2,N3,N4,X,Y,Z,F)

A call to this subroutine causes the execution of an intersection test between two objects. N1,N2 is the name of the first object, and N3,N4 the name of the second. Both objects must have been previously defined.

J is a return parameter that indicates the outcome of the test.

J = 1 The objects intersect.

J = 2 The system cannot decide whether the objects intersect. This return (which does not normally occur) indicates that the number of steps necessary to complete the test has exceeded a prespecified bound.

J = 3 The objects do not intersect.

J = 5 This is an error return meaning that one of the objects referenced in the call has not been defined. No test is executed in this case.

J = 6 System error

The return parameters X,Y,Z,F are the coordinates of the point where the pseudo-characteristic function has last been evaluated, and the value of the function. When the objects intersect, the point lies inside the intersection of the objects, and F must be < 0. In the other cases the point has no geometric significance.

IDTSS (J,N1,N2,N3,N4,X,Y,Z,F)

A call to IDTSS causes the execution of an intersection test between the segment N1,N2 and the object N3,N4. The meaning of the other parameters is analogous to that of the IDTOS parameters.

IDTOM (J,N1,N2,N3,N4,X,Y,Z,F)

A call to IDTOM causes the execution of a sequence of intersection tests. The object N1,N2 is successively tested against all other objects in the system, taken in the order in which they have been defined, until either

a) it is found to intersect some other object (or be "undecidable" with respect to it); or

b) it is found not to intersect any other object.

Note that in case (a) the sequence of tests is terminated as soon as an intersection (or undecidable case) is detected.

The meaning of the return parameters is as follows:

J = 1 The object N1,N2 intersects some other object. The name of the latter is returned in N3,N4. The return parameters X,Y,Z,F contain the coordinates of a point lying in the intersection and the value of the p-c function there.

J = 2 The system cannot complete the test. The name of the "undecidable" object is returned in N3,N4.

J = 3 The object N1,N2 does not intersect any other object. (This is also the return when N1,N2 is the only object that has been defined; this result is vacuously true since no test is executed.)

J = 5 Error return. The object N1,N2 has not been defined.

J = 6 System error

IDTSM (J,N1,N2,N3,N4,X,Y,Z,F)

This causes the execution of a sequence of intersection tests between the segment N1,N2 and all the objects that have been defined. The execution of the test and the meaning of the other parameters is analogous to that of the IDTOM subroutines.

J. PROGRAM PARAMETERS

All parameters used by ID/3D for defining and testing objects and segments are set at initialization time. Their values may be changed if desired, by means of calls to

IDPR1 (T1,T2,T3,T4,T5,T6,T7,T8,T9,T10)

and similar calls to IDPR2, IDPR3 and IDPR4. For card input, the keywords PR1, PR2, PR3 and PR4 are used. Although the calls must reference 10 parameters, fewer than 10 are actually used. The values of the parameters in the calls must be real, and these values are converted by the program to either integer or double precision form.

The only program parameter that may be of direct interest to the user is T6 in IDPR1. If T6 is set to 0. (which is the initial setting), the intersection test is executed in the normal manner. If this parameter is set to 1., the program will find the minimum of the p-c function for the particular object-object or segment-object pair, regardless whether there is intersection or not. The value of this minimum gives an indication of the distance or the extent of overlap between the things being tested: a large positive value indicates a large distance, and a large negative value indicates a large overlap.

When changing this parameter, the user should specify the standard initial values for the remaining ones thus:

CALL IDPR1 (60.,3.,60.,25.,4.,T6,0.,0.,0.,0.).

The other program parameters are discussed in the section on Systems Material.

K. CARD INPUT

The subroutine IDCARD, which has no parameters, is used for reading input from cards. This subroutine reads cards in sets of 50 or until an END card appears, prints the input that it has read, then executes it. If a card contains an invalid keyword it is ignored; otherwise, each card results in one subroutine call. Cards are read under control of the format (5A4,10F6.3). The correspondence between card fields and input parameters is as follows:

cc	
1-8	N1,N2
9-12	keyword
13-20	N3,N4
21-26	T1
27-32	T2
33-38	T3
39-44	T4
45-50	T5
51-56	T6
57-62	T7
63-68	T8
69-74	T9
75-80	T10

When the subroutine corresponding to a given keyword does not require all the parameters listed above, the extra parameters (as read from a card) are ignored; those card fields may be left blank. There are, of course, no card fields corresponding to the output parameters. For example, with the keywords TOS or TSS the parameters N3,N4, must be supplied in the input card because they correspond to input

parameters of IDTOS or IDTSS; vice versa with TOM or TSM the field in cc 13-20 is ignored because N3,N4 are output parameters of IDTOM or IDTSM.

The keywords must be left justified in the keyword field, and the object names (including blanks) must appear in exactly the same form in successive references.

IV. OPERATING CHARACTERISTICS

A. STORAGE REQUIREMENTS AND PROGRAM BOUNDS

The program is written in FORTRAN and requires at least 128K bytes of storage to compile and execute. The source deck consists of approximately 2300 statements.

The approximate core requirements (in bytes) for the main subdivisions of the program are as follows:

	BPS	OS FORTRAN G
IDCARD, KAL, and IDEU	10,100	10,200
ID26	13,300	13,400
IDEX and its subroutines	46,500	50,900
COMMON	20,900	20,900
	90,800	95,400

The subroutines IDCARD, KAL, and IDEU need not be loaded if they are not used. The subroutines in the group ID26 may also be selectively loaded if desired.

To explain some of the program bounds the concept of subcomponent must be introducted. Some of the previously defined components (e.g., the sphere) are bounded by a single surface and represented internally by a single algebraic expression, while other components require several algebraic expressions; for example, the cylinder is bounded by

two surfaces: an infinite cylinder and a slab that intersects it. We then say that the sphere has one subcomponent and the cylinder has two. The number of subcomponents for each of the components in ID/3D is shown in the table below:

Component	Number of subcomponents
Half-space	1
Slab	1
Sphere	1
Ellipsoid	1
Cylinder	2
Cone	2
Paraboloid	2
Hyperboloid	2
Box	3

The bounds on the number of objects, subcomponents and segments can now be stated as follows:

Maximum number of objects in the system	29
Maximum number of subcomponents per object	15
Maximum number of subcomponents in the system	119
Maximum number of segments in the system	10

The above bounds are based on the sizes of the tables in COMMON where the various parameters are stored. Although not directly relevant to the user, the following information may be of interest. For each object, storage is used both for the parameters of the object itself and for those of each of its subcomponents. The amounts of storage used are as follows:

For each object	108 bytes
For each subcomponent	124 bytes
For each segment	[illegible] bytes

These storage requirements are determined in part by the fact that all floating parameters are stored (and processed) in double precision. The storage requirements could be reduced slightly if the program were written in a language that has access to bytes and half-words; they could be reduced substantially, at some cost in processing speed, by the use of list processing rather than fixed format tables.

B. TIMING

The times required to perform some typical functions when the program is executed on the IBM 360/50 are given below.

a) Segment and object definition.
The time to define and remove a segment is 2.8 milliseconds. The time to define and remove an object depends on the components of the object; the time per component varies from 4 ms. for a sphere to 17 ms. for a parallelepiped with faces parallel to the coordinate planes to 26 ms. for a parallelepiped in an arbitrary orientation.

b) Segment-object intersection test.
The time can be as low as 3.5 ms. when the segment does not intersect the sphere containing the object. Otherwise it can vary over a fairly wide range (6 to 75 ms.). Typical values for 1- or 2- component objects are as

follows: when the distance or amount of overlap is of the order of .1 times the linear dimensions of the object: 20 ms.; when the distance or overlap is of the order of .001 times the linear dimension: 30 ms.

c) Object-object intersection test.

When the spheres containing the objects do not overlap the time is 2.7 ms. When these spheres do overlap, the test time is affected by the following factors:

- it varies directly with the number of components comprising the objects;
- it is usually much lower when the objects are symmetrically placed;
- it increases slowly as the distance or amount of overlap decreases.

In a large number of tests between pairs of asymmetrically placed objects, each consisting usually of 1 or 2 components, test times were observed between 10 ms. and 2.5 seconds. The following are typical times, where the distance or amount of overlap is expressed as a fraction of the linear dimensions of the objects:

distance = .1	250 ms.
distance = .001	500 ms.
overlap = .1	100 ms.
overlap = .001	300 ms.

The above times were obtained with the ID/3D program compiled under FORTRAN G. If ID/3D is compiled under BPS FORTRAN, these times are increased by about 20 percent. Under FORTRAN H these times are reduced by about 15 percent.

V. EXAMPLES AND APPLICATIONS

A. SAMPLE CARD INPUT

The listing below represents a sample input deck which is also included in the program tape. The following aspects of ID/3D are illustrated:

1. Various input errors detected by the program, and one type of input error (multiple definition of PAR1) not detected.

2. Starting at the definition of OCTPRSM1, many correct object definitions and object-object tests are shown.

3. The several tests involving HYP show the dependence of the tests on the region being used.

4. The tests involving WEDGE illustrate the change in tolerance.

5. There follows a set of segment-object tests.

6. Starting at the last initialization, the effect of the output level parameter is illustrated.

The instructions for running the sample card input are given in the section on Operating Instructions.

```
A       INIT    0.
        SLA     0.      0.      0.      1.      0.      0.      2.
        SLA     0.      0.      0.      1.      1.      0.      2.
        SLA     0.      0.      0.      0.      1.      0.      2.
        SLA     0.      0.      0.      -1.     1.      0.      2.
        HSP     0.      0.      2.5     0.      0.      1.      5.
        SPH     -2.     0.      0.      1.
        ELL     0.      0.      0.      1.
        ELL     0.0     0.0     0.0     3.0     2.0     1.0
        PAR     0.0     0.0     0.0     3.0     2.0     1.0     -90.0   45.0    90.0
        CYL     0.      0.      0.      1.      2.      2.
        BOX     0.      0.      0.      2.      2.      3.
        BOX     1.      2.      5.      5.      5.      5.
BOX       BOX   0.      0.      0.      1.      2.      1.
        BOX     2.1     6.      1.3     2.2     1.2     5.1
BOX1    BOX     0.      0.      0.      1.      1.      1.      0.      0.      0.      0.
BOX1    TOS SPH1        0.      0.      0.      1.      1.      1.      0.      0.      0.      0.
SPH1    TOS BOX1
SPH1    SPH
 BOX1   TOS SPH1        0.      0.      0.      5.      0.      0.      0.      0.      0.      0.
SPH2    TOM SPH1
        TOM SPH2
        REMO    :
        REM
        REMO
        REMO
        SEG
LINE1   SEG             0.      0.      0.      6.      6.      6.      0.      0.      0.      0.
LINE1   TSS LINE2       0.      0.      0.      6.      6.      6.      0.      0.      0.      0.
LINE2   TSS LINE1
LINE2   SEG
BOX2    BOX             2.      2.      2.      1.      1.      1.      0.      0.      0.      0.
LINE3   TSM LINE1       0.      0.      0.      5.      5.      5.      0.      0.      0.      0.
BOX2    TSS LINE1
        REMS
        REMS
        REMS
        CLR
OCTPRSM1SLA             0.      0.      0.      1.      0.      0.      2.
        SLA             0.      0.      0.      1.      1.      0.      2.
        SLA             0.      0.      0.      0.      1.      0.      2.
        SLA             0.      0.      0.      -1.     1.      0.      2.
OCTPRSM2BOX             0.      0.      2.5     0.      0.      1.      5.
        BOX             4.      2.      2.5     2.      2.      2.5
OCTPRSM1TOS OCTPRSM2    4.      2.      2.5     2.      2.      2.5     45.
HEM1    SPH             -2.     2.      0.      1.
        HSP             -2.     0.      0.      1.
HEM1    TOM
CYL     CYL             0.      4.05    1.      .1      .1      20.     90.     90.
CYL     TOM
        CLR
ELL1    ELL             0.0     0.0     0.0     3.0     2.0     1.0
ELL2    ELL             0.0     0.0     0.0     3.0     2.0     1.0     -90.0   45.0    90.0
ELL4    ELL             4.0     0.0     4.0     3.0     2.0     1.0     -90.0   45.0    90.0
ELL4    TOM
        REMO
ELL2    TOM
        REMO
ELL1    TOM
        REMO
PAR1    PAR             0.      0.      0.      1.      2.      2.
```

```
PAR1      PAR          3.      0.      0.      1.      2.      2.
PAR1      PAR          5.      0.      0.      2.      1.      2.
PAR1      TOM
          CLR
PAR1      PAR          0.      0.      0.      1.      2.      2.
PAR2      PAR          3.      0.      0.      1.      2.      2.
PAR3      PAR          5.      0.      0.      2.      1.      2.
HYP       HYP          0.      0.      8.      5.      5.      8.          130.        6.
          REG          10.     0.      0.      1.
HYP       TOM
          REG          10.     0.      0.      5.5
HYP       TOM
          REG          10.     0.      0.      6.5
HYP       TOM
          REG          10.     0.      0.      9.5
HYP       TOM
          CLR
          REG          0.      0.      0.      10.
WEDGE     CYL          0.      0.      0.      2.      2.      3.
          HSP          0.      1.      0.      0.      -3.     1.
          HSP          0.      -1.     0.      0.      3.      1.
CONE1     CON          0.      1.002   0.      1.      1.      5.          -90.
CONE2     CON          0.      -.998   0.      1.      1.      5.          90.
WEDGE     TOS CONE1
WEDGE     TOS CONE2
          REMO
          REMO
          TOL          .0001
CONE1     CON          0.      1.002   0.      1.      1.      5.          -90.
CONE2     CON          0.      -.998   0.      1.      1.      5.          90.
WEDGE     TOS CONE1
WEDGE     TOS CONE2
          CLR
ELL1      ELL          0.0     0.0     0.0     8.0     6.0     1.0
HEMI      SPH          -2.     2.      0.      1.
          HSP          -2.     0.      0.      1.
SPH3      SPH          7.      7.      7.      .002
SEG1      SEG          0.0     0.0     12.0    9.0     0.0     0.0
SEG1      TSS ELL1
SEG1      TSM
          REMS
SEG1      SEG          0.      0.      12.     0.      0.      11.
SEG1      TSS ELL1
SEG1      TSM
          REMS
SEG1      SEG          0.0     0.0     12.0    9.0     0.0     11.0
SEG1      TSS ELL1
SEG1      TSM
          REMS
SEG1      SEG          0.0     0.0     12.0    5.0     0.0     0.0
SEG1      TSS ELL1
SEG1      TSM
SEG3      SEG          -7.     0.      .98     2.      0.      .99
SEG3      TSS ELL1
NULL      SEG          1.      1.      1.      1.      1.      1.
NULL      TSS ELL1
          INIT         1.
OCTPRSM1  SLA          0.      0.      0.      1.      0.      0.      2.
          SLA          0.      0.      0.      1.      1.      0.      2.
          SLA          0.      0.      0.      0.      1.      0.      2.
          SLA          0.      0.      0.      -1.     1.      0.      2.
          SLA          0.      0.      2.5     0.      0.      1.      5.
OCTPRSM2  BOX          4.      2.      2.5     2.      2.      2.5
          BOX          4.      2.      2.5     2.      2.      2.5     45.
PAR2      PAR          3.      0.      0.      1.      2.      2.
LINE      SEG          2.007   4.      2.      2.007   -2.5    1.9
```

```
OCTPRSM1TOS  OCTPRSM2
LINE     TSS OCTPRSM1
         OUT          2.
OCTPRSM1TOS  OCTPRSM2
LINE     TSS OCTPRSM1
         OUT          3.
OCTPRSM1TOS  OCTPRSM2
LINE     TSS OCTPRSM1
         END
```

B. SAMPLE SUBROUTINE INPUT: A PIPE PLACEMENT PROBLEM

The program listed below illustrates how data can be passed to ID/3D by means of subroutine calls. The output of the program is also given. The program is intended to solve a simple placement problem in which a straight segment of pipe is to clear two obstacles. The problem is solved twice: first the pipe is represented by a line segment, then by a cylinder. The objects to be cleared are a hemisphere and a truncated cone shown in two projections in Figure 3.

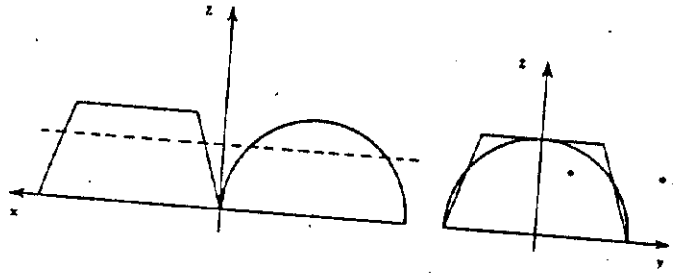

Figure 3

The segment extends from X = -6. to X = 6., lies in the Place Z = 2., and is to be placed by varying the value of its Y coordinate. The procedure for placing the segment is essentially a binary search: first the segment is placed at

Y = 4. where it obviously does not intersect either object; then it is moved to Y = 1. where it intersects both objects; then it is successively moved by half the amount of the previous move either out or in, depending whether the previous test produced intersections or not. The procedure for placing the cylinder is entirely analogous.

```
 BPS FORTRAN IVD  COMPILER  VERSION  1  LEVEL  0 JUNE  1965

BEGIN COMPILATION
 S.0001          CALL IDINIT (-1.)                                                    SMPL0001
 S.0002          CALL IDSPH(J,193,0,    -3.,0.,0.,3.)                                 SMPL0002
 S.0003          CALL IDHSP(J,  0,0,    0.,0.,0.,0.,0.,1.)                            SMPL0003
 S.0004          CALL IDCON(J,194,0,    3.,0.,8.,3.,3.,8.,0.,180.,0.)                 SMPL0004
 S.0005          CALL IDHSP(J,  0,0,    0.,0.,3.,0.,0.,-1.)                           SMPL0005
 S.0006          CALL IDEND                                                           SMPL0006
 S.0007          WRITE (6,2)                                                          SMPL0007
 S.0008        2 FORMAT (1H1,30X,14HLINE PLACEMENT)                                   SMPL0008
 S.0009          Y=4.                                                                 SMPL0009
 S.0010          DY=6.                                                                SMPL0010
 S.0011          DO 30 N=1,25                                                         SMPL0011
 S.0012          CALL IDSEG(J,226,0,    -6.,Y,2.,6.,Y,2.)                             SMPL0012
 S.0013          CALL IDTSS(J,226,0,193,0,    XR,YR,ZR,FR)                            SMPL0013
 S.0014          GO TO (20,80,10,80,80,80),J                                          SMPL0014
 S.0015       10 CALL IDTSS(J,226,0,194,0,    XR,YR,ZR,FR)                            SMPL0015
 S.0016          GO TO (20,80,12,80,80,80),J                                          SMPL0016
 S.0017       12 IF (DY-.01) 40,40,14                                                 SMPL0017
 S.0018       14 DY=.5*DY                                                             SMPL0018
 S.0019          Y=Y-DY                                                               SMPL0019
 S.0020          GO TO 28                                                             SMPL0020
 S.0021       20 DY=.5*DY                                                             SMPL0021
 S.0022          Y=Y+DY                                                               SMPL0022
 S.0023       28 CALL IDREMS                                                          SMPL0023
 S.0024       30 CONTINUE                                                             SMPL0024
 S.0025          GO TO 80                                                             SMPL0025
 S.0026       40 WRITE (6,41) Y,N                                                     SMPL0026
 S.0027       41 FORMAT(1H0,14HCLEAR LINE. Y=,F9.5,10X,2HN=,I2)                       SMPL0027
 S.0028          GO TO 90                                                             SMPL0028
 S.0029       80 WRITE (6,81) Y,N                                                     SMPL0029
 S.0030       81 FORMAT(1H0,29HERROR OR UNDECIDABLE CASE. Y=,F9.5,10X,2HN=,I2)        SMPL0030
 S.0031       90 WRITE (6,91)                                                         SMPL0031
 S.0032       91 FORMAT(1H0,30X,18HEND LINE PLACEMENT)                                SMPL0032
 S.0033          WRITE (6,102)                                                        SMPL0033
 S.0034      102 FORMAT(1H1,30X,18HCYLINDER PLACEMENT)                                SMPL0034
 S.0035          Y=4.                                                                 SMPL0035
 S.0036          DY=6.                                                                SMPL0036
 S.0037          DO 130 N=1,25                                                        SMPL0037
 S.0038          CALL IDCYL(J,195,0,    0.,Y,2.,.1,.1,6.,90.,90.,0.)                  SMPL0038
 S.0039          CALL IDTOS(J,195,0,193,0,    XR,YR,ZR,FR)                            SMPL0039
 S.0040          GO TO (120,180,110,180,180,180),J                                    SMPL0040
 S.0041      110 CALL IDTOS(J,195,0,194,0,    XR,YR,ZR,FR)                            SMPL0041
 S.0042          GO TO (120,180,112,180,180,180),J                                    SMPL0042
 S.0043      112 IF (DY-.01) 140,140,114                                              SMPL0043
 S.0044      114 DY=.5*DY                                                             SMPL0044
 S.0045          Y=Y-DY                                                               SMPL0045
 S.0046          GO TO 128                                                            SMPL0046
 S.0047      120 DY=.5*DY                                                             SMPL0047
 S.0048          Y=Y+DY                                                               SMPL0048
 S.0049      128 CALL IDREMO                                                          SMPL0049
 S.0050      130 CONTINUE                                                             SMPL0050
 S.0051          GO TO 180                                                            SMPL0051
 S.0052      140 WRITE (6,141) Y,N                                                    SMPL0052
 S.0053      141 FORMAT (1H0,18HCLEAR CYLINDER. Y=,F9.5,10X,2HN=,I2)                  SMPL0053
 S.0054          GO TO 190                                                            SMPL0054
 S.0055      180 WRITE (6,81) Y,N                                                     SMPL0055
 S.0056      190 WRITE (6,191)                                                        SMPL0056
 S.0057      191 FORMAT(1H0,30X,22HEND CYLINDER PLACEMENT/1H1)                        SMPL0057
 S.0058          STOP                                                                 SMPL0058
 S.0059          END                                                                  SMPL0059
```

LINE PLACEMENT

CLEAR LINE. Y= 2.25098

N=12

END LINE PLACEMENT

CYLINDER PLACEMENT

CLEAR CYLINDER. Y= 2.36963

N=13

END CYLINDER PLACEMENT

The following remarks refer to individual statements in the listing.

S.0001 — The output control parameter (hereafter referred to as P) is set to -1., so the only output is that produced by this program. For debugging the program one would use P = 0.

S.0002-S.0003 — Definition of hemisphere. The name of the object is 193, 0, which prints as bbbAbbbb.

S.0004-S.0005 — Definition of truncated cone. The vertex of the cone is at (3.,0.,8.) and its height is 8. Since the standard attitude of the cone is with the vertex down, it must be rotated by means of the Eulerian angles (0., 180., 0.) in order to bring the base down; since the first of the three angles is 0., the line of nodes will coincide with the X-axis, and the second angle specifies a rotation of 180 degrees about the X-axis.

S.0006 — Force immediate printing of message relating to definition of object (printing occurs only if $P \geq 0.$)

S.0007-S.0008 — Print heading.

S.0009 — Initial value of Y.

S.0010 — Twice initial Y increment.

S.0011 — Loop for placement of segment.

S.0012 — Definition of segment.

S.0013 — Test of segment against hemisphere. The return parameters XR,YR,ZR,FR are not used in the rest of the program.

S.0014 — If J = 1, indicating that it intersects the hemisphere, no test against the truncated cone is executed. If J = 3, segment is to be tested against cone. If undecidable or error return, procedure is terminated.

S.0015 — Test segment against truncated cone.

S.0017 — If last displacement (to non-intersecting position) was $\leq .01$, the desired position of the segment has been obtained.

S.0018-S.0019 — Halve displacement and decrement Y.

S.0021-S.0022 — Halve displacement and increment Y.

S.0023 — Remove segment. Since the same name is used in S.0013 for the different positions of the segment, it is essential that the segment be removed before being redefined. Also, the system cannot contain more than 10 segments.

S.0025 — If the desired placement is not obtained in 25 iterations, the procedure is terminated. (After being halved 25 times, DY is of the order of 10^{-7} times Y, hence any further iterations would not affect the value of Y.)

The following remarks refer to individual statements in the listing.

S.0001 The output control parameter (hereafter referred to as P) is set to -1., so the only output is that produced by this program. For debugging the program one would use P = 0.

S.0002-S.0003 Definition of hemisphere. The name of the object is 193, 0, which prints as bbbAbbbb.

S.0004-S.0005 Definition of truncated cone. The vertex of the cone is at (3.,0.,8.) and its height is 8. Since the standard attitude of the cone is with the vertex down, it must be rotated by means of the Eulerian angles (0., 180., 0.) in order to bring the base down; since the first of the three angles is 0., the line of nodes will coincide with the X-axis, and the second angle specifies a rotation of 180 degrees about the X-axis.

S.0006 Force immediate printing of message relating to definition of object (printing occurs only if $P \geq 0$.)

S.0007-S.0008 Print heading.

S.0009 Initial value of Y.

S.0010 Twice initial Y increment.

S.0011 Loop for placement of segment.

S.0012 Definition of segment.

S.0013 Test of segment against hemisphere. The return parameters XR,YR,ZR,FR are not used in the rest of the program.

S.0014 If J = 1, indicating that it intersects the hemisphere, no test against the truncated cone is executed. If J = 3, segment is to be tested against cone. If undecidable or error return, procedure is terminated.

S.0015 Test segment against truncated cone.

S.0017 If last displacement (to non-intersecting position) was $\leq .01$, the desired position of the segment has been obtained.

S.0018-S.0019 Halve displacement and decrement Y.

S.0021-S.0022 Halve displacement and increment Y.

S.0023 Remove segment. Since the same name is used in S.0013 for the different positions of the segment, it is essential that the segment be removed before being redefined. Also, the system cannot contain more than 10 segments.

S.0025 If the desired placement is not obtained in 25 iterations, the procedure is terminated. (After being halved 25 times, DY is of the order of 10^{-7} times Y, hence any further iterations would not affect the value of Y.)

S.0033 Beginning of second part of program, for placement of cylinder.

S.0038 Definition of cylinder. Its radius is .1 and its center-line is positioned like the segment in the first part of the program.

The correctness of the results may be verified geometrically. In the plane Z = 0 the maximum value of Y is $\sqrt{5}$ = 2.2361 for the hemisphere and 2.25 for the truncated cone. Hence, the clear line should have a Y value between 2.25 and $\sqrt{3.1^2 4.}$ = 2.3685, while a cylinder tangent to the truncated cone has Y = 2.25 + .1 $\sqrt{1+1/16}$ = 2.3531. Hence, the clear cylinder should have a Y value between 2.3685 and 2.3785.

C. SOLUTION OF THE HIDDEN LINE PROBLEM

As explained in the introduction, the problem is to determine which portions of (a) the edges of an object and (b) the intersections of the surfaces bounding the object are visible from a given viewing point. The term "edge" is used here to mean a line where the line of sight is tangent to a surface bounding the object. See Figure 4.

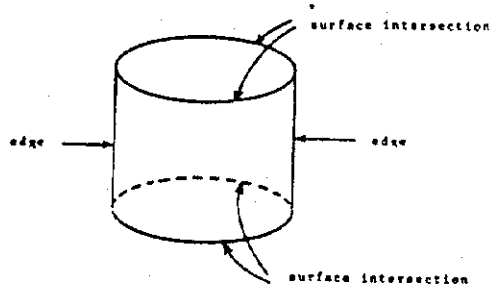

Figure 4. Edges and Surface Intersections of a Cylinder.

The ID/3D program itself does not compute the edges and surface intersections: they have to be obtained by an independent procedure which should compute for each line a sequence of closely spaced points, the closeness depending on the desired accuracy of the image to be displayed.

The procedure shown below, together with the explanatory notes following it, gives a solution of the hidden line problem. The essential parts of the procedure have been coded in FORTRAN and found to give correct results in a number of test cases.

HIDDEN LINE PROCEDURE

```
1.  For each i
2.      Compute list PH_i
3.  Set TOLERANCE
4.  For each i
5.      Define object O_i
6.  For each i
7.      Generate points P_k on edges and surface intersections
        of O_i
8.      For each k
9.          Define segment S_k joining P_k to VP
10.         For each j in PH_i (except j=i if P_k is on edge of
            O_i)
11.             Test for intersection of S_k and O_j
12.             Intersection? Yes   P_k is invisible
13.                 +No             Go to α
14.             Continue j loop
15.         P_k is invisible
16.         α:  Continue k loop
```


NOTES TO HIDDEN LINE PROCEDURE

- The underlined statements are executed in ID/3D and the others are in the calling program.

- The numbers preceding the following notes refer to the statement numbers in the procedure.

- It is assumed that a set of objects O_i (i = 1, ...,n) and a viewing point or viewing direction VP are given. It is also assumed that the linear dimensions of the objects are of the order of 1 unit.

2. The list PH_i contains the indices of all the objects that may hide parts of O_i, including i itself. The simplest way to set up these lists is to enter in each of them every index. However, in order to avoid redundant tests, it is desirable to keep these lists short. This may be done by any of three methods:

   a) by a priori knowledge about the spatial arrangement of the objects;
   b) by surrounding each object with a sphere, projecting the spheres from VP onto a plane, and calculating whether the projected images intersect; the latter step can be done directly or by using ID/3D (e.g. replacing the projected ellipses by elliptical cylinders of arbitrary thickness); if the projections of two objects do not intersect, then neither object can hide the other;
   c) in some cases it may be possible to shorten the lists during the execution of the program; since all objects

are convex, it is impossible for two objects to each hide part of the other; thus if it is found that a point on O_i is hidden by O_j, i should be removed from the list PH_j.

5. A distinction must be made depending on whether this procedure is used for testing for surface intersections that may be hidden by the object itself, or not. The latter case may occur when there are more direct methods of doing the test, and does not require special precautions. In the former case, however, it should be noted that the p-c function used in ID/3D has the effect of replacing the surfaces bounding an object by an approximating surface that lies slightly outside the original surfaces but contains the surface intersections. In order to make the surface intersections visible in ID/3D (when they are visible), it is necessary to define the objects slightly smaller than they actually are, but to compute the exact positions of the surface intersection points. The suggested procedure is to set a very small value for the tolerance (e.g. .0001) and to reduce the dimensions of the objects by an amount larger than the tolerance but small compared with the dimensions of the objects (e.g. .001). The situation is illustrated in Figure 5 where the rectangles represent cross sections of a box. The vertices of the rectangle are the cross sections of what are here called the "surface intersections" of the box, usually called "edges".

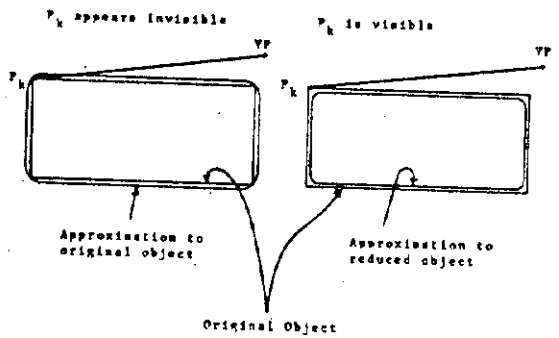

Figure 5.

INTERSECTION DETECTION IN THREE DIMENSIONS

USER INFORMATION

VI. DETAILS OF MATHEMATICAL METHOD

This section contains a formal description of the intersection detection procedure and a discussion of some of its properties.

A. THE INTERSECTION DETECTION PROCEDURE

1. A number of objects are defined. An object is the intersection of $k \geq 1$ regions R_i where each R_i is defined by an inequality

$$g_i(x,y,z) \leq 0$$

and the function g_i has the following properties:

a) it is convex
b) it is differentiable
c) it is normalized so that $|\text{grad } g_i| \geq 1$ on the surface $g_i = 0$
d) given a bounded region of space, grad g_i is bounded there. This property is actually a consequence of a) and b), but it is listed separately for purposes of later discussion.

It is also assumed that every object is contained in a bounded region of space $\bar{R}$, even though the individual R_i need not be bounded. For example, a hemisphere may be defined as the intersection of a sphere and a half-space. The region $\bar{R}$ is used for establishing the bound on grad g_i mentioned in d) above.

2. The pseudo-characteristic function G of the intersection of n objects is defined in terms of the g_i functions for the objects by

$$v_i = \sqrt{g_i^2 + t^2} + g_i \qquad (1)$$

$$V = \Sigma v_i \qquad (2)$$

$$G = \frac{1}{2}\left(V - \frac{t^2}{V}\right) + c \qquad (3)$$

where t and c are small positive numbers. The summation in (2) contains one term for each region R_i of each object.

3. A sequence of points approaching the minimum of G is constructed by means of a descent procedure. The search is terminated as soon as
   a) a negative value of G is obtained, indicating that there is a (non-empty) intersection; or
   b) the minimum of G is obtained and it is positive, indicating that there is no intersection; from the bounds on the sizes of the objects it is possible to compute lower bounds for G, and the search can be stopped if a positive lower bound is established.

4. As a special case, the problem of whether a line segment intersects an object can be solved (approximately) by finding the minimum on the segment of the pseudo-characteristic function of the object.

B. REMARKS ON THE PSEUDO-CHARACTERISTIC FUNCTION METHOD

1. The central idea of the method is that of applying the transformation (1) to each of the functions g_i that define the various R_i regions. The function v_i is always positive and it is an increasing function of g_i. If $t \ll 1$, v_i also has the property that

$$v_i = \begin{cases} 2g_i + O(t^2) & \text{when } g_i > 0 \\ t & g_i = 0 \\ O(t^2) & g_i < 0 \end{cases}$$

In other words, except for the vicinity of the surface $g_i = 0$, v_i is large compared with t outside R_i, and small compared with t inside R_i. Consequently, the function V given by (2) is small compared with t when all the g_i are sufficiently negative, i.e., at those points in the intersection of the R_i that lie sufficiently far from the boundaries $g_i = 0$; and $V \geq t$ outside that intersection, where at least one of the g_i is ≥ 0.

The right hand side of equation (3) contains an expression in V and a "correction term" c. Disregarding for the time being the latter, and considering the expression

$$G^* = \frac{1}{2}\left(V - \frac{t^2}{V}\right),$$

one can easily verify that the transformation from V to G* is the inverse of the transformation from g_i to v_i.

It can also be seen that

$$G^* \begin{cases} <0 \\ =0 \\ >0 \end{cases} \text{ when } \begin{cases} V < t \\ V = t \\ V > t. \end{cases}$$

Hence, except for the vicinity of the surfaces $g_i = 0$, G* will be negative inside the intersection of the regions $g_i < 0$ and positive outside that intersection.

2. The nature of the approximation inherent in the method will now be discussed in detail, and it will be shown how the parameters t and c can be used to control its closeness and direction. Let I denote the intersection of N regions R_i, and let B denote the boundary of I. The points of B can be classified according to the number of the $g_i \leq 0$ inequalities in which the equal sign holds. A point will be here called a j-point if the equal sign holds in exactly j inequalities. (In "normal" cases 1-points lie on surfaces, 2-points on curves, and higher order points are isolated. For example, with a segment of a sphere the base and the circular cap are made up of 1-points, the circumference of the base is made up of 2-points, and there are no boundary points of order higher than 2. In the case of a parallelepiped the vertices are 3-points. An exception to the normal case occurs when the set of functions g_i is functionally dependent; it is then possible to have bounding surfaces made up of points of order higher than 1, etc.) Let $\bar{R}$ be a region containing I, let M be a bound on all the $|\text{grad } g_i|$ in this region, and assume that $t \ll 1/M$. The following order-of-magnitude considerations can be made about the expression for the gradient of G:

$$\text{grad } G = \frac{1}{2}\left(1 + \frac{t^2}{V^2}\right) \cdot \sum \left[\left(\frac{g_i}{\sqrt{g_i^2 + t^2}} + 1\right) \text{ grad } g_i\right]$$

- whenever a g_i is sufficiently negative, the corresponding term in square brackets is $O(t^2)$
- whenever a $g_i = 0$, the corresponding term in square brackets has the magnitude $|\text{grad } g_i| \geq 1$
- if $g_i = 0$ for exactly j values of i and the remaining g_i are sufficiently small,

and
$$V = jt + O(t^2)$$

$$\frac{1}{2}\left(1 + \frac{t^2}{V^2}\right) = \frac{1}{2}\left(1 + \frac{1}{j^2}\right) + O(t).$$

Substituting the above estimate for V in the expression for grad G and for G* one can reach the following conclusions: given a j-point where $g_{i_1} = g_{i_2} = \ldots = g_{i_j} = 0$ and assuming that the point is sufficiently distant from boundary points of higher order; the following equations hold:

$$\text{grad } G = \frac{1}{2}\left(1 + \frac{1}{j^2}\right) \sum_{s=1}^{j} \text{grad}_{i_s} + O(t) \qquad (4)$$

$$G^* = \frac{1}{2}\left(j - \frac{1}{j}\right)t + O(t^2). \qquad (5)$$

In particular at a 1-point $G^* = O(t^2)$ and the sum in equation (4) reduces to a single term, so that

$$|\text{grad } G^*| = |\text{grad } G| \geq 1 + O(t),$$

while at boundary points of order higher than 1, $G^* > 0$. Hence, the surface $G^* = 0$ lies within a distance $O(t^2)$ of the 1-points of B, except in the neighborhood of higher order boundary points, where it lies inside I.

In order to control the position of the surface $G = 0$ with respect to B, one can now set the parameter c in equation (3). If one sets

$$c = \frac{1}{2}\left(j - \frac{1}{j}\right)t \ ,$$

so that

$$G = G^* + c$$

$$= \frac{1}{2}\left(V - \frac{t^2}{V}\right) - \frac{1}{2}\left(j - \frac{1}{j}\right)t$$

then the surface $G = 0$ will lie within a distance $O(t^2)$ of the j-points of B, and it will lie inside I near the points of B of order higher than j and outside I near the points of B of order less than j. Estimates of the distance from B to the surface $G = 0$ for the various boundary points can be obtained from (4) and (5). In particular at a 1-point

$$G = -\frac{1}{2}\left(j - \frac{1}{j}\right)t$$

and, because of the lower bound on the gradient,

$$\frac{1}{2}\left(j - \frac{1}{j}\right)t$$

is an approximate upper bound to the distance in question.

3. The role of the assumptions made in B.1. about the form of the functions g_i and the boundedness of the objects will now be discussed.

The boundedness of the objects is not a serious restriction from a practical viewpoint. The reason for making this assumption is that it makes it possible to establish bounds to the region where the descent procedure searches for negative values of G.

The convexity assumption is the most restrictive, since it implies that only convex objects can be defined. It can be proved that G is a convex function of the g_i and hence of the coordinate variables. Hence, the assumptions of convexity and boundedness imply further that
(a) G has exactly one minimum,
(b) it is possible to compute lower bounds for the values of G without actually finding its minimum.
These two properties make it possible to use a relatively simple and efficient search procedure to determine whether G ever takes on negative values. Without the convexity assumption these two properties would not hold, and it would be necessary to have a procedure to find the absolute minimum of a function that may have several relative minima; this is in general very difficult.

The assumption of differentiability is a minor restriction in practical cases. If one is dealing with an object bounded by a piecewise differentiable surface, the several differentiable pieces are usually represented by different analytic expressions, and one should assign a different g_i to each of the expressions. The assumption of differentiability makes it possible to use a gradient method in the search for negative values of G and the computation of lower bounds to G.

VII. NOTES

1. The reasons for requiring the specification of a region are the following:

   (a) While it is possible to define unbounded objects, e.g. half-spaces, the procedure for intersection detection works only in a bounded region.

   (b) It offers the user the possibility of testing more selectively for potential intersections.

   (c) A reasonably small region will usually result in greater program efficiency.

2. Starting from a given set of geometric shapes (components), a natural way to define a greater variety of shapes is to combine the components by taking intersections, unions, and relative complements. Since ID/3D is designed to test for intersections of objects, the capability of defining objects as intersections of components comes essentially for free.

   The capability of defining objects as unions of components (or, more generally, as unions of intersections of components) does not present mathematical difficulties; thus if one wants to test the union of A and B against C, one can simply test A against C, then (if necessary) B against C. It would, however, require a more complex form of in-

put, a more elaborate naming capability, and an internal structure based on lists rather than tables.

The introduction of relative complements would raise more serious difficulties, since it would give rise to non-convex p-c functions. See [4] below.

3. Fletcher, R. and Powell, M.J.D. "A Rapidly Convergent Method for Minimization," The Computer Journal, 6, 163 (1963)

4. The p-c function method works efficiently with objects that are convex and have a convex p-c function because in this case

   (a) any p-c function resulting from the intersection of such object has only one minimum.

   (b) it is often possible, by means of lower bound computation, to establish that the minimum of the p-c function is positive without actually computing the minimum.

When the p-c function is non-convex (and this is always the case with non-convex objects) neither of the above statements holds: the p-c function may have several relative mimima, and it is usually impossible to establish lower bounds for them. An algorithm to decide whether such a function is ever negative would be very difficult to construct and probably very time consuming.

5. The integer function KAL (I,J,K,L), where I,J,K,L are the numerical values of the EBCDIC codes for four characters, computes an integer that under the format (A4)

prints the four characters. For example, given that the codes for A and B are 193 and 194, a call KAL (193, 194, 193, 194) will compute the integer that prints ABAB.

6. The use of Eulerian angles to represent a rotation in space is discussed in most textbooks on Classical Mechanics or Analytical Dynamics; see e.g.

Goldstein, H. (1950) Classical Mechanics, Addison-Wesley

Houston, W.V. (1948) Principles of Mathematical Physics, McGraw-Hill. 2nd Edition

The convention used here as to the placement of the angles is as follows: let XYZ be the absolute coordinate system (fixed in space) and X'Y'Z' a coordinate system fixed with respect to the object; then

T7 is the angle from X to the line of nodes,
T8 is the angle from Z to Z',
T9 is the angle from the line of nodes to X'.

Hence the cosines of the angles between the two sets of axes are as shown in the following table:

	X	Y	Z
X'	cos(T7)cos(T9) -sin(T7)sin(T9)cos(T8)	sin(T7)cos(T9) +cos(T7)sin(T9)cos(T8)	sin(T9)sin(T8)
Y'	-cos(T7)sin(T9) -sin(T7)cos(T9)cos(T8)	-sin(T7)sin(T9) +cos(T7)cos(T9)cos(T8)	cos(T9)sin(T8)
Z'	sin(T7)sin(T8)	-cos(T7)sin(T8)	cos(T8)

7. Since the calculation of Eulerian angles is at times cumbersome, the subroutine IDEU has been provided to enable the user to specify the orientation of a component in terms of the direction numbers of its axes. If IDEU is used, a call to IDEU should be executed before a call to a component subroutine. The format is

   IDEU (A1,A2,A3,B1,B2,B3,T7,T8,T9)

   where A1 through B3 are input parameters and T7, T8, T9 are output parameters; the latter are, in fact, the Eulerian angles. The parameters A1, A2, A3 are direction numbers of the X' axis, and B1, B2, B3 are direction numbers of the Z' axis. The direction numbers need not be normalized. If the vectors (A1, A2, A3) and (B1, B2, B3) are not orthogonal and non-null vectors, the input is inconsistent and IDEU takes the following action:

   (a) if (B1, B2, B3) is a null vector, no rotation takes place, i.e. T7 = T8 = T9 = 0.

   (b) if (A1, A2, A3) is a null vector or is parallel to (B1, B2, B3), X' is taken perpendicular to Z' in the XY plane

   (c) if (A1, A2, A3) is not orthogonal to (B1, B2, B3), the projection of (A1, A2, A3) in the plane perpendicular to (B1, B2, B3) determines X'.

8. It may occasionally happen that the calculations in ID/3D give rise to exponent underflows. These are not errors, but they may cause the operating system to print interrupt messages. It is up to the user to disable these interrupts if he so desires.

INTERSECTION DETECTION IN THREE DIMENSIONS

OPERATING INSTRUCTIONS

The program tape contains three files (see section on Magnetic Tape Key) from which three decks can be punched by using either the BPS or the OS utilities.

1. To Obtain ID/3D Program

   The following job control statement may be used to obtain the card decks, assuming that SYSCP is the external name of the card punch, and that the program tape is mounted on unit 182.

```
//OSUT   JOB  NYSC,MSGLEVEL=1   O/S 360 UTILITY TAPE TO PUNCH 9TR
//       EXEC  PGM=IEBPTPCH
//SYSPRINT     DD  SYSOUT=A
//SYSUT1  DD  UNIT=(182,,DEFER),VOLUME=SER=TAPE,LABEL=(1,NL),        X
//            DSNAME=INPUT,DCB=(,RECFM=FB,LRECL=80,BLKSIZE=1600),    X
//            DISP=(OLD,DELETE)
//SYSUT2 DD   UNIT=SYSCP
//SYSIN  DD   *
        PUNCH TYPORG=PS,MAXFLDS=1
        RECORD   FIELD=(80,1,,1)
/*
//OSUT   JOB  NYSC,MSGLEVEL=1   O/S 360 UTILITY TAPE TO PUNCH 9TR
//       EXEC  PGM=IEBPTPCH
//SYSPRINT     DD  SYSOUT=A
//SYSUT1  DD  UNIT=(182,,DEFER),VOLUME=SER=TAPE,LABEL=(2,NL),        X
//            DSNAME=INPUT,DCB=(,RECFM=FB,LRECL=80,BLKSIZE=1600),    X
//            DISP=(OLD,DELETE)
//SYSUT2 DD   UNIT=SYSCP
//SYSIN  DD   *
        PUNCH TYPORG=PS,MAXFLDS=1
        RECORD   FIELD=(80,1,,1)
/*
//OSUT   JOB  NYSC,MSGLEVEL=1   O/S 360 UTILITY TAPE TO PUNCH 9TR
//       EXEC  PGM=IEBPTPCH
//SYSPRINT     DD  SYSOUT=A
//SYSUT1  DD  UNIT=(182,,DEFER),VOLUME=SER=TAPE,LABEL=(3,NL),        X
//            DSNAME=INPUT,DCB=(,RECFM=FB,LRECL=80,BLKSIZE=1600),    X
//            DISP=(OLD,DELETE)
//SYSUT2 DD   UNIT=SYSCP
//SYSIN  DD   *
        PUNCH TYPORG=PS,MAXFLDS=1
        RECORD   FIELD=(80,1,,1)
/*
```

2. To Run Sample Card Input

A program containing the statement

```
CALL IDCARD
```

must be provided. The following control and FORTRAN statements may be used for this purpose.

```
//J1    JOB     MSGLEVEL=1
//      EXEC    FORTGCLG
//FORT.SYSIN DD *
      CALL IDCARD
      STOP
      END
```

ID/3D program deck

```
/*
//GO.SYSIN DD *
```

Sample data cards

```
/*
```

3. To Run Sample Subroutine Input

The following control statements may be used.

```
//J2    JOB     MSGLEVEL=1
//      EXEC    FORTGCLG
//FORT.SYSIN DD *
```

Sample subroutine deck

ID/3D program deck

```
/*
```


INTERSECTION DETECTION IN THREE DIMENSIONS

SYSTEMS MATERIAL

I. LAYOUT OF COMMON

The following chart shows the layout of COMMON. The size of the arrays is shown by the unindented entries in the column "Name of Variable" (e.g. MEFF(50)), whereas the indented entries (e.g. MEFF(1)) refer to individual array elements. In the columns for the various subroutines the letter "S" means that the variable is set in that subroutine; "U" means that it is only used; and "W" means that it is used only in WRITE statements.

Layout of COMMON

Byte (hex.)	Word (dec.)	Name of Variable	Alternate Names	Meaning	ID26	IDEX	SUDO	SMALL	STRAN	SINTER	DESC	DRMIN	SGTRAN	SGINT	SCMIN	SEV
0	0	VX,VY,VZ		coordinates of point	U	W	GAP1	GAP1	GAP1	GAP1	W		GAP1			S
18	6	QQ		p-c function value	U	W					U	U		U	U	S
20	8	QX,QY,QZ		p-c gradient components							U	U		U	U	S
38	14	QA		not used												
40	16	P1-P10		numeric input parameters	S	S	U		U		GAP1	GAP1	U	GAP1	GAP1	GAP1
90	36	NA1,NA2		object name input parameters	S	U	GAP2		GAP2				GAP2			
98	38	NA3,NA4		object name input parameters	S	U										
A0	40	NN1,NN2		object name return parameters	U	S										
A8	42	NBX		branch number in IDEX	S	U										
AC	43	JJR		return parameter	U	S			S	S	S			S	S	S
B0	44	NSUB		not used	not											
B4	45	KO		output level indicator	used	S			U	U			U			
B8	46	KU		output level indicator		S				U	U	U		U	U	U
BC	47	EFF(17)		not used			GAP3	GAP2	GAP3				GAP3			GAP2
100	64	MEFF(50)		test statistics												
		MEFF(1-9)		single object test statistics												
100	64	MEFF(1)	N	steps		U				S	S					
104	65	2	NABR	abnormal returns from DRMIN		U				S	S					
108	66	3	NINV	inversions		U				S	S					
10C	67	4	NPARS(1)	evaluations of p-c functions		U				S	S					
110	68	5	2	extrapolations		U				S	S					
114	69	6	3	linear interpolations		U				S	S					
118	70	7	4	Newton methods		U				S	S					
11C	71	8	5	tangent crossings		U				S	S					
120	72	9	6	binary searches		U				S	S					
124	73	MEFF(10-20)		not used												
150	84	MEFF(21-29)		multiple object test statistics: cumulative values of MEFF(1-9)		S										

Layout of COMMON

Byte (hex.)	Word (dec.)	Name of Variable	Alternate Names	Meaning	ID26	IDEX	SUDO	SMALL	STRAN	SINTER	DESC	DRMIN	SGTRAN	SGINT	SGMIN	SEV
174	93	MEFF(30)		not used												
		MEFF(31-34)		single segment test statistics												
178	94	MEFF(31)	NP2	p-c function evaluations		U								S	S	
17C	95	MEFF(32)	NLI	linear interpolations		U								S	S	
180	96	MEFF(33)	NTC	tangent crossings		U								S	S	
184	97	MEFF(34)	NBS	binary searches		U								S	S	
188	98	MEFF(35-40)		not used												
1A0	104	MEFF(41-44)		multiple segment test statistics: cumulative values of MEFF(31-34)		S										
1B0	108	MEFF(45-50)		not used												
1C8	114	CPICT(10)		tolerance and related statistics							GAP2	GAP2			GAP2	
1C8	114	CPICT(1)	CPI	π/180		S	U		U							
1D0	116	(2)	CT	tolerance		S	GAP4		U							
1D8	118	(3)	CTT	CT**2		S			GAP4							U
1E0	120	(4)		CM(1)*CT		S				U				U		
1E8	122	(5)	CT75	.75*CT		S										U
1F0	124	(6)	CTT5	.5*CTT		S										U
1F8	126	CPICT(7-10)		not used												
218	134	CM(10)		double pr. test parameters										GAP2A		GAP3
218	134	CM(1)		used in computing CPICT(4)		S										
220	136	CM(2)		loss of precision parameter		S				U						
228	138	CM(3)	SM	step multiplier		S						U				
230	140	CM(4)	BC	bound on cosine		S						U				
238	142	CM(5)		coeff. for step accuracy parameter		S				U						
240	144	CM(6)	TH1	threshold for Newton method bypass		S						U				
248	146	CM(7)		high bound on step accuracy		S				U						
250	148	CM(8)		low bound on step accuracy		S				U						
258	150	CM(9-10)		not used												

Layout of COMMON

Byte (hex.)	Word (dec.)	Name of Variable	Alternate Names	Meaning	ID26	IDEX	SUDO	SMALL	STRAN	SINTER	DESC	DRMIN	SGTRAN	SGINT	SGMIN	SEV
258	154	KM(10)		integer test parameters												
260	154	KM(1)	NMAX	maximum number of steps		S					U					
26C	155	KM(2)	IB	step accuracy selector		S						U				
270	156	KM(3)	LX	maximum number of extrapolations		S						U				
274	157	KM(4)	LI	maximum number of interpolations		S						U				
278	158	KM(5)	LCTC	consecutive tangent crossings		S						U				
27C	159	KM(6)		minimum forcing parameter						U				U		
280	160	KM(7-10)		not used												
290	164	CSG(10)		double pr. parameters for segment test						CAP2	CAP3	CAP3		CAP2B		
290	164	CSG(1)	THD1	threshold for lim. interp. bypass		S									U	
298	165	CSG(2)	THD2	threshold for tang. cross. bypass		S									U	
2A0	168	CSG(3-10)		not used												
2E0	184	KSG(10)		integer params. for segment test												
2E0	184	KSG(1)	LB3	maximum number of iterations		S									U	
2E4	185	KSG(2)	LB4	consecutive tangent crossings		S									U	
2E8	186	KSG(3-10)		not used												
308	194	KMID(64)		DESC-DRMIN communication area											not	
308	194	KMID(1)	NPAR(1),NEV	evaluations							U	S			used	
30C	195	KMID(2)	NPAR(2),NX	extrapolations							U	S				
310	196	KMID(3)	NPAR(3),NLI	linear interpolations							U	S				
314	197	KMID(4)	NPAR(4),NNM	Newton methods							U	S				
318	198	KMID(5)	NPAR(5),NTC	tangent crossings							U	S				
31C	199	KMID(6)	NPAR(6),NBS	binary searches							U	S				
320	200	KMID(7-64)		not used												
408	258	KHI		no. of subcomponents for p-c fn. eval.						S	not	not		S		U
40C	259	KLIST(31)		list of subcomponents for p-c fn. eval.						S	used	used		S		U
488	290	KVAC(3)		vacuous object indicators		W		S		U				U		CAP4
500	320	KSN(3)		no. of subcomponents per object		S		CAP3		U				U		
578	350	KSURF(30,15)		lists of subcomponents by object		S				U				U		
C80	800	KTYPE(120)		type of subcomponent		S										U
E60	920	KCONV(120)		not used												

Layout of COMMON

Byte (hex.)	Word (dec.)	Name of Variable	Alternate Names	Meaning	ID26	IDEX	SUDO	SMALL	STRAN	SINTER	DESC	DRMIN	SGTRAN	SGINT	SGMIN	SEV
1040	1040	CSX(15)		coordinates of center of component sphere		GAP1	S	S	GAP3	GAP3				GAP3		
1078	1070	CSY(15)					S	S								
1130	1100	CSZ(15)					S	S								
11A8	1130	CSR(15)		radius of component sphere			S	S								
1220	1160	CPX(30)		coordinates of center of object sphere		W	not	S		U				U		
1310	1220	CPY(30)				W	used	S		U				U		
1400	1280	CPZ(30)				W		S		U				U		
14F0	1340	CPR(30)		radius of object sphere		W		S		U				U		
15B0	1400	CA(120,15)		subcomponent parameters		W		not	S	not				GAP4		U
4E20	5000	SGX1(10)		coordinates of first end point of segment		not		used	not	used			S	U		not
4E70	5020	SGY1(10)				used			used				S	U		used
4EC0	5040	SGZ1(10)											S	U		
4F10	5060	SGX2(10)		coordinates of second end point of segment									S	U		
4F60	5080	SGY2(10)											S	U		
4FB0	5100	SGZ2(10)											S	U		
5000	5120	SGDX(10)		differences between end point coordinates									S	U		
5050	5140	SGDY(10)											S	U		
50A0	5160	SGDZ(10)											S	U		
50F0	5180	SGLS(10)		length of segment squared									S	U		
5140	5200	SGL(10)		length of segment									S	U		

INTERSECTION DETECTION IN THREE DIMENSIONS

SYSTEMS MATERIAL

II. COMPUTATION OF SPHERES ENCLOSING OBJECTS

The input parameters T1-T10 are mapped by the subroutines ID26 into the parameters P1-P10. In the case of the components subject to rotation (ELL through BOX), P1-P6 correspond to T1-T6 in the natural order, but for the other parameters the mapping is

T7 into P8
T8 into P9
T9 into P10
T10 into P7

Thus the Eulerian angles for a rotation in space are in P8, P9,P10.

A. SPHERES ENCLOSING COMPONENTS

The subroutine SUDO computes the radius and the center of the smallest sphere containing each component as shown in the table below. These values are stored in CSR(J) and (CSX(J), CSY(J), CSZ(J)), where J is the component number. If the component is unbounded these values are set to zero. In the table on the following page m = max (P4,P5).

Component Type	CSR	(CSX,CSY,CSZ)
1	0	(0,0,0)
2	0	(0,0,0)
3	P4	(P1,P2,P3)
4	max (P4,P5,P6)	(P1,P2,P3)
5	$\sqrt{m^2 + P6^2}$	(P1,P2,P3)
6, 7, 8	if m>P6, CSR=m if m≤P6, CSR=$\frac{m^2+P6^2}{2P6}$	(P1 + a sinP8sinP9, P2 - a cosP8sinP9, P3 + a cosP9) where a = min (CSR,P6)
9	$\sqrt{P4^2 + P5^2 + P6^2}$	(P1,P2,P3)

B. SPHERES ENCLOSING OBJECTS

The subroutine SMALL computes the center and radius of a small sphere containing an object by pairwise comparison of the spheres containing its components, as explained in the comments in the listing. When two spheres overlap, the computation of the smallest sphere containing their intersection gives rise to three cases. Let r_1 and r_2 denote the radii of the spheres, assume that $r_1 \geq r_2$, and let d denote the distance between their centers. Then

a) if $r_1 - r_2 \geq d$, the larger sphere contains the smaller one, and the latter is the desired sphere;

b) otherwise, if $r_1^2 - r_2^2 \geq d^2$, the smaller sphere is still the desired sphere;

c) otherwise, the desired sphere has center $\alpha C_1+(1-\alpha)C_2$ and radius $\sqrt{r_2^2 - (\sigma\cdot d)^2}$ where

$$\sigma = .5\left(1 - \frac{r_1^2 - r_2^2}{d^2}\right)$$

and C_1 and C_2 denote the centers of the two spheres.

III. COMPUTATION OF THE PSEUDO-CHARACTERISTIC FUNCTION

The computation of the p-c function and its gradient involves four steps, of which the first two are executed (by IDEX and STRAN respectively) when an object is defined, and the last two are executed (by SEV) when an intersection test is made:

a) break down a component into its subcomponents,
b) compute the subcomponent coefficients,
c) evaluate the g function for a subcomponent,
d) evaluate the p-c function.

The second and third step are closely related and will be discussed together in the sequel.

A. COMPONENT AND SUBCOMPONENT TYPES. LINEARIZATION.

Each component is represented internally by one or more subcomponents. For some of the subcomponents it is possible to choose between two forms of the g function, one of which is "linearized" in the sense that the value of g is an asymptotically linear function of the distance from the surface g = 0. For example, the g function for an ellipse has the non-linear form

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1$$

and the linearized form

$$\sqrt{\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 + 1.25^2} - 1.25 \quad .$$

The numbers (types) associated with the various subcomponents are shown in the following table, where the position of the slabs is given before rotation and translation.

Type non-lin.	Type lin.	Description
	1	half-space
2	22	slab
3	23	sphere
4	24	ellipsoid
5	25	infinite cylinder
	6	hyperboloid representing cone (see 8)
7		infinite paraboloid
	8	one sheet of elliptic hyperboloid
9	29	slab, center at origin, perpendicular to X-axis
10	30	slab, center at origin, perpendicular to Y-axis
11	31	slab, center at origin, perpendicular to Z-axis
12	32	slab in upper half-space, perpendicular to Z-axis

Table of Subcomponents

The correspondence between components and subcomponents is given by either of the arrays KTT and KTL, where each non-zero entry refers to a subcomponent. These arrays are set at initialization time. The selection between the arrays KTT and KTL is determined for each component (at object definition time) by the contents of the corresponding entry of the array KTS. An entry of 0 indicates that KTT is to be selected. At initialization time KTS is set to 1,0,0,0,0,0,0,0,0. The contents of KTS (except for KTS(1)) can be changed by means of a call to IDPR4, where non-zero values for the parameter 2 through 9 will produce non-zero values in the corresponding positions in KTS.

Component Name	Component Type L	Subcomponent Type KTT(L,M)			Subcomponent Type KTL(L,M)		
HSP	1	1	0	0	1	0	0
SLA	2	2	0	0	22	0	0
SPH	3	3	0	0	23	0	0
ELL	4	4	0	0	24	0	0
CYL	5	5	11	0	25	31	0
CON	6	6	12	0	6	32	0
PAR	7	7	12	0	7	32	0
HYP	8	8	12	0	8	32	0
BOX	9	9	10	11	29	30	31

B. COEFFICIENT COMPUTATION AND g-FUNCTION EVALUATION

The subcomponent coefficients are computed by STRAN and stored in the array CA, and they are used by SEV to evaluate the g function and its gradient. These two processes are described below for the individual subcomponent types. The description of the function and gradient evaluation does not give every detail of the code, where more temporary variables may be used, but should be sufficient to an understanding of it.

To simplify the notation, the following correspondence is established between the symbols used here and the variables that appear in the subroutines STRAN and SEV

Symbols used here	Symbols used in STRAN, SEV	Meaning
$p_1, p_2, \ldots$	P1,P2,...	input parameters
$a_1, a_2, \ldots$	CA(I,1),CA(I,2),...	coefficients of i-th subcomponent
$e_{11}, e_{12}, \ldots$	E11,E12,...	rotation coefficients
$t_1, t_2, \ldots$	T1,T2,...	temporaries
g	G	g_1 function
g_x, g_y, g_z	GX,GY,GZ	components of grad g_1

Type 1. Half-space

The function to be evaluated is

$$g = - \frac{p_4(x-p_1) + p_5(y-p_2) + p_6(z-p_3)}{\sqrt{p_4^2 + p_5^2 + p_6^2}}$$

Coefficient computation:

$$t_1 = -1 \; / \; \sqrt{p_4^2 + p_5^2 + p_6^2}$$

$$a_1 = t_1 p_4$$

$$a_2 = t_1 p_5$$

$$a_3 = t_1 p_6$$

$$a_4 = -t_1(p_1 p_4 + p_2 p_5 + p_3 p_6)$$

Function and gradient evaluation:

$$g = a_1 x + a_2 y + a_3 z + a_4$$

$$g_x = a_1$$

$$g_y = a_2$$

$$g_z = a_3$$

Type 2. Slab

The function to be evaluated is

$$g = \frac{1}{2p_7} \left\{ \left[\frac{p_4(x-p_1) + p_5(y-p_2) + p_6(z-p_3)}{\sqrt{p_4^2 + p_5^2 + p_6^2}} \right]^2 - p_7^2 \right\}$$

Coefficient computation:

$$t_1 = 1 \; / \; \sqrt{2p_7(p_4^2 + p_5^2 + p_6^2)}$$

$$a_1 = t_1 p_4$$

$$a_2 = t_1 p_5$$

$$a_3 = t_1 p_6$$

$$a_4 = -t_1(p_1 p_4 + p_2 p_5 + p_3 p_6)$$

$$a_5 = -.5 p_7$$

Function and gradient evaluation:

$$t_1 = a_1x + a_2y + a_3z + a_4$$

$$g = t_1^2 + a_5$$

$$g_x = 2t_1a_1$$

$$g_y = 2t_1a_2$$

$$g_z = 2t_1a_3$$

Type 22. Slab, linearized

The function to be evaluated is

$$g = \sqrt{\left[\frac{p_4(x-p_1) + p_5(y-p_2) + p_6(z-p_3)}{\sqrt{p_4^2 + p_5^2 + p_6^2}}\right]^2 + .5625p_7^2 - 1.25p_7}$$

Coefficient computation:

$$t_1 = 1/\sqrt{p_4^2 + p_5^2 + p_6^2}$$

$$a_1 = t_1p_4$$

$$a_2 = t_2p_4$$

$$a_3 = t_3p_4$$

$$a_4 = -t_1(p_1p_4 + p_2p_5 + p_3p_6)$$

$$a_5 = .5625\ p_7^2$$

$$a_6 = -1.25\ p_7$$

Function and gradient evaluation:

$$t_1 = a_1x + a_2y + a_3z + a_4$$

$$t_2 = \sqrt{t_1^2 + a_5}$$

$$g = t_2 + a_6$$

$$g_x = (t_1/t_2)a_1$$

$$g_y = (t_1/t_2)a_2$$

$$g_z = (t_1/t_2)a_3$$

Type 3. Sphere

The function to be evaluated is

$$g = \frac{1}{2p_4}\left[(x-p_1)^2 + (y-p_2)^2 + (z-p_3)^2 - p_4^2\right]$$

Coefficient computation:

$$a_1 = -p_1$$

$$a_2 = -p_2$$

$$a_3 = -p_3$$

$$a_4 = -p_4^2$$

$$a_5 = \frac{1}{2p_4}$$

Function and gradient evaluation:

$t_1 = x + a_1$

$t_2 = y + a_2$

$t_3 = z + a_3$

$g = a_5(t_1^2 + t_2^2 + t_3^2 + a_4)$

$t_4 = 2a_5$

$g_x = t_4 t_1$

$g_y = t_4 t_2$

$g_z = t_4 t_3$

Type 23. Sphere, linearized

The function to be evaluated is

$$g = \sqrt{(x-p_1)^2 + (y-p_2)^2 + (z-p_3)^2 + .5625p_4^2} - 1.25p_4$$

Coefficient computation:

$a_1 = -p_1$

$a_2 = -p_2$

$a_3 = -p_3$

$a_4 = .5625p_4^2$

$a_5 = -1.25p_4$

Function and gradient evaluation:

$t_1 = x + a_1$

$t_2 = y + a_2$

$t_3 = z + a_3$

$t_4 = \sqrt{t_1^2 + t_2^2 + t_3^2 + a_4}$

$g = t_4 + a_5$

$g_x = t_1/t_4$

$g_y = t_2/t_4$

$g_z = t_3/t_4$

Quadric Surfaces

The following subcomponents are bounded by quadric surfaces, and some of the internal processing applied to them is independent of the specific type:

type 4	ellipsoid
type 24	ellipsoid, linearized
type 5	cylinder
type 25	cylinder, linearized
type 6	cone
type 7	paraboloid
type 8	hyperboloid

Type 6 is treated by setting P7 = tolerance, and is then processed as a hyperboloid.

The expressions for the g functions for the other types will now be derived, rather than given directly, since these subcomponents are subject to rotation and translation.

If no rotation and translation were applied, the g functions would have the following form, except for a normalizing factor:

Type 4 $$\frac{x^2}{p_4^2} + \frac{y^2}{p_5^2} + \frac{z^2}{p_6^2} - 1$$

Type 24 $$\sqrt{\frac{x^2}{p_4^2} + \frac{y^2}{p_5^2} + \frac{z^2}{p_6^2} - 1 + 1.5625} - 1.25$$

Type 5 $$\frac{x^2}{p_4^2} + \frac{y^2}{p_5^2} - 1$$

Type 25 $$\sqrt{\frac{x^2}{p_4^2} + \frac{y^2}{p_5^2} - 1 + 1.5625} - 1.25$$

Type 7 $$\frac{x^2}{p_4^2} + \frac{y^2}{p_5^2} - \frac{z}{p_6^2}$$

Type 8 $$\sqrt{\frac{x^2}{p_4^2} + \frac{y^2}{p_5^2} + \frac{p_7^2}{p_5(p_6+2p_7)}} - \frac{z + p_7}{\sqrt{p_6(p_6+2p_7)}}$$

In order to discuss the effect of a rotation on the quadratic expressions above (and neglecting for type 8 the term outside the square root), let $x_1x_2x_3$ denote the coordinate system fixed in space and $u_1u_2u_3$ the coordinate system fixed with respect to the object. Let e_{ij} be the rotation matrix such that

$$u_i = \sum_j e_{ij}x_j$$

The e's are computed from the Eulerian angles as follows:

$$\phi = (\pi/180)p_8 \qquad \theta = (\pi/180)p_9 \qquad \psi = (\pi/180)p_{10}$$

$$\begin{aligned}
e_{11} &= \cos\phi\cos\psi - \sin\phi\sin\psi\cos\theta \\
e_{12} &= \cos\phi\sin\psi + \sin\phi\cos\psi\cos\theta \\
e_{13} &= \sin\phi\sin\theta \\
e_{21} &= -\sin\phi\cos\psi - \cos\phi\sin\psi\cos\theta \\
e_{22} &= -\sin\phi\sin\psi + \cos\phi\cos\psi\cos\theta \\
e_{23} &= \cos\phi\sin\theta \\
e_{31} &= \sin\psi\sin\theta \\
e_{32} &= -\cos\psi\sin\theta \\
e_{33} &= \cos\theta
\end{aligned}$$

In terms of the u's the functions to be evaluated can be written as

$$\Sigma A_{ij}u_iu_j + \Sigma B_iu_i + C ,$$

and expressed in terms of the x's as

$$\sum_{ij} A_{ij}\left(\sum_k e_{ik}x_k\right)\left(\sum_\ell e_{j\ell}x_\ell\right) + \sum_i B_i\left(\sum_k e_{ik}x_k\right) + C .$$

For the subcomponents used here the only coefficients that may be non-zero are

$$A_{11} \quad A_{22} \quad A_{33} \quad B_3 \quad C$$

and they will be denoted respectively by

$$t_1 \quad t_2 \quad t_3 \quad t_4 \quad t_5 .$$

If x_1, x_2, x_3 are now replaced by x,y,z, and all the necessary simplifications are carried out, the expression for the rotated quadric becomes

$$a_1x^2+a_2y^2+a_3z^2+a_4xy+a_5xz+a_6yz+t_4e_{31}x+t_4e_{32}y+t_4e_{33}z+t_5$$

where

$$(*)\begin{cases} a_1 = t_1e_{11}^2 + t_2e_{21}^2 + t_3e_{31}^2 \\ a_2 = t_1e_{12}^2 + t_2e_{22}^2 + t_3e_{32}^2 \\ a_3 = t_1e_{13}^2 + t_2e_{23}^2 + t_3e_{33}^2 \\ a_4 = 2(t_1e_{11}e_{12} + t_2e_{21}e_{22} + t_3e_{31}e_{32}) \\ a_5 = 2(t_1e_{11}e_{13} + t_2e_{21}e_{23} + t_3e_{31}e_{33}) \\ a_6 = 2(t_1e_{12}e_{13} + t_2e_{22}e_{23} + t_3e_{32}e_{33}) \end{cases}$$

For a translation of coordinates one must now replace x,y,z by $x-p_1$, $y-p_2$, $z-p_3$ respectively. This does not affect the quadratic terms, so the final form is

$$Q \equiv a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5xz + a_6yz + a_7x + a_8y + a_9z + a_{10}$$

where a_1 through a_6 are the same as above, and

$$(*)\begin{cases} a_7 = t_4e_{31} - 2a_1p_1 - a_4p_2 - a_5p_3 \\ a_8 = t_4e_{32} - 2a_2p_2 - a_4p_1 - a_6p_3 \\ a_9 = t_4e_{33} - 2a_3p_3 - a_5p_1 - a_6p_2 \\ a_{10} = a_1p_1^2 + a_2p_2^2 + a_3p_3^2 + a_4p_1p_2 + a_5p_1p_3 + a_6p_2p_3 \\ \qquad -t_4(e_{31}p_1 + e_{32}p_2 + e_{33}p_3) + t_5 \end{cases}$$

The computation of the coefficients a_i for the individual types requires the values of the variables t_1 through t_5. They are given in the following table, which shows also the values used for the normalizing factor a_{11}.

Type	t_1	t_2	t_3	t_4	t_5	a_{11}
4	$1/p_4^2$	$1/p_5^2$	$1/p_6^2$	0	-1	.5 max(p_4,p_5,p_6)
24						max(p_4,p_5,p_6)
5	$1/p_4^2$	$1/p_5^2$	0	0	-1	.5 max(p_4,p_5)
25						max(p_4,p_5)
7	$1/p_4^2$	$1/p_5^2$	0	$-1/p_6$	0	max(.5p_4,.5p_5,p_6)
8	$1/p_4^2$	$1/p_5^2$	0	0	$p_7^2/(p_6(p_6+2p_7))$	$p_6 + p_7$

The following additional provisions for coefficient computation are made:

1. In the special case when P8=P9=P10 and there is no rotation, the computation of a_1 through a_{10} is done by using simpler formulas equivalent to (*).

2. For types 24 and 25 the value of a_{10} given by (*) is replaced by $a_{10}+1.5625$, and the additional coefficient a_{12} = -1.25 is set.

3. For <u>type 8</u> after the computation of a_1 through a_{11} the following steps are executed

$$t_4 = -\sqrt{1/(p_6(p_6+2p_7))}$$

$$t_5 = t_4 p_7$$

$$a_{12} = t_4 e_{31}$$

$$a_{13} = t_4 e_{32}$$

$$a_{14} = t_4 e_{33}$$

$$a_{15} = t_4(e_{31}p_1+e_{32}p_2+e_{33}p_3)+t_5$$

The evaluation of the g function and its gradient are done according to the following formulas, where Q is the value of the quadratic function shown above:

$$t_2 = 2a_1 x + a_4 y + a_5 z + a_7$$

$$t_3 = 2a_2 y + a_4 x + a_6 z + a_8$$

$$t_4 = 2a_3 z + a_5 x + a_6 y + a_9$$

Types 4,5,7:

$$g = a_{11} Q$$

$$g_x = a_{11} t_2$$

$$g_y = a_{11} t_3$$

$$g_z = a_{11} t_4$$

Types 24,25:

$$g = a_{11}(\sqrt{Q} + a_{12})$$

$$t_5 = .5\, a_{11}/\sqrt{Q}$$

$$g_x = t_5 t_2$$

$$g_y = t_5 t_3$$

$$g_z = t_5 t_4$$

Type 8:

$$g = a_{11}(Q+a_{12}x+a_{13}y+a_{14}z+a_{15})$$

$$t_1 = .5/\sqrt{Q}$$

$$g_x = a_{11}(t_1 t_2+a_{12})$$

$$g_y = a_{11}(t_1 t_3+a_{13})$$

$$g_z = a_{11}(t_1 t_4+a_{14})$$

<u>Slabs</u>

The subcomponents of type 9 through 12 are slabs and those of type 29 through 32 are slabs in linearized form. Their g functions are as follows.

<u>Type 9</u>

$$g = \frac{1}{2p_4}\left\{\left[e_{11}(x-p_1)+e_{12}(y-p_2)+p_{13}(z-p_3)\right]^2 - p_4^2\right\}$$

Type 10

$$g = \frac{1}{2p_5}\left\{\left[e_{21}(x-p_1)+e_{22}(y-p_2)+e_{23}(z-p_3)\right]^2 - p_5^2\right\}$$

Type 11

$$g = \frac{1}{2p_6}\left\{\left[e_{31}(x-p_1)+e_{32}(y-p_2)+e_{33}(z-p_3)\right]^2 - p_6^2\right\}$$

Type 12

$$g = \frac{1}{p_6}\left\{\left[e_{31}(x-p_1)+e_{32}(y-p_2)+e_{33}(z-p_3) - .5p_6\right]^2 - .25p_6^2\right\}$$

Type 29

$$g = \sqrt{\left[e_{11}(x-p_1)+e_{12}(y-p_2)+e_{13}(z-p_3)\right]^2 + .5625p_4^2 - 1.25p_4}$$

Type 30

$$g = \sqrt{\left[e_{21}(x-p_1)+e_{22}(y-p_2)+e_{23}(z-p_3)\right]^2 + .5625p_5^2 - 1.25p_5}$$

Type 31

$$g = \sqrt{\left[e_{31}(x-p_1)+e_{32}(y-p_2)+e_{33}(z-p_3)\right]^2 + .5625p_6^2 - 1.25p_6}$$

Type 32

$$g = \sqrt{\left[e_{31}(x-p_1)+e_{32}(y-p_2)+e_{33}(z-p_3) - .5p_6\right]^2 + .140625p_6^2 - .625p_6}$$

Coefficient computation for types 9 through 12:

Type→	9	10	11	12
t_1	$1/\sqrt{2p_4}$	$1/\sqrt{2p_5}$	$1/\sqrt{2p_6}$	$1/\sqrt{p_6}$
t_2	e_{11}	e_{21}	e_{31}	e_{31}
t_3	e_{12}	e_{22}	e_{32}	e_{32}
t_4	e_{13}	e_{23}	e_{33}	e_{33}
a_5	$-.5p_4$	$-.5p_5$	$-.5p_6$	$-.25p_6$
a_4	$-t_1(t_2p_1+t_3p_2+t_4p_3)$			$-t_1(t_2p_1+t_3p_2+t_4p_3+.5p_6)$
a_1	t_1t_2			
a_2	t_1t_3			
a_3	t_1t_4			

The function and gradient evaluation is the same as for type 2.

Coefficient computation for types 29 through 32:

Type→	29	30	31	32
a_1	e_{11}	e_{21}	e_{31}	e_{31}
a_2	e_{12}	e_{22}	e_{32}	e_{32}
a_3	e_{13}	e_{23}	e_{33}	e_{33}
a_5	$.5625p_4^2$	$.5625p_5^2$	$.5625p_6^2$	$.140625p_6^2$
a_6	$-1.25p_4$	$-1.25p_5$	$-1.25p_6$	$-.625p_6$
a_4	$-(a_1p_1+a_2p_2+a_3p_3)$			$-(a_1p_1+a_2p_2+a_3p_3+.5p_6)$

The function and gradient evaluation is the same as for type 22.

C. PSEUDO-CHARACTERISTIC FUNCTION EVALUATION

The formulas for the p-c function and its gradient are given in section VI of the User Information. The p-c function involves a sum containing one term for each of the relevant subcomponents. The indices of these subcomponents are placed in the array KLIST by the subroutine SINTER or SGINT prior to the evaluation of the p-c function (cf. next section). The subroutine SEV evaluates both the g function for each component and the p-c function. The correspondence between the symbols used in this write-up and those used in the code is given on the next page.

Symbols used in write-up	Symbols used in code
x,y,z	CX,CY,CZ
g or g_i	G
v_i	CV
V	V
p-c function or G	QQ
g_x, g_y, g_z or grad g	GX,GY,GZ
grad v_i	CWX,CWY,CWZ
grad V	WX,WY,WZ
grad G	QX,QY,QZ

INTERSECTION DETECTION IN THREE DIMENSIONS

SYSTEMS MATERIAL

IV. OBJECT-OBJECT INTERSECTION TEST

The first phase of the object-object test is done by the subroutine SINTER, which first determines whether the spheres enclosing the two objects intersect. If they do not, neither do the objects intersect; if they do, the smallest sphere (say SS) containing their intersection is computed by the method explained in Section II.B. A test is then executed to decide whether SS intersects the region sphere. If so, a further test determines whether

(a) SS lies entirely inside the region sphere, or

(b) SS lies partly outside the region sphere.

Several special cases arise when either or both of the object-enclosing spheres are unbounded.

When the above tests do not exclude the possibility of the objects intersecting inside the region sphere, the p-c function method must be used, and SINTER enters the indices of the subcomponents comprising the two objects in the array KLIST. In case (b) above, the index of the region sphere is also entered. SINTER then calls the subroutine DESC.

The method used in DESC is explained in the reference given in note [3] in the User Information. In summary it consists of computing a sequence of points approaching the minimum of the p-c function. Each iteration involves two phases: first DESC computes the direction of a half-line on which the next point is to be located and the length of a tentative step toward that point; then DRMIN computes the point itself, i.e.

the point on the given half-line where the p-c function attains it minimum.

The minimum (not the directional minimum) of the p-c function is deemed to have been reached if the length of its gradient is less that EPS. The setting of EPS is discussed in the section on Program Parameters. If the test cannot be concluded within NMAX iterations, it is considered undecidable.

The computation of the directional minimum in DRMIN is also divided into two phases: first a point has to be located on the far side of the minimum, i.e. where the directional derivative is positive (it is always negative at the starting point); then the "interval of uncertainty" between two points on opposite sides of the minimum is iteratively shortened.

If the tentative step length computed by DESC leads to a point on the far side of the minimum, the first phase is concluded. Otherwise successive step increments are added until such a point is reached; except that if the point is not reached within LX iterations the test is considered undecidable. The parameter SM is used in computing the above step increments.

Once two points have been found where the directional derivative has opposite signs, the following four methods are used in order to locate its zero, with various rules (explained below) to decide which method to use next:

(a) Linear interpolation on the values of the directional derivative.

(b) Modified Newton's method. This is explained with reference to Figure 6, where the same notation is used as in the code, except for the labels given to the points:

STI : length of step from starting point to end point of interval of uncertainty where directional derivative is negative;

DL : length of interval of uncertainty;

DLI.DL2: subintervals of DL terminating at point being computed.

The two cases illustrated are those where at the last computed point (P1) the directional derivative is negative (indicated in the code by I=1). In Figure 6(a) the point P1 is above the line P1'-P2 (indicated in the code by J1=1), and the new point (P) is given by the intersection of the line P1'-P1 with the axis. In Figure 6(b), P1 is below the line P1'-P2 (J1=2), and P is taken as the leftmost of the following two points: the point where P1'-P1 intersects the axis and the point where a parabola with vertex at P1 going through P2 intersects the axis.

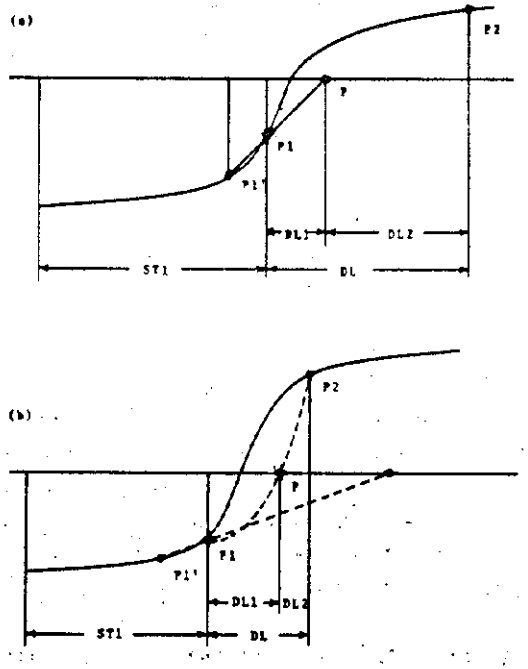

Figure 6. Modified Newton's Method

(c) Tangent-crossing method. This method uses the values of the p-c function as well as those of the directional derivative. It is illustrated in Figure 7, where the upper curve represents the p-c function and the lower curve the directional derivative. The abscissa of the tangent-crossing point is obtained by solving for DL1 the system of equations.

$$y - F1 = DOT1 \cdot DL1$$
$$y - F2 = DOT2 \cdot (DL1 - DL)$$

where F1,F2 are the function values and DOT1,DOT2 are the directional derivative values. The solution is

$$DL1 = \frac{DL.DOT2 - (F2-F1)}{DOT2 - DOT1}$$

In the code the quantity (DOT2-DOT1)/DL is denoted by SLOPE.

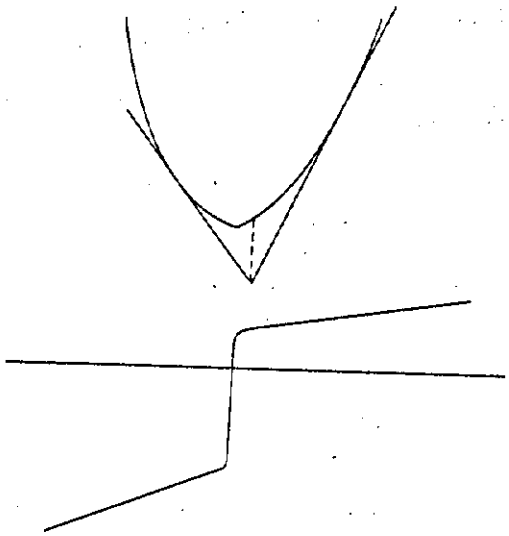

Figure 7. Tangent-Crossing Method

(d) Halving the interval of uncertainty (i.e. binary search).

The following rules determine which of the four interpolation methods is used:

- apply linear interpolation; if ratio (DODO) of last computed value of directional derivative to previously computed value having same sign is less than TH1, apply Newton's method; else apply tangent crossing method;
- after Newton's method, if at last computed point the directional derivative has same sign as at immediately preceding point, apply tangent crossing; else repeat linear interpolation;
- tangent crossing is applied LCTC times in succession, followed by linear interpolation;
- if at any time DL1 or DL2 is less that SHORT, apply binary search, then repeat linear interpolation;
- if DL1 or DL2 is less than SHORT when binary search is being applied, give abnormal return.

The test to decide whether the last computed point is sufficiently close to the minimum is explained in the comments in the code; except that if this criterion cannot be met within LI interpolations, the object-object test is considered undecidable (see also section VI).

INTERSECTION DETECTION IN THREE DIMENSIONS

SYSTEMS MATERIAL

V. SEGMENT-OBJECT INTERSECTION TEST

The segment-object intersection test is executed by SGINT and SGMIN. Many of the techniques used are similar to those used for the object-object intersection test. The following points are worthy of note.

1. The smallest sphere containing the intersection of the sphere enclosing the object and the region sphere is computed. Its center has coordinates CX, CY, CZ and the square of its radius is RBARS.

2. The end points of the segment have coordinates X1, Y1, Z1 and X2, Y2, Z2. To find whether the line containing the segment intersects the sphere, the product of the length of the segment times the radius of the sphere is compared with twice the area of the triangle determined by the segment and the center of the sphere (actually in the code the squares of these quantities are compared).

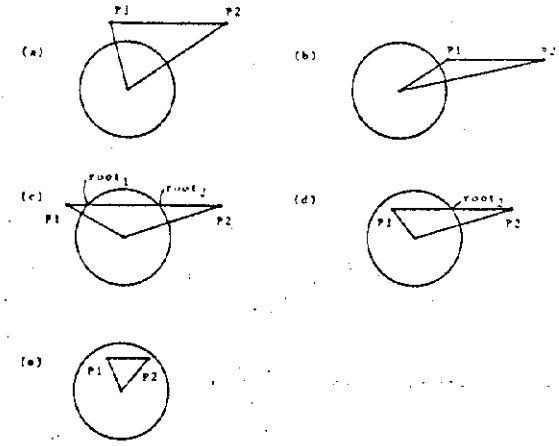

Figure 8. Relative Positions of Segment and Sphere

3. If the line intersects the sphere, the endpoint of the segment nearer the center of the sphere is selected. Figure 8 illustrates the case where P1 is that point. The next test (Figure 8(b)) deals with the case that P1 is outside the sphere and the angle at P1 is obtuse.

4. In the remaining cases (c, d and e) the segment intersects the sphere. If one or both of its endpoints are outside the sphere, their coordinates are replaced by those of the corresponding intercept of the segment and the sphere. For example, in case (d) the coordinates of the point $\underline{root}_2$ are stored in X2, Y2, Z2.

5. The steps above determine a segment (either the original one or a portion of it) where the minimum of the p-c function is to be sought. To test whether the minimum occurs at an end point, the direction of the gradient at the end point is compared with the direction of the segment.

6. The minimum of the p-c function on the segment is now approached by SGMIN. The methods used are linear interpolation on the directional derivative, tangent crossing on the function curve, and binary search using the directional derivative. Linear interpolation is used as long as it produces a ratio (DODO) of current to previous value of the directional derivative less than THD1. Otherwise, tangent crossing is applied LB4 times, then linear interpolation is applied again. Binary search is used only when tangent crossing produces a subinterval smaller than THD2 times the previous interval. The test is considered undecidable if the procedure iterates more than LB3 times. The minimum of the p-c function on the segment is deemed to have been found if the directional derivative is less than EPS. The value of EPS at any given time is equal to CPICT(4) divided by the current interval of uncertainty. Regarding CPICT(4), see next section.

VI. PROGRAM PARAMETERS

The following table gives the initial settings of the program parameters and describes briefly their use.

Common Name	Local Name	Initial Setting	Where Used	Meaning
KM(1)	NMAX	60	DESC	max. number of iterations
KM(2)	IB	3	DRMIN	= 1: use cosine bound test = 2: use step accuracy test = 3: use both
KM(3)	LX	60	DRMIN	max. number of extrapolations
KM(4)	LI	25	DRMIN	max. number of interpolations
KM(5)	LCTC	4	DRMIN	consecutive tangent crossings
KM(6)		0	SINTER SGINT etc.	+1: normal intersection test, = 1: find minimum of p-c function
CM(1)		.5	IDEX	used in computing CPICT(4);see below
CM(2)		16.	SINTER	used in computing loss of precision parameter SHORT=10.**(-CM(2))* DMAX1(CX,CY,CZ,RBAR)
CM(3)	SM	6.	DRMIN	step multiplier
CM(4)	BC	1.D-10	DRMIN	cosine bound
CM(5)		1.	SINTER	step accuracy factor;see below
CM(6)	TH1	.5	DRMIN	threshold on DODO for bypassing Newton's method
CM(7)		1.D-3	SINTER	upper bound on BR; see below

CM(8)	1.D-3	SINTER	lower bound on BR; see below
KSG(1) LB3	60	SGMIN	max. number of interpolations
KSG(2) LB4	4	SGMIN	consecutive tangent crossings
CSG(1) THD1	.5	SGMIN	threshold on DODO for switching to tangent crossing method
CSG(2) THD2	1.D-5	SGMIN	loss of precision parameter for switching to binary search

In DESC and DRMIN the minimum of the p-c function is deemed to have been found if at a given point the gradient of the p-c function is less than

```
EPS = CPICT(4)/(2.*RBAR)
```

where

```
CPICT(4) = CM(1)*CT ;
```

CT is the tolerance and RBAR is the radius of the sphere containing the intersection of the objects.

In DRMIN two criteria may be used for deciding that the directional minimum has been located: the cosine bound test and the step accuracy test. The former means that the cosine of the angle between the step direction and the gradient direction is less than BC. The latter means that the ratio of the interval of uncertainty to the length of the step is less than

```
BR = DMIN1(DMAX1(CM(8),CT*CM(5)/RBAR),CM(7)) .
```

The values of KM(1) through KM(6) can be changed by means of a call to IDPR1; the first six subroutine parameters correspond to these program parameters. In a call to IDPR2 the first eight parameters specify the values of CM(1) through CM(8). In a call to IDPR3 the first, second, fifth and sixth parameter specify KSG(1), KSG(2), CSG(1) and CSG(2) respectively.