9950-695

Report No. 955322 JPL

PLASIM: A COMPUTER CODE FOR SIMULATING CHARGE-EXCHANGE PLASMA PROPAGATION

Final Report and Program Documentation

by

Raymond S. Robinson
William D. Deininger
Dale R. Winder
and
Harold R. Kaufman

January 1982

Department of Physics
Colorado State University
Fort Collins, CO 80523

This code was developed pursuant to Jet Propulsion Laboratory Contract No. 955322.

Report No. 955322 JPL

PLASIM: A COMPUTER CODE FOR SIMULATING CHARGE-EXCHANGE PLASMA PROPAGATION

Final Report and Program Documentation

by

Raymond S. Robinson
William D. Deininger
Dale R. Winder
and
Harold R. Kaufman

January 1982

Department of Physics
Colorado State University
Fort Collins, CO 80523

This code was developed pursuant to Jet Propulsion Laboratory Contract No. 955322.

ABSTRACT

The propagation of the charge-exchange plasma for an electrostatic ion thruster is crucial in determining the interaction of that plasma with the associated spacecraft. A model that describes this plasma and its propagation is described, together with a computer code based on this model.

The structure and calling sequence of the code, named PLASIM, is described. An explanation of the program's input and output is included, together with samples of both. The code is written in ANSI Standard Fortran IV.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Ion thrusters can be used in a variety of primary and auxiliary space-propulsion applications. A thruster produces a charge-exchange plasma which can interact with various systems of the spacecraft. In order to understand these possible interactions, a detailed knowledge of the plasma propagation is required.

The production of charge-exchange ions by thrusters has been understood for some time.[1] Fast ions from the thruster interact with slow neutrals that are also escaping, resulting in the production of ions that initially have only a thermal velocity. The electric fields within the ion beam cause these ions to move approximately radially out of the ion beam. These charge-exchange ions leave the ion beam along with electrons supplied by the neutralizer, the combination constituting the charge-exchange plasma. The propagation of the charge-exchange plasma depends on several factors, including the initial thermal energy of the ions, the distribution of ion production along the beam, and the potentials and geometry of neighboring spacecraft surfaces.

In the THEORY section of this report, the geometry of an idealized spacecraft with an ion thruster is described, together with the simplifications and definitions used in modeling the ion beam. The distribution function used for charge-exchange ion production is also presented, along with the barometric equation that relates the variation in plasma potential to the variation in plasma density. The numerical methods and approximations used for the calculations are then discussed. This section describes the main calculation subroutine, CALC, and the displacement calculation subroutine, CALCD.

In the PROGRAM STRUCTURE section, a flowchart is provided that diagrams the calling sequence of the modules; also presented are detailed descriptions of each of the modules. A guide to using the program is presented in the INPUT AND OUTPUT section of this report. Descriptions of the calculated results are presented in the INPUT AND OUTPUT and VERIFICATION sections.

Also presented is a method of obtaining better resolution in the upstream region. The high-resolution option of the program simulates only the upstream region. This option utilizes previously calculated trajectories as boundaries for the region to be simulated at higher resolution (see notes in the computer code).

The VERIFICATION section of this report also compares experimental and analytic results with those obtained by the computer code. Factors limiting simulation accuracy are also discussed.

An analytic solution is derived for the case of an infinitely long cylindrical beam with a uniform distribution of charge-exchange ion production along the beam. Expressions are obtained for the radial variations in ion density and velocity, permitting a direct comparison with results from the computer code. This analytic solution is described in APPENDIX A and used in the comparison described above.

It should be noted that this final report provides a complete description of the program and supersedes previous reports.[2,3] All of the information necessary to use the program is contained herein. A glossary of the variables used in the computer code is provided in APPENDIX C and the computer code is listed in APPENDIX D.

THEORY

The interaction of an ion thruster with other components of an electrically propelled spacecraft through the plasma surrounding a spacecraft has been studied for some time. The transport of electrons from the ion beam to a solar-array surface was treated first by Knauer, et al.[4] as an electron space-charge-flow problem. Measured electron currents, though, were found to be much higher than calculated by Knauer. The difference was due to the presence of a charge-exchange plasma.

Charge-exchange ions are produced when fast beam ions pass near slow escaping neutrals. The fast neutrals that result usually present no problem, and escape following the directions they had as ions. The slow charge-exchange ions that are produced, though, initially have only the velocity of the thermal neutrals. Small electric fields within the ion beam result in the charge-exchange ions leaving the beam in approximately radial directions. These charge-exchange ions, together with some escaping electrons, form the charge-exchange plasma that surrounds an electrically propelled spacecraft.

The production rate for the charge-exchange ions was first calculated by Staggs, et al.[1] The capability of the charge-exchange plasma to transport electrons to other parts of the spacecraft was experimentally evaluated by Worlock, et al.[5] Some detailed trajectories of charge-exchange ions have been examined by Komatsu, et al.[6] Experimental studies of the charge-exchange plasma distribution, particularly upstream of the ion-beam direction, have been conducted by Kaufman,[7-9] and Carruth, et al.[10-12] Several studies included a correlation of plasma properties in terms of the distance from the thruster and the angle relative to the

beam direction.[9-11] Theoretical studies of the charge exchange plasma have been carried out by Robinson, et al.[3,13] and Katz, et al.[14] The latter treat the ions, numerically, as a cold fluid in contrast to the use of calculated ion trajectories and density gradients.

The physical processes involved in the charge-exchange plasma have become well understood as a result of the various studies that have been conducted. The electron population outside of the beam agrees with the "barometric" equation

$$n_e = n_{e,ref} \exp(-qV/kT_e) \qquad (1)$$

which was introduced by Sellen, et al.[15] and verified by Ogawa, et al.[16,17] for the population within the beam and by Kaufman[7] for the population in the charge-exchange plasma. The plasma potential V in Eq. (1) is taken to be zero at the reference electron density $n_{e,ref}$. The electron temperature T_e in the charge-exchange plasma has been found to be about half the value in the ion beam.[7] The electron temperature in the ion beam varies with thruster size and ranges from about 7 eV for a 5 cm thruster to 5 eV for 15 cm and 0.35 eV for 30 cm. Also, q is the elementary unit of charge and k is Boltzman's constant.

The experimental validity of Eq. (1) is consistent with the low density and long mean-free paths in the charge-exchange plasma. The decreasing plasma density with increasing distance from the thruster forms a potential well for the electrons, so that many transits of this region are probable before an electron escapes. The many transits permit randomization of the electron population to a single Maxwellian distribution by Coulomb collisions.

The extent of the charge-exchange plasma is large compared to the Debye shielding distance, which means that the electron density must everywhere be equal to the ion density. Inasmuch as the ions only move outward from the thruster, their motion is essentially collisionless and governed by the potential distribution from Eq. (1).

The approach used in this study has been to assume a cylindrical, axially symmetric ion beam, with the charge-exchange ions leaving the beam with a uniform velocity in the radial direction. The coordinates and simulation boundaries are defined in Fig. 1. The current density of these charge-exchange ions at the cylindrical beam boundary is a function of the distance downstream from the thruster. The total charge-exchange current is distributed among the total number of trajectories, with this total number specified as an input parameter, N. Approximately fifty percent of the charge-exchange ions are generated within one beam radius of the downstream end of the thruster, so about half of the specified trajectories will initially start in this region.

A trajectory represents the path of a single representative ion on which acceleration is produced by electric fields in the plasma. These fields correspond to potential gradients induced by gradients in the plasma density, as indicated by Eq. (1). Density gradients are used in two separate calculations. The component of the density gradient along the path provides a potential gradient which serves to change the ion velocity in that direction, while the component of the density gradient normal to the path provides a potential gradient which modifies the direction of the path. The forces on the ions acting normal and parallel to the path are resolved into x and z components to obtain the resultant force acting on the ion.

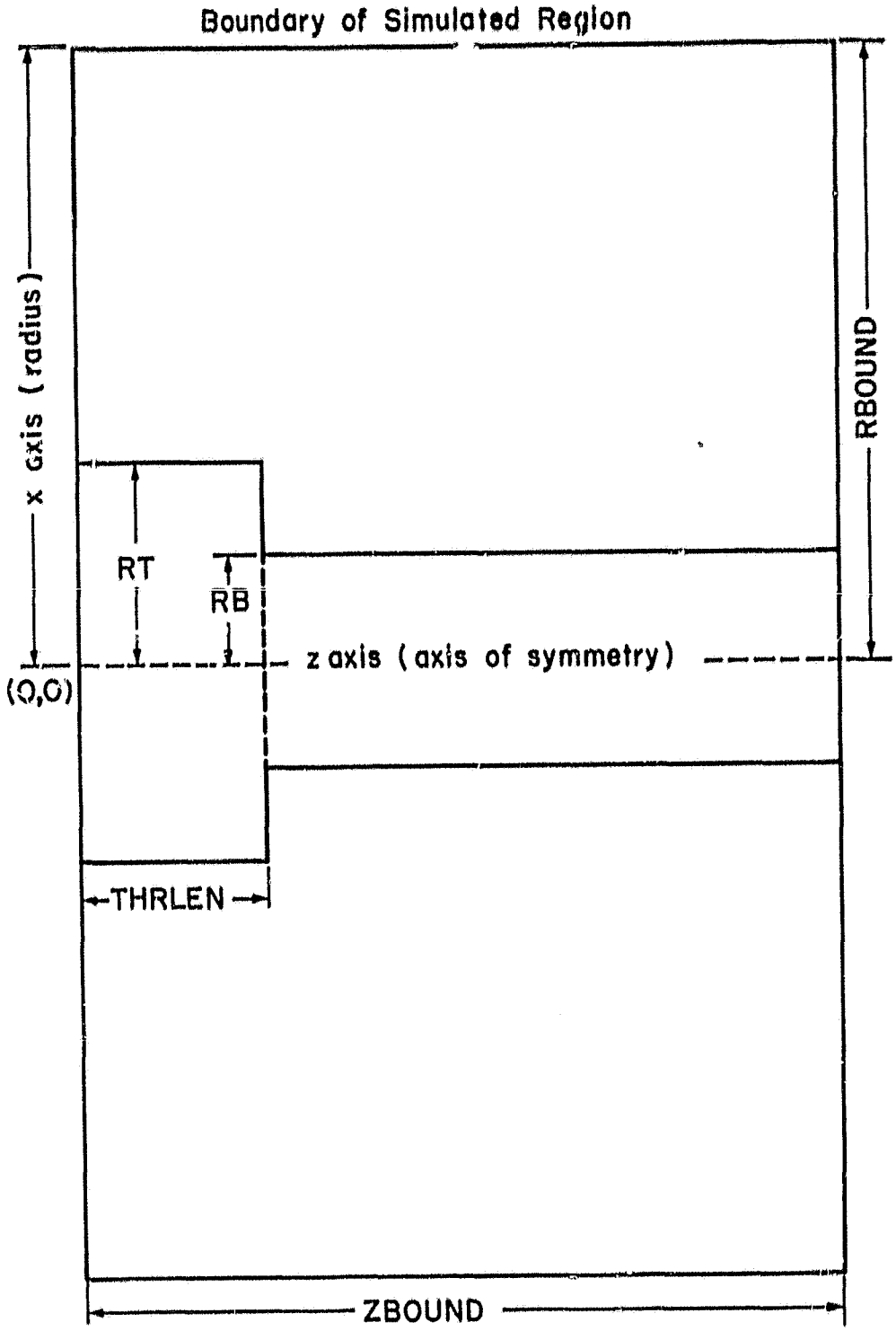
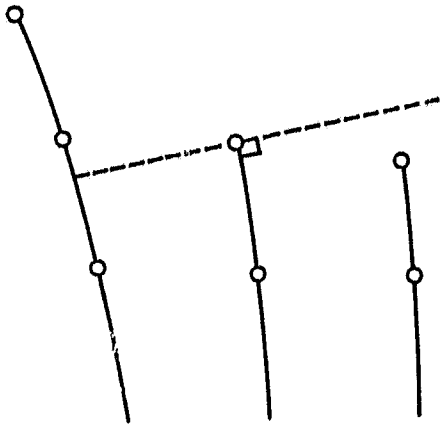
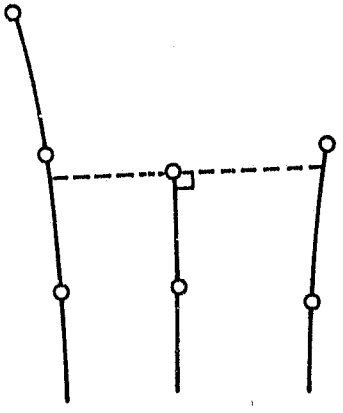


Fig. 1. Coordinate system and dimensions for simulation.

In the simulation used herein, the ion path is represented with a stepwise progression away from the beam and the trajectories are advanced from left to right starting from the end of the thruster. It is assumed that, with small enough step sizes, following the ions through one pass of calculations is sufficient. (The validity of this assumption is partially checked later by comparison with experimental and analytic results.) From a physical viewpoint, the ions are moving at, or above, acoustic velocity, so disturbances should not propagate in the upstream direction. Also, the extent of the plasma is very large compared to the Debye shielding distance, hence electric fields at the flow boundaries should not extend into the bulk of the plasma. The distances to neighboring paths are important parameters, in that they are used to determine densities. As indicated in Fig. 2, a normal to the path being incremented is extended in both directions. This normal is used to calculate the distances to neighboring paths, on the right and left of the path being incremented. In this report, right and left are defined in terms of relationships upon leaving the ion beam, with the viewing direction in the direction of charge-exchange ion motion. The distances to the neighboring paths are obtained by calculating the distance from the path being incremented, along the normal to this path, to the neighboring path where the normal intersects it. If the neighboring path was not intersected by a normal, as in the right side of Fig. 2a, then the neighboring path is extended linearly from the last interval and the distance is calculated. If the normal intersects a neighboring path below the final two ion positions on that path, as is the case for the left hand paths in Figs. 2a and 2b, back stepping is used. This allows the distance to be calculated to the line the normal actually intersects.



(a) Linear extrapolation used on right, back stepping used on left.



(b) Paths intersected both sides (path iterated again), back stepping used on left.

Fig. 2. Geometry for evaluations of distances between paths.

If intersections are found on both the right and left, as in Fig. 2b (linear extrapolation not used), the path currently being iterated will be iterated again so it can keep in step with its neighbors. Finally, if the normal intersects the neighboring path between the final two ion positions, as in the right hand path of Fig. 2b, the intersection point is used as it is.

The density is inversely proportional to the distance between neighboring paths, the ion velocity and the radial distance. The latter relationship is due to the axial symmetry and the use of only one trajectory for each axial location. The density on the left is thus given by

$$n_L = C/\Delta d_L x v_i, \tag{2}$$

where C is a constant depending on operating conditions and the number of trajectories specified, Δd_L is the distance between the path being incremented and the path on the left, x is the radius and v_i is the ion velocity. The density on the right is defined in a similar manner, except that n_R and Δd_R are used. After the radial distance, x, and the ion velocity, v_i, are calculated, the densities to the right and left are calculated using Eq. (2). The two quantities, n_L and n_R, are averaged to get the plasma density at the point under consideration.

The plasma potentials to the right and left are then calculated using Eq. (1). Here n_e represents the density just calculated to the right or left and $n_{e,ref}$ represents the initial density to the right or left of the path being incremented. The force normal to the path direction is then,

$$F_\perp = -q\Delta V_\perp/\Delta d_s \tag{3}$$

where $\Delta V_\perp$ is the difference between the potentials on the right and left and Δd_s is the smaller of Δd_L and Δd_R. This choice for Δd_s was included to accommodate radical changes in the displacements arising when the perpendicular displacement intersects a boundary. The effect of this choice as compared to averaging the displacements was found to have a negligible effect on interior paths. This force is resolved into x and z components.

The force acting parallel to the path can be calculated in a fashion similar to Eq. (3),

$$F_{||} = -q\Delta V_{||}/\Delta d_p \tag{4}$$

where $\Delta V_{||}$ is obtained through Eq. (1) with potentials being those at the point presently under consideration and the previous point and Δd_p the distance between these two points. This force is also resolved into x and z components.

The normal and parallel forces can also be set equal to the rate of change of momentum:

$$F_{\perp,||} = m\Delta v_{\perp,||}/\Delta t \tag{5}$$

with m the ion mass, $\Delta v_{\perp,||}$ the velocity component generated normal or parallel to the path direction, and Δt the size of the time interval used in the iteration. Equating these two force expressions yields

$$\Delta v_{\perp,||} = F_{\perp,||}\Delta t/m \ . \tag{6}$$

These are the velocity changes normal and parallel to the path direction for the present iteration. Similar expressions hold for the x and z

components of the velocity, Δv_x and Δv_z, where $F_{x,z}$ is used instead of $F_{\perp,\parallel}$. The total velocity and its components are calculated from Δv_x and Δv_z. The new ion position is then calculated, using linear expressions, thereby incrementing the path.

It was necessary to consider several special cases in the execution of the displacement algorithm. Three were mentioned in reference to Figs. 2a and 2b in discussing the calculation of distances to neighboring paths. Those were linear extrapolation, back stepping and iterating a path again if intersections were found on both sides. Other cases involve the extreme right and left trajectories that have not intercepted a boundary. Without special treatment, these cases would result in an undefined density on one side of the path because the normal will intercept a boundary instead of another path. The boundary is treated as another path with one exception. If at any time a path would be repelled by the boundary, the direction is left unchanged. This approximates a plasma sheath which would be present at such a boundary.

In general, both the distance between trajectories and the distance from a trajectory to a boundary will be much larger than the Debye distance. The accuracy of the simulation should therefore be considered questionable at any location where the distance between trajectories approaches the distance to a boundary. A better approximation in such a location might be obtained by extrapolating from deeper within the charge-exchange plasma. It would also be possible to use more trajectories, so that the space between them would be reduced.

The distribution of charge-exchange ion production along the axis is assumed to be proportional to the neutral density on the axis. This neutral density for a single thruster (no overlap of neutral effluxes from adjacent thrusters) is[7,8] (see Appendix B)

$$n(z) = n_o \left(1 - \frac{z}{(z^2 + r_b^2)^{1/2}}\right) , \tag{7}$$

where r_b is the beam radius and n_o is a constant for a given combination of beam diameter and neutral loss rate. This function decreases rapidly with increasing z, due to the rapid divergence of neutral atom paths in free molecular flow. The beam radius, r_b, is an important parameter in this simulation, because approximately half of the total charge-exchange production occurs within about one beam radius of the thruster. This means that half of the charge-exchange trajectories will originate within the same distance.

In determining the locations for the origin of ion trajectories along the axis, the integral of Eq. (7) is used

$$\int_o^\infty n(z)dz = n_o r_b . \tag{8}$$

The region simulated is finite, so that not all of the integral can be represented. The region to be simulated was defined so that 95 percent of $n_o r_b$ is contained within this region. For N trajectories making up $0.95\ n_o r_b$, with each trajectory located at the median of the density interval that it represents, the following expression holds

$$\frac{1}{2} \sum_{i=0}^{2N} \int_{z_i}^{z_{i+1}} n(z)dz = N \int_{z_i}^{z_{i+1}} n(z)dz = 0.95\ n_o r_b/2 . \tag{9}$$

For the first trajectory, the starting point is at the right end of the thruster (left end of the ion beam, see Fig. 1). To calculate each successive z, the expression used is

$$0.95 r_b/2N = z_{i+1} - (z_{i+1}^2 + r_b^2)^{1/2} - z_i + (z_i^2 + r_b^2)^{1/2} . \tag{10}$$

For the first trajectory, i = 0 and i + 1 = 1. The value of z_o is the right end of the thruster and the first trajectory is started at z_1. The second trajectory is started at z_3, third at z_5, and so forth.

The initial velocity upon leaving the ion beam is the Bohm velocity,

$$v_B = (kT_e/m_i)^{1/2} , \tag{11}$$

where T_e is the electron temperature in the ion beam and m_i is the ion mass. The constant, C, in Eq. (2) is obtained using the following procedure. The total production rate of charge-exchange ions, for a uniform beam current density profile, is given by[6]

$$\dot{N}_{ce} = 2J_b^2(1-\eta_u)\sigma_{ce}/\pi r_b \eta_u q^2 \bar{v}_o , \tag{12}$$

where J_b is the ion-beam current (A), η_u is the propellant utilization, σ_{ce} is the charge-exchange cross section (m^2), r_b is the beam radius (m), q is the absolute electronic charge (C), and $\bar{v}_o$ is the mean neutral thermal velocity, $(8kT_o/\pi m_o)^{1/2}$ (m/sec). With typical values for Hg neutrals and singly charged ions used,

$$\dot{N}_{ce} = 6.18 \times 10^{16} J_b^2(1-\eta_u)/r_b\eta_u . \tag{13}$$

The charge-exchange cross section usually decreases with ion energy. The value used for Eq. (13) corresponds to Hg^+ ions at about 1,000 eV.

The plasma density can be related to this production rate by

$$n = \dot{N}_{ce}/2\pi\Delta d_m x v_i N , \tag{14}$$

where Δd_m is the local mean spacing between trajectories (m), x is the radius (m), v_i is the local ion velocity (m/sec) and N is the number of trajectories simulated. Substituting for $\dot{N}_{ce}$,

$$n = \left(\frac{J_b^2 (1-\eta_\mu) \sigma_{ce}}{N r_b \eta_\mu q^2 \pi^2 \bar{v}_o} \right) \quad \frac{1}{x \Delta d_m v_i}$$

where the quantity enclosed in the parentheses is the constant C.

PROGRAM STRUCTURE

The simulation is performed by a computer program written in standard Fortran IV that is listed in APPENDIX D. The main driver program, PLASIM, establishes the sequencing of computations in the simulation and makes calls to the major subroutines which perform the required calculations. This overall flow is illustrated with a flow chart in Fig. 3.

First, all of the fixed parameters are stored in the COMMON blocks by the subprogram BLOCK DATA. The control parameters which define the type of computation, disposition of results and termination, along with physical data used in the simulation, such as the thruster and accelerator system dimensions and the plasma characteristics, are input through subroutine READER. The parameters and data are read from card images.

For a typical simulation, the next call is to subroutine INIT, which initializes the constants and arrays, calculates the coordinates of ion trajectories (paths) at the beam edge and performs the first iteration, thus calculating the second set of coordinates on the paths. Calls are then made to subroutine WRIT to output the heading, thruster schematic, initial parameters and results of the initialization and first iteration.

PLASIM next begins the staging and successive iterations by calls to subroutine CALC. Subroutine WRIT is also called to output a heading for information on the progress of the computation. After the completion of a specified number of iterations, termed a stage, the contents of the coordinate and density arrays are written to an external mass storage file, PATHS. The arrays are reinitialized and the next stage commenced.

ORIGINAL PAGE IS OF POOR QUALITY

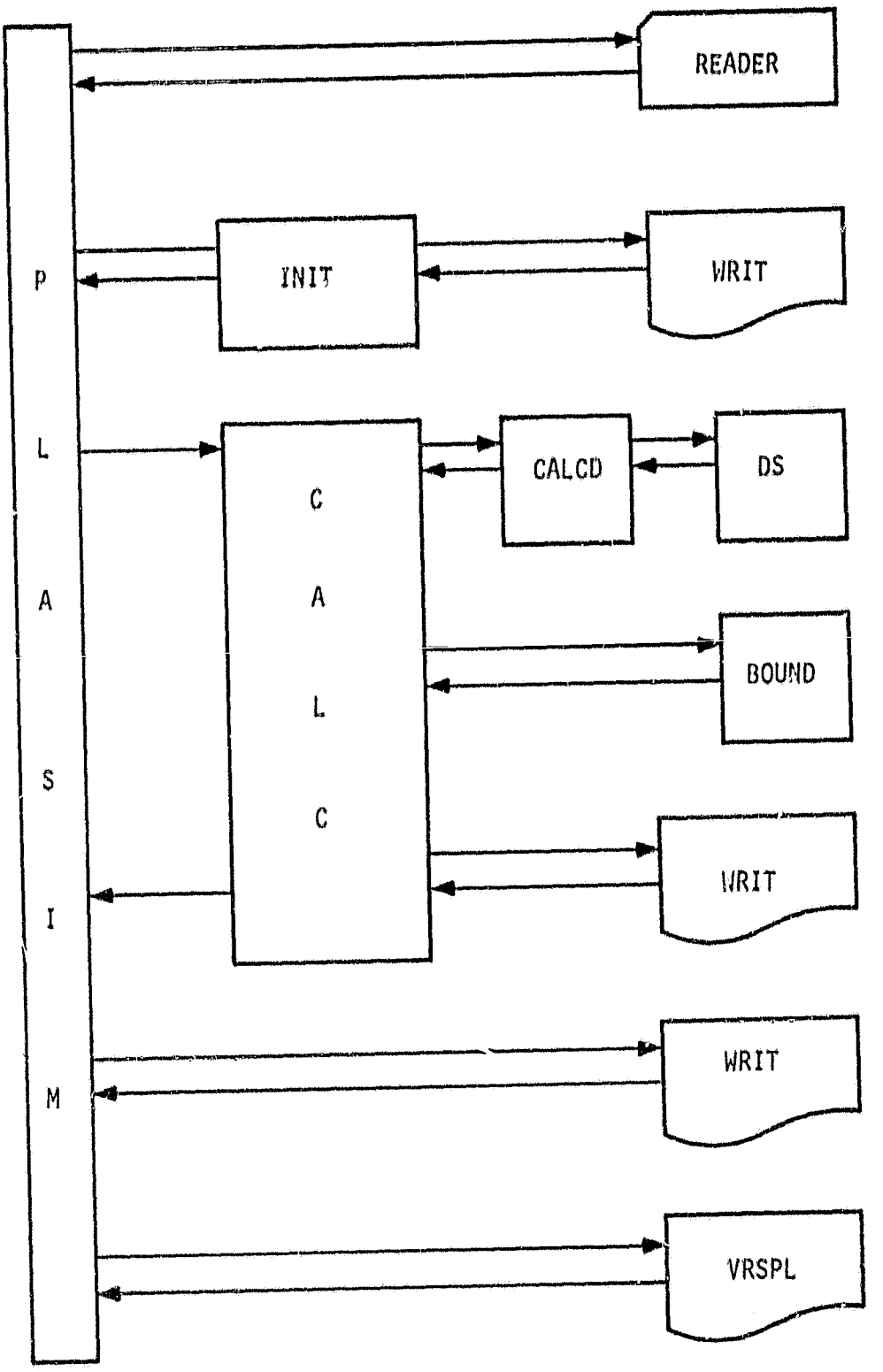


Fig. 3. Calling sequence for program PLASIM.

After completion of all the stages, the plotting routines are called according to the status of the control parameters set at the beginning. Another call to READER is made to determine the next action, either another simulation run or a normal termination.

The main calculation subroutine, CALC, sets up an accounting procedure for the paths, the iteration number, the activity of the path, and the boundary condition, in addition to carrying out calculations leading to new ion positions. Displacements from the current path to adjacent paths are returned by a call to subroutine CALCD and then the densities on both sides of the present path are calculated. The potentials, on both sides of the path and at the present and prior ion locations, are obtained, followed by the calculation of forces acting perpendicular and parallel to the path, respectively. The forces are resolved into z and x components to calculate the new z and x components of the ion velocities. A new ion location is computed from the z and x velocities and a call is made to subroutine BOUND to ascertain if it is within the simulation boundaries.

Two plotting methods are included, one for a line printer, LNPLT, and one for a Versatec electrostatic plotter, VRSPL. The subroutines called in VRSPL are described in the appropriate literature.[18] These may be not only device-dependent, but site-specific as well. They are included to indicate a possible preparation procedure to use in presenting the simulation results in graphical form.

There are several error detection segments in the code which output informative messages whenever unrealistic values appear in certain variables (see subroutine BLOCK DATA).

There are three types of simulations defined by the control parameter KEY. KEY=-1 generates a normal simulation with a non-uniform initial density distribution, KEY=-1 generates a simulation with a uniform initial density distribution and KEY>0 generates an upstream simulation which utilizes a given path from a previous run as a boundary and simulates paths upstream from that path. A call with KEY=0 causes a normal termination including normal clearing of the plotting buffers. The following subsections discuss the various subroutines in detail.

PLASIM

This is the main driver routine for the simulation of a charge-exchange plasma surrounding a broad beam ion thruster. This routine controls and sequences the computation and defines the program structure and staging by calling various subroutines. It is outlined in Fig. 4. The first statement (two lines) of the program is a non-ANSI statement which declares the files required for input and output. This will be converted to a comment statement to prevent errors because of non-standard FORTRAN.

READER: The first call is to subroutine READER which reads in the run specifications and information. INIT: The next call is to subroutine INIT which initializes various parameters and performs the first iteration. CALC: The next call is to subroutine CALC which computes the ion positions along the trajectories. CALC is called NUMIT times. This completes the first stage. (NUMIT is the maximum number of iterations to be performed on any given path during any particular stage.) Information for the first (NUMIT - 10) iterations is written to an external file, PATHS. Core memory is then reinitialized by transfering the results of the final ten iterations to the core space for the

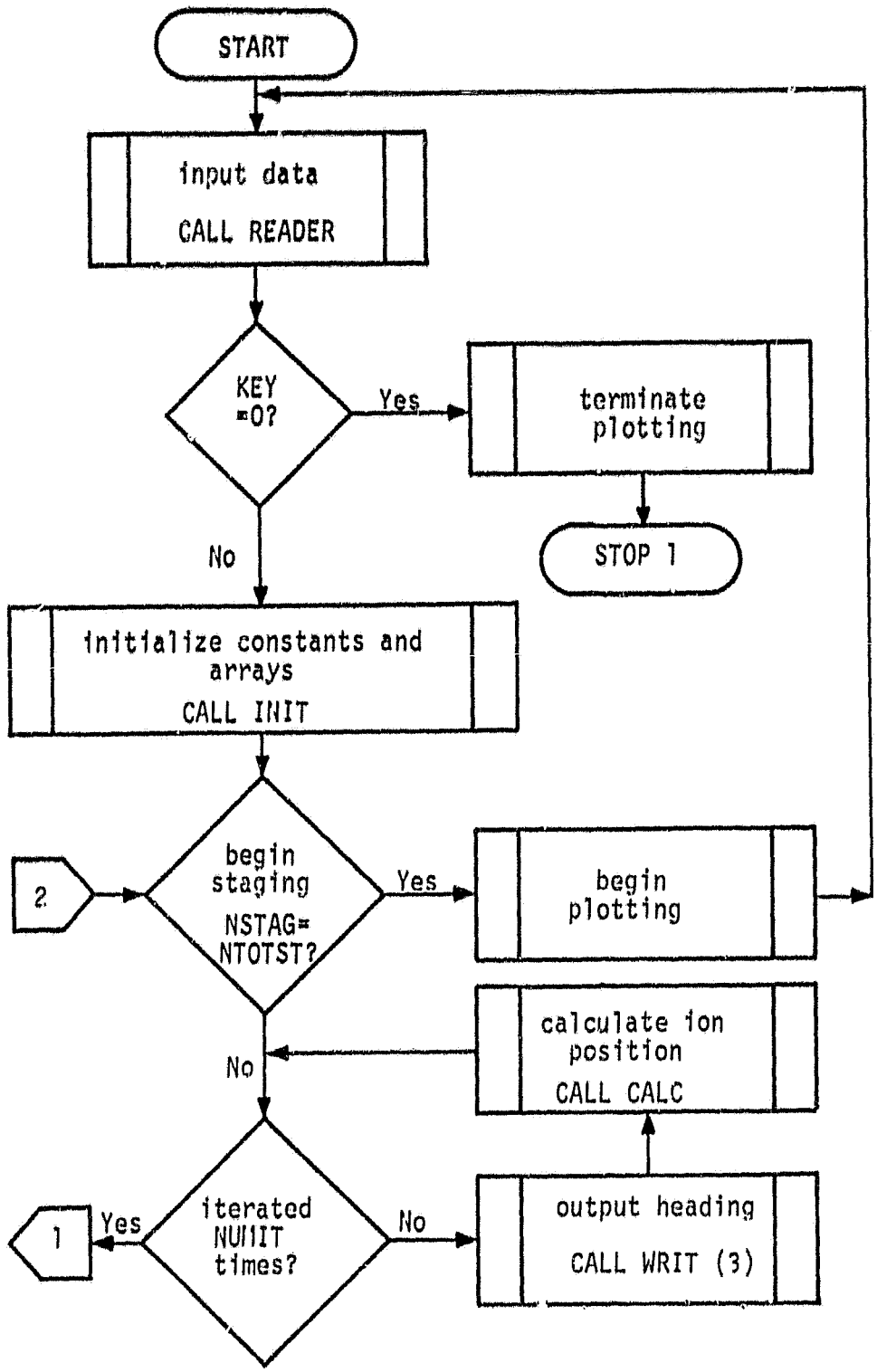


Fig. 4. Flow chart of main driver routine, PLASIM.

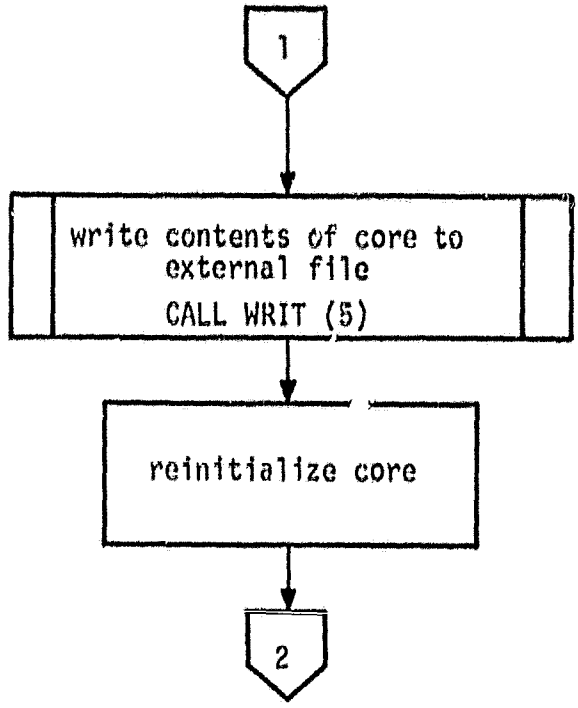


Fig. 4. Continued, PLASIM.

initial ten. The results of these ten iterations act as a base for another set of (NUMIT - 10) iterations. CALC is called another (NUMIT - 10) times to fill core storage with another set of PATH coordinates which completes the second stage. Then another set of (NUMIT - 10) results are written to PATHS, core is reinitialized again and another stage is run. This is done until all paths intersect boundaries. Finally, the plot indicator, ICLPLT, is checked to determine which plotting routine to use to output a plot of the path coordinates, if any.

Two blank cards are always placed at the end of the data deck to signal a stop. The word size dependent variables are IF1, IF2, INFO, ITITL, and IW. See BLOCK DATA for instructions on word size dependent variables and code generated error messages.

BLOCK DATA

This subprogram loads data into labeled common storage at compile time through the data statements. Error message information and the instructions on word size dependence have been included. This dependence was motivated by the requirement for transfer of alphanumeric data for labels to the plotting subroutines of Versatec and IMSL. The basic unit for input was chosen to be 80 characters, a full data card. These 80 characters must be packed continuously into words of an array which will be transferred to the external plotting subroutines. The arrangement described below accomplishes this but requires a change in the DATA statements for computers with word size not equal to 10 characters.

The word size dependent variables are: IF1, IF2, IW, INFO, and ITITL. IW is the number of words required to generate 80 characters. To convert to a machine using different word size, modify only IW, IF1 and IF2 in first two data statements below. The third data statement

is for I/O buffers and the fourth data statement is for values of constants.

```
DATA IW, IF1(1), IF1(2) /8, 6H(8A10), 1H /
DATA IF2(1), IF2(2), IF2(3), IF2(4) /10H(8A10/2(4A,4H10)),1H ,1H /
DATA IN, IOUT, IPATHS /5, 6, 7/
DATA BK, Q, PI /1.3806E-23, 1.602E-19, 3.141593/
```

When a code generated error message is called, the first line of the error output will be of the form,

****** ERROR NNN ******

where 'NNN' is one of the following integers,

207	See Subroutine WRIT
410	See Function Subroutine DS
412	See Function Subroutine DS
521	See Subroutine CALCD
522	See Subroutine CALCD
523	See Subroutine CALCD
524	See Subroutine CALCD
525	See Subroutine CALCD
526	See Subroutine CALCD
527	See Subroutine CALC
528	See Subroutine CALCD
529	See Subroutine CALCD
530	See Subroutine CALC
610	See Subroutine BOUND
612	See Subroutine BOUND

and the referenced subroutine is the calling routine.

READER

See INPUT AND OUTPUT section.

INIT

This subroutine initializes the necessary variables and performs the first iteration. It is outlined in Fig. 5. The constants, which include the Bohm velocity, velocity of the thermal neutrals and the step size, are defined first. Then the z and x coordinates of the ion

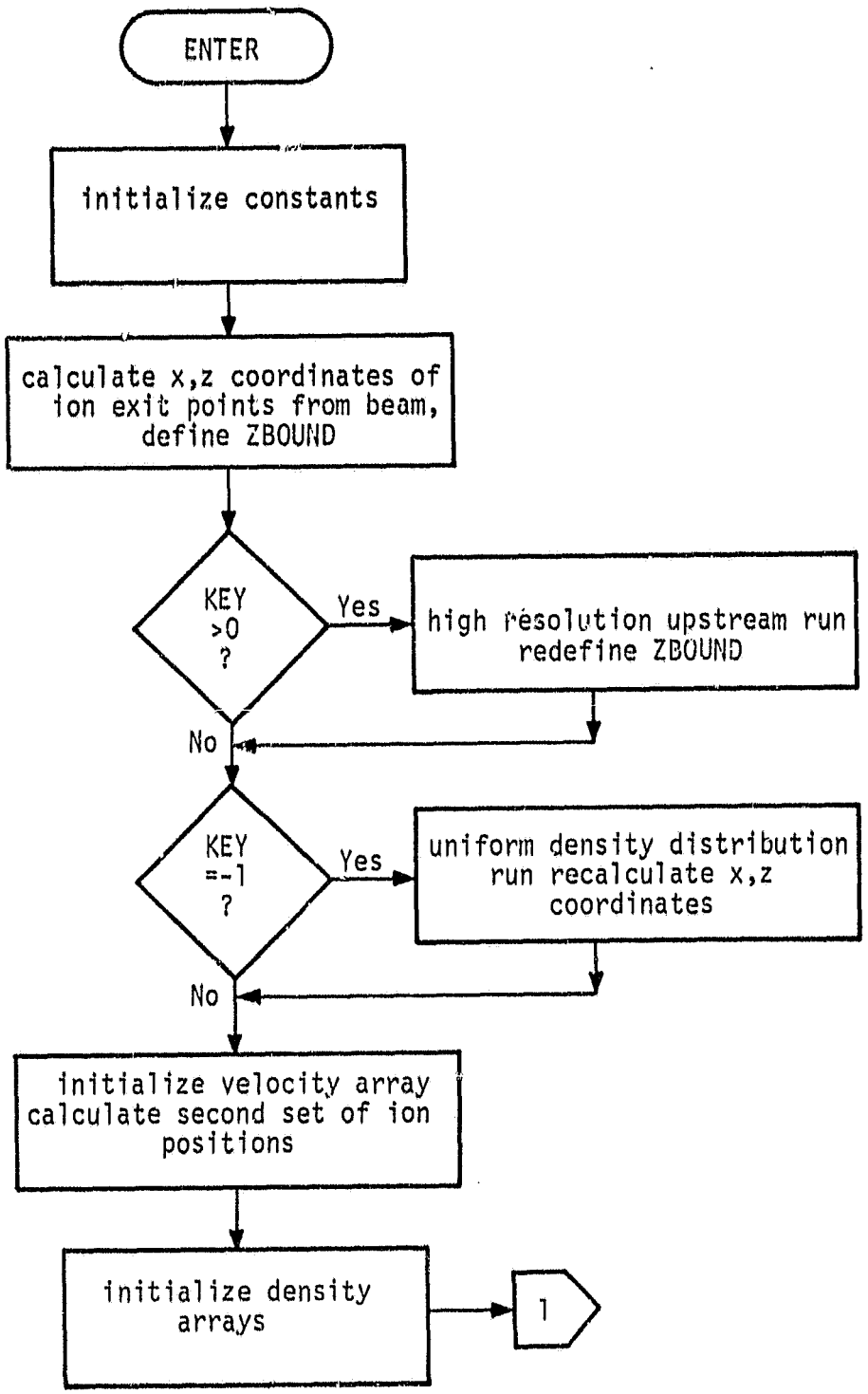


Fig. 5. Flow chart of initialization routine, INIT.

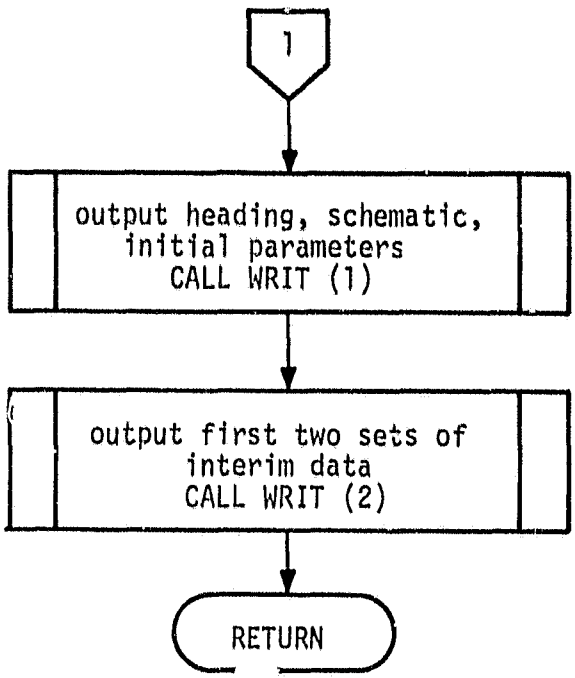


Fig. 5. Continued, INIT.

trajectory exit points are obtained and ZBOUND is defined. If this is a high resolution upstream run (KEY>0), ZBOUND is redefined. If this is a uniform density run (KEY=-1), the ion trajectory exit points are redefined to be uniformly spaced. Following this, the velocity arrays and iteration counter are initialized, the second ion positions on each path are calculated and all the paths are set to active. The initial densities between the paths and the initial densities on the paths are then calculated. Finally, the heading, thruster schematic, initial parameters and results of the initialization and first iteration are output, then control is returned to PLASIM

CALC

This is the main calculation routine. It is outlined in Fig. 6. This subroutine uses the arrays in blank common along with subroutine CALCD to determine the next position of the ion being considered. The densities and potentials to the right and left of the present path are calculated first. Then the force acting perpendicular to the present path is calculated. If the boundaries repel the path, the perpendicular force is set equal to zero. The potentials and forces acting parallel to the present path are obtained next. The total force (sum of perpendicular and parallel components) acting on the ion is calculated and then the velocity components along the x and z axes are obtained. The speed on path one is normalized to 1.2 times the speed on path two when the speed on path one is 20% greater than the speed on path two. Finally the next ion position is calculated. Subroutine BOUND is called to make sure the new ion position is inside the boundaries. The results are printed every ICLWRT iterations. IFLAG1 is checked (see CALCD) to see if intersections were found to both the right and the left. If intersections were found to both sides, the path is iterated again.

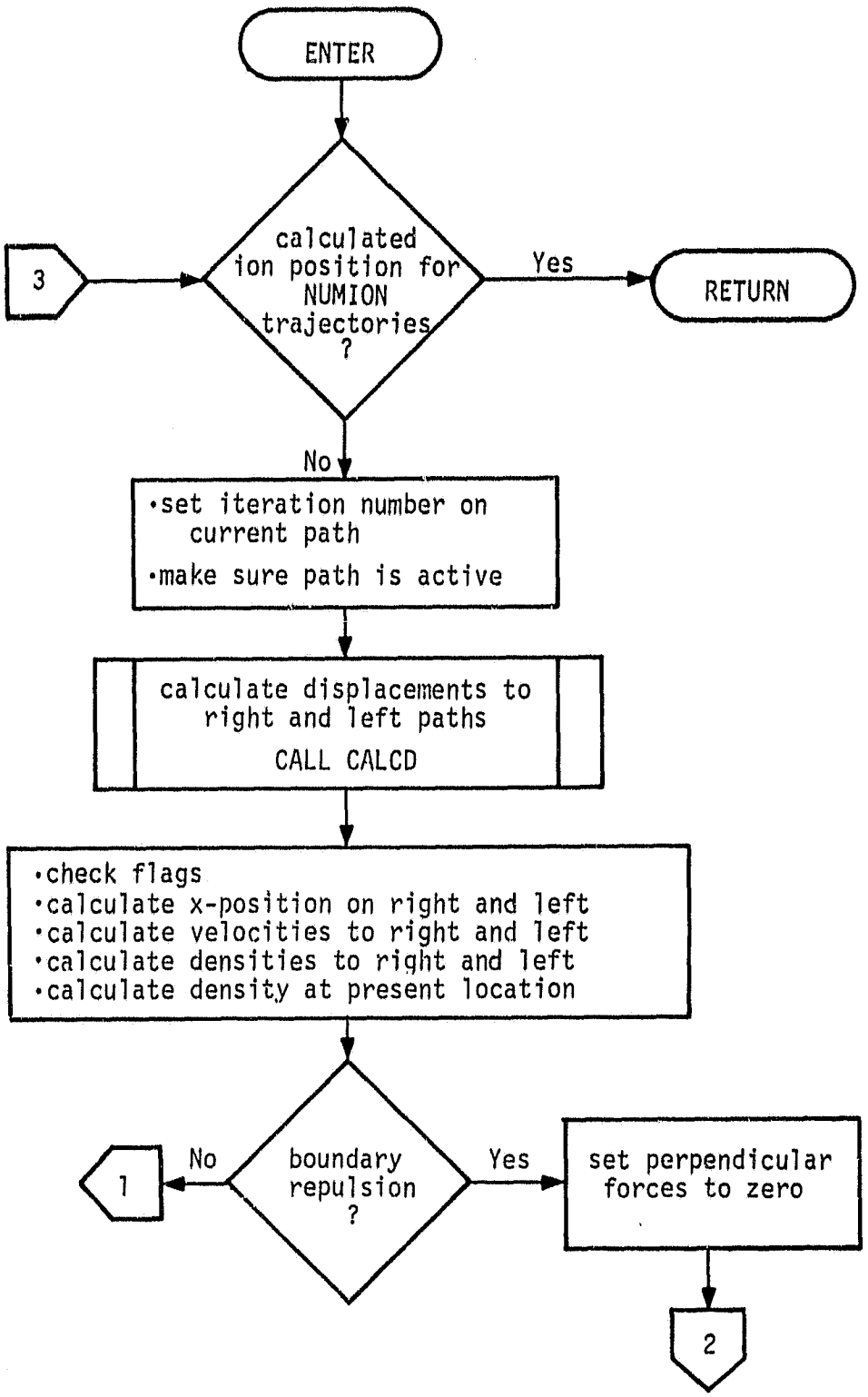


Fig. 6. Flow chart of main calculation routine, CALC.

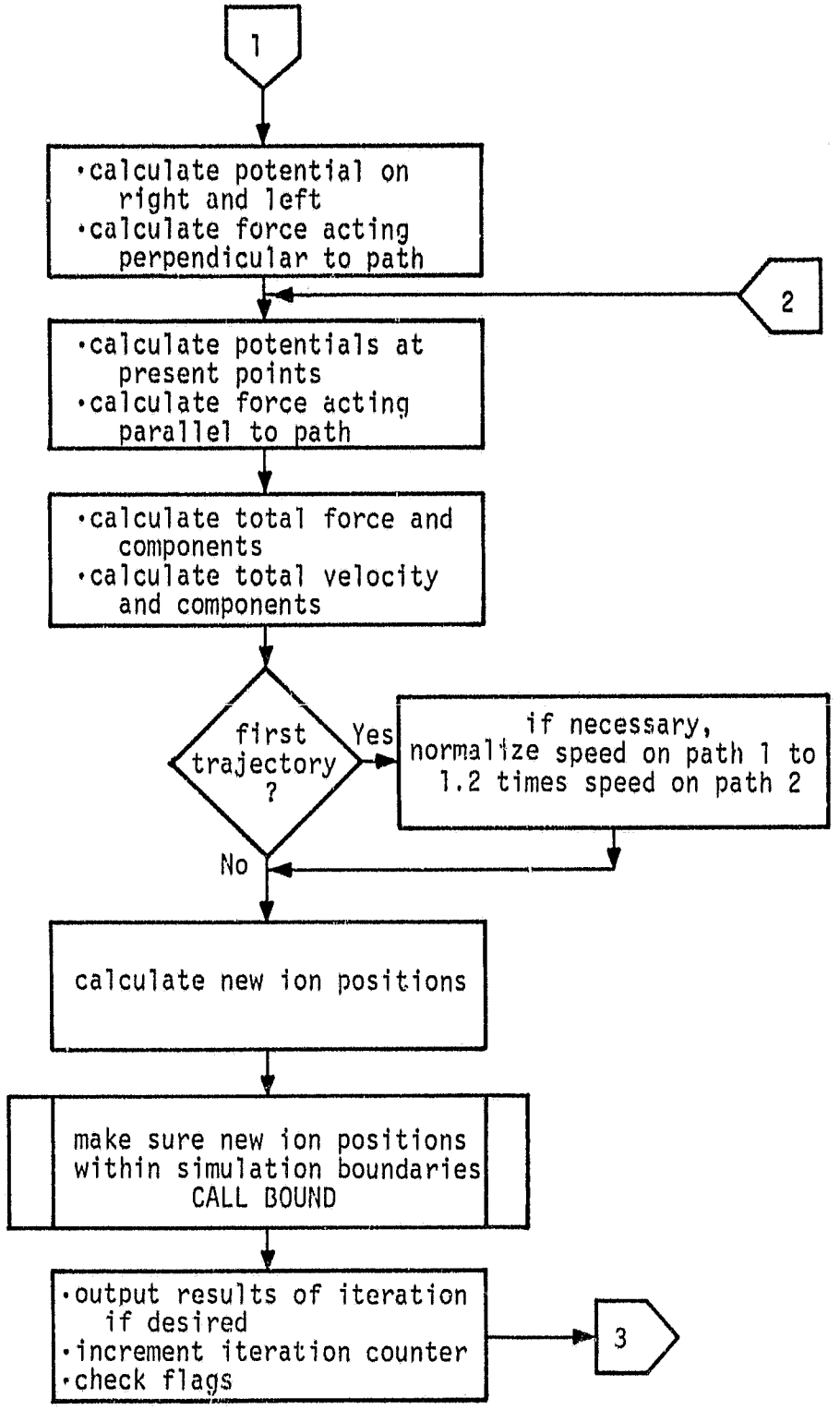


Fig. 6. Continued, CALC.

CALCD

This subroutine, which is outlined in Fig. 7, computes the perpendicular displacement from the present path to the adjacent paths. The flags are defined in this routine.

First, the flags are initialized and then the slope of the line perpendicular to the present path, at its end point, is calculated. If the path under consideration is the right- or left-most active path, function subroutine DS is used to get the displacement to the boundary.

When an interior path is under consideration, displacements are obtained to the right- and left-hand paths. Here, the slope of the adjacent path, between the last two ion positions, is calculated. The intersection point between the line perpendicular to the path under consideration and the line formed by the last two points of the adjacent path, is calculated. Tests are run to determine if linear extrapolation (adjacent path extended to find an intersection point) or back stepping (interpolation between points on the adjacent path previous to the last two points) is to be used. The flags are set, the displacement of the adjacent path is calculated and then control is returned to subroutine CALC. This subroutine returns the values of the flags, the displacements to the right and left and the angle that the perpendicular makes with the horizontal, to subroutine CALC.

BOUND

This subroutine checks the point (z,x) to see if it lies inside the defined boundaries of the simulation. If (z,x) does lie inside the defined boundaries, no changes are made and control is returned to subroutine CALC. If the point lies outside the boundaries, the path status is set equal to the iteration number. In addition, if (z,x) lies on the

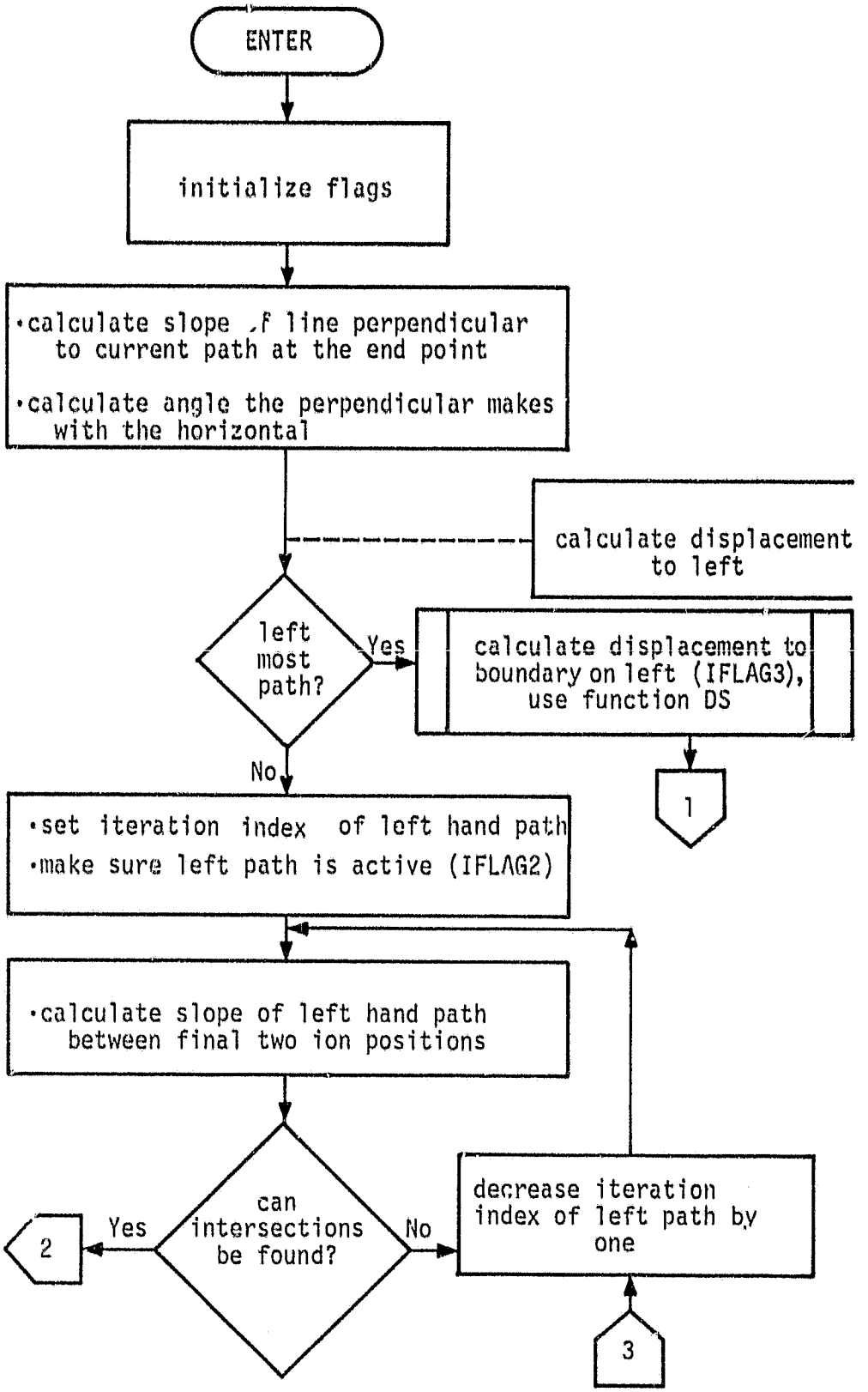


Fig. 7. Flow chart of displacement calculation routine, CALCD.

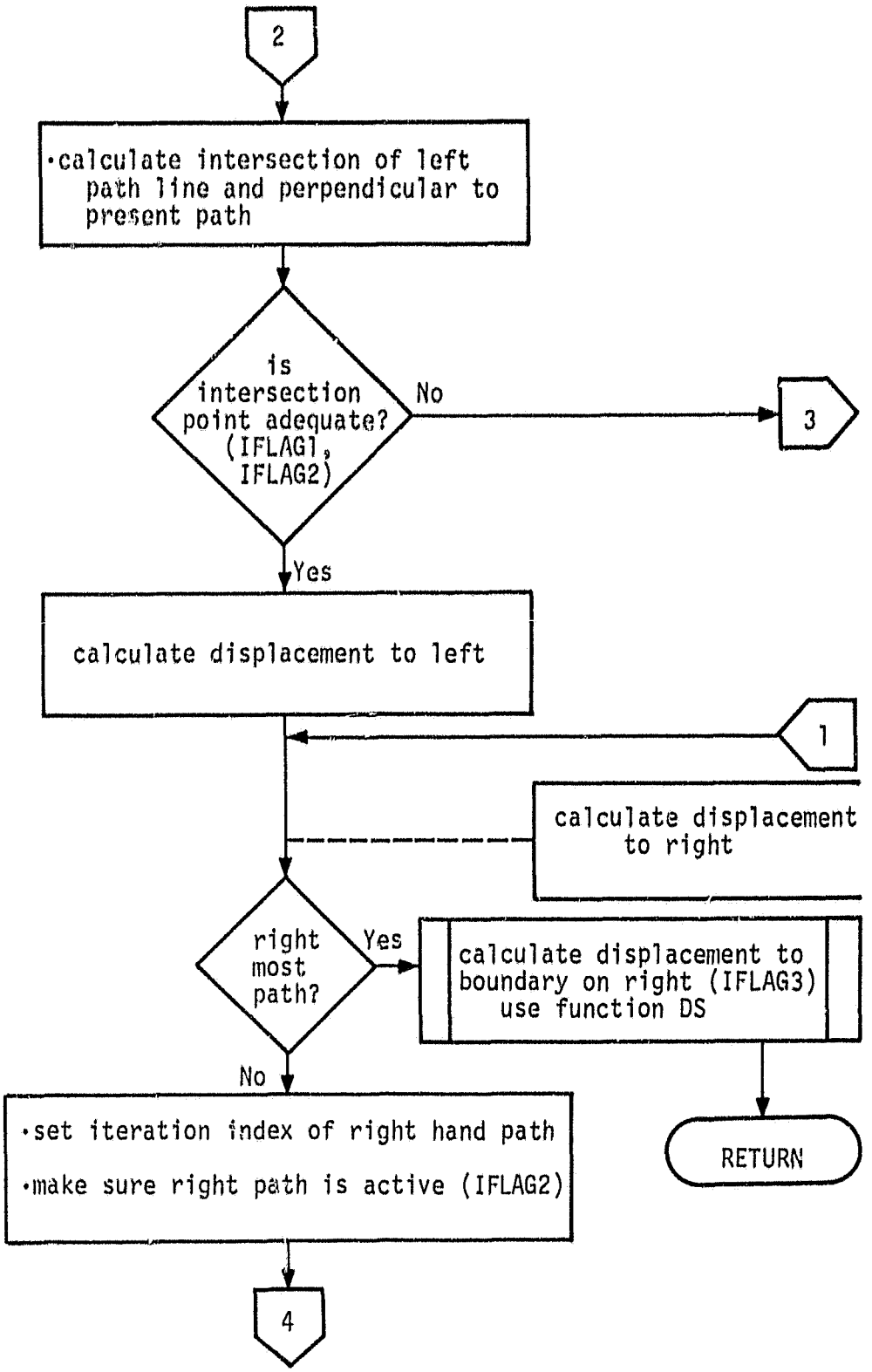


Fig. 7. Continued, CALCD.

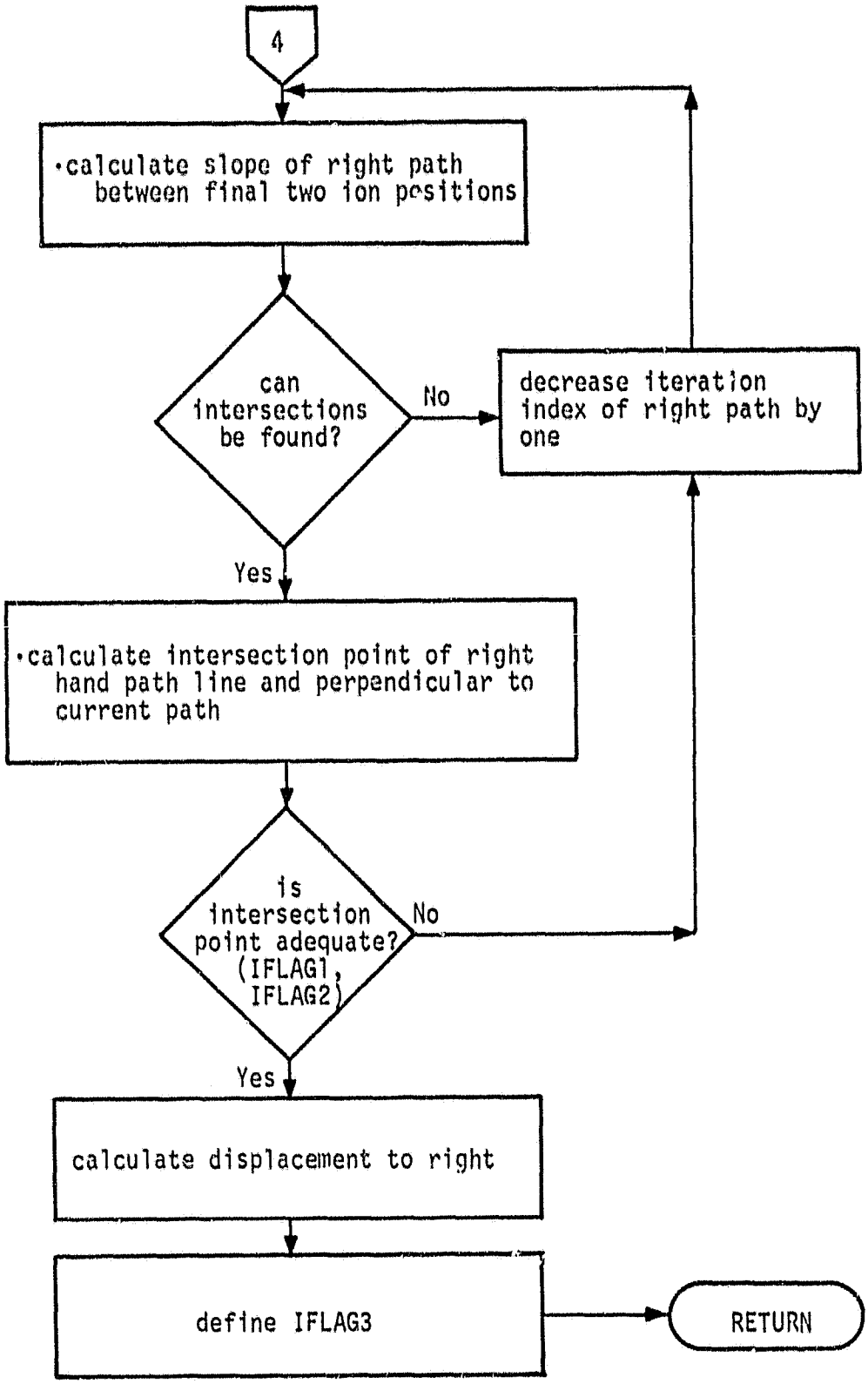


Fig. 7. Continued, CALCD.

ORIGINAL PAGE IS
OF POOR QUALITY

first or last active path and lies outside the boundaries, no changes are made and control is returned to subroutine CALC.

WRIT

This is the main output routine. It consists of five subsections with the value of the passed parameter, KE, determining which section is called. KE=1: the first three pages of output are generated, consisting of the heading, a thruster schematic and the constants and input data. KE=2: the interim status of the major variables is output from subroutine INIT. KE=3: the heading is printed for the results of one pass of calculations. KE=4: a file of path coordinates is created to be used in a high resolution upstream run. KE=5: a file of position-density triplets is created to be used for plotting (see INPUT AND OUTPUT section).

DS

This function subroutine finds the perpendicular displacement from the first or last active path to the boundary. CALCD constructs a perpendicular to the present path at the present point (z,x) with a slope of SLOPEP. Function DS then extrapolates this perpendicular of slope SLOPEP to the left or right (depending on whether the first or last active path is considered) and finds the z and x intercepts (ZINT and XINT) along a boundary line. ZINT and XINT are checked to see if they lie on or between the boundary endpoints on the boundary line. If they do, the displacement is calculated, if they do not, ZINT and XINT are calculated along the next boundary line and tested again. This continues until adequate intersection points are found or all boundaries have been considered in which case an error message is output. The perpendicular displacement is returned to subroutine CALCD.

VRSPL and VRSPLT

These subroutines set up the data in ZION and XION for transfer to the utility plotting package, Versatec. They also draw a schematic of the thruster and beam. Labels for the graph and its axes are also transferred and involve the word-size dependent variables. VRSPL plots from the external file, PATHS, whereas VRSPLT plots from core memory. These routines are device dependent and may require considerable modification at other installations.

LNPLT and PLOTW

LNPLT sets up the data in ZION and XION for transfer to the utility plotting routine PLOTW which utilizes the line printer. It is also dependent on the type of printer available and may require considerable modification at other installations.

ORIGINAL PAGE IS
OF POOR QUALITY

INPUT AND OUTPUT

For each simulation run, seven data cards (or card images) are read by READER. On the second card, if KEY=0, the reading sequence is interrupted and control returns to PLASIM, where normal termination ensues. The input data cards have the following formats and parameters which are defined in the following subsection and in APPENDIX C.

Card 1, FORMAT (see the first data card in BLOCK DATA)
Content: Description of the run, up to 80 characters.

Card 2, FORMAT (7I10)
Content: NUMION, NUMIT, KEY, ICLWRT, ICLPLT, NTOTST, ICLERR.

Card 3, FORMAT (6F10.5)
Content: RB, RBOUND, RT, THRLEN, BMCLR, UTIL.

Card 4, FORMAT (5E10.3)
Content: TELIN, TELOUT, TTHNEU, CEXSEC, UMSION.

Card 5, FORMAT (3F10.3)
Content: TIMEMU, XVELMU, ZVELMU.

Card 6, FORMAT (see the second data card in BLOCK DATA)
Content: Label for graphical output.

Card 7, FORMAT (see the second data card in BLOCK DATA)
Content: Axis labels for graphical output.

Due to the dimensions in the COMMON BLOCK, the values of the variables NUMION and NUMIT should not exceed 41 and 150, respectively. Also NUMIT should not be less than 11 and the total number of iterations performed should be less than 8888. The temperatures input on the fourth data card should be in units of eV, all other variables require SI units.

The printed output begins with three pages of identification including a schematic of the simulation region and a statement of the parameters defining the conditions of the run. (See WRIT(1).)

The next two segments consist of INTERIM STATUS, a report of the coordinates, velocity and density for each path after the initialization and first iteration, as prepared by INIT. (See WRIT(2).) A provision is made in subroutine CALC to output the counters, coordinates, velocity and density values for each iteration so that the progress of the simulation may be followed in detail. If the results of each iteration are not desired, the value of the input variable ICLWRT on data card 2 should be set to a value other than one. The value chosen determines how many iterations must occur before these results are output. WRIT(3) outputs the heading for the information output in subroutine CALC.

Graphical output consists of an outline drawing of the thruster and beam boundaries and the paths generated by the computation in the form of dots or lines. The paths are graphs of the coordinates contained in the arrays ZION, XION. These arrays are each a double-subscripted array in which the first subscript identifies the ion path number, and the second one identifies the iteration. The density array is also available for plotting, but provision for such has not been installed.

An appropriate relative scale in terms of the number of paths and the iteration step size must be established to properly model the upstream regions. Very small steps compared to the path spacing are inappropriate as are very large steps compared to the path spacing. The scaling ratio $S = \Delta d/v_B \Delta t$ compares the path spacing to the axial step size. Appropriate values of this ratio, used herein, for the various thruster sizes are: 5 cm thruster: S = 2.9, 15 cm thruster: S = 3.9, 30 cm thruster: S = 3.6. These values were obtained by taking Δd as the separation between the 14th and 15th paths, along the beam edge, v_B as the Bohm velocity and Δt as the time step for each configuration given in the SAMPLE INPUT section. It should be noted that the value of the scaling ratio, S = 2.9, for the

5 cm thruster was restricted by computer CPU time allocations. The modeling of the upstream region for a 5 cm thruster would be better if more stages were used because for 10 stages, S = 3.2 and for 11 stages S = 3.5.

Sample Input

A typical set of input parameters for program PLASIM, to simulate a 5 cm ion thruster using mercury (Hg) propellant, are,

Data Card	Content
1	5 cm thruster, non-uniform density distribution, basic configuration.
2	40 150 -2 20 3 9 1
3	.025 .60 .08 .50 .05 .71
4	7.0E+00 3.5E+00 4.7E-02 6.0E-19 3.34E-23
5	.75 1.0 0.0
6	Propagation of a charge-exchange plasma, 5 cm thruster
7	Distance along beam axis (meters) Radial distance (meters)

A typical set of input parameters for program PLASIM, to simulate a 15 cm thruster using mercury (Hg) propellant, are

Data Card	Content
1	15 cm thruster, non-uniform density distribution, basic configuration.
2	40 150 -2 20 3 4 1
3	.075 .60 .12 .30 .63 .85
4	5.0E+00 2.5E+00 4.7E-02 6.0E-19 3.34E-23
5	.75 1.0 0.0
6	Propagation of a charge-exchange plasma, 15 cm thruster.
7	Distance along beam axis (meters) Radial distance (meters).

A typical set of input parameters for program PLASIM to simulate a 30 cm ion thruster using mercury (Hg) propellant are,

Data Card	Content
1	30 cm thruster, non-uniform density distribution, basic configuration.
2	40 150 -2 20 3 2 1
3	.14 .60 .20 .40 2.0 0.9
4	0.350E+00 0.175E+00 4.7E-02 6.0E-19 3.34E-23
5	.75 1.0 0.0
6	Propagation of a charge-exchange plasma, 30 cm thruster.
7	Distance along beam axis (meters) Radial distance (meters).

The values given on data cards 3 and 4, in all of the above cases, correspond to the experimental values for those quantities.

VERIFICATION

Two studies, one analytic and one experimental, are used for the verification of the computer code presented herein. The analytic study was conducted as a support activity for the development of the subject computer code, and is reported in more detail in APPENDIX A.

Analytic Solution

The analytic solution is for the case of an infinitely long cylindrical ion beam with a uniform production of charge-exchange ions along the beam. The density and potential variations are restricted to be only radial so that an analytic solution could be obtained in a straightforward manner.

A computer solution was obtained using the uniform ion density option of the program (KEY=-1) and is shown in Fig. 8. The dots occur every 60 iterations. The radial density and velocity from the analytic solution are compared to the radial density and velocity from the simulation, as a function of radial distance, in Figs. 9 and 10, respectively. The agreement between the analytic solution and the computer simulation for the velocity and the density is excellent.

Experimental Solution

Experimental surveys of the plasma density[8,9] are shown in Figs. 11 and 12 along with comparisons to the computer code for 5 cm and 15 cm thrusters, respectively. The operating conditions used in the experimental studies were duplicated to obtain the isodensity contours generated by the computer simulation. Figure 13 shows the isodensity contours generated by the computer simulation for a 30 cm thruster. No experimental data is presently available for a 30 cm thruster. The ion

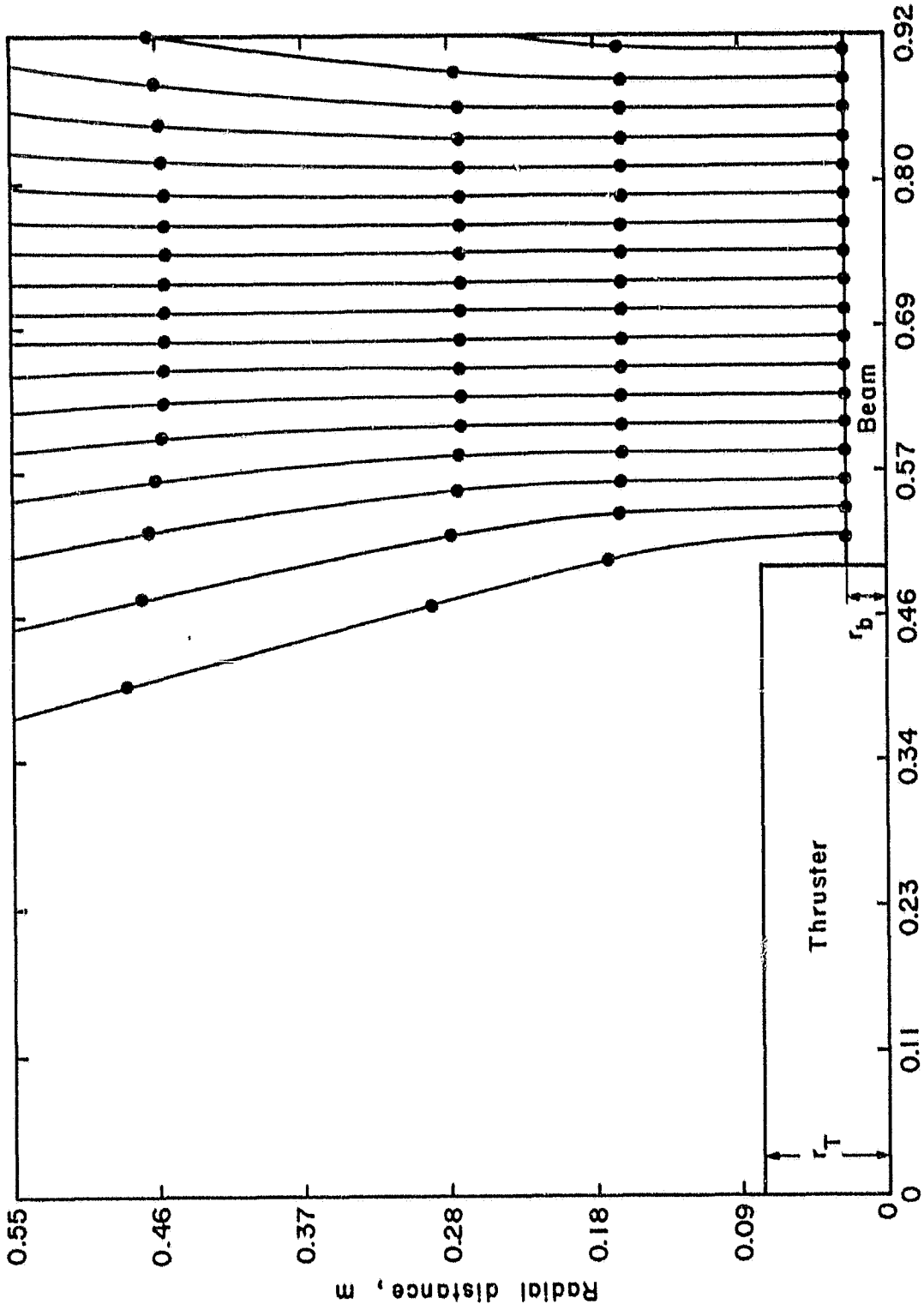


Fig. 8. Simulation for uniform density of charge-exchange ion production. Twenty trajectories simulated in three stages, electron temperature 7.0 eV in ion beam and 3.5 eV in charge-exchange plasma for 5 cm thruster.

ORIGINAL PAGE IS
OF POOR QUALITY

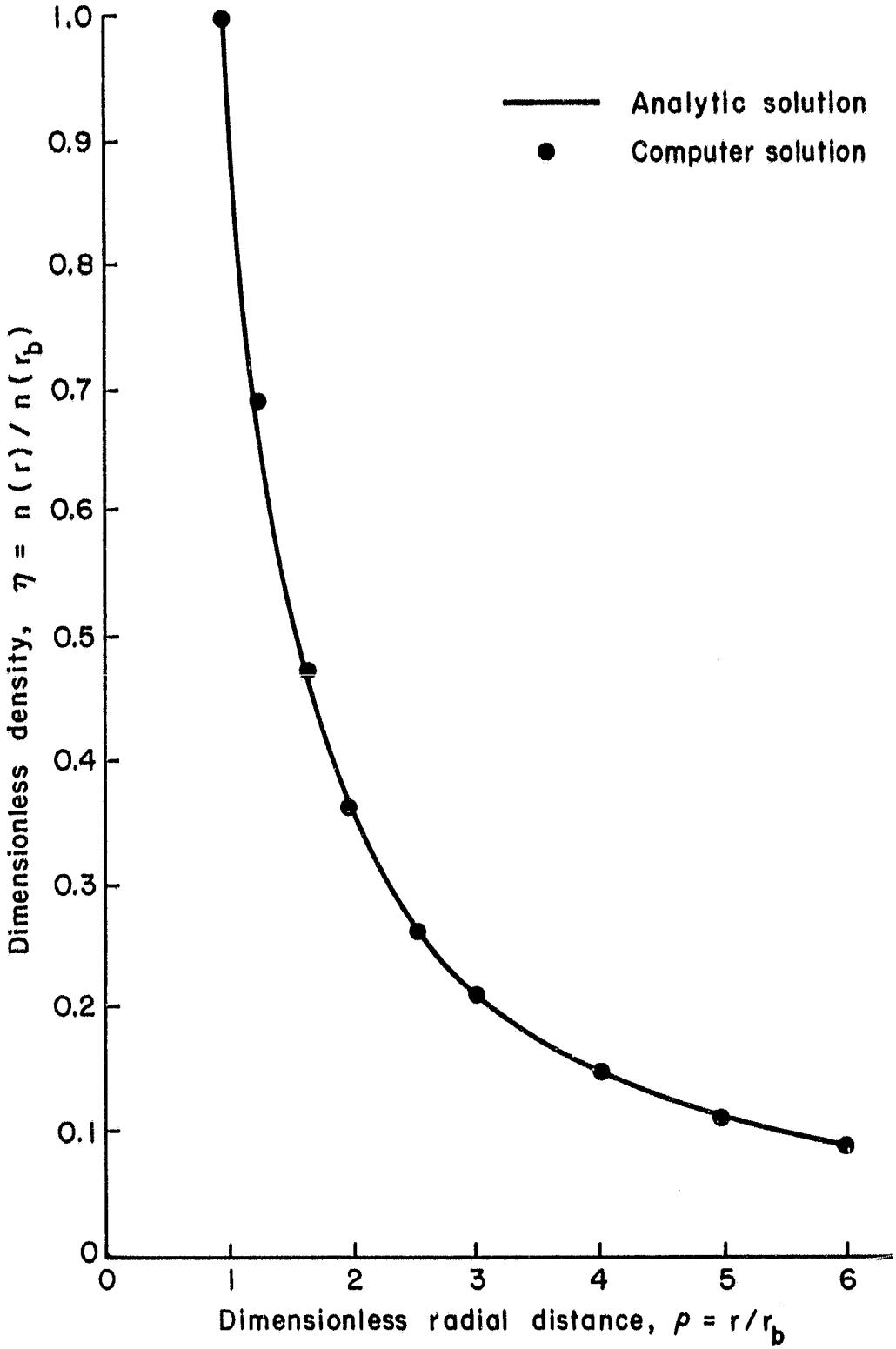


Fig. 9. Comparison of radial densities calculated using the computer code and analytic solution.

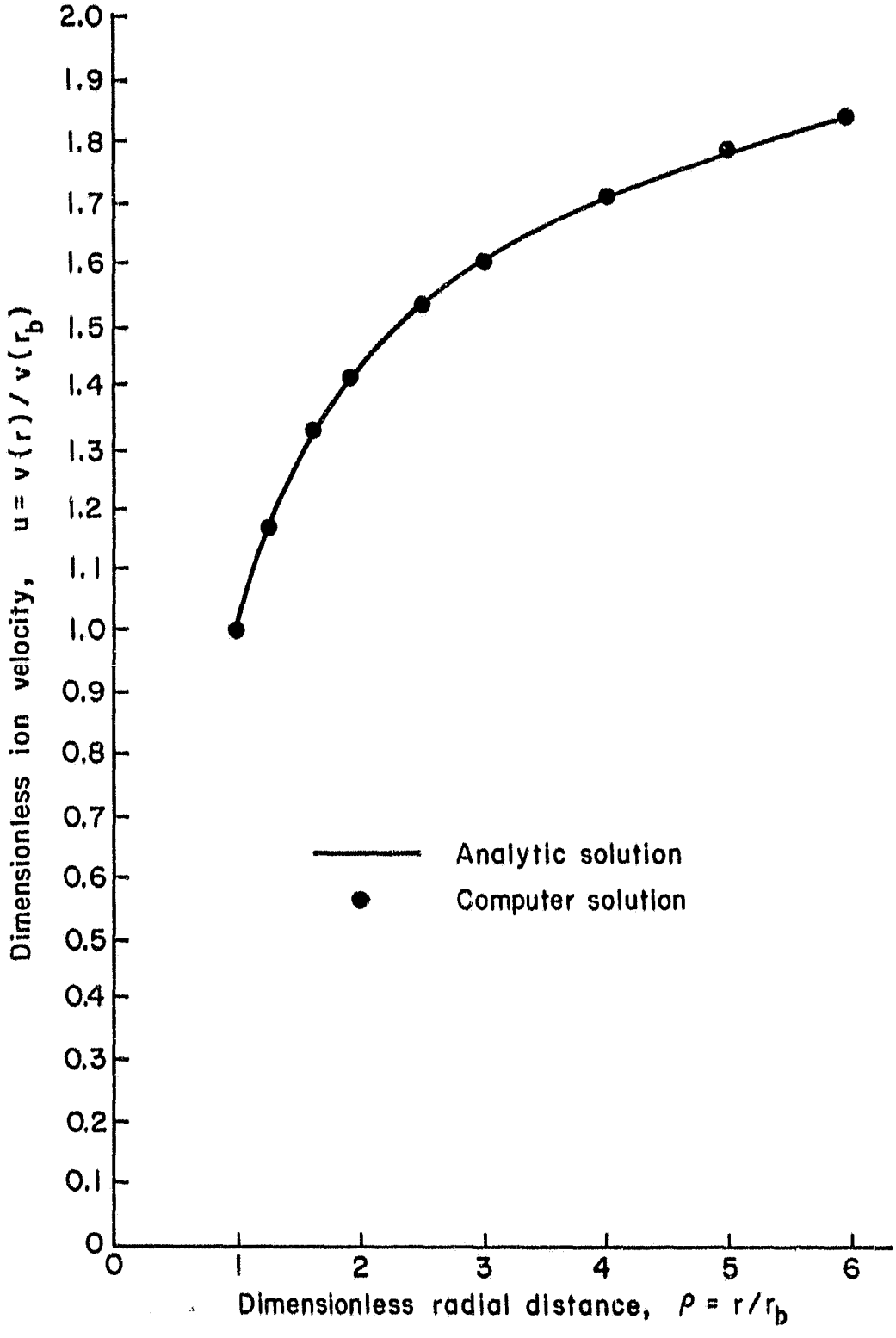


Fig. 10. Comparison of radial velocities calculated using the computer code and analytic solution.

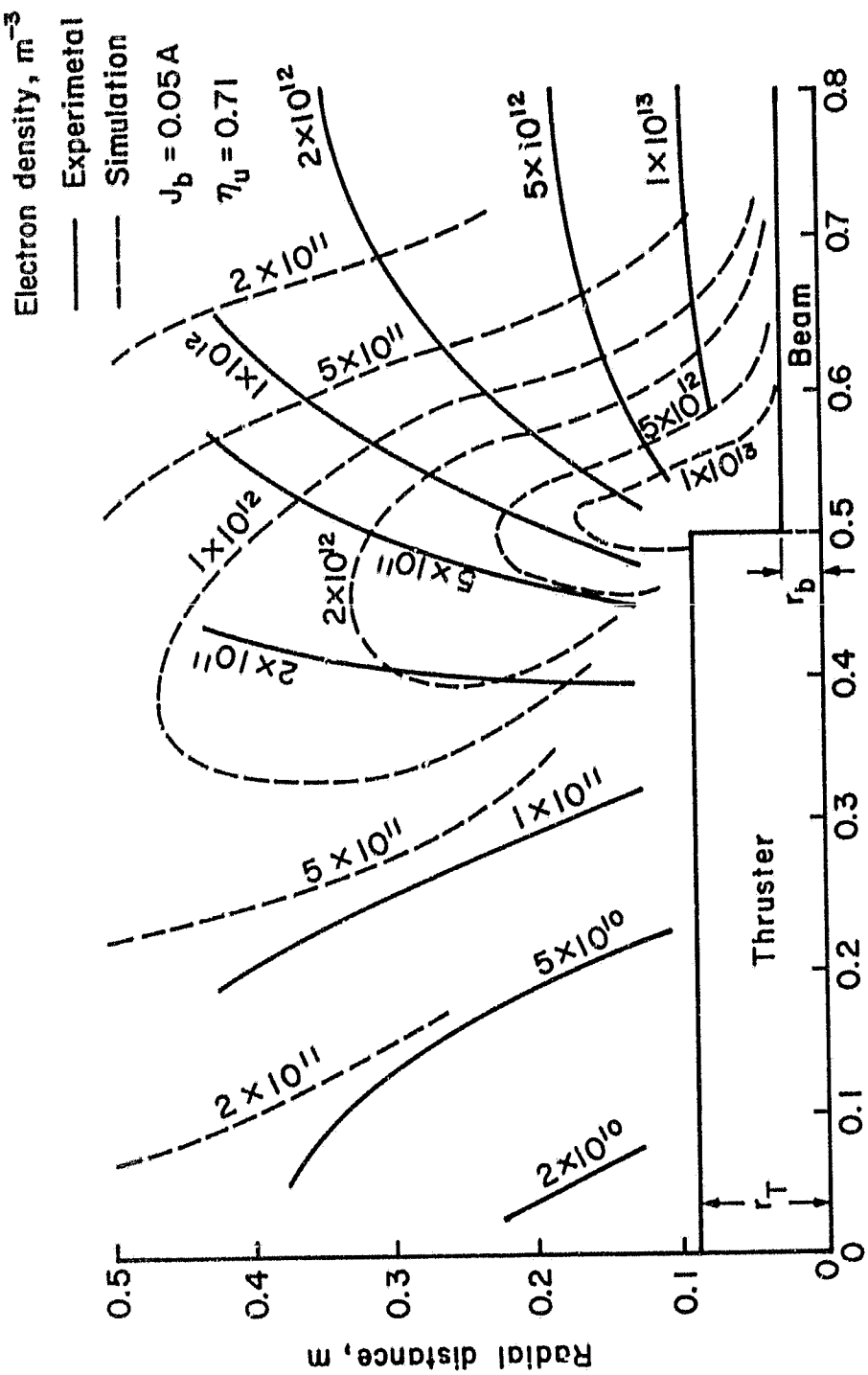


Fig. 11. Simulated and experimental surveys of electron density for a 5 cm thruster.

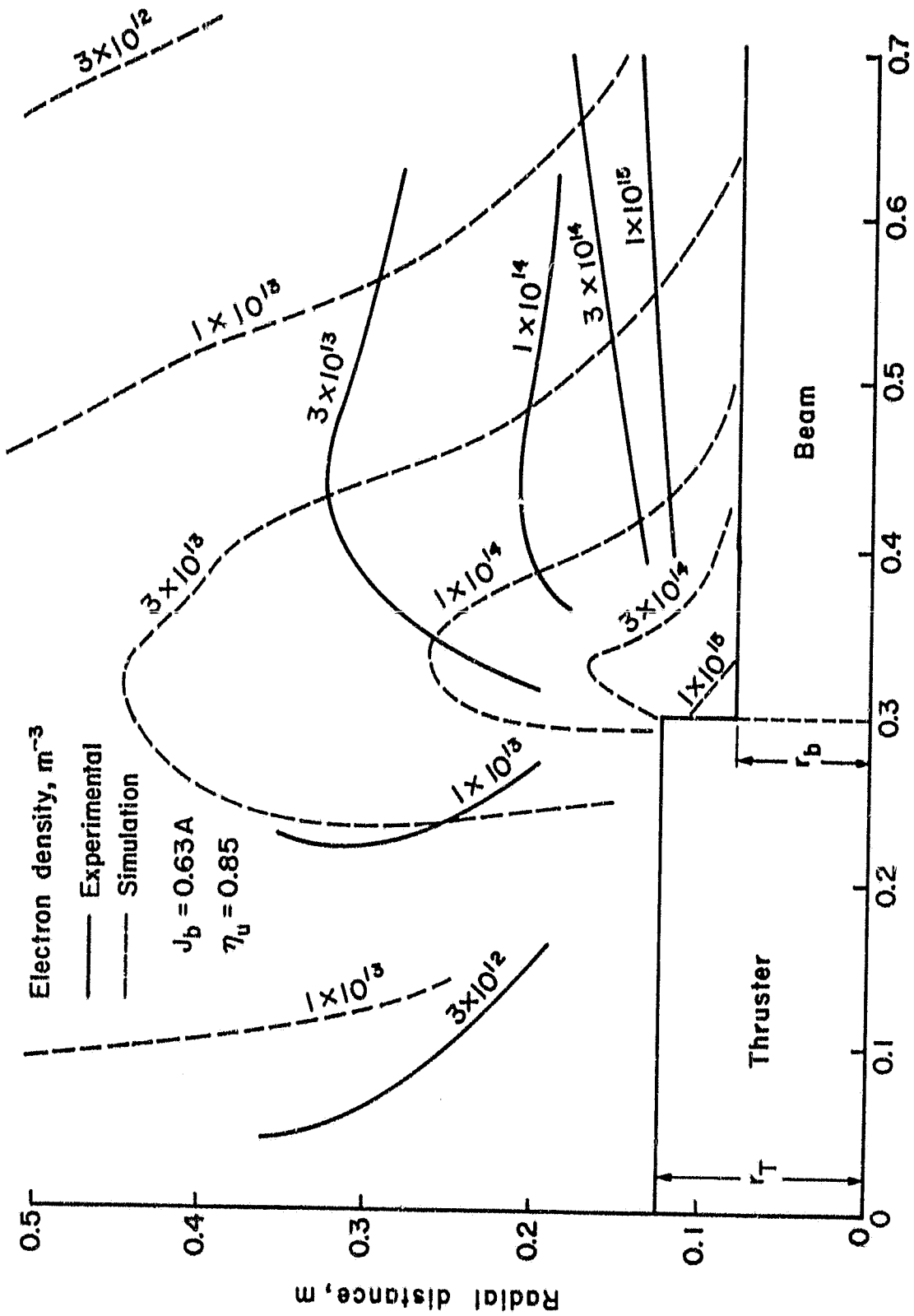


Fig. 12. Simulated and experimental surveys of electron density for a 15 cm thruster.

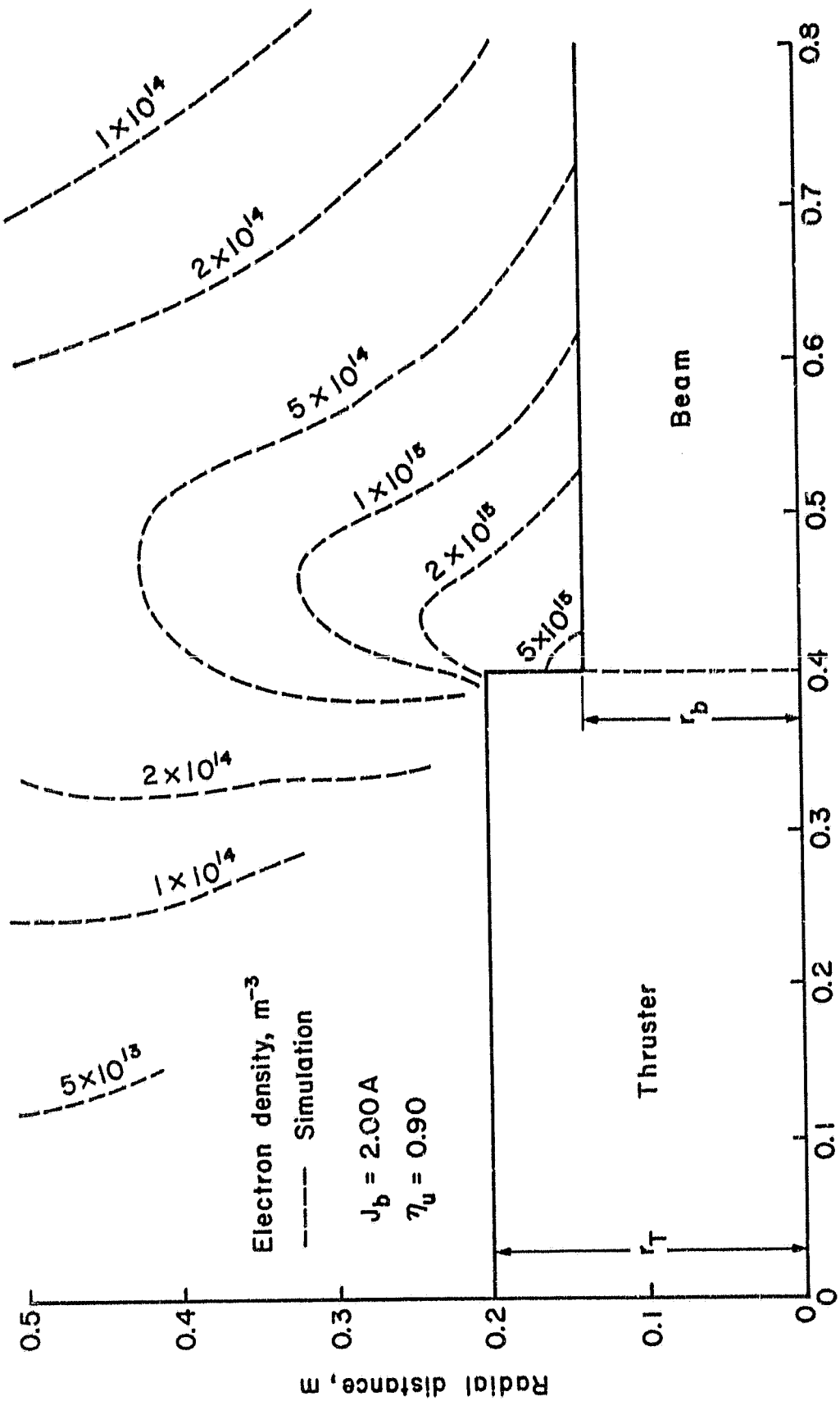


Fig. 13. Simulated survey of electron density for a 30 cm thruster.

trajectory directions obtained using the simulation are in good agreement with the experimental measurements of Carruth and Brady.[10]

Figures 14, 15 and 16 show typical ion trajectories generated by the simulation using the data in the SAMPLE INPUT section for a 5 cm, 15 cm and 30 cm thruster, respectively. The dots occur every 60 iterations.

Figure 17 is a simulation of the 15 cm thruster for use in comparing the computer code contained herein to that discussed in relation to Fig. 8 of the October 1980 report.[3]

Limitations in Use

Major factors affecting the accuracy of the simulation obtained are the number of ion paths used, the number of iterations performed and the time interval used. When the number of ion paths simulated is increased, either more iterations must be used or the time interval size decreased through use of the variable TIMEMU. This is necessary to keep the spacing between the paths comparable to the distance the ion travels in one iteration, this is accomplished using the scaling ratio, S. If care is not taken in doing this, path crossings will sometimes occur, especially among the paths within one beam radius of the thruster. These path crossings result from plasma properties changing so rapidly that the error in a path location will exceed the local path spacing. The procedure used in carrying out the simulation depends on a "laminar" path structure, that is, no intersection of paths. The existence of any crossed paths, therefore, invalidates local calculations of densities and the other related parameters.

Other limitations are imposed by core storage allocations, external file space, and the CPU time available in a particular machine.

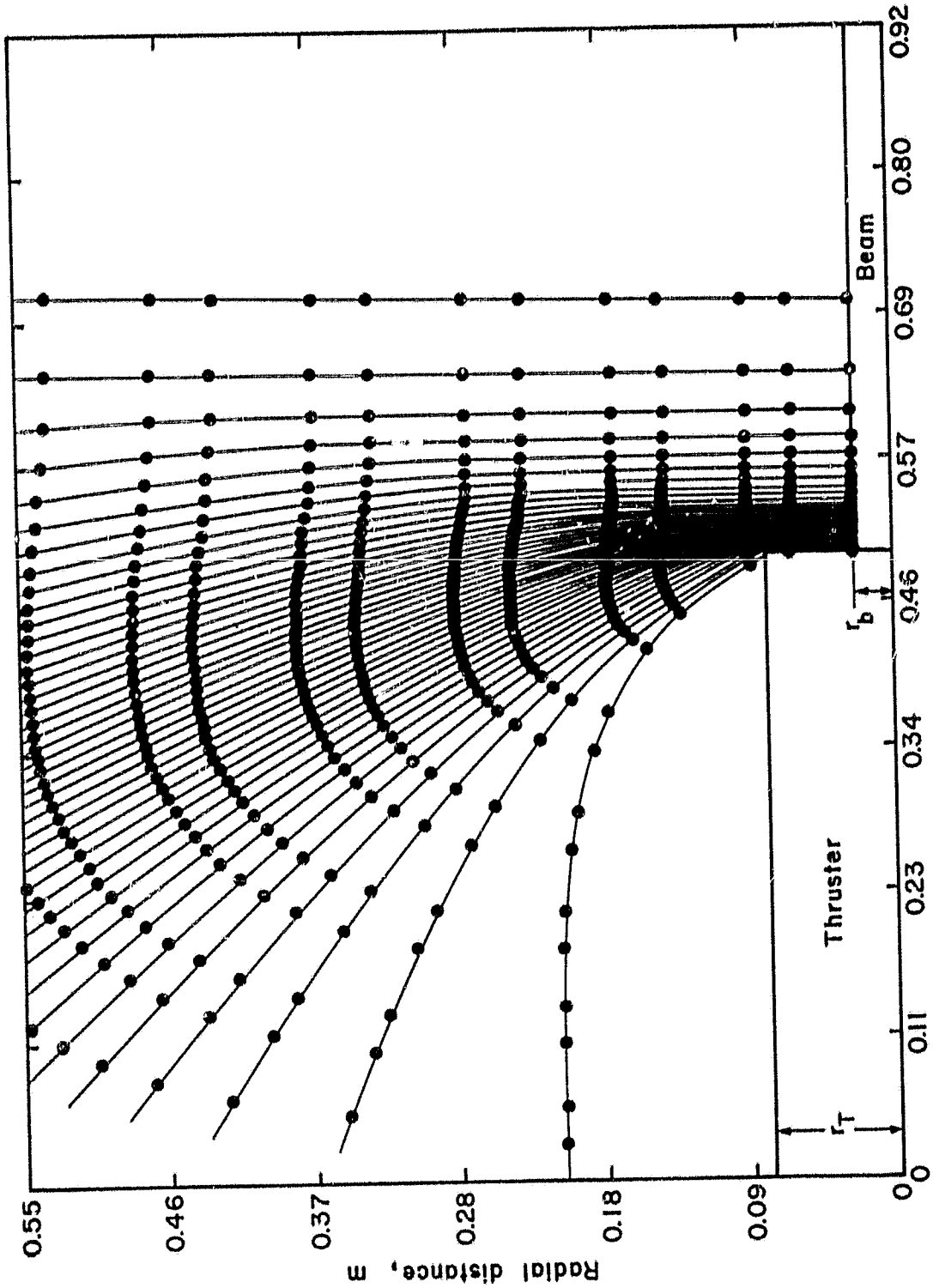


Fig. 14. Ion trajectories generated using data from the SAMPLE INPUT section for a 5 cm thruster.

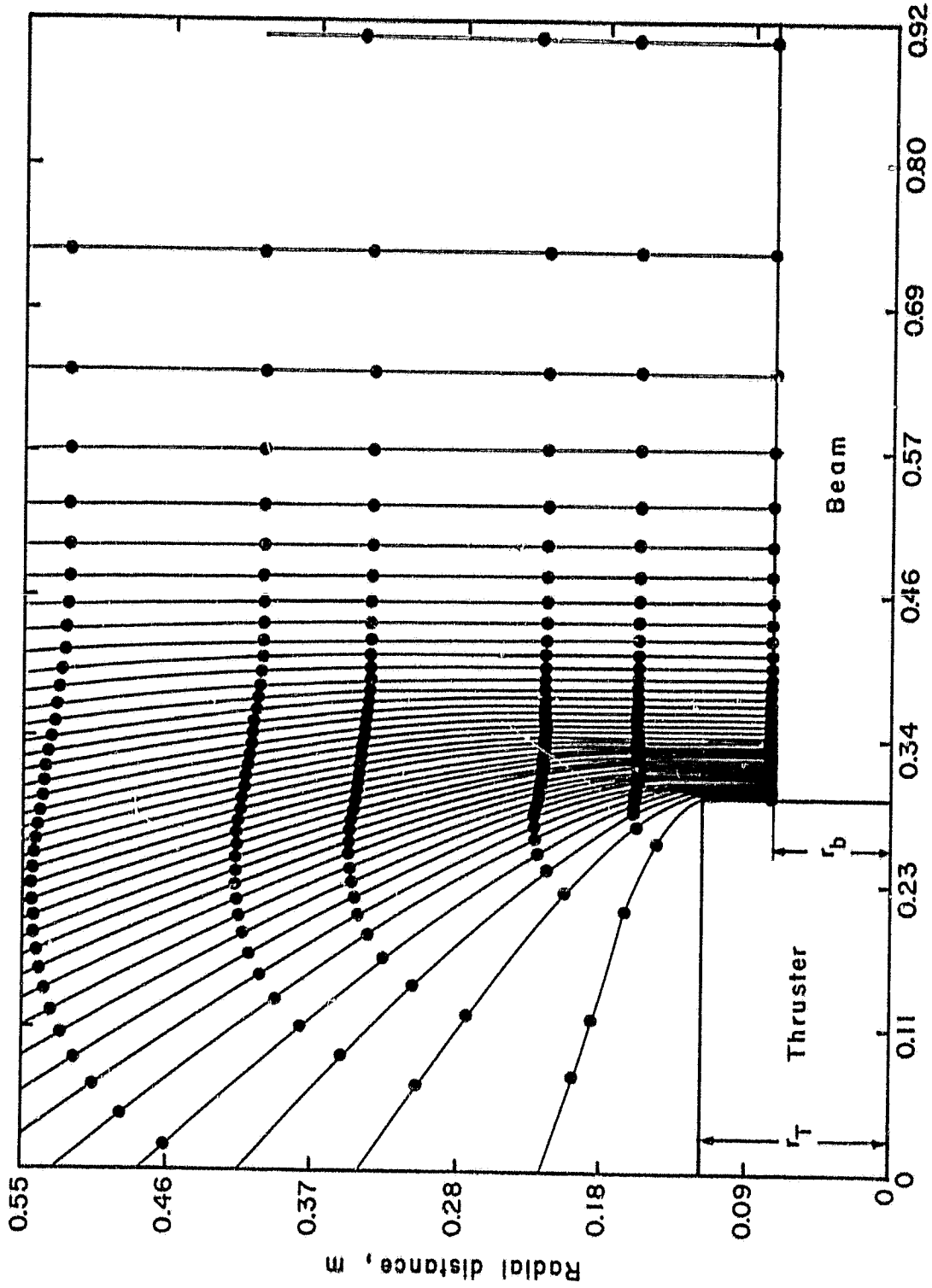


Fig. 15, Ion trajectories generated using data from the SAMPLE INPUT section for a 15 cm thruster.

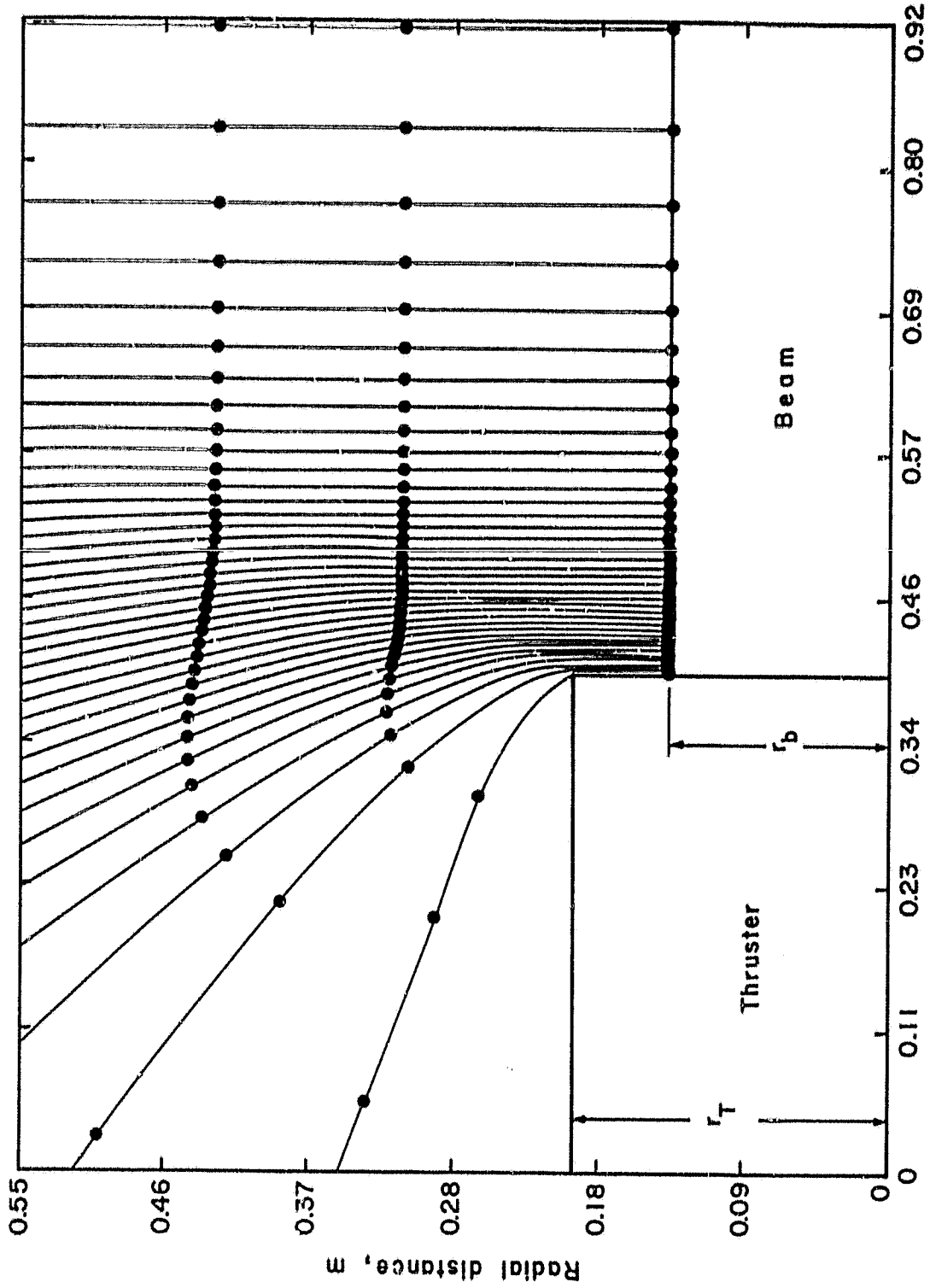


Fig. 16. Ion trajectories generated using data from the SAMPLE INPUT section for a 30 cm thruster.

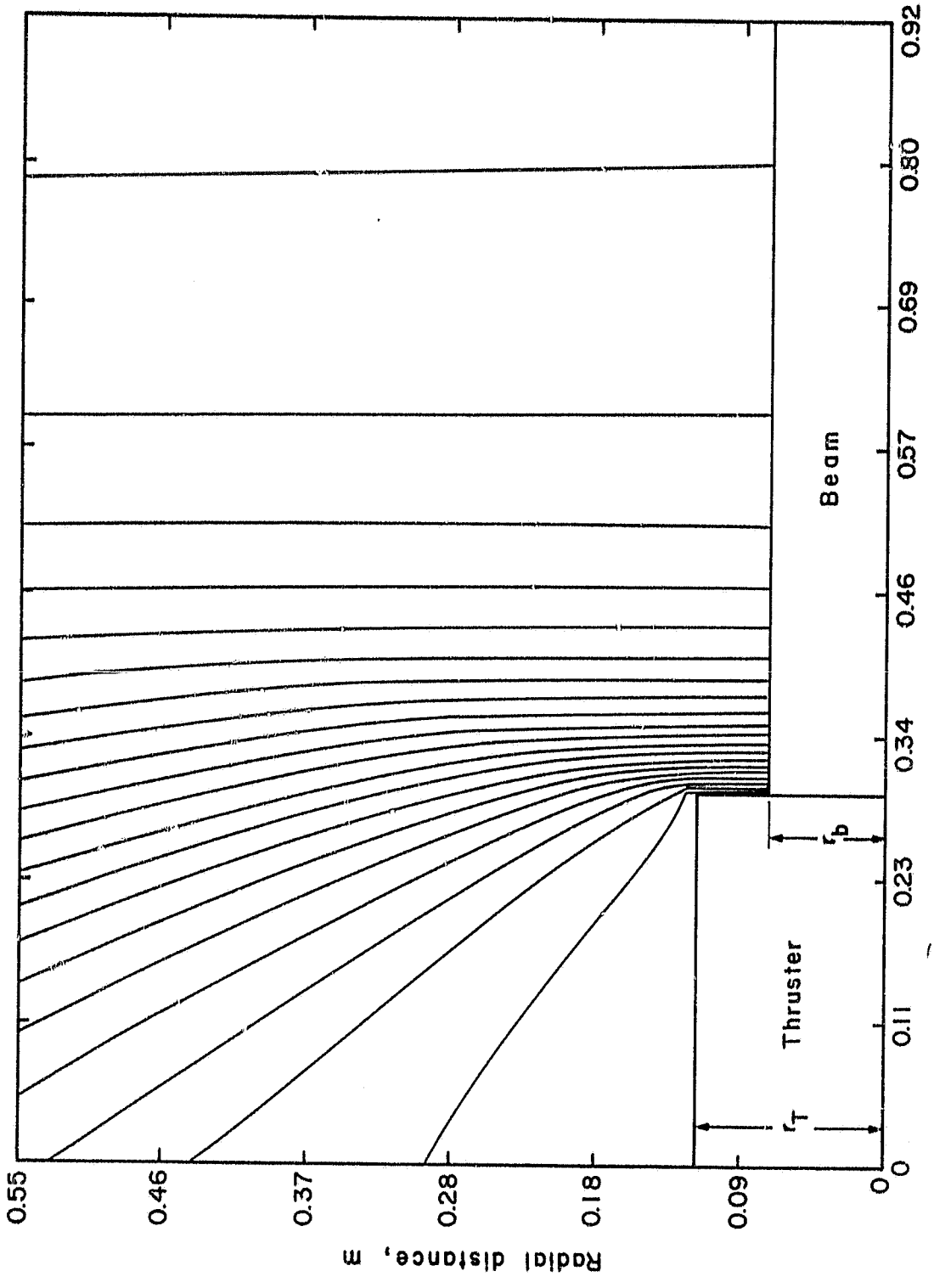


Fig. 17. Ion trajectories generated by the computer simulation for a 15 cm thruster, one stage, 20 ion trajectories and 30 iterations. Other quantities same as in SAMPLE INPUT section.

The present code takes approximately 20 seconds to compile, 150 seconds to execute three stages with NUMION=40, NUMIT=150 and ICLWRT=1, occupies about $140K_8$ bytes of central memory and writes approximately 125,000 characters on an external file for three stages, as above, on a Control Data Cyber 171 computer.

It should be noted that the code could be significantly simplified and shortened if it were translated to Fortran 77 (Fortran V).

REFERENCES

1. J. F. Staggs, W. P. Gula, and W. R. Kerslake, "Distribution of Neutral Atoms and Charge-Exchange Ions Downstream of an Ion Thruster," J. Spacecr. Rockets, Vol. 5, pp. 159-164 (1968).

2. J. E. Barnes, R. S. Robinson, and H. R. Kaufman, "Computer Code for Charge-Exchange Plasma Propagation," Contractor Report, JPL Contract No. 955322, March 1980.

3. R. S. Robinson and H. R. Kaufman, "Computer Code for Charge-Exchange Plasma Propagation," Contractor Report, JPL Contract No. 955322, October 1980.

4. W. Knauer, J. R. Bayless, G. Todd, and J. W. Ward, NASA Contr. Rep. CR-72675, May 1970.

5. R. Worlock, G. Trump, J. M. Sellen, and R. F. Kemp, AIAA Paper No. 73-1101 (1973).

6. G. K. Komatsu, R. K. Cole, D. K. Hoffmaster, and J. M. Sellen, "Charge-Exchange Ion Formation and Motion in Mercury Ion Engine Thrust Beams," AIAA Paper No. 75-428 (1975).

7. H. R. Kaufman, "Charge-Exchange Plasma Generated by an Ion Thruster," NASA Contractor Report, CR-134844, June 1975.

8. H. R. Kaufman, "Interaction of a Solar Array with an Ion Thruster Due to the Charge-Exchange Plasma," NASA Contractor Report, CR-135099, October 1976.

9. H. R. Kaufman, "Charge-Exchange Plasma Generated by an Ion Thruster," NASA Contractor Report, CR-135318, December 1977.

10. M. R. Carruth, Jr. and M. E. Brady, "Measurement of the Charge-Exchange Plasma Flow from an Ion Thruster," J. Spacecr. Rockets, Vol. 18, No. 5, pp. 458-461 (1981).

11. M. R. Carruth, Jr. and Y. S. Kuo, "Ion Thruster Plume Effects on Spacecraft Surfaces," AIAA PaperNo. 80-1228 (1980).

12. M. R. Carruth, Jr., "Facility Produced Charge-Exchange Ions," JPL Publication 80-92, pp. 147-166, January 1981.

13. R. S. Robinson, H. R. Kaufman, and D. R. Winder, "Simulation of Charge-Exchange Plasma Propagation Near an Ion Thruster Propelled Spacecraft," AIAA Paper No. 81-0744, April 1981.

14. I. Katz, D. E. Parks, M. J. Mendell, and G. W. Schnuelle, "Parasitic Current Losses Due to Solar Electric Propulsion Generated Plasmas," AIAA Paper No. 81-0740, April 1981.

15. J. M. Sellen, Jr., W. Bernstein, and R. F. Kemp, Rev. Sci. Instr., Vol. 36, pp. 316-322 (1965).

16. H. S. Ogawa, R. K. Cole, and J. M. Sellen, Jr., "Measurements of Equilibration Between a Plasma "Thrust" Beam and a Dilute "Space" Plasma," AIAA Paper No. 69-263 (1969).

17. H. S. Ogawa, R. K. Cole, and J. M. Sellen, Jr., "Factors in the Electrostatic Equilibration Between a Plasma Thrust Beam and the Ambient Space Plasma," AIAA Paper No. 70-1142 (1970).

18. VERSAPLOT-07 Graphics Programming Manual, 50028-90001, Pub. No. 5921-01, December 1976, VERSATEC, a XEROX Co., Santa Clara, California.

19. H. R. Kaufman and G. C. Isaacson, "The Interactions of Solar Arrays with Electric Thrusters," AIAA Paper No. 76-1051 (1976).

APPENDIX A

ANALYTIC SOLUTION

A theoretical benchmark is valuable for verification of the computer code developed to model the charge-exchange plasma propagation in the vicinity of an operating ion thruster. An analytic solution is developed herein for that purpose.

A cylindrical ion beam is assumed with a length very much greater than r_b, the beam radius. The current density representing positive charge-exchange ion production in the beam is assumed uniform along the beam.

In the region exterior to the beam, three basic physical conditions are assumed to hold for the ion population and/or the plasma as a whole. The first is continuity of ion current represented by

$$\vec{\nabla} \cdot \vec{j} = 0 \tag{A1}$$

where $\vec{j}$ is the ion current density. The barometric equation is also used to relate plasma density to local potential V

$$n = n_{o,ref} \exp[e(V-V_o)/kT_e] \tag{A2}$$

where V_o is the potential at the reference density $n_{o,ref}$ and T_e is the electron temperature in the region exterior to the beam. Finally, energy conservation for singly charged ions is represented by

$$|\vec{v}| = (v_o^2 - 2e(V-V_o)/m_i)^{1/2} \tag{A3}$$

where $\vec{v}$ is the ion velocity and m_i is the ion mass. As a boundary condition at the beam edge, ions are assumed to have acquired the Bohm velocity

$$v_o = v_B = (kT_{e_B}/m_i)^{1/2} \tag{A4}$$

where T_{e_B} is the electron temperature in the beam.

The ion current density is related to the streaming velocity by

$$\vec{j} = ne\vec{v} . \tag{A5}$$

For the assumed symmetry, the velocity is radial and is

$$\vec{v} = v(r)\hat{r} . \tag{A6}$$

In cylindrical coordinates, Eq. (A1) can be written with the substitution of Eq. (A5) as

$$\frac{1}{r} n(r)v(r) + n(r) \frac{\partial v(r)}{\partial V} \frac{\partial V(r)}{\partial r} + v(r) \frac{\partial n(r)}{\partial V} \frac{\partial V(r)}{\partial r} = 0 \tag{A7}$$

which can be solved for V(r) by eliminating n(r) and v(r) using Eqs. (A2) and (A3). The solution can then be written as

$$(1 - e(V-V_o)/kT_e)^{1/2} \exp(-e(V-V_o)/kT_e) = r/r_b . \tag{A8}$$

The density and velocity can be calculated using Eq. (A8) along with Eq. (A3) or Eq. (A4).

The solution in terms of potential, density and velocity is displayed in dimensionless form in Figs. 1A, 2A, and 3A.

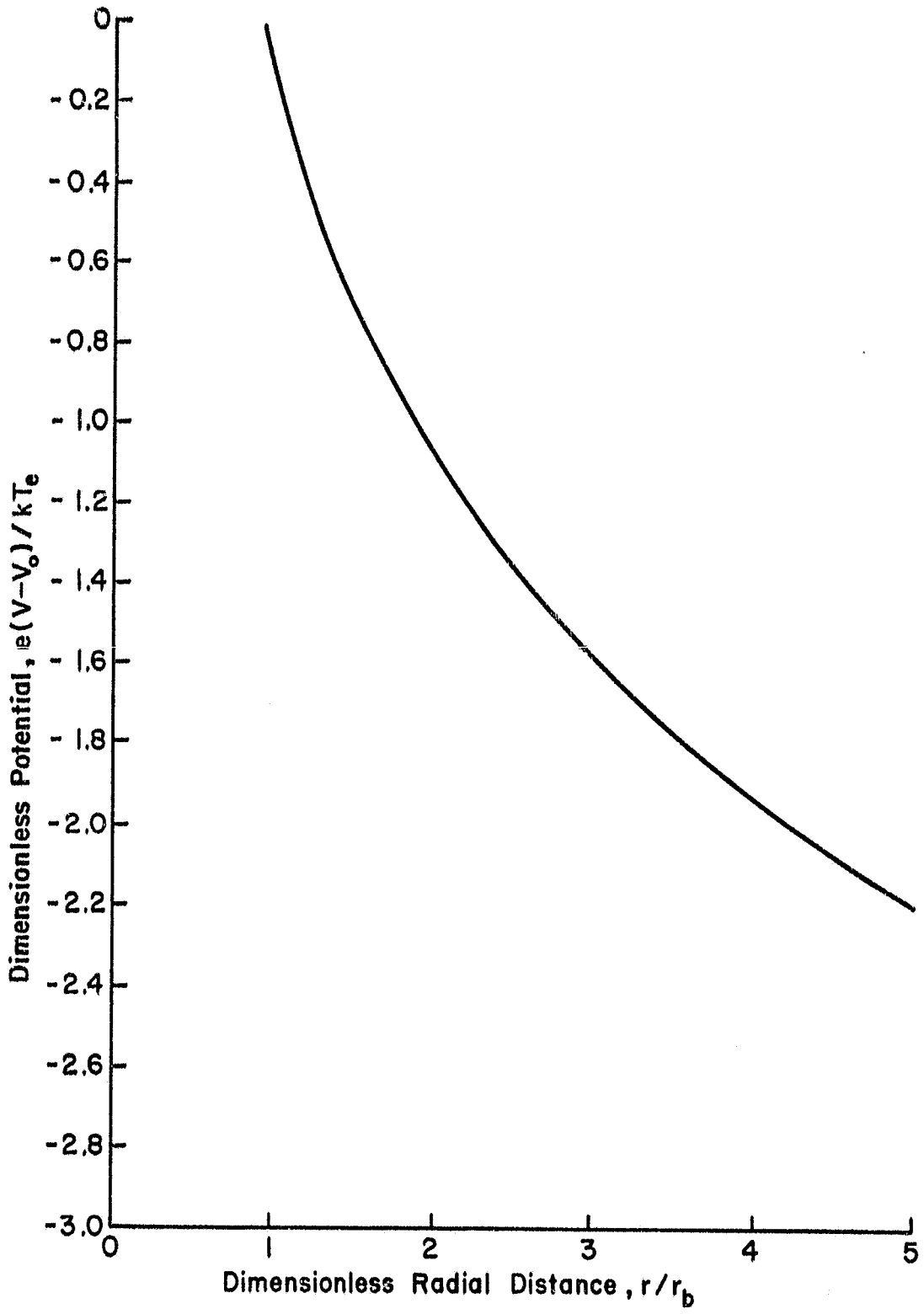


Fig. 1A. Potential as a function of radial distance.

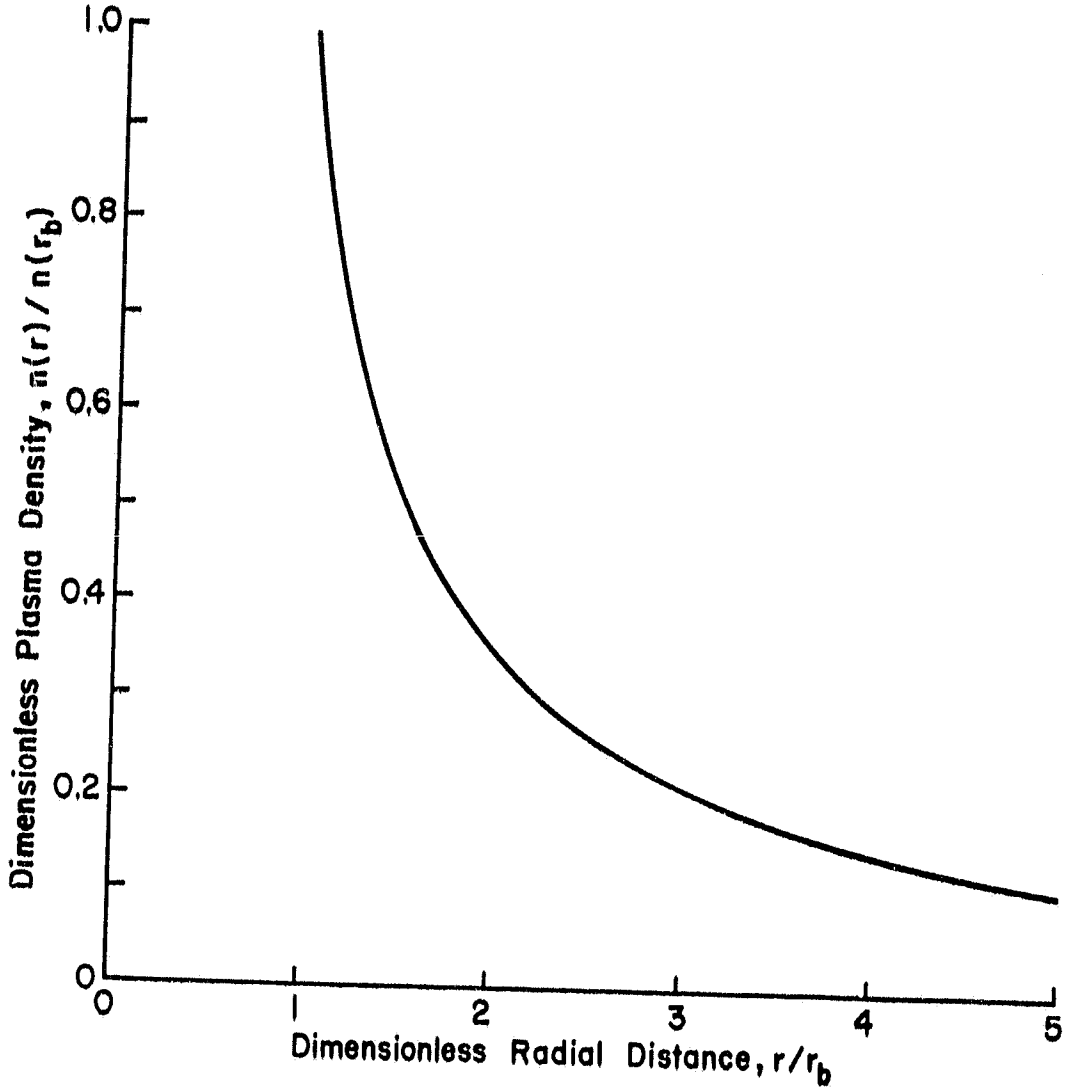


Fig. 2A. Density as a function of radial distance.

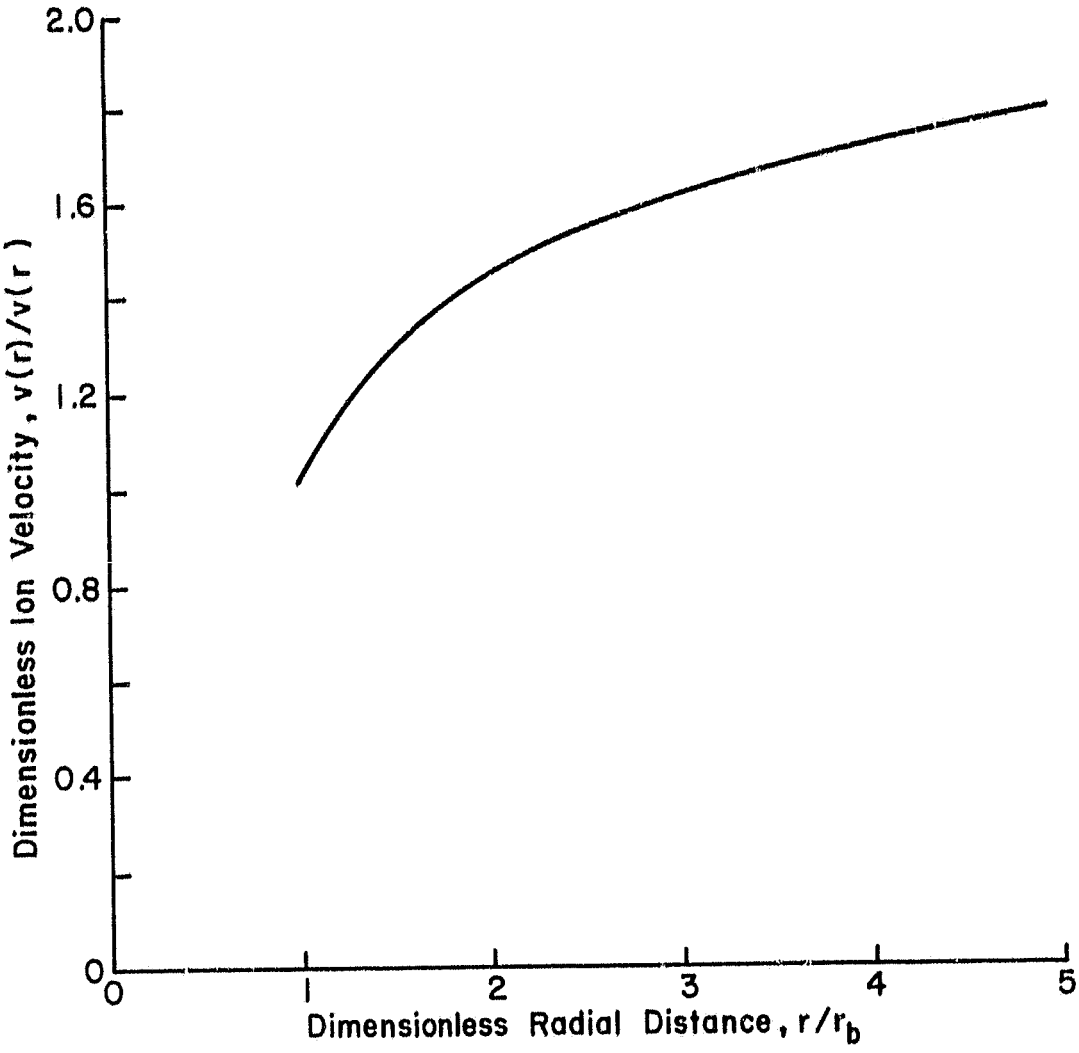


Fig. 3A. Velocity as a function of radial distance.

APPENDIX B

BEAM CURRENT DENSITY PROFILES

Determination of the total rate of charge exchange ion production in the beam volume must take into account both the beam current density profile of the ion thruster and the spatial distribution of neutral propellant atoms in the path of the ion beam.

For theoretical calculations a simplified ion beam current density distribution is usually chosen and axial symmetry assumed. A closed-form solution exists only for the very simplest distribution which has been used for conservative estimates of charge exchange ion production.[19] In this case the current density is given by a Dirac delta function,

$$j_b = J_b \delta(\vec{r}) \ , \tag{B1}$$

where J_b is the total beam current and $\delta(\vec{r})$ is the two-dimensional Dirac delta function. This expression essentially places all of the beam current on the thruster axis where the neutral density follows a simple function, thus allowing a closed-form solution.

A current density profile that is more accurate for the larger, multipole thrusters is a uniform beam current density

$$j_b = J_b/\pi r_b^2 \ , \tag{B2}$$

where r_b is the beam radius. Another profile that approximates the beam of some divergent field thrusters is a parabolic profile

$$j_b = 2J_b(1 - r^2/r_b^2)\pi r_b^2 \ , \tag{B3}$$

where r is the distance from the beam axis.

The expressions given above for beam current density are normalized such that

$$J_b = \int_0^{2\pi} \int_0^{r_b} j_b r dr d\theta \ . \tag{B4}$$

A Gaussian profile is also used by some workers where

$$j_b = (J_b/\pi r_b^2) e^{-(r/r_b)^2} \ , \tag{B5}$$

subject to the normalization:

$$J_b = \int_0^{2\pi} \int_0^{\infty} j_b r dr d\theta \ . \tag{B6}$$

The uniform, parabolic and Gaussian profiles can also accommodate a defined rate of beam spreading as it propagates. The simulation described herein does not include effects from beam spreading.

Neutral gas leaving the accelerator system is taken to be in free molecular flow so that the effective neutral density at an arbitrary point such as that shown in Fig. 1B is proportional to the subtended solid angle of the ion optics as viewed from the point (r,θ,z). Effective neutral densities were calculated numerically for an r-z matrix with a resolution of 0.1 r_b. For the near field, where about half of the charge exchange takes place, the calculated densities are given in dimensionless form in Table 1B.

The calculated distribution of neutral propellant along with the beam current density profile allow a calculation of charge exchange ion production rate per unit length as a function of distance from the ion optics.

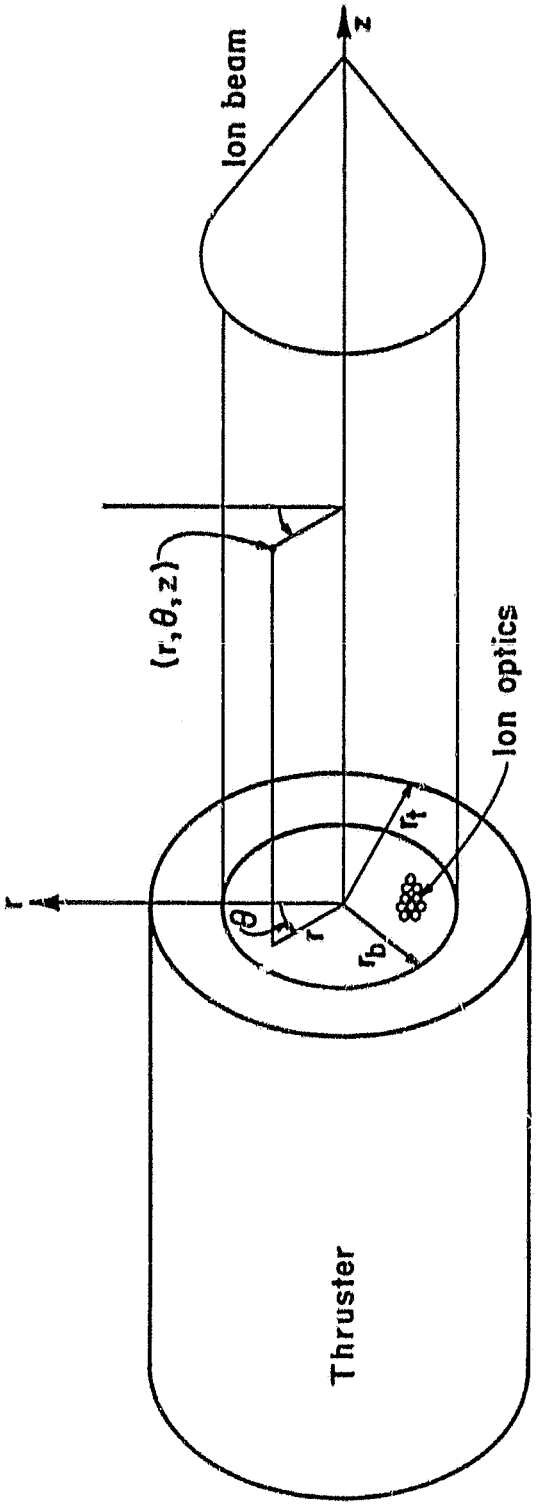


Fig. 1B. Coordinate system for thruster and ion beam geometry.

Table 1B. Density of Neutral Propellant Efflux, $n(r,z)/n_{o,ref}$

	z/r_b									
r/r_b	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.0	4.52×10^{-1}	4.03×10^{-1}	3.57×10^{-1}	3.15×10^{-1}	2.77×10^{-1}	2.43×10^{-1}	2.14×10^{-1}	1.88×10^{-1}	1.66×10^{-1}	1.47×10^{-1}
0.1	4.51	4.02	3.56	3.14	2.76	2.42	2.13	1.87	1.65	1.46
0.2	4.49	4.00	3.53	3.11	2.73	2.39	2.10	1.85	1.63	1.44
0.3	4.47	3.96	3.48	3.05	2.67	2.34	2.05	1.80	1.59	1.41
0.4	4.43	3.90	3.40	2.97	2.59	2.26	1.98	1.74	1.54	1.36
0.5	4.38	3.81	3.29	2.85	2.47	2.15	1.89	1.66	1.47	1.31
0.6	4.31	3.68	3.14	2.69	2.33	2.02	1.77	1.57	1.39	1.24
0.7	4.18	3.47	2.92	2.48	2.14	1.87	1.64	1.45	1.30	1.16
0.8	3.94	3.15	2.61	2.21	1.92	1.69	1.49	1.33	1.19	1.08
0.9	3.41	2.64	2.20	1.90	1.67	1.49	1.33	1.20	1.09	9.86×10^{-2}
1.0	2.15	1.91	1.72	1.55	1.41	1.28	1.17	1.07	9.75×10^{-2}	8.94

As a first approximation, the charge exchange production rate can be calculated in closed form if the Dirac delta function is used for the ion beam profile. The neutral density along the beam axis is given by

$$n_o(z) = \frac{n_{o,ref}}{2}\left(1 - \frac{z}{\sqrt{z^2 + r_b^2}}\right), \tag{B7}$$

where $n_{o,ref}$ is defined as the density that would provide the correct loss rate of neutral propellant through an opening of the same area as the beam. The neutral loss rate is then

$$\dot{N}_o = r_b^2 n_{o,ref}\sqrt{\frac{\pi k T_o}{2m_o}}\ , \tag{B8}$$

where T_o is the propellant temperature, m_o is the propellant mass, and k is Boltzmann's constant. The charge exchange ion production rate per unit length is thus

$$\dot{N}_{CE}(z) = J_b Q n(z)/e\ , \tag{B9}$$

for small total production rates. Integrating to obtain the total production rate gives

$$N_{CE_T} = J_b Q n_{o,ref} r_b/2e\ , \tag{B10}$$

As the assumed ion beam profile becomes less peaked, progressing from a Dirac delta function through Gaussian and parabolic functions to a uniform distribution, the production rate of charge exchange ions

will diminish as more of the beam passes through the peripheral regions of lower neutral propellant density. The two extremes in production rates as a function of distance thus use a delta function and a uniform function for the beam profile. The results for these two extremes are shown in Fig. 2B. Table 2B gives the calculated total charge exchange ion production rates for these two extremes, as well as the intermediate parabolic ion beam profile. The ion beam profile is clearly not a dominant parameter for charge exchange ion production.

The simulation developed to model the charge exchange plasma propagation can be modified for an arbitrary input for the production rate as a function of distance.

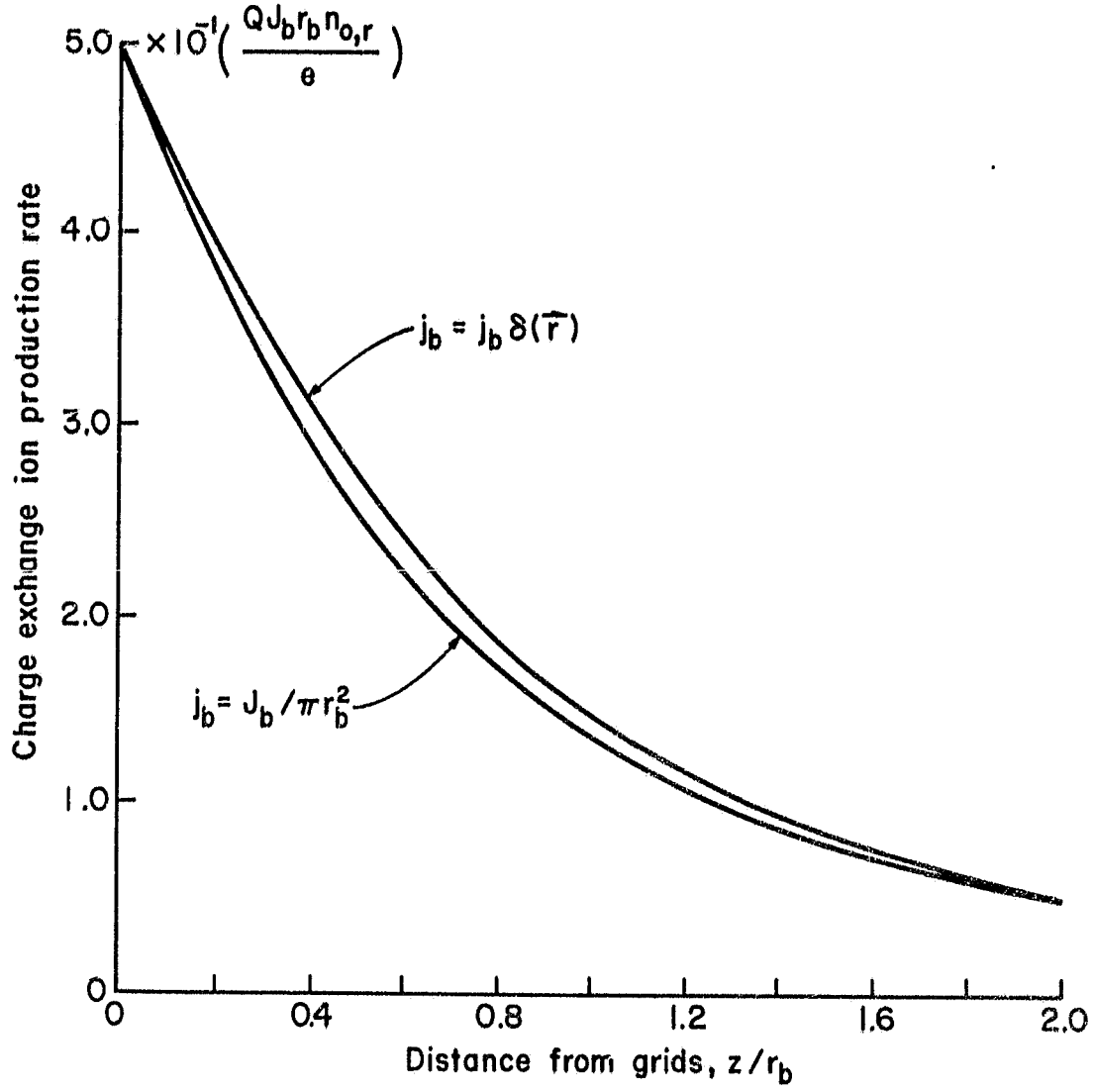


Fig. 2B. Charge-exchange ion production rates as a function of distance from the grids for extremes of possible beam current density distributions.

Table 2B. Charge-Exchange Ion Production Rates for Different Beam Current Density Profiles.

Profile	Production Rate ($J_b Qn_{o,ref} r_b/2e$)
$j_b \delta(\vec{r})$	1.00
$2J_b(1 - r^2/r_b^2)/\pi r_b^2$	0.97
$J_b/\pi r_b^2$	0.94

APPENDIX C

GLOSSARY OF VARIABLES FOR PLASIM

This glossary contains definitions of the variables used in the driver program PLASIM, the subroutines BLOCK DATA, READER, INIT, CALC, CALCD, BOUND, and WRIT and the function subroutine DS, including all the variables in the COMMON blocks. The variables used in the plotting subroutines, VRSPLT, VRSPL, LNPLT, and PLOTW are not defined here.

BK: Boltzman's constant (J/K).

BMCUR: Beam current (AMPS).

CEXSEC: Charge exchange cross section (METERS SQUARED).

C: Time interval divided by mass of ion (TIME/UMSION).

COSPAR: Cosine of angle between line parallel to path and horizontal.

COSPER: Cosine of angle between line perpendicular to path and horizontal.

DELTAX: Difference between two x coordinates on present path.

DELTAZ: Difference between two z coordinates on present path.

DELTLX: Difference between two x coordinates on left path.

DELTLZ: Difference between two z coordinates on left path.

DELTRX: Difference between two x coordinates on right path.

DELTRZ: Difference between two x coordinates on right path.

DELTRZ: Difference between two z coordinates on right path.

DN(I,N): Density on path I at iteration N.

DNI(I): Initial density between paths (I-1) and (I).

DNL: Density to left side of current path.

DNOB: Constant used in calculation of the densities, combination of quantities including; BMCUR, CEXSEC, RB, NUMION, UTIL, Q and VELNEU.

DNR: Density to right side of present path.

DS: Perpendicular displacement from present path to boundary.

DSPLL: Displacement to left hand path.

DSPLR: Displacement to right hand path.

DUMMYP, DUMMYL, DUMMYR: Dummy variables used in the calculation of the intercepts. Contain intermediate results.

DSPLIP: Displacement of ion along path.

F: Total force acting on ion.

FPAR: Force acting parallel to the path.

FPARX: Component of FPAR acting in x direction.

FPARZ: Component of FPAR acting in z direction.

FPER: Force acting perpendicular to the path.

FPERX: Component of FPER acting in x direction.

FPERZ: Component of FPER acting in z direction.

FX: Component of F acting in x direction (FPERX + FPARX).

FZ: Component of F acting in z direction (FPERZ + FPARZ).

I: Allows do loop index, II, to be passed through COMMON, path index.

ICLERR: Used to determine if code generated error messages, for non-fatal errors, are written out.
= 0, Write messages.
= 1, Do not write messages.

ICLPLT: Used to determine which (if any) plotting routine is used;
= 1, Use subroutine VRSPLT.
= 2, Use subroutine LNPLT.
= 3, Use both VRSPLT and LNPLT.
= anything else, No plots.

ICLWRT: Frequency with which results of CALC are output. Write statement in CALC called after every ICLWRT number of iterations.

IDEF: Used to determine when the x and z coordinates of an ion trajectory exit point are defined (every other time).

IDXNEW: New index for right or left-most path.

IFLAG1: Used to determine if intersections were found to both the left and the right without using linear extrapolation,
= 0, Linear extrapolation not used.
= 1, Linear extrapolation used on left.
= 2, Linear extrapolation used on right.
= 3, Linear extrapolation used in both cases.

IFLAG2: Error flag, tells whether or not the neighboring paths are active,
= 0, No error, path active.
> 0, Error exists, value references program statement where error condition originated.
= -1, Path to left is not active.
= -2, Path to right is not active.
= -3, Neither path (right or left) is active.
(Negative values do not indicate an error condition.)

IFLAG3: Used to determine when there is a boundary on the right or left (or both) of the current path.
= 1, No boundary intersected by the perpendicular to the current path on either side.
= 2, Boundary intersected on left.
= 3, Boundary intersected on right.
= 4, Boundary intersected on both the left and right.

IFLAG4: Trajectory one renormalization flag,
= 0, Continue iterating as usual.
= 1, Recalculate position, velocity and density of ion on path 1.

IFVAR: Dummy variable used as an argument in an if statement.

IF1: A format for alpha-numeric data I/O.

IF2: A format for alpha-numeric data I/O.

II: Do loop index.

IL: Index (I) of left-most active path.

IN: Device code for input unit, used in read statements.

INFO(K): Array storing information describing run.

INITDO: Initial do loop index for DO 80 N which calls CALC.

INITIT: Initial iteration index (stage dependent).

IOUT: Device code for output unit, used in write statements.

IPAG: Page number for output.

IPATHS: Device code for output to external files.

IR: Index (I) of right-most active path.

ISAT: Used as path status holder for reading IPATHS and in defining right boundary for high resolution upstream run.

ISTAT(I): Status of path I,
= 0, Path is active.
= 1 to NMAX, Path is not active, value is iteration number where boundary was intersected.
= 8888, Path is not active, error condition.

ITITL(K): Array storing graph title and axis labels.

ITTOTN: Total number of iterations to be performed.

IW: Number of words required to generate 80 characters,
= 8 For computers with a word size of 10.
= 14 For computers with a word size of 6.
(Used in COMMON /IO/ and Routines BLOCK DATA, READER, WRIT, VRSPLT and VRSPL).

J: Used to determine when WRITE-435 is executed (CALC).

KE: Calling parameter for subroutine WRIT,
= 1, Output heading, initial information and data.
= 2, Output interim status of main variables.
= 3, Finish of a pass, output results, start of new pass.
= 4, Create file of path coordinates.
= 5, Create file of position - density triplets.

KEY: Used to determine type of run,
= 0, Finish plots and terminate (2 blank cards will do).
> 0, High resolution upstream pass, right most boundary is defined using KEY'TH path of file IPATHS (used to define ZBOUND).
= -1, First pass, uniform density distribution.
< -1, Regular run; first pass, normal non-uniform density distribution.

LAB: Label containing computer code name, used for output.

LASTDO: Dummy variable denoting last value of do loop index.

LASTIT: Final (last) iteration index (path status dependent).

LR: Determines which side is being considered (DS),
= 1, Looking to the left.
= 2, Looking to the right.

N: (Working) number of iterations on present path, I, total number of iterations for the present stage.

NIP(I): Total number of completed iterations on path I.

NITP: Iteration pass number, used for output.

NL: (Working) number of iterations on left hand path.

NLM1: NL minus one (NL-1), used for indexing left-hand path.

NMAX: NUMPRE divided by four (NUMPRE / 4) (see WRIT(4)).

NR: (Working) number of iterations on right hand path.

NRM1: LR minus one (NR-1), used for indexing right-hand path.

NSTAG: Allows do loop index NSTAGE to be passed through COMMON.

NSTAGE: Program stage number.

NSTGMU: Used to add 10 to N after first stage, (N stage multiplier),
NSTAG = 1; NSTGMU = 0
NSTAG > 1; NSTGMU = 1

NTOTST: Maximum number of stages to be run.

NUMION: Number of ion paths.

NUMIT: Maximum number of iterations to be performed on any one path during any one stage.

NUMPRE: Number of ion paths (NUMION) from run which created file IPATHS (see WRIT(4)).

NUMTRI: Number of triplets output to be external file.

NUM1: Number of ion paths plus one (NUMION + 1).

NUM2: Two times the number of ion paths plus one (2*NUMION + 1).

PI: Geometrically defined constant, 3.14159265...

PIOV2: PI over (divided by) 2.

Q: Elementary unit of charge (C).

RB: Radius of the beam (METERS).

RBOUND: Radial boundary in positive x direction (METERS).

RB2: Beam radius squared (RB ** 2).

RB95N: 95 percent of the beam radius divided by 2 times the number of ion paths (RB * .95 / (2 * NUMION)).

RT: Radius of the thruster (METERS).

SINOV2: SINPER over (divided by) 2.

SINPAR: Sine of angle between line parallel to path and horizontal.

SINPER: Sine of angle between line perpendicular to path and horizontal.

SLOPEL: Slope of path on left between two "working" points.

SLOPEP: Slope of line perpendicular to current path at endpoint.

SLOPER: Slope of path on right between two "working" points.

SPACER: Initial distance between paths in uniform distribution.

TELIN: Temperature of electrons in the ion beam (EV).

TELOUT: Temperature of electrons outside the ion beam (EV).

THETAP: Angle between line with slope SLOPEP and horizontal.

THRLEN: Thruster length (METERS).

TIME: Time interval, defines iteration step size.

TIMEMU: Time multiplier, used to define the time step in terms of multiple of RBOUND / (VELBOH * NUMIT).

TTHNEU: Temperature of thermal neutrals in chamber (EV).

UMSION: Mass of ions (propellant) (KILOGRAMS).

UTIL: Utilization factor of propellant (part of propellant turned into ions).

VCURR: Plasma potential at present point on path I.

VELBET: Present total velocity between path under consideration and neighboring path.

VELBOH: Bohm velocity.

VELBTL: Present total velocity between path under consideration and path on left.

VELBTR: Present total velocity between path under consideration and path on right.

VELNEU: Thermal velocity of the neutrals in the chamber.

VELTOT: Present total velocity of ion.

VELTT2: Present total velocity of ion along path 2.

VELTX(I): Present total velocity component in x direction.

VELTZ(I): Present total velocity component in z direction.

VELX: Velocity contribution for this iteration along x direction.

VELZ: Velocity contribution for this iteration along z direction.

VL: Plasma potential on left side of present path.

VPREV: Plasma potential at previous point on path I.

VR: Plasma potential on right side of present path.

X: X coordinate of point to be tested.

XINT: X intersection point on boundary line.

XINTL: X intersection, to left, of lines SLOPEP and SLOPEL.

XINTR: X intersection, to right, of lines SLOPEP and SLOPER.

XION(I,1): X coordinate of ion trajectory exit point I.

XION(I,2): First x coordinate of ion after leaving ion beam along trajectory I.

XION(I,N): Present x position of ion.

XION(I,N+1): Newly calculated x position of ion.

XPOSL: X-position (coordinate) half way along the perpendicular displacement (DSPLL) to the neighboring path on the left.

XPOSR: X-position (coordinate) half way along the perpendicular displacement (DSPLR) to the neighboring path on the right.

XVELMU: X velocity multiplier, used to define initial x velocity in terms of some multiple of the Bohm velocity.

Z: Z coordinate of point to be tested.

ZBOUND: Z boundary to right of thruster.

ZCURR: Current z increment, used to get ion exit points.

ZINT: Z intersection point on boundary line.

ZINTL: Z intersection, to left, of lines SLOPEP and SLOPEL.

ZINTR: Z intersection, to right, of lines SLOPEP and SLOPER.

ZION(I,1): Z coordinate of ion trajectory exit point I.

ZION(I,2): First z coordinate of ion after leaving ion beam along trajectory I.

ZION(I,N): Present z position of ion.

ZION(I,N+1): Newly calculated z position of ion.

ZMAX: Maximum z value on path KEY of IPATHS, defines ZBOUND for high resolution upstream run.

ZPREV: Previous z increment, used to get ion exit points.

ZVELMU: Z velocity multiplier, used to define initial z velocity in terms of some multiple of the Bohm velocity.

APPENDIX D

COMPUTER CODE LISTING FOR PLASIM

```
      PROGRAM PLASIM (INPUT, OUTPUT, TAPE5 = INPUT, TAPE6 = OUTPUT,
     1               NPARAM, PATHS, TAPE7 = PATHS, DEBUG = OUTPUT)
C
C     ************  DRIVER ROUTINE  ************
C
C
C     PROGRAM DESCRIPTION: PROGRAMMER - WILLIAM DEININGER, 9 - 30 - 81
C     PLACE NOTES ON REVISIONS HERE: (INCLUDE DATE, INITIALS, LOCATION
C     AND CHANGE MADE  *** PLEASE ***)
C
C         THIS IS THE DRIVER ROUTINE FOR THE SIMULATION OF A CHARGE -
C     EXCHANGE PLASMA SURROUNDING A BROAD BEAM ION THRUSTER.  THIS ROUTINE
C     DEFINES THE PROGRAM STRUCTURE AND STAGING BY CALLING VARIOUS
C     SUBROUTINES.  READER:  THE FIRST CALL IS TO SUBROUTINE READER WHICH
C     READS IN THE RUN SPECIFICATIONS AND INFORMATION.  INIT:  THE NEXT
C     CALL IS TO SUBROUTINE INIT WHICH INITIALIZES VARIOUS PARAMETERS AND
C     CONSTANTS, DEFINES THE ION EXIT POINTS ALONG THE BEAM EDGE AND
C     PERFORMS THE FIRST ITERATION.  CALC:  THE NEXT CALL IS TO
C     SUBROUTINE CALC WHICH COMPUTES THE ION POSITIONS ALONG THE
C     TRAJECTORIES.  CALC IS CALLED NUMIT TIMES.  THIS COMPLETES THE
C     FIRST STAGE.  (NUMIT IS THE MAXIMUM NUMBER OF ITERATIONS TO BE
C     PERFORMED ON ANY GIVEN PATH DURING ANY PARTICULAR STAGE).
C     INFORMATION FOR THE FIRST (NUMIT - 10) ITERATIONS IS WRITTEN TO AN
C     EXTERNAL FILE, PATHS.  CORE MEMORY IS THEN REINITIALIZED BY
C     TRANSFERING THE RESULTS OF THE FINAL TEN ITERATIONS TO THE CORE SPACE
C     FOR THE INITIAL TEN.  THE RESULTS OF THESE TEN ITERATIONS ACT AS A
C     BASE FOR ANOTHER SET OF (NUMIT - 10) ITERATIONS.  CALC IS CALLED
C     ANOTHER (NUMIT - 10) TIMES TO FILL CORE STORAGE  WITH ANOTHER SET OF
C     PATH COORDINATES WHICH COMPLETES THE SECOND STAGE.  THEN CORE IS
C     REINITIALIZED AGAIN AND ANOTHER STAGE IS RUN.  THIS IS DONE UNTIL
C     THE DESIRED TOTAL NUMBER OF ITERATIONS IS PERFORMED, OR UNTIL
C     ALL PATHS INTERSECT BOUNDARIES.  FINALLY, THE PLOT INDICATOR,
C     'ICLPLT', IS CHECKED TO DETERMINE WHICH PLOTTING ROUTINE TO USE TO
C     OUTPUT A PLOT OF THE PATH COORDINATES, IF ANY.
C
C     *** NOTE ***
C               ALWAYS PUT TWO BLANK CARDS AT THE END OF THE DATA DECK TO
C               SIGNAL A STOP.  THE WORD SIZE DEPENDENT VARIABLES ARE:
C               IF1, IF2, INFO, ITITL AND IW.
C               SEE BLOCK DATA FOR INSTRUCTIONS ON WORD SIZE
C               DEPENDENT VARIABLES AND CODE GENERATED ERROR MESSAGES.
C
C     ********  VARIABLE DICTIONARY  ********
C
C     DN(K,M)    : DENSITY ON PATH K, AT ITERATION M.
C     ICLPLT     : USED TO DETERMINE WHICH (IF ANY) PLOTTING ROUTINE IS
C                  USED     = 1              , USE SUBROUTINE VRSPLT.
C                           = 2              , USE SUBROUTINE LNPLT.
C                           = 3              , USE BOTH VRSPLT AND LNPLT.
C                           = ANYTHING ELSE  , NO PLOTS.
C     INITDO     : INITIAL DO LOOP INDEX FOR 'DO 80 N' WHICH CALLS 'CALC'.
C     ISTAT(K)   : STATUS OF PATH K,
C                  = 0          , PATH IS ACTIVE.
C                  = 1 TO NMAX  , PATH IS NOT ACTIVE, VALUE IS ITERATION
C                                 NUMBER WHERE BOUNDARY WAS CROSSED.
C                  = AAAA       , PATH IS NOT ACTIVE, ERROR CONDITION.
C     KEY        : USED TO DETERMINE TYPE OF RUN,
C                  = 0 , FINISH PLOTS AND TERMINATE.
```

```
C                > 0 , HIGH RESOLUTION UPSTREAM PASS, RIGHT MOST
C                      BOUNDARY IS DEFINED USING "KEY'TH" PATH OF FILE
C                      IPATHS (USED TO DEFINE ZBOUND).
C                = -1, FIRST PASS, UNIFORM DISTRIBUTION.
C                < -1, REGULAR RUN, FIRST PASS, NON-UNIFORM DISTRIBUTION.
C   NSTAG      : ALLOWS DO LOOP INDEX 'NSTAGE' TO BE PASSED THROUGH
C                COMMON.
C   NSTAGE     : PROGRAM STAGE NUMBER.
C   NTOTST     : MAXIMUM NUMBER OF STAGES TO BE RUN.
C   NUMION     : NUMBER OF ION PATHS TO BE SIMULATED.
C   NUMIT      : MAXIMUM NUMBER OF ITERATIONS TO BE PERFORMED ON ANY ONE
C                PATH DURING ANY ONE STAGE.
C   XION(K,M)  : X POSITION OF ION ON PATH K AT ITERATION M.
C   ZION(K,M)  : Z POSITION OF ION ON PATH K AT ITERATION M.
C
C   ***  END OF PROGRAM DESCRIPTION  ***
C
C   PROGRAM DECLARATION STATEMENTS.
C   BLANK COMMON FOR LARGE ARRAYS, IO = INPUT-OUTPUT, PARAM = PARAMETERS.
C
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151),
     1    NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(20),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON / PARAM / N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELROH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4    DNOR,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
C
C   INPUT PARAMETERS, SPECIFICATIONS, INFORMATION.  INITIALIZE VARIABLES.
C   CHECK KEY TO SIGNAL STOP.
C
    1 CALL READER
      INITDO = 2
      NSTAG  = 1
C   ********************************************
C   PLOTTING ROUTINE VRSPLT PLOTS FROM CORE MEMORY
C      IF (KEY .EQ. 0)  CALL VRSPLT
C   ********************************************
      IF (KEY .EQ. 0)  CALL VRSPL
      IF (KEY .EQ. 0)  STOP 1
C
C   PERFORM INITIALIZATION.
C
      CALL INIT
C
C   BEGIN STAGING.  THE DO LOOP INDEX 'NSTAGE' GIVES THE STAGE NUMBER.
C
      DO 100 NSTAGE = 1, NTOTST
         NSTAG = NSTAGE
         IF (NSTAG .EQ. 1)  GO TO 70
         INITDO = 10
C
C   REINITIALIZATION PROCEDURE.  THE RESULTS FOR THE FINAL TEN ITERATIONS
C   ARE TRANSFERED TO THE CORE AREA FOR THE FIRST TEN ITERATIONS.  THESE
C   TEN ITERATIONS ARE USED AS A BASE FOR ANOTHER (NUMIT - 10) ITERATIONS.
C   THE TOTAL ITERATION PER PATH COUNTER IS REDUCED BY ONE TO ENSURE
```

```
C  THAT EACH STAGE DOES THE PROPER NUMBER OF ITERATIONS.
C
      DO 50 M = 1, 10
         DO 40 K = 1, NUMION
            IF (ISTAT(K) .NE. 0)  GO TO 40
            MM          = M + 141
            XION(K,M)   = XION(K,MM)
            ZION(K,M)   = ZION(K,MM)
            IF (M .EQ. 10)  GO TO 30
            DN(K,M)     = DN(K,MM)
            GO TO 40
   30          DN(K,M)    = 0.0
   40    CONTINUE
   50 CONTINUE
      DO 60 I = 1, NUMZON
         NIP(I) = NIP(I) - 1
   60 CONTINUE
C
C  ITERATE "NUMIT" TIMES TO COMPLETE THE FIRST STAGE.  ITERATE
C  "NUMIT - 10" TIMES TO COMPLETE THE SUCCESSIVE STAGES.  THIS
C  SECTION CALCULATES THE ION POSITIONS ALONG THE PATHS.
C
   70 CONTINUE
      DO 80 NN = INIT00, NUMIT
         N = NN
         CALL WRIT(3)
         CALL CALC
   80 CONTINUE
C
C  WRITE INFORMATION FOR FIRST (NUMIT - 10) ITERATIONS IN CORE
C  TO EXTERNAL FILE.  BEGIN NEW STAGE IF DESIRED.
C
C  ************************************
C  VRSPLT AND LNPLT PLOT FROM CORE MEMORY
C        CALL VRSPLT
C        IF (ICLPLT .GT. 1)  CALL LNPLT
C  ************************************
      CALL WRIT(5)
  100 CONTINUE
C
C  DETERMINE PLOTTING ROUTINE TO BE USED, IF ANY.  AFTER PLOTTING CHECK
C  FOR MORE INPUT (READ MORE DATA IF ANY).
C
      IF (ICLPLT .EQ. 1)  CALL VRSPL
      IF (ICLPLT .EQ. 3)  CALL VRSPL
      GO TO 1
      END
```

```
CARD NR. SEVERITY  DETAILS    DIAGNOSIS OF PROBLEM

       1   ANSI               THIS STATEMENT TYPE IS NON-ANSI.
```

```
      BLOCK DATA
C
C     **********  "COMMON" DATA ENTRY ROUTINE  **********
C
C     PROGRAM DESCRIPTION; PROGRAMMER - WILLIAM DEININGER.
C     REVISIONS: (INCLUDE DATE, INITIALS AND DESCRIBE CHANGE ** PLEASE **)
C
C     THIS PROGRAM LOADS DATA INTO LABLED COMMON STORAGE AT COMPILE TIME
C     THROUGH THE DATA STATEMENTS.
C
C     ********  VARIABLE DICTIONARY  ********
C
C     BK      : BOLTZMAN'S CONSTANT (J/K).
C     IF1     : A FORMAT FOR ALPHA-NUMERIC DATA I/O.
C     IF2     : A FORMAT FOR ALPHA-NUMERIC DATA I/O.
C     IN      : DEVICE CODE FOR INPUT UNIT, USED IN READ STATEMENTS.
C     IOUT    : DEVICE CODE FOR OUTPUT UNIT, USED IN WRITE STATEMENTS.
C     IPATHS  : DEVICE CODE FOR OUTPUT TO EXTERNAL FILES.
C     IW      : NUMBER OF WORDS REQUIRED TO GENERATE 80 CHARACTERS,
C               = 8  FOR COMPUTERS WITH A WORD SIZE OF 10
C               = 14 FOR COMPUTERS WITH A WORD SIZE OF 6
C               (USED IN "COMMON /IO/" AND ROUTINES "BLOCK DATA", "READER",
C               "WRIT", "VRSPLT" AND "VRSPL")
C     PI      : GEOMETRICALLY DEFINED CONSTANT, 3.14159265...
C     Q       : ELEMENTARY UNIT OF CHARGE (C).
C
C     ***  END OF PROGRAM DESCRIPTION AND DICTIONARY  ***
C
C     PROGRAM DECLARATION STATEMENTS.
C     BLANK COMMON FOR LARGE ARRAYS, IO = INPUT-OUTPUT, PARAM = PARAMETERS.
C
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151),
     1    NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON / PARAM / N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4    DNOB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
C
C     NOTE: WORD SIZE DEPENDENT VARIABLES;
C           IF1, IF2, IW, INFO, ITITL
C           IW IS THE NUMBER OF WORDS REQUIRED TO GENERATE 80 CHARACTERS.
C           TO CONVERT TO MACHINE USING DIFFERENT WORD SIZE, MODIFY ONLY
C           IW, IF1 AND IF2 IN FIRST TWO DATA STATEMENTS BELOW.
C     THIRD DATA STATEMENT FOR I/O BUFFERS, FOURTH DATA STATEMENT FOR
C     VALUES OF CONSTANTS.
C
      DATA IW, IF1(1), IF1(2) /8, 6H(8A10), 1H /
      DATA IF2(1), IF2(2), IF2(3), IF2(4) /10H(8A10/2(4A,4H10)),1H ,1H /
      DATA IN, IOUT, IPATHS /5, 6, 7/
      DATA BK, Q, PI /1.3806E-23, 1.602E-19, 3.141593/
C
C     INFORMATION ON PROGRAM GENERATED ERRORS MESSAGES;
C     ------------------------------------------------
C     WHEN ERROR CALLED, FIRST LINE OF ERROR OUTPUT WILL BE OF THE FORM,
C                  ******  ERROR NNN  ******
```

```
      C   WHERE 'NNN' IS ONE OF THE FOLLOWING INTEGERS,
      C        207      SEE SUBROUTINE WRIT
   60 C        410      SEE FUNCTION SUBROUTINE OS
      C        412      SEE FUNCTION SUBROUTINE OS
      C        521      SEE SUBROUTINE CALCD
      C        522      SEE SUBROUTINE CALCD
      C        523      SEE SUBROUTINE CALCD
   65 C        524      SEE SUBROUTINE CALCD
      C        525      SEE SUBROUTINE CALCD
      C        526      SEE SUBROUTINE CALCD
      C        527      SEE SUBROUTINE CALC
      C        528      SEE SUBROUTINE CALCD
   70 C        529      SEE SUBROUTINE CALCD
      C        530      SEE SUBROUTINE CALC
      C        610      SEE SUBROUTINE BOUND
      C        612      SEE SUBROUTINE BOUND
      C   AND THE REFERENCED SUBROUTINE IS WHERE THE ERROR IS CALLED FROM.
   75 C   'FATAL' ERRORS ARE GENERALLY FATAL TO PARTICULAR PATH ONLY.
            END
```

```
      SUBROUTINE READER
C
C  ************  INPUT (READ) ROUTINE  ************
C
C  PROGRAM DESCRIPTION: PROGRAMMER - WILLIAM DEININGER, 7 - 9 - 81.
C  REVISIONS: (INCLUDE DATE, INITIALS AND DESCRIBE CHANGE ** PLEASE **)
C
C     THIS SUBROUTINE READS IN THE RUN DESCRIPTION AND SPECIFICATIONS,
C  BOUNDARY SPECIFICATIONS, PROPELLANT AND PLASMA SPECIFICATIONS, GRAPH
C  LABLES AND READS FILE IPATHS (IF NECESSARY) TO SUPPLY THE NECESSARY
C  INFORMATION FOR HIGH RESOLUTION UPSTREAM RUNS.
C
C  ********  VARIABLE DICTIONARY  ********
C
C  BMCUR     : BEAM CURRENT (AMPS).
C  CEXSEC    : CHARGE EXCHANGE CROSS SECTION (METERS SQUARED).
C  ICLPLT    : USED TO DETERMINE WHICH (IF ANY) PLOTTING ROUTINE IS
C              USED:   = 1              , USE SUBROUTINE VRSPLT.
C                      = 2              , USE SUBROUTINE LNPLT.
C                      = 3              , USE BOTH VRSPLT AND LNPLT.
C                      = ANYTHING ELSE  , NO PLOTS
C  ICLWRT    : FREQUENCY WITH WHICH RESULTS OF CALC ARE OUTPUT.  WRITE
C              STATEMENT IN CALC CALLED AFTER EVERY "ICLWRT" NUMBER
C              OF ITERATIONS.
C  ICLERR    : USED TO DETERMINE IF CODE GENERATED ERROR MESSAGES, FOR
C              NON - FATAL ERRORS, ARE WRITTEN OUT.
C              = 0,    WRITE MESSAGES.
C              = 1,    DO NOT WRITE MESSAGES.
C  INFO(K)   : ARRAY STORING INFORMATION DESCRIBING RUN.
C  ISAT      : USED AS PATH STATUS HOLDER FOR READING IPATHS AND IN
C              DEFINING RIGHT BOUNDARY FOR HIGH RESOLUTION UPSTREAM RUN.
C  ISTAT(I)  : STATUS OF PATH I,
C              = 0          , PATH IS ACTIVE.
C              = 1 TO NMAX  , PATH IS NOT ACTIVE, VALUE IS ITERATION
C                             NUMBER WHERE BOUNDARY WAS CROSSED.
C              = 8888       , PATH IS NOT ACTIVE, ERROR CONDITION.
C  ITITL(K)  : ARRAY STORING GRAPH TITLE AND AXIS LABLES.
C  KEY       : USED TO DETERMINE TYPE OF RUN,
C              = 0 , FINISH PLOTS AND TERMINATE (2 BLANK CARDS WILL DO).
C              > 0 , HIGH RESOLUTION UPSTREAM PASS, RIGHT MOST BOUNDARY
C                    IS DEFINED USING KEY'TH PATH OF FILE IPATHS (USED
C                    TO DEFINE ZBOUND). (SEE NOTE BELOW).
C              = -1, FIRST PASS, UNIFORM DISTRIBUTION.
C              < -1, REGULAR RUN; FIRST PASS, NORMAL NON-UNIFORM
C                    DISTRIBUTION.
C  NIP(I)    : TOTAL NUMBER OF COMPLETED ITERATIONS ON PATH I.
C  NMAX      : NUMPRE DIVIDED BY FOUR (NUMPRE / 4) (SEE WRIT(4)).
C  NTOTST    : TOTAL NUMBER OF STAGES TO BE RUN.
C  NUM1      : NUMBER OF ION PATHS PLUS ONE (NUMION + 1).
C  NUMION    : NUMBER OF ION PATHS.
C  NUMIT     : MAXIMUM NUMBER OF ITERATIONS TO BE PREFORMED ON ANY
C              ONE PATH DURING ANY ONE STAGE.
C  NUMPRE    : NUMBER OF ION PATHS (NUMION) FROM RUN WHICH CREATED FILE
C              IPATHS (SEE WRIT(4)).
C  RB        : RADIUS OF THE BEAM (METERS).
C  RBOUND    : RADIAL BOUNDARY IN POSITIVE X DIRECTION (METERS).
C  RT        : RADIUS OF THE THRUSTER (METERS).
```

```
C   TELIN     : TEMPERATURE OF ELECTRONS IN THE ION BEAM (EV).
C   TELOUT    : TEMPERATURE OF ELECTRONS OUTSIDE THE ION BEAM (EV).
C   THRLEN    : THRUSTER LENGTH (METERS).
C   TIMEMU    : TIME MULTIPLIER, USED TO DEFINE THE TIME STEP IN TERMS
C               OF SOME MULTIBLE OF  RBOUND / (VELBOH * NUMIT).
C   TTHNEU    : TEMPERATURE OF THERMAL NEUTRALS IN CHAMBER (EV).
C   UMSION    : MASS OF IONS (PROPELLANT) (KILOGRAMS).
C   UTIL      : UTILIZATION FACTOR OF PROPELLANT (PART OF PROPELLANT
C               TURNED INTO IONS).
C   XVELMU    : X VELOCITY MULTIPLIER, USED TO DEFINE INITIAL VELOCITY
C               IN TERMS OF SOME MULTIBLE OF THE BOHM VELOCITY.
C   ZVELMU    : Z VELOCITY MULTIPLIER, USED TO DEFINE INITIAL VELOCITY
C               IN TERMS OF SOME MULTIBLE OF THE BOHM VELOCITY.
C
C   ***  END OF PROGRAM DESCRIPTION AND DICTIONARY  ***
C
C   PROGRAM DECLARATION STATEMENTS.
C   BLANK COMMON FOR LARGE ARRAYS, IO = INPUT-OUTPUT, PARAM = PARAMETERS.
C
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151),
     1    NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON / PARAM / N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4    DNOB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
C
C   READ IN 80 COLUMNS OF INFORMATION DESCRIBING RUN.  FIRST CHARACTER
C   SHOULD BE A BLANK FOR PRINTER LINE CONTROL.
C
      READ (IN, IF1) (INFO(K), K = 1, IW)
C
C   READ IN RUN SPECIFICATIONS AND PARAMETERS.
C
      READ (IN, 12) NUMION, NUMIT, KEY, ICLWRT, ICLPLT, NTOTST, ICLERR
   12 FORMAT (7I10)
      IF (KEY .EQ. 0) RETURN
      NUM1 = NUMION + 1
C
C   READ IN BOUNDARY SPECIFICATIONS.
C
      READ (IN, 13) RB, RBOUND, RT, THRLEN, BMCUR, UTIL
   13 FORMAT (6F10.5)
C
C   READ IN PROPELLANT AND PLASMA SPECIFICATIONS.  TEMPERATURES (FIRST
C   THREE VARIABLES LISTED) SHOULD BE INPUT IN ELECTRON-VOLTS.  THEY WILL
C   BE CONVERTED TO THE DESIRED UNITS FOR CALCULATION BY THE CODE.
C
      READ (IN, 14) TELIN, TELOUT, TTHNEU, CEXSEC, UMSION
   14 FORMAT (5E10.3)
      TELIN   = TELIN * Q
      TELOUT  = TELOUT * Q
      TTHNEU  = TTHNEU * Q
C
C   READ IN THE TIME MULTIPLIER AND THE X AND Z VELOCITY MULTIPLIERS
C   (SEE ABOVE DEFINITIONS).
```

```
      C
         READ (IN, 15) TIMEMU, XVELMU, ZVELMU
      15 FORMAT (3F10.3)
      C
      C  READ TWO CARDS FOR GRAPH, 80 COLUMNS FOR TITLE AND 40 COLUMNS EACH
      C  FOR AXIS LABLES.  (WORD SIZE DEPENDENT AREA)
      C
         IW2 = IW * 2
         READ (IN, IF?) (ITITL(K), K = 1, IW2)
      C
      C  TEST TO SEE IF THIS IS A HIGH RESOLUTION UPSTREAM RUN.  IF NOT,
      C  RETURN TO DRIVER, IF SO, READ FILE IPATHS AND SET UP BOUNDARY.
      C
         IF (KEY) 99, 99, 30
      C
      C  REWIND FILE IPATHS SO IT CAN BE READ FOR ANOTHER PASS.  THEN READ
      C  THE NUMBER OF ION PATHS IN IPATHS TO BE READ, THE TOTAL NUMBER OF
      C  ION PATHS MAKING UP THE FILE IPATHS (NUMPRE = NUMION FROM BEFORE),
      C  THE STATUS OF THE PATHS AND THE ARRAY OF PATH COORDINATES.
      C  ****************
      C  ***** NOTE *****
      C  ****************
      C        DUE TO THE DEFINITION OF NMAX (SEE WRIT(4)), KEY MUST BE LESS
      C        THAN OR EQUAL TO NMAX FOR A HIGH RESOLUTION UPSTREAM RUN.
      CWD      ASSIGNMENT OF DEVICE CODE SHOULD BE MODIFIED SO AS NOT TO
      CWD      INTERFER WITH STAGING AND WRIT(5). (SEE WRIT(4))
      C
         30    REWIND IPATHS
               READ (IPATHS) NMAX, NUMPRE
               READ (IPATHS) (ISTAT(I), I = 1, NMAX)
               DO 50 I = 1, NMAX
                 ISAT = ISTAT(I)
                 READ (IPATHS) (ZION(I,NN), XION(I,NN), NN = 1, ISAT)
         50    CONTINUE
      C
      C  DEFINITION OF THE RIGHT BOUNDARY (USED TO DEFINE ZBOUND).  PATH FOR
      C  RIGHT BOUNDARY IS DEFINED BY TRAJECTORY "KEY" OF FILE IPATHS ALONG
      C  WITH THE PATH STATUS AND ITERATIONS ON THE PATH.
      C
               ISAT = ISTAT(KEY)
               DO 70 NN = 1, ISAT
                 ZION(NUM1, NN) = ZION(KEY, NN)
                 XION(NUM1, NN) = XION(KEY, NN)
         70    CONTINUE
               ISTAT(NUM1)      = ISAT
               NIP(NUM1)        = ISAT
         99 CONTINUE
            RETURN
            END
```

```
      SUBROUTINE INIT
C
C  ************  INITIALIZATION ROUTINE  ************
C
C  PROGRAM DESCRIPTION: PROGRAMMER - WILLIAM DEININGER, 7 - 2 - 81
C  REVISIONS: (INCLUDE DATE, INITIALS AND DESCRIBE CHANGE ** PLEASE **)
C
C      THIS SUBROUTINE INITIALIZES THE NECESSARY VARIABLES, CALCULATES
C  THE COORDINATES OF THE ION TRAJECTORY EXIT POINTS FROM THE BEAM
C  EDGE AND PERFORMS THE FIRST ITERATION.  FIRST THE CONSTANTS ARE
C  DEFINED, THEN THE ION EXIT POINTS, THE INITIAL Z AND X POSITIONS
C  AND 'ZBOUND' ARE CALCULATED.  THEN THE TOTAL VELOCITY COMPONENTS
C  AND THE ITERATION NUMBER PER PATH ARE INITIALIZED.  THE NEXT ION
C  POSITIONS ARE CALCULATED AND ALL THE ION PATHS ARE SET TO ACTIVE.
C  THE INITIAL DENSITIES ARE CALCULATED AND FINALLY THE RESULTS OF THE
C  INITIALIZATION AND FIRST ITERATION ARE OUTPUT.
C
C  ********  VARIABLE DICTIONARY  ********
C
C  DNI(I)    : INITIAL DENSITY BETWEEN PATHS (I-1) AND (I).
C  DNOB      : CONSTANT USED IN CALCULATION OF THE DENSITIES, COMBINATION
C              OF QUANTITIES INCLUDING: BMCUR, CEXSEC, RB, NUMION,
C              UTIL, Q AND VELNEU.
C  IDEF      : USED TO DETERMINE WHEN THE X AND Z COORDINATES OF AN ION
C              TRAJECTORY EXIT POINT ARE DEFINED (EVERY OTHER TIME).
C  IL        : INDEX (I) OF LEFT MOST ACTIVE PATH.
C  IR        : INDEX (I) OF RIGHT MOST ACTIVE PATH.
C  ISAT      : ISTAT(I) FOR KEY'TH PATH IN UPSTREAM RUN.
C  ISTAT(I)  : STATUS OF PATH I,
C              = 0          , PATH IS ACTIVE.
C              = 1 TO NUMIT , PATH IS NOT ACTIVE, VALUE IS ITERATION
C                             NUMBER WHERE BOUNDARY WAS CROSSED.
C              = 8888       , PATH IS NOT ACTIVE, ERROR CONDITION.
C  NIP(I)    : TOTAL NUMBER OF ITERATIONS PERFORMED ON PATH I.
C  NUM1      : NUMBER OF ION PATHS PLUS ONE (NUMION + 1).
C  NUM2      : TWO TIMES THE NUMBER OF ION PATHS PLUS ONE (2*NUMION + 1).
C  PIOV2     : PI OVER (DIVIDED BY) 2.
C  RB2       : BEAM RADIUS SQUARED (RB ** 2).
C  RB95N     : 95 PERCENT OF THE BEAM RADIUS DIVIDED BY 2 TIMES THE
C              NUMBER OF ION PATHS (RB * .95 / (2 * NUMION)).
C  SPACER    : INITIAL DISTANCE BETWEEN PATHS IN UNIFORM DISTRIBUTION.
C  TIME      : TIME INTERVAL, DEFINES ITERATION STEP SIZE.
C  VELBET    : PRESENT TOTAL VELOCITY BETWEEN PATH UNDER CONSIDERATION
C              AND NEIGHBORING PATH.
C  VELBOH    : BOHM VELOCITY.
C  VELNEU    : THERMAL VELOCITY OF THE NEUTRALS IN THE CHAMBER.
C  VELTX(I)  : PRESENT TOTAL VELOCITY COMPONENT IN X DIRECTION.
C  VELTZ(I)  : PRESENT TOTAL VELOCITY COMPONENT IN Z DIRECTION.
C  XION(I,1): X COORDINATE OF ION TRAJECTORY EXIT POINT I.
C  XION(I,2): FIRST X COORDINATE OF ION AFTER LEAVING ION BEAM ALONG
C              TRAJECTORY I.
C  ZBOUND    : Z BOUNDARY TO RIGHT OF THRUSTER.
C  ZCURR     : PRESENT Z INCREMENT, USED TO GET ION EXIT POINTS.
C  ZION(I,1): Z COORDINATE OF ION TRAJECTORY EXIT POINT I.
C  ZION(I,2): FIRST Z COORDINATE OF ION AFTER LEAVING ION BEAM ALONG
C              TRAJECTORY I.
C  ZMAX      : MAXIMUM Z VALUE ON PATH "KEY" OF IPATHS, DEFINES ZBOUND
```

```
C                FOR HIGH RESOLUTION UPSTREAM RUN.
C   ZPREV     : PREVIOUS Z INCREMENT, USED TO GET ION EXIT POINTS.
C
C   ***  END OF PROGRAM DESCRIPTION AND DICTIONARY  ***
C
C   PROGRAM DECLARATION STATEMENTS.
C   BLANK COMMON FOR LARGE ARRAYS, IO = INPUT-OUTPUT, PARAM = PARAMETERS.
C
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151),
     1    NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(20),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON / PARAM / N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4    DNOB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
C
C   DEFINE CONSTANTS INCLUDING BOHM VELOCITY, THERMAL VELOCITY OF THE
C   NEUTRALS AND THE TIME INTERVAL.
C
      IL        = 1
      IR        = NUMION
      NUM1      = NUMION + 1
      NUM2      = 2 * NUMION + 1
      PIOV2     = PI / 2.0
      RB2       = RB ** 2
      RB95N     = RB * 0.95 / (2.0 * FLOAT(NUMION))
      VELBOH    = SQRT (TELIN / UMSION)
      VELNEU    = SQRT (TTHNEU / UMSION)
      ZPREV     = 0.0
      DNOB      = ((BMCUR ** 2) * CEXSEC * (1.0 - UTIL)) / (FLOAT(NUMION)
     6            * RB * UTIL * VELNEU * (Q ** 2) * (PI ** 2.0))
      TIME      = TIMEMU * RBOUND / (VELBOH * FLOAT(NUMIT * NTOTST))
C
C   CALCULATION OF THE X AND Z COORINATES OF THE ION TRAJECTORY EXIT
C   POINTS FROM THE BEAM.  GIVES A NON-UNIFORM DISTRIBUTION OF ION EXIT
C   POINTS.  (2 * NUMION + 1) POINTS ARE CALCULATED, THE EVEN POINTS
C   ARE USED AS ION EXIT POINTS.  THE LAST VALUE CALCULATED,
C   (2 * NUMION + 1)TH POINT, DEFINES ZBOUND.
C
      DO 100 II = 1, NUM2
         ZCURR          = 0.5 * (RB95N + ZPREV - SQRT (ZPREV ** 2 + RB2)) -
     7      0.5 * RB2 / (RB95N + ZPREV - SQRT(ZPREV ** 2 + RB2))
         IDEF           = MOD (II, 2)
         IF (IDEF .EQ. 0)  GO TO 95
            I           = (II + 1) / 2
            XION(I,1)= RB
            ZION(I,1)= THRLEN + ZCURR
   95    ZPREV          = ZCURR
  100 CONTINUE
      ZBOUND            = ZION(NUM1, 1)
C
C   IF THIS IS A HIGH RESOLUTION UPSTREAM RUN, ZBOUND MUST BE REDEFINED.
C   THE LARGEST ZION() VALUE ON THE KEY'TH PATH IS USED AS ZBOUND.
C
      IF (KEY)  120, 120, 105
  105    ZMAX = ZION (KEY, 1)
```

```
115           ISAT = ISTAT(I)
              DO 110 M = 2, ISAT
                IF (ZION(KEY, M) .LE. ZMAX)  GO TO 110
                  ZMAX = ZION(NUM1, M)
        110   CONTINUE
120           ZBOUND = ZMAX
      C
      C  IF KEY EQUALS -1, A UNIFORM DENSITY DISTRIBUTION RUN IS DONE.
      C  THE INTERVAL BETWEEN THRLEN AND ZBOUND IS BROKEN INTO NUMION+1
      C  EQUAL INTERVALS.  THE ION PATHS START AT THE END OF EACH INTERVAL.
125   C
        120 IF (KEY .NE. -1)  GO TO 125
              SPACER       = (ZBOUND - THRLEN) / FLOAT(NUM1)
              ZION(1,1)    = THRLEN + SPACER
              DO 122 I = 2, NUMION
130             ZION(I,1)  = ZION(I-1, 1) + SPACER
        122   CONTINUE
        125 CONTINUE
      C
      C  INITIALIZATION OF THE TOTAL VELOCITY COMPONENTS AND THE TOTAL NUMBER
135   C  OF ITERATIONS PER PATH COUNTER (NIP(I)).  CALCULATE NEXT ION
      C  POSITIONS AND SET ALL ION PATHS ACTIVE.
      C
              DO 130 I = 1, NUMION
                VELTX(I)   = VELBOH * XVELMU
140             VELTZ(I)   = VELBOH * ZVELMU
                NIP(I)     = 2
                XION(I,2)  = XION(I,1) + VELTX(I) * TIME
                ZION(I,2)  = ZION(I,1) + VELTZ(I) * TIME
                ISTAT(I)   = 0
145     130 CONTINUE
      C
      C  CALCULATION OF THE INITIAL DENSITIES.
      C  THE FOLLOWING GETS THE INITIAL DENSITIES BETWEEN THE PATHS.
      C
150           DO 160 I = 1, NUM1
                IF (I .EQ. 1) GO TO 152
                  IF (I .EQ. NUM1) GO TO 156
                    VELBET = (SQRT (VELTX(I-1) ** 2 + VELTZ(I-1) ** 2) +
          8         SQRT (VELTX(I) ** 2 + VELTZ(I) ** 2)) / 2.0
155                 DNI(I)= DNOR / (((XION(I, 1) + XION(I-1, 1)) / 2.0)
          9         * (ZION(I, 1) - ZION(I-1, 1)) * VELBET)
                    GO TO 160
        152       DNI(1)   = DNOR / (XION(1, 1) * (ZION(1, 1) - THRLEN)
            1     * (SQRT (VELTX(I) ** 2 + VELTZ(I) ** 2)))
160               GO TO 160
        156     DNI(NUM1)   = DNOR / (XION(I-1, 1) * (ZBOUND -ZION(I-1, 1))
              2 * (SQRT (VELTX(I-1) ** 2 + VELTZ(I-1) ** 2)))
        160 CONTINUE
      C
165   C  INITIALIZATION OF THE DENSITY ARRAY
      C
              DO 180 I = 2, NUM1
                DN(I-1, 1)  = (DNI(I) + DNI(I-1)) / 2.0
        180 CONTINUE
170   C
      C  RETURN TO DRIVER AFTER OUTPUTTING HEADING, SCHEMATIC OF THRUSTER,
```

```
      C  INITIAL PARAMETERS AND THE TWO SETS OF INFORMATION GENERATED BY
      C  THIS SUBROUTINE.
      C
  175       TELIN  = TELIN / 0
            TELOUT = TELOUT / 0
            TTHNEU = TTHNEU / 0
            CALL WRIT(1)
            TELIN  = TELIN * 0
  180       TELOUT = TELOUT * 0
            TTHNEU = TTHNEU * 0
            N  = 1
            CALL WRIT(2)
            N  = 2
  185       CALL WRIT(2)
            RETURN
            END
```

```
      SUBROUTINE CALC
C
C  ************  CALCULATION ROUTINE  ************
C
C  PROGRAM DESCRIPTION; PROGRAMMER - WILLIAM DEININGER, 5 - 26 - 81
C  REVISIONS: (INCLUDE DATE, INITIALS AND DESCRIBE CHANGE ** PLEASE **)
C
C     THIS SUBROUTINE USES THE ARRAYS IN BLANK COMMON
C  ALONG WITH SUBROUTINE CALCD (GETS DISPLACEMENTS TO RIGHT AND LEFT)
C  TO DETERMINE THE NEXT POSITION OF THE ION BEING CONSIDERED.  THE
C  DENSITIES AND POTENTIALS TO THE RIGHT AND LEFT OF THE CURRENT PATH
C  ARE CALCULATED FIRST.  THEN THE FORCE ACTING PERPENDICULAR TO THE
C  CURRENT PATH IS CALCULATED.  THE POTENTIALS AND FORCES ACTING
C  PARALLEL TO THE CURRENT PATH ARE OBTAINED NEXT.  THE TOTAL FORCE
C  (SUM OF PERPENDICULAR AND PARALLEL COMPONENTS) ACTING ON THE ION IS
C  CALCULATED AND THEN THE VELOCITY COMPONENTS ALONG THE X AND Z AXES
C  ARE OBTAINED.  FINALLY THE NEXT ION POSITION IS CALCULATED.  SUB-
C  ROUTINE BOUND IS CALLED TO MAKE SURE THE NEW ION POSITION IS INSIDE
C  THE BOUNDARIES.  THE RESULTS ARE PRINTED (EVERY "ICLWRT" TIMES) AND
C  FLAG1 IS CHECKED (SEE CALCD) TO SEE IF INTERSECTIONS WERE FOUND TO
C  BOTH THE RIGHT AND THE LEFT.  IF INTERSECTIONS WERE FOUND TO BOTH
C  SIDES, PATH I IS ITERATED AGAIN.
C
C  ********  VARIABLE DICTIONARY  ********
C
C     C      : TIME INTERVAL DIVIDED BY MASS OF ION (TIME / UMSION)
C     COSPAR : COSINE OF ANGLE BETWEEN LINE PARALLEL TO PATH AND HORIZONTAL.
C     COSPER : COSINE OF ANGLE BETWEEN LINE PERPENDICULAR TO PATH AND
C              HORIZONTAL.
C     DN(I,N): DENSITY ON PATH I AT ITERATION N.
C     DNL    : DENSITY TO LEFT SIDE OF CURRENT PATH.
C     DNOB   : CONSTANT USED IN THE DENSITY CALCULATIONS (C IN THE
C              REPORTS).
C     DNR    : DENSITY TO RIGHT SIDE OF CURRENT PATH.
C     DSPLIP : DISPLACEMENT OF ION ALONG PATH.
C     F      : TOTAL FORCE ACTING ON ION.
C     FPAR   : FORCE ACTING PARALLEL TO THE PATH.
C     FPARX  : COMPONENT OF FPAR ACTING IN X DIRECTION.
C     FPARZ  : COMPONENT OF FPAR ACTING IN Z DIRECTION.
C     FPER   : FORCE ACTING PERPENDICULAR TO THE PATH.
C     FPERX  : COMPONENT OF FPER ACTING IN X DIRECTION.
C     FPERZ  : COMPONENT OF FPER ACTING IN Z DIRECTION.
C     FX     : COMPONENT OF F ACTING IN X DIRECTION (FPERX + FPARX).
C     FZ     : COMPONENT OF F ACTING IN Z DIRECTION (FPERZ + FPARZ).
C     I      : ALLOWS DO LOOP INDEX, II, TO BE PASSED THROUGH COMMON.
C              PATH INDEX.
C     IFLAG4 : TRAJECTORY ONE RENORMALIZATION FLAG,
C              = 0,   CONTINUE ITERATING AS USUAL.
C              = 1,   RECALCULATE POSITION, VELOCITY AND DENSITY OF ION
C                     ON PATH 1.
C     IFVAR  : DUMMY VARIABLE USED AS ONE ARGUMENT IN AN IF STATEMENT.
C     J      : USED TO DETERMINE WHEN WRITE-435 IS EXECUTED.
C     N      : (WORKING) NUMBER OF ITERATIONS ON PRESENT PATH, I, TOTAL
C              NUMBER OF ITERATIONS FOR THE PRESENT STAGE.
C     NIP(I) : TOTAL NUMBER OF COMPLETED ITERATIONS ON PATH I.
C     NSTGMU : USED TO ADD 10 TO 'N' AFTER FIRST STAGE,
C              (N STAGE MULTIPLIER)
```

```
C            NSTAG = 1 ;     NSTGMU = 0
C            NSTAG > 1 ;     NSTGMU = 1
C  SINOV2 : SINPER OVER (DIVIDED BY) 2.
C  SINPAR : SINE OF ANGLE BETWEEN LINE PARALLEL TO PATH AND HORIZONTAL.
C  SINPER : SINE OF ANGLE BETWEEN LINE PERPENDICULAR TO PATH AND
C           HORIZONTAL.
C  VCURR  : PLASMA POTENTIAL AT CURRENT POINT ON PATH I.
C  VELBTL : PRESENT TOTAL VELOCITY BETWEEN PATH UNDER CONSIDERATION
C           AND PATH ON THE LEFT.
C  VELBTR : PRESENT TOTAL VELOCITY BETWEEN PATH UNDER CONSIDERATION
C           AND PATH ON THE RIGHT.
C  VELTOT : CURRENT TOTAL VELOCITY OF ION.
C  VELTT2 : CURRENT TOTAL VELOCITY OF ION ALONG PATH 2.
C  VELTX(I): CURRENT TOTAL VELOCITY COMPONENT IN X DIRECTION.
C  VELTZ(I): CURRENT TOTAL VELOCITY COMPONENT IN Z DIRECTION.
C  VELX   : VELOCITY CONTRIBUTION FOR THIS ITERATION ALONG X DIRECTION.
C  VELZ   : VELOCITY CONTRIBUTION FOR THIS ITERATION ALONG Z DIRECTION.
C  VL     : PLASMA POTENTIAL ON LEFT SIDE OF CURRENT PATH.
C  VPREV  : PLASMA POTENTIAL AT PREVIOUS POINT ON PATH I.
C  VR     : PLASMA POTENTIAL ON RIGHT SIDE OF CURRENT PATH.
C  XPOSL  : X-POSITION (COORDINATE) HALF WAY ALONG THE PERPENDICULAR
C           DISPLACEMENT (DSPLL) TO THE NEIGHBORING PATH ON THE LEFT.
C  XPOSR  : X-POSITION (COORDINATE) HALF WAY ALONG THE PERPENDICULAR
C           DISPLACEMENT (DSPLR) TO THE NEIGHBORING PATH ON THE RIGHT.
C  XION(I,N): CURRENT X POSITION OF ION.
C  XION(I,N+1): NEXT (NEW) X POSITION OF ION.
C  ZION(I,N): CURRENT Z POSITION OF ION.
C  ZION(I,N+1): NEXT (NEW) Z POSITION OF ION.
C
C  ***  END OF PROGRAM DESCRIPTION AND DICTIONARY  ***
C
C  PROGRAM DECLARATION STATEMENTS.
C  BLANK COMMON FOR LARGE ARRAYS, IO = INPUT-OUTPUT, PARAM = PARAMETERS.
C
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151),
     1    NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON / PARAM / N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4    DNDB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
C
C  DEFINE CONSTANTS, BEGIN ITERATION OF EACH PATH, TEST FOR PATH ONE
C  RENORMALIZATION AND SET THE CURRENT "WORKING" NUMBER OF ITERATIONS
C  ON PATH I.  MAKE SURE THE TOTAL NUMBER OF ITERATIONS PERFORMED ON
C  PATH I IS NOT TOO LARGE FOR THE CURRENT STAGE AND SEE IF CURRENT
C  PATH IS ACTIVE.
C       NOTE:  ANY QUANTITIES OPERATED ON BY "AINT" AND MULTIPLIED
C              OR DIVIDED BY 1 TIMES SOME POWER OF TEN,  ARE BEING
C              TRUNCATED TO AVOID COMPUTER ROUND-OFF ERROR.
C
      C      = AINT (TIME / (UMSION * 1.0E+15))
      C      = C * 1.0E+15
      NSTGMU = 0
      IF (NSTAG .GT. 1)  NSTGMU = 1
      IFLAG4 = 0
```

```
      DO 450 II = 1, NUMION
         I = II
         IF (I .EQ. 2)  IFLAG4 = 1
         IF (NIP(1) .GT. ((NSTAG * NUMIT) - ((NSTAG - 1) * 10)))
     1   IFLAG4 = 0
  300    N = NIP(I) - (NSTAG - 1) * (NUMIT - 10)
         IF (NIP(I) .GT. ((NSTAG * NUMIT) - ((NSTAG - 1) * 10)))
     2   GO TO 450
         IF (ISTAT(I) .NE. 0) GO TO 450
C
C    GET THE DISPLACEMENTS TO THE RIGHT AND LEFT, CHECK THE ERROR
C    FLAG, BOUNDARY INTERSECTION FLAG, MAKE SURE PARTICLE TRAJECTORY
C    PATHS ARE SMOOTH AND CALCULATE THE NEEDED TRIGONOMETRIC FUNCTIONS
C    OF THETAP.
C
         CALL CALCD (I, IFLAG1, IFLAG2, IFLAG3, DSPLR, DSPLL, THETAP)
         IF (IFLAG2 .GT. 0) GO TO 499
         IF (IFLAG3 .LT. 0 .OR. IFLAG3 .GE. 5)  GO TO 497
         IF (IFLAG3 .EQ. 4)  GO TO 460
         COSPAR  = AINT (COS (PIOV2 - THETAP) * 1.0E+03)
         COSPER  = AINT (COS (THETAP) * 1.0E+03)
         SINPAR  = AINT (SIN (PIOV2 - THETAP) * 1.0E+03)
         SINPER  = AINT (SIN (THETAP) * 1.0E+03)
         COSPAR  = COSPAR / 1.0E+03
         COSPER  = COSPER / 1.0E+03
         SINPAR  = SINPAR / 1.0E+03
         SINPER  = SINPER / 1.0E+03
C
C    CALCULATION OF THE LEFT AND RIGHT X-POSITIONS HALF WAY ALONG THE
C    PERPENDICULAR DISPLACEMENTS TO THE NEIGHBORING PATHS
C    (X-COORDINATES AT CENTER OF DENSITY CELLS).  "IF STATEMENT"
C    DETERMINES THE DIRECTION OF PATH PROPAGATION.
C
         SINOV2 = SINPER / 2.0
         IF (ZION(I,N) - ZION(I,N-1))  314, 310, 318
C
  310      XPOSL = XION(I,N)
           XPOSR = XION(I,N)
           GO TO 320
C
  314      XPOSL = XION(I,N) - DSPLL * SINOV2
           XPOSR = XION(I,N) + DSPLR * SINOV2
           GO TO 320
C
  318      XPOSL = XION(I,N) + DSPLL * SINOV2
           XPOSR = XION(I,N) - DSPLR * SINOV2
C
C    CALCULATION OF THE TOTAL VELOCITIES BETWEEN THE PRESENT PATH
C    AND THE PATHS TO THE RIGHT AND LEFT.
C
  320      IF (I .EQ. 1)  GO TO 322
              VELTL = (SQRT (VELTX(I-1) ** 2 + VELTZ(I-1) ** 2)
     3        + SQRT (VELTX(I) ** 2 + VELTZ(I) ** 2)) / 2.0
              IF (I .EQ. NUMION) GO TO 323
  321      VELTR = (SQRT (VELTX(I) ** 2 + VELTZ(I) ** 2) +
     4     SQRT(VELTX(I+1) ** 2 + VELTZ(I+1) ** 2)) / 2.0
           GO TO 325
```

```
      322          VELBTL = SQRT (VELTX(I) ** 2 + VELTZ(I) ** 2)
                   GO TO 321
      323          VELBTR = SQRT (VELTX(I) ** 2 + VELTZ(I) ** 2)
175   C
      C    CALCULATION OF THE DENSITIES TO THE RIGHT AND LEFT, AND THE
      C    AVERAGE DENSITY AT THE CURRENT POINT.  MAKE SURE DNL AND DNR
      C    ARE NOT ZERO.
      C
180     325     DNL      = AINT (DNOB / (XPOSL * DSPLL * VELBTL))
              DNR      = AINT (DNOB / (XPOSR * DSPLR * VELBTR))
              DN(I,N) = (DNL + DNR) / 2.0
              IF (DNL .LE. 0.0 .OR. DNR .LE. 0.0)  GO TO 495
      C
185   C  WHEN A BOUNDARY IS INTERSECTED ON THE LEFT OR RIGHT, THE DISPLACE -
      C  MENTS HAVE TO BE CHECKED TO MAKE SURE NO BOUNDARY REPULSION EXISTS.
      C  IF THE DISPLACEMENT FROM THE CURRENT PATH TO THE BOUNDARY IS LESS
      C  THEN THE DISPLACEMENT BETWEEN THE CURRENT PATH AND THE NEIGHBORING
      C  PATH, THE PERPENDICULAR FORCE IS ZEROED, OTHERWISE THE PATH
190   C  PROPAGATES AS USUAL.
      C
              IF (IFLAG3 .EQ. 1)  GO TO 340
              IF (VELT7(I) .EQ. 0.0)  GO TO 340
                IF (IFLAG3 .EQ. 2)  GO TO 328
195             IF (IFLAG3 .EQ. 3)  GO TO 335
                  GO TO 497
      328       IF (DSPLL .LE. DSPLR)  GO TO 330
                  GO TO 340
      330       FPERX = 0.0
200             FPERZ = 0.0
                GO TO 370
      335       IF (DSPLR .LE. DSPLL)  GO TO 330
      C
      C    CALCULATION OF THE POTENTIALS TO THE RIGHT AND LEFT.  THE FORCE,
205   C    AND ITS COMPONENTS, ACTING PERPENDICULAR TO THE PATH IS THEN
      C    CALCULATED USING THE SMALLER OF THE TWO PERPENDICULAR DIS -
      C    PLACEMENTS AS THE DIVISOR.
      C
      340     VL       = AINT ((ALOG (DNL / DNI(I)) * TELOUT / Q ) *
210     5             1.0E+04)
              VR       = AINT ((ALOG (DNR / DNI(I+1)) * TELOUT / Q) *
        6             1.0E+04)
              VL       = VL / 1.0E+04
              VR       = VR / 1.0E+04
215           IF (DSPLL .LE. DSPLR)  GO TO 345
                FPER   = AINT ((Q * (VL - VR) / DSPLR) * 1.0E+20)
                GO TO 350
      345     FPER     = AINT ((Q * (VL - VR) / DSPLL) * 1.0E+20)
      350     FPER     = FPER / 1.0E+20
220           FPERX    = FPER * SINPER
              FPERZ    = FPER * COSPER
      C
      C    CALCULATION OF THE POTENTIALS AND THE FORCE AND ITS COMPONENTS
      C    ACTING PARALLEL TO THE PATH.
225   C
      370     VCURR    = ALOG (DN(I,N) / DN(I,1)) * TELOUT / Q
              VPREV    = ALOG (DN(I,N-1) / DN(I,1)) * TELOUT / Q
              DSPLIP   = SQRT ((XION(I,N) - XION(I,N-1)) ** 2 + (ZION(I,N)
```

C-2

```
     7            - ZION(I,N-1)) ** 2)
230          FPAR    = AINT ((0 * (VPREV - VCURR) / DSPLIP) * 1.0E+20)
             FPAR    = FPAR / 1.0E+20
             FPARX   = FPAR * SINPAR
             FPARZ   = FPAR * COSPAR
C
235  C    CALCULATION OF TOTAL FORCE AND COMPONENTS.
C
             FX      = FPARX + FPERX
             FZ      = FPARZ + FPERZ
             F       = SQRT (FX ** 2 + FZ ** 2)
240  C
C    CALCULATION OF VELOCITY COMPONENTS FOR THIS ITERATION, TOTAL
C    VELOCITY COMPONENTS AND TOTAL VELOCITY FOR PATH I.
C
             VELX    = FX * C
245          VELZ    = FZ * C
       390   VELTX(I)= VELTX(I) + VELX
             VELTZ(I)= VELTZ(I) + VELZ
             VELTOT  = SQRT (VELTX(I) ** 2 + VELTZ(I) ** 2)
C
250  C    THIS SECTION MODIFIES THE VELOCITY ON THE FIRST PATH SO THAT ITS
C    NORMALIZED WITH RESPECT TO 1.2 TIMES THE VELOCITY ON THE SECOND
C    PATH IN TERMS OF MAGNITUDE.  THE DIRECTION OF THE TRAJECTORY IS
C    LEFT UNCHANGED.  THIS NORMALIZATION ONLY OCCURS WHEN THE VELOCITY
C    ON PATH 1 IS MORE THAN 20 PRECENT GREATER THEN THE VELOCITY ON
255  C    PATH 2.  THE VALUE OF 20 PERCENT IS ARBITARY AND IS BASED ON
C    WHAT GIVES THE SMOOTHEST TRAJECTORIES.  THIS PREVENTS PATH ONE
C    FROM ACCELERATING TO FAST.
C
             IF (I .EQ. 1 .AND. IFLAG4 .EQ. 1)  GO TO 398
260             GO TO 410
       398   VELTT2 = SQRT (VELTX(I+1) ** 2 + VELTZ(I+1) ** 2) * 1.2
             IF (VELTOT - VELTT2)  410, 410, 400
       400      VELTX(I) = (VELTX(I) * VELTT2) / VELTOT
                VELTZ(I) = (VELTZ(I) * VELTT2) / VELTOT
265             VELTOT = SQRT (VELTX(I) ** 2 + VELTZ(I) ** 2)
C
C    CALCULATION OF THE NEXT ION POSITION (USES LINEAR APPROXIMATION),
C    MAKE SURE ITS INSIDE THE BOUNDARIES.
C
270    410   XION(I,N+1) = VELTX(I) * TIME + XION(I,N)
             ZION(I,N+1) = VELTZ(I) * TIME + ZION(I,N)
             CALL BOUND (ZION(I,N+1), XION(I,N+1), I)
C
C    WRITE THE RESULTS EVERY "ICLWRT" TIMES.  IF ISTAT(I) IS NON-ZERO,
275  C    WRITE THE RESULTS.  INCREASE NIP(I) BY ONE FOR NEXT PASS.
C    IF PATH I REACHED A BOUNDARY ON THIS ITERATION, SET NIP(I) TO ITS
C    FINAL VALUE.  IF TO MANY ITERATIONS HAVE OCCURED SET ISTAT(I) = N.
C    TEST TO SEE IF PATH I NEEDS TO BE ITERATED AGAIN (TEST IFLAG1) AND
C    CHECK TO SEE IF ITERATION LIMIT HAS BEEN REACHED.  CHECK IFLAG4
280  C    FOR RENORMALIZATION OF PATH 1.
C
       420   IF (ICLWRT .LE. 0)  GO TO 440
             J = MOD (NIP(I), ICLWRT)
             IF (ISTAT(I) .NE. 0)  GO TO 425
285             IF (J .NE. 0) GO TO 440
```

```
      425         WRITE (IOUT, 435) I, N, NIP(I), ISTAT(I), ZION(I,N+1),
     8            XION(I,N+1), VELTZ(I), VELTX(I), VELTOT, DN(I,N)
     435          FORMAT (1X, I3, 1X, 3(I4, 2X), 6(E13.6, 2X))
      440     NIP(I)  = NIP(I) + 1
290           IF (ISTAT(I) .NE. 0)  NIP(I) = NIP(I) - 2
              IFVAR = (NSTAG * NUMIT + 1) - ((NSTAG - 1) * 9)
              IF (NIP(I) .GT. IFVAR)  ISTAT(I) = N
              IF (I .EQ. 1 .AND. IFLAG4 .EQ.1)  GO TO 445
              IF (IFLAG1 .EQ. 0) GO TO 300
295           IF (IFLAG4 .EQ. 0)  GO TO 450
                I = 1
                IF (ISTAT(I) .NE. 0)  GO TO 445
                NIP(I) = NIP(I) - 1
                GO TO 300
300     445     IFLAG4 = 0
      450 CONTINUE
          NSTGMU = 1
          RETURN
      C
305   C  IF A BOUNDARY IS INTERSECTED ON BOTH SIDES OF THE PATH (TO THE RIGHT
      C  AND LEFT) ALL FORCES ARE SET EQUAL TO ZERO CAUSING THE X AND Z
      C  VELOCITY CONTRIBUTIONS FOR THIS ITERATION TO BE ZERO.  THE PATH
      C  PROPAGATES LINEARLY.  THE DENSITY IS TREATED AS A CONSTANT.
      C
310     460 VELX = 0.0
          VELZ = 0.0
          DN(I,N) = DN(I,N-1)
          GO TO 390
      C
315   C  *******  ERROR EXITS AND ERROR CONDITIONS  *******
      C
      C    - ERROR 527 - IFLAG3 IMPROPERLY DEFINED, FATAL.
      C
      C    - ERROR 530 - DNL OR DNR EQUAL ZERO OR LESS THEN ZERO.  CAUSES
320   C                  VL OR VR TO BLOW UP, FATAL.
      C
      495 IFLAG2 = 325
          WRITE (IOUT, 530)  IFLAG1, IFLAG2, IFLAG3, I, N, DNL, DNR,
         9                   DSPLL, DSPLR, XPOSL, XPOSR
325       GO TO 499
      497 WRITE (IOUT, 527)  IFLAG1, IFLAG2, IFLAG3, I, N
      499 ISTAT(I) = 8888
          GO TO 420
      527 FORMAT (/,11X,23H****** ERROR 527 ******,/,/,11X,
330      1        33HIFLAG3 IMPROPERLY DEFINED (FATAL),/,11X,
         2        27HCALLED FROM SUBROUTINE CALC,/,11X,8HIFLAG1 =,I5,
         3        10H  IFLAG2 =,I5,10H  IFLAG3 =,I5,5H  I =,I5,5H  N =,I5)
      530 FORMAT (/,11X,23H****** ERROR 530 ******,/,/,11X,
         1        45HDNL OR DNR LESS THEN OR EQUAL TO ZERO (FATAL),/,11X,
335      2        27HCALLED FROM SUBROUTINE CALC,/,11X,
         3        8HIFLAG1 =,I5,10H  IFLAG2 =,I5,10H  IFLAG3 =,I5,5H  I =,
         4        I5,5H  N =,I5,7H  DNL =,E10.3,7H  DNR =,E10.3,/,11X,
         5        9H  DSPLL =,F10.3,9H  DSPLR =,E10.3,9H  XPOSL =,E10.3,
         6        9H  XPOSR =,E10.3)
340       END
```

```
      SUBROUTINE CALCD (I, IFLAG1, IFLAG2, IFLAG3, DSPLR, DSPLL, THETAP)
C
C ******* CALCULATION OF THE DISPLACEMENTS TO THE RIGHT AND LEFT *******
C
C  PROGRAM DESCRIPTION : PROGRAMER - WILLIAM DEININGER, 5 - 19 - 81
C  REVISIONS : (INCLUDE DATE, INITIALS AND DESCRIBE CHANGE **PLEASE**)
C
C     THIS SUBROUTINE USES THE ARRAYS ZION(), XION() AND NIP(I) TO GET
C  THE DISPLACEMENTS FROM THE CURRENT PATH TO THE LEFT AND RIGHT HAND
C  PATHS.  THE DISPLACEMENT TO THE LEFT IS DENOTED "DSPLL" AND TO THE
C  RIGHT "DSPLR".  FIRST THE PERPENDICULAR IS OBTAINED (SLOPEP), THEN
C  THE LINE FORMED BY THE FINAL TWO POINTS OF THE PATH ON THE LEFT
C  IS OBTAINED (SLOPEL), AND FINALLY THE X AND Z INTERSECTIONS, TO
C  THE LEFT, ARE CALCULATED (XINTL, ZINTL).  TESTS ARE RUN TO MAKE SURE
C  THE INTERSECTION POINTS ARE "GOOD" AND TO DETERMINE IF LINEAR
C  EXTRAPOLATION WAS USED.  THE DISPLACEMENT TO THE LEFT HAND PATH
C  (DSPLL) IS THEN CALCULATED.  LIKEWISE FOR THE RIGHT HAND
C  DISPLACEMENT (DSPLR).  FUNCTION (SUBROUTINE) DS IS CALLED TO GET
C  THE DISPLACEMENTS FOR THE SPECIAL CASES, EG:  BOUNDARIES ON THE
C  RIGHT OR LEFT.
C
C  ---IFLAG1 IS USED TO DETERMINE IF INTERSECTIONS WERE FOUND TO BOTH
C  THE LEFT AND THE RIGHT WITHOUT USING LINEAR EXTRAPOLATION,
C        IFLAG1 = 0        LINEAR EXTRAPOLATION NOT USED.
C        IFLAG1 = 1        LINEAR EXTRAPOLATION USED ON LEFT.
C        IFLAG1 = 2        LINEAR EXTRAPOLATION USED ON RIGHT.
C        IFLAG1 = 3        LINEAR EXTRAPOLATION USED IN BOTH CASES.
C  IF LINEAR EXTRAPOLATION IS NOT USED, THE CURRENT PATH NEEDS TO BE
C  ITERATED AGAIN DUE TO ITS SMALLER ACCELERATION.
C
C  ---IFLAG2 IS THE ERROR FLAG AND TELLS WHETHER OR NOT THE NEIGHBORING
C  PATHS ARE ACTIVE.
C        IFLAG2 = 0        NO ERROR.
C        IFLAG2 > 0        ERROR EXISTS, VALUE REFERENCES PROGRAM
C                          STATEMENT WHERE ERROR CONDITION ORGINATED.
C        IFLAG2 = -1       PATH TO LEFT IS NOT ACTIVE.
C        IFLAG2 = -2       PATH TO RIGHT IS NOT ACTIVE.
C        IFLAG2 = -3       NEITHER PATH (RIGHT OR LEFT) IS ACTIVE.
C
C  ---IFLAG3 IS USED TO DETERMINE WHEN THERE IS A BOUNDARY ON THE RIGHT
C  OR LEFT (OR BOTH) OF THE CURRENT PATH.
C        IFLAG3 = 1        NO BOUNDARY INTERSECTED BY THE PERPENDICULAR TO
C                          THE CURRENT PATH ON EITHER SIDE.
C        IFLAG3 = 2        BOUNDARY INTERSECTED ON LEFT.
C        IFLAG3 = 3        BOUNDARY INTERSECTED ON RIGHT.
C        IFLAG3 = 4        BOUNDARY INTERSECTED ON BOTH THE LEFT AND RIGHT
C
C
C  THIS SUBROUTINE RETURNS : IFLAG1, IFLAG2, IFLAG3, DSPLL, DSPLR,
C  AND THETAP.
C
C     *****  VARIABLE DICTIONARY  *****
C
C  DELTAX: DIFFERENCE BETWEEN TWO X COORDINATES ON CURRENT PATH.
C  DELTAZ: DIFFERENCE BETWEEN TWO Z COORDINATES ON CURRENT PATH.
C  DELTLX: DIFFERENCE BETWEEN TWO X COORDINATES ON LEFT PATH.
C  DELTLZ: DIFFERENCE BETWEEN TWO Z COORDINATES ON LEFT PATH.
```

```
C  DELTRX: DIFFERENCE BETWEEN TWO X COORDINATES ON RIGHT PATH.
C  DELTRZ: DIFFERENCE BETWEEN TWO Z COORDINATES ON RIGHT PATH.
C  DSPLL : SEE ABOVE COMMENTS.
C  DSPLR : SEE ABOVE COMMENTS.
C  DUMMYP, DUMMYL, DUMMYR : DUMMY VARIABLES USED IN THE CALCULATION
C          OF THE INTERCEPTS.  CONTAIN INTERMEDIATE RESULTS.
C  IFLAG1: SEE ABOVE COMMENTS.
C  IFLAG2: SEE ABOVE COMMENTS.
C  IFLAG3: SEE ABOVE COMMENTS.
C  IFVAR : DUMMY VARIABLE USED AS AN ARGUMENT IN AN IF STATEMENT.
C  NIP(I): TOTAL NUMBER OF COMPLETED ITERATIONS ON PATH I.
C  N     : (WORKING) NUMBER OF ITERATIONS ON PRESENT PATH, TOTAL
C          NUMBER OF ITERATIONS FOR PRESENT STAGE.
C  NL    : (WORKING) NUMBER OF ITERATIONS ON LEFT HAND PATH.
C  NLM1  : NL MINUS 1 (NL - 1), USED FOR INDEXING LEFT HAND PATH.
C  NR    : (WORKING) NUMBER OF ITERATIONS ON RIGHT HAND PATH.
C  NRM1  : NR MINUS 1 (NR - 1), USED FOR INDEXING RIGHT HAND PATH.
C  SLOPEL: SLOPE OF PATH ON LEFT BETWEEN TWO "WORKING" POINTS.
C  SLOPEP: SLOPE OF LINE PERPENDICULAR TO CURRENT PATH AT ENDPOINT.
C  SLOPER: SLOPE OF PATH ON RIGHT BETWEEN TWO "WORKING" POINTS.
C  THETAP: ANGLE BETWEEN LINE WITH SLOPE "SLOPEP" AND HORIZONTAL.
C  XINTL : X INTERSECTION, TO LEFT, OF LINES "SLOPEP" AND "SLOPEL".
C  XINTR : X INTERSECTION, TO RIGHT, OF LINES "SLOPEP" AND "SLOPER".
C  ZINTL : Z INTERSECTION, TO LEFT, OF LINES "SLOPEP" AND "SLOPEL".
C  ZINTR : Z INTERSECTION, TO RIGHT, OF LINES "SLOPEP" AND "SLOPER".
C
C  ***  END OF PROGRAM DESCRIPTION AND DICTIONARY  ***
C
C  PROGRAM DECLARATION STATEMENTS.
C  BLANK COMMON FOR LARGE ARRAYS, IO = INPUT-OUTPUT, PARAM = PARAMETERS.
C
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151)
     1,NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,IYITL(28),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON / PARAM / N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4    DNOB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
C
C  INITIALIZE NECESSARY VARIABLES (FLAGS).
C
      IFLAG1  = 0
      IFLAG2  = 0
      IFLAG3  = 0
C
C  CALCULATE SLOPE OF LINE PERPENDICULAR TO THE CURRENT PATH AT END-
C  POINT OF CURRENT PATH (NEGATIVE RECIPROCAL OF SLOPE BETWEEN LAST
C  TWO POINTS ON CURRENT PATH) AND ANGLE THIS LINE MAKES WITH HORIZONTAL
C
      DELTAZ = ZION(I,N-1) - ZION(I,N)
      DELTAX = XION(I,N) - XION(I,N-1)
      IF (DELTAX .EQ. 0.0) DELTAX = 1.0E-16
      SLOPEP = DELTAZ / DELTAX
      THETAP = ATAN(SLOPEP)
      DUMMYP = XION(I,N) - SLOPEP * ZION(I,N)
C
```

```
115  C   TWO SUBSECTIONS FOLLOW;  THE FIRST SUBSECTION OBTAINS THE INTER-
     C   SECTION POINT AND DISPLACEMENT TO THE LEFT,  THE SECOND SUBSECTION
     C   OBTAINS THE INTERSECTION POINT AND DISPLACEMENT TO THE RIGHT.
     C
     C   CALCULATIONS FOR LEFT.
120  C   ~~~~~~~~~~~~~~~~~~~~~~
     C
     C     CHECK TO SEE IF THIS IS THE FIRST ACTIVE PATH AND INITIALIZE
     C     THE COUNTER FOR THE LEFT HAND PATH.  THE FIRST ACTIVE PATH MUST
     C     BE HANDLED AS A SPECIAL CASE (IL = 1).  CHECK TO SEE IF LEFT HAND
125  C     PATH IS ACTIVE.
     C
           IF (I - IL) 492, 460, 307
       307 NL   = NIP(I-1) - (NSTAG - 1) * (NUMIT - 9)
           NLM1 = NL - 1
130        IF (ISTAT(I-1)) 470, 311, 470
     C
     C     CALCULATE SLOPE OF PATH ON LEFT BETWEEN FINAL TWO (OR TWO
     C     "WORKING") POINTS.
     C
135    311 DELTLZ = ZION(I-1,NL) - ZION(I-1,NLM1)
           DELTLX = XION(I-1,NL) - XION(I-1,NLM1)
           IF (DELTLZ .EQ. 0.0) DELTLZ = 1.0E-16
           SLOPEL = DELTLX / DELTLZ
     C
140  C     MAKE SURE INTERSECTION CAN BE FOUND, THE SLOPES OF THE TWO LINES
     C     CAN NOT BE THE SAME.  STATEMENTS 312 AND THOSE RIGHT BELOW,
     C     BACKSTEP THE LEFT HAND PATH ONE ITERATION, ALLOWING A NEW SLOPE
     C     TO BE CALCULATED.  IF BACKSTEPPING IS NOT NEEDED, THE
     C     INTERSECTION POINTS ARE CALCULATED.
145  C
           IF (SLOPEP - SLOPEL) 313, 312, 313
       312   IF (IFLAG2 .EQ. 312)  GO TO 481
             IF (IFLAG2 .EQ. 311)  GO TO 481
             NL   = NL - 1
150          IF (NL .EQ. 1) GO TO 480
             NLM1 = NL - 1
             GO TO 311
       313 DUMMYL = XION(I-1,NL) - SLOPEL * ZION(I-1,NL)
           ZINTL  = (DUMMYL - DUMMYP) / (SLOPEP - SLOPEL)
155        XINTL  = SLOPEP * ZINTL + DUMMYP
     C
     C     TEST TO SEE IF INTERSECTION POINTS ARE GOOD.  FIRST FIND DIRECTION
     C     OF PATH PROPGATION, 320 IMPLIES NEGITIVE Z DIRECTION, 325 IMPLIES
     C     POSITIVE Z DIRECTION, 330 IMPLIES VERTICAL (UP OR DOWN) DIRECTION.
160  C     THEN TESTS ARE RUN TO SEE IF THE INTERSECTIONS ARE "GOOD", IF
     C     LINEAR EXTRAPOLATION IS USED (SETS IFLAG1), OR IF BACK STEPPING
     C     IS NEEDED.  THESE TESTS ARE RUN IN ALL CASES.
     C
           IF (ZION(I-1,NL) - ZION(I-1,NLM1))  320, 330, 325
165  C
       320   IF (ZION(I-1,NL) .LE. ZINTL .AND. ZINTL .LE.
          6  ZION(I-1,NLM1)) GO TO 355
             IF (ZINTL .LT. ZION(I-1,NL)) GO TO 353
       322   IF (ZINTL .GT. ZION(I-1,NLM1)) GO TO 312
170          IFLAG2 = 322
             GO TO 486
```

```
      C
        325       IF (ZION(I-1,NLM1) .LE. ZINTL .AND. ZINTL .LE.
           7      ZION(I-1,NL)) GO TO 355
175               IF (ZINTL .GT. ZION(I-1,NL)) GO TO 353
        327       IF (ZINTL .LT. ZION(I-1,NLM1)) GO TO 312
                  IFLAG2 = 327
                  GO TO 486
      C
180   C    WHEN STATEMENT 330 IS CALLED WE HAVE TO TEST THE X COMPONENTS
      C    TO FIND THE X DIRECTION OF PROPAGATION, THEN SEE IF THE
      C    INTERSECTIONS ARE "GOOD", IF LINEAR EXTRAPOLATION IS USED,
      C    OR IF BACK STEPPING IS NEEDED.
      C
185     330       IF (XION(I-1,NL) - XION(I-1,NLM1)) 340, 484, 345
      C
        340         IF (XION(I-1,NL) .LE. XINTL .AND. XINTL .LE.
           8        XION(I-1,NLM1)) GO TO 355
                    IF (XINTL .LT. XION(I-1,NL)) GO TO 353
190     342         IF (XINTL .GT. XION(I-1,NLM1)) GO TO 312
                    IFLAG2 = 342
                    GO TO 486
      C
        345         IF (XION(I-1,NLM1) .LE. XINTL .AND. XINTL .LE.
195        9        XION(I-1,NL)) GO TO 355
                    IF (XINTL .GT. XION(I-1,NL)) GO TO 353
        347         IF (XINTL .LT. XION(I-1,NLM1)) GO TO 312
                    IFLAG2 = 347
                    GO TO 486
200   C
      C    STATEMENT 350 RESETS IFLAG2 IF STATEMENT 484 IS REFERENCED.
      C    STATEMENT 353 SETS IFLAG1, STATEMENT 355 CALCULATES THE
      C    DISPLACEMENT TO THE LEFT.  MAKE SURE IFLAG2 IS SET PROPERLY.
      C
205     350           IFLAG2 = 0
                      GO TO 355
        353           IFLAG1 = IFLAG1 + 1
        355           DSPLL = SQRT ((XINTL - XION(I,N)) ** 2 +
           1                  (ZINTL - ZION(I,N)) ** 2)
210                   IF (IFLAG2 .GT. 0)  IFLAG2 = IFLAG2 - 312
      C
      C  CALCULATIONS FOR RIGHT.
      C  ======================
      C
215   C    CHECK TO SEE IF THIS IS THE LAST ACTIVE PATH AND INITIALIZE THE
      C    COUNTER FOR THE RIGHT HAND PATH.  THE LAST ACTIVE PATH MUST
      C    BE HANDLED AS A SPECIAL CASE (IR = NUMION).  CHECK TO SEE IF THE
      C    LEFT HAND PATH IS ACTIVE.
      C
220     358     IF (IR - I) 492, 461, 359
        359     NR    = NIP(I+1) + (NSTAG - 1) * (NUMIT - 9)
                NRM1 = NR - 1
                IF (ISTAT(I+1)) 472, 361, 472
      C
225   C    CALCULATE SLOPE OF PATH ON RIGHT BETWEEN FINAL TWO (OR TWO
      C    "WORKING") POINTS.
      C
        361     DELTRZ = ZION(I+1,NR) - ZION(I+1,NRM1)
```

```
            DELTRX = XION(I+1,NR) - XION(I+1,NRM1)
230         IF (DELTRZ .EQ. 0.0) DELTRZ = 1.0E-16
            SLOPER = DELTRX / DELTRZ
      C
      C     MAKE SURE INTERSECTIONS CAN BE FOUND, THE SLOPES OF THE TWO LINES
      C     CAN NOT BE THE SAME.  STATEMENTS 362 BACKSTEP THE RIGHT HAND PATH
235   C     ONE ITERATION, ALLOWING A NEW SLOPE TO BE CALCULATED.  IF
      C     BACKSTEPPING IS NOT NEEDED, THE INTERSECTION POINTS ARE
      C     CALCULATED.
      C
            IF (SLOPEP - SLOPER) 363, 362, 363
240     362   IF (IFLAG2 .GE. 359 .AND. IFLAG2 .LE. 362)  GO TO 483
              NR = NR - 1
              IF (NR .EQ. 1) GO TO 482
              NRM1 = NR - 1
              GO TO 361
245     363   DUMMYR = XION(I+1,NR) - SLOPER * ZION(I+1,NR)
              ZINTR = (DUMMYR - DUMMYP) / (SLOPEP - SLOPER)
              XINTR = SLOPEP * ZINTR + DUMMYP
      C
      C     TEST TO SEE IF INTERSECTION POINTS ARE GOOD.  FIRST FIND DIRECTION
250   C     OF PATH PROPAGATION, 370 IMPLIES NEGITIVE Z DIRECTION, 375 IMPLIES
      C     POSITIVE Z DIRECTION, 380 IMPLIES VERTICAL (UP OR DOWN) DIRECTION.
      C     AS BEFORE, TESTS ARE RUN TO SEE IF THE INTERSECTIONS ARE "GOOD",
      C     IF LINEAR EXTRAPOLATION IS USED (SETS IFLAG1), OR IF BACK STEPPING
      C     IS NEEDED.  THESE TESTS ARE RUN IN ALL CASES.
255   C
            IF (ZION(I+1,NR) - ZION(I+1,NRM1)) 370, 380, 375
      C
        370   IF (ZION(I+1,NR) .LE. ZINTR .AND. ZINTR .LE.
           2  ZION(I+1,NRM1)) GO TO 397
260           IF (ZINTR .LT. ZION(I+1,NR)) GO TO 395
        372   IF (ZINTR .GT. ZION(I+1,NRM1)) GO TO 362
              IFLAG2 = 372
              GO TO 488
      C
265     375   IF (ZION(I+1,NRM1) .LE. ZINTR .AND. ZINTR .LE.
           3  ZION(I+1,NR)) GO TO 397
              IF (ZINTR .GT. ZION(I+1,NR)) GO TO 395
        377   IF (ZINTR .LT. ZION(I+1,NRM1)) GO TO 362
              IFLAG2 = 377
270           GO TO 488
      C
      C     WHEN STATEMENT 380 IS CALLED, WE HAVE TO TEST THE X COMPONENTS
      C     IN THE SAME MANOR AS WHEN STATEMENT 330 WAS CALLED TO LOOK TO
      C     THE LEFT.  NEED TO TEST FOR "GOOD" INTERSECTIONS, LINEAR
275   C     EXTRAPOLATION AND BACK STEPPING.
      C
        380   IF (XION(I+1,NR) - XION(I+1,NRM1)) 385, 490, 390
      C
        385     IF (XION(I+1,NR) .LE. XINTR .AND. XINTR .LE.
280        4    XION(I+1,NRM1)) GO TO 397
                IF (XINTR .LT. XION(I+1,NR)) GO TO 395
        387     IF (XINTR .GT. XION(I+1,NRM1)) GO TO 362
                IFLAG2 = 387
                GO TO 488
285   C
```

```
      390         IF (XION(I+1,NRM1) .LE. XINTR .AND. XINTR .LE.
     5            XION(I+1,NR)) GO TO 397
                 IF (XINTR .GT. XION(I+1,NR)) GO TO 395
      392        IF (XINTR .LT. XION(I+1,NRM1)) GO TO 362
290              IFLAG2 = 392
                 GO TO 488
C
C      STATEMENT 394 RESETS IFLAG2 IF STATEMENT 490 IS REFFRENCED.
C      STATEMENT 395 SETS IFLAG1, STATEMENT 397 CALCULATES THE
295    C      DISPLACEMENT TO THE RIGHT.  MAKE SURE IFLAG2 AND IFLAG3 ARE
C      PROPERLY SET.
C
      394        IFLAG2 = 0
                 GO TO 397
300   395        IFLAG1 = IFLAG1 + 2
      397        DSPLR = SQRT ((XINTR - XION(I,N)) ** 2 +
     6                    (ZINTR - ZION(I,N)) ** 2)
                 IF (IFLAG2 .GT. 0)  IFLAG2 = IFLAG2 - 362
                 IF (IFLAG3 .EQ. 2)  GO TO 405
305                GO TO 418
      405        IF (IFLAG2 .NE. -2 .AND. IFLAG1 .NE. 2)  GO TO 450
                 IFLAG3 = 4
                 GO TO 450
C
310   C  DEFINE IFLAG3.
      C  ----------
C
      C  THE FOLLOWING SECTION EXAIMES IFLAG1 AND IFLAG2 SO THAT IFLAG3 CAN BE
      C  DEFINED.  IFLAG3 IS USED IN CALC TO DETERMINE WHEN BOUNDARY REPULSION
315   C  EXISTS SO THAT THE NESCESSARY FORCES CAN BE ZEROED.  STATEMENT 430:
      C  BOUNDARIES INTERSECTED ON BOTH SIDES,  STATEMENT 435:  BOUNDARY
      C  INTERSECTED ON RIGHT,  STATEMENT 440:  BOUNDARY INTERSECTED ON LEFT,
      C  STATEMENT 445:  NO BOUNDARIES INTERSECTED.
C
320   418        IF (IFLAG2)  420, 445, 496
      420          IF (IFLAG1)  494, 445, 423
      423            IF (IFLAG2 .EQ. -1 .AND. IFLAG1 .EQ. 2)  GO TO 445
                     IF (IFLAG2 .EQ. -1)  GO TO 440
                       IF (IFLAG2 .EQ. -2 .AND. IFLAG1 .EQ. 1)  GO TO 445
325                    IF (IFLAG2 .EQ. -2)  GO TO 435
                       IF (IFLAG2 .NE. -3)  GO TO 496
                         IF (IFLAG1 .EQ. 1)  GO TO 440
                         IF (IFLAG1 .EQ. 2)  GO TO 435
                         IF (IFLAG1 .EQ. 3)  GO TO 430
330                        GO TO 494
      430                  IFLAG3 = 4
                           GO TO 450
      435                IFLAG3 = 3
                         GO TO 450
335   440              IFLAG3 = 2
                       GO TO 450
      445          IFLAG3 = 1
      450 IF (DSPLL .GT. 0.25)  GO TO 502
      452 IF (DSPLR .GT. 0.25)  GO TO 504
340   455 RETURN
C
      C  CALCULATIONS FOR (BOUNDARY) SPECIAL CASES.
```

```
      C  ------------------------------------------
      C
      C
      C     STATEMENTS 460 AND 461 CALCULATE THE DISPLACEMENT TO THE LEFT
      C     OF THE FIRST ACTIVE PATH TO THE BOUNDARY AND TO THE RIGHT OF THE
      C     LAST ACTIVE PATH TO THE BOUNDARY, RESPECTIVELY.  WHEN STATEMENT
      C     461 IS CALLED IFLAG3 HAS TO BE PROPERLY SET.
      C
        460     DSPLL = DS (ZION(I,N), XION(I,N), SLOPEP, 0, I)
                IF (DSPLL .EQ. 1.0E+20)  GO TO 498
                IFLAG3 = 2
                GO TO 358
      C
        461     DSPLR = DS (ZION(I,N), XION(I,N), SLOPEP, 1, I)
                IF (DSPLR .EQ. 1.0E+20)  GO TO 500
                IF (IFLAG2 .NE. -1 .AND. IFLAG1 .NE. 1)  GO TO 465
                  IFLAG3 = 4
                  GO TO 466
        465     IFLAG3 = 3
        466     RETURN
      C
      C  OTHER TESTS AND ASSIGNMENTS.
      C  ----------------------------
      C
      C     DEFINE IFLAG2 IN CASES WHERE TESTED PATH IS INACTIVE.
      C
        470 IFLAG2 = -1
            GO TO 311
        472 IF (IFLAG2)  475, 478, 496
        475 IFLAG2 = -3
            GO TO 361
        478 IFLAG2 = -2
            GO TO 361
      C
      C     BACK-STEPPING LOGIC WHEN MORE THEN TEN BACKSTEPS ARE NEEDED ON
      C     A PARTICULAR PATH.  ALLOWS USE OF INFORMATION IN CORE STORAGE
      C     NOT YET OVER WRITTEN WITH NEW RESULTS.
      C
      C     FOR LEFT-HAND PATH,
      C
        480 IFLAG2 = IFLAG2 + 312
            IFVAR = NIP(I-1) - (NSTAG - 1) * (NUMIT - 9)
            IF (IFVAR .GE. 140)  GO TO 506
            NLM1 = 141
            GO TO 311
      C
        481 IFVAR = NIP(I-1) - (NSTAG - 1) * (NUMIT - 9)
            IF (NLM1 .LE. (IFVAR + 2))  GO TO 506
            NL = NLM1
            NLM1 = NLM1 - 1
            GO TO 311
      C
      C     FOR RIGHT-HAND PATH,
      C
        482 IFLAG2 = IFLAG2 + 362
            IFVAR = NIP(I+1) - (NSTAG - 1) * (NUMIT - 9)
            IF (IFVAR .GE. 140)  GO TO 508
            NRM1 = 141
```

```
400         GO TO 361
      C
        483 IFVAR = NIP(I+1) - (NSTAG - 1) * (NUMIT - 9)
            IF (NRM1 .LE. (IFVAR + 2))  GO TO 508
            NR = NRM1
405         NRM1 = NRM1 - 1
            GO TO 361
      C
      C   ERROR CONDITIONS.
      C   ================
410   C
      C   ***** STATEMENTS 484 THROUGH 508 ARE VARIOUS ERROR EXITS. *****
      C   THE VALUE OF IFLAG2 DETERMINES IF VALUES OUTPUT ARE FOR RIGHT OR
      C   LEFT, SINCE VALUE OF IFLAG2 REFERENCES A PROGRAM STATEMENT.
      C
415   C     - ERROR 521 - STATEMENTS 506, 508 MEAN INDICES NL, NR WERE BACK-
      C                   STEPPED (138 TIMES) UNTIL CORE STORAGE NO LONGER
      C                   CONTAINED VALUES THAT WERE CALCULATED DURING THE
      C                   PREVIOUS STAGE.  CORRECT COMPARISIONS CAN NOT BE
      C                   MADE PAST THIS POINT. (FATAL).
420   C
      C     - ERROR 522 - STATEMENTS 486, 488 MEAN THAT THE INTERSECTION POINTS
      C                   CONSIDERED IN STATEMENTS 322, 327, 342, 347, 372,
      C                   377, 387 AND 392 DO NOT SATISFY ANY LOGICAL
      C                   CRITERION.  UNPHYSICAL INTERSECTIONS (FATAL).
425   C
      C     - ERROR 523 - STATEMENTS 484, 490 INDICATE THAT POINTS N AND N-1
      C                   ARE THE SAME.  ION ON LEFT OR RIGHT HAND PATH,
      C                   RESPECTIVILY, DID NOT MOVE.  UNPHYSICAL UNLESS ION
      C                   HAS ZERO VELOCITY AND THERE IS NO NET FORCE
430   C                   ACTING ON IT (FATAL).
      C
      C     - ERROR 524 - STATEMENT 492 INDICATES THAT I IS LESS THAN ONE OR
      C                   GREATER THEN NUMION.  SHOULD NOT OCCUR SINCE I IS
      C                   DO LOOP INDEX (FATAL).
435   C
      C     - ERROR 525 - IFLAG1 IMPROPERLY DEFINED (FATAL).
      C
      C     - ERROR 526 - IFLAG2 IMPROPERLY DEFINED (FATAL).
      C
440   C     - ERROR 528 - DSPLL OR DSPLR COULD NOT BE DEFINED (FATAL).
      C
      C     - ERROR 529 - DSPLL OR DSPLR UNUSUALLY LARGE (NON-FATAL).
      C
        484 IFLAG2 = 484
445         IF (VELTX(I-1) .EQ. 0.0 .AND. VELTZ(I-1) .EQ. 0.0)
           7GO TO 350
            WRITE (IOUT, 523) IFLAG2, I, NL, N, ZION(I-1,NL), ZION(I-1,NLM1),
           8    XION(I-1,NL), XION(I-1,NLM1), ZINTL, XINTL, SLOPEP, SLOPEL
            GO TO 510
450     486 WRITE (IOUT, 522) IFLAG2, I, NL, N, ZION(I-1,NL), ZION(I-1,NLM1),
           9    XION(I-1,NL), XION(I-1,NLM1), ZINTL, XINTL, SLOPEP, SLOPEL
            GO TO 510
        488 WRITE (IOUT, 522) IFLAG2, I, NR, N, ZION(I+1,NR), ZION(I+1,NRM1),
           1    XION(I+1,NR), XION(I+1,NRM1), ZINTR, XINTR, SLOPEP, SLOPER
455         GO TO 510
        490 IFLAG2 = 490
```

```
            IF (VELTX(I+1) .EQ. 0.0 .AND. VELTZ(I+1) .EQ. 0.0)
           2GO TO 394
            WRITE (IOUT, 523) IFLAG2, I, NR, N, ZION(I+1,NR), ZION(I+1,NRM1),
460        3    XION(I+1,NR), XION(I+1,NRM1), ZINTR, XINTR, SLOPEP, SLOPER
            GO TO 510
          492 WRITE (IOUT, 524) I, IL, IR, NUMION
            GO TO 510
          494 WRITE (IOUT, 525)  IFLAG1, I, N
465         GO TO 510
          496 WRITE (IOUT, 526)  IFLAG2, I, N
            GO TO 510
          498 IFLAG2 = 460
            WRITE (IOUT, 528)  IFLAG2, I, N
470         GO TO 510
          500 IFLAG2 = 461
            WRITE (IOUT, 528)  IFLAG2, I, N
            GO TO 510
          502 IF (ICLERR .EQ. 1)  GO TO 452
475         IFLAG2 = 355
            WRITE (IOUT, 529) IFLAG2, I, N, NL, DSPLL, SLOPEL, SLOPEP, XINTL,
           4                  ZINTL, DUMMYL, DUMMYP, XION(I,N), ZION(I,N)
            IFLAG2 = 0
            GO TO 452
480       504 IF (ICLERR .EQ. 1)  GO TO 510
            IFLAG2 = 397
            WRITE (IOUT, 529) IFLAG2, I, N, NR, DSPLR, SLOPER, SLOPEP, XINTR,
           5                  ZINTP, DUMMYR, DUMMYP, XION(I,N), ZION(I,N)
            IFLAG2 = 0
485       506 WRITE (IOUT, 521) IFLAG2, I, NL, N, SLOPEP, SLOPEL, ZINTL,
           6   XINTL, DUMMYP, DUMMYL, XION(I-1,NL), XION(I-1,NLM1),
           7   ZION(I-1,NL), ZION(I-1,NLM1), DSPLL, THETAP
            GO TO 510
          508 WRITE (IOUT, 521) IFLAG2, I, NR, NRM1, N, SLOPEP, SLOPER, ZINTR,
490        8   XINTR, DUMMYP, DUMMYR, XION(I+1,NR), XION(I+1,NRM1),
           9   ZION(I+1,NR), ZION(I+1,NRM1), DSPLP, THETAP
            GO TO 510
          510 RETURN
      C
495   C     ERROR CONDITION FORMATS, ERROR NUMBER IS FORMAT NUMBER.
      C
          521 FORMAT (/,11X,23H****** ERROR 521 ******,/,/,11X,
             1        36HNL OR NR BACK STEPPED TO FAR (FATAL),/,11X,
             2        28HCALLED FROM SUBROUTINE CALCD,/,11X,
500          3        8HIFLAG2 =,I5,5H  I =,I5,13H  N(L OR R) =,I5,9H  N(L OR ,
             4        6HR)M1 =,I5,5H  N =,I5,10H  SLOPEP =,E9.3,12H  SLOPE(L OR,
             5        5H R) =,E9.3,/,11X,16H  ZINT(L OR R) =,E9.3,/14HXINT(L OR,
             6        5H R) =,E9.3,10H  DUMMYP =,E9.3,17H  DUMMY(L OR R) =,E9.3,
             7        /,11X,33H  XION(I (= OR +) 1, N(L OR R)) =,E9.3,
505          8        33HXION(I (= OR +) 1, N(L OR R)M1) =,E9.3,
             9        33H  ZION(I (= OR +) 1, N(L OR R)) =,/,11X,E9.3,
             1        33HZION(I (= OR +) 1, N(L OR R)M1) =,E9.3,
             2        16H  DSPL(L OR R) =,E9.3,10H  THETAP =,E9.3)
          522 FORMAT (/,11X,23H****** ERROR 522 ******,/,/,11X,
510          1        38HUNPHYSICAL INTERSECTION POINTS (FATAL),/,11X,
             2        28HCALLED FROM SUBROUTINE CALCD,/,11X,
             3        8HIFLAG2 =,I5,5H  I =,I5,13H  N(L OR R) =,I5,
             4        5H  N =,I5,33H  ZION(I (= OR +) 1, N(L OR R)) =,E9.3,/,11X,
```

```
     5       33HZION(I (= OR +) 1, N(L OR R)M1) =,E9.3,
515  6       33H  XION(I (= OR +) 1, N(L OR R)) =,E9.3,/,11X,
     7       33HXION(I (= OR +) 1, N(L OR R)M1) =,E9.3,
     8       16H  ZINT(L OR R) =,E9.3,16H  XINT(L OR R) =,E9.3,/,11X,
     9       8HSLOPEP =,F9.3,17H  SLOPE(L OR R) =,E9.3)
   523 FORMAT (/,11X,23H****** ERROR 523 ******,/,/,11X,
520    1       45HION POSITIONS ARE THE SAME, NO MOTION (FATAL),/,11X,
       2       28HCALLED FROM SUBROUTINE CALCD,/,11X,
       3       8HIFLAG2 =,I5,5H  I =,I5,13H  N(L OR R) =,I5,
       4       5H  N =,I5,33H  ZION(I (= OR +) 1, N(L OR R)) =,E9.3,/,11X,
       5       33HZION(I (= OR +) 1, N(L OR R)M1) =,E9.3,
525    6       33H  XION(I (= OR +) 1, N(L OR R)) =,E9.3,/,11X,
       7       33HXION(I (= OR +) 1, N(L OR R)M1) =,E9.3,
       8       16H  ZINT(L OR R) =,E9.3,16H  XINT(L OR R) =,E9.3,/,11X,
       9       8HSLOPEP =,F9.3,17H  SLOPE(L OR R) =,E9.3)
   524 FORMAT (/,11X,23H****** ERROR 524 ******,/,/,11X,
530    1       30HDO LOOP INDEX MODIFIED (FATAL),/,11X,
       2       28HCALLED FROM SUBROUTINE CALCD,/,11X,
       3       3HI =,I5,6H  IL =,I5,6H  IR =,I5,10H  NUMION =,I5)
   525 FORMAT (/,11X,23H****** ERROR 525 ******,/,/,11X,
       1       33HIFLAG1 IMPROPERLY DEFINED (FATAL),/,11X,
535    2       28HCALLED FROM SUBROUTINE CALCD,/,11X,
       3       8HIFLAG1 =,I5,5H  I =,I5,5H  N =,I5)
   526 FORMAT (/,11X,23H****** ERROR 526 ******,/,/,11X,
       1       33HIFLAG2 IMPROPERLY DEFINED (FATAL),/,11X,
       2       28HCALLED FROM SUBROUTINE CALCD,/,11X,
540    3       8HIFLAG2 =,I5,5H  I =,I5,5H  N =,I5)
   528 FORMAT (/,11X,23H****** ERROR 528 ******,/,/,11X,
       1       43HDSPLL OR DSPLR COULD NOT BE DEFINED (FATAL),/,11X,
       2       28HCALLED FROM SUBROUTINE CALCD,/,11X,
       3       8HIFLAG2 =,I5,5H  I =,I5,5H  N =,I5)
545 529 FORMAT (/,11X,23H****** ERROR 529 ******,/,/,11X,
       1       32HDSPLL OR DSPLR LARGE (NON-FATAL),/,11X,
       2       28HCALLED FROM SUBROUTINE CALCD,/,11X,
       3       8HIFLAG2 =,I5,5H  I =,I5,5H  N =,I5,13H  N(L OR R) =,
       4       I5,16H  DSPL(L OR R) =,E9.3,17H  SLOPE(L OR R) =,E9.3,
550    5       /,11X,8HSLOPEP =,E9.3,16H  XINT(L OR R) =,E9.3,
       6       16H  ZINT(L OR R) =,E9.3,17H  DUMMY(L OR R) =,E9.3,
       7       10H  DUMMYP =,E9.3,/,11X,11HXION(I,N) =,E9.3,
       8       13H  ZION(I,N) =,E9.3)
        END
```

```
      SUBROUTINE BOUND (Z, X, I)
C
C     ************  BOUNDARY CHECK ROUTINE  ************
C
C     PROGRAM DESCRIPTION; PROGRAMMER = WILLIAM DEININGER, 1 - 8 - 82
C     REVISIONS: (INCLUDE DATE, INITIALS AND DESCRIBE CHANGE ** PLEASE **)
C
C          THIS SUBROUTINE CHECKS THE POINT (Z,X) TO SEE IF IT LIES INSIDE
C     THE DEFINED BOUNDARIES OF THE SIMULATION.  IF (Z,X) DOES LIE INSIDE
C     THE DEFINED BOUNDARIES, NO CHANGES ARE MADE AND CONTROL IS RETURNED
C     TO SUBROUTINE CALC.  IF (Z,X) LIES OUTSIDE THE DEFINED BOUNDARIES,
C     THE PATH STATUS IS SET EQUAL TO THE ITERATION NUMBER.  IN ADDITION,
C     IF (Z,X) LIES ON THE FIRST OR LAST ACTIVE PATH AND LIES OUTSIDE THE
C     BOUNDARIES, THE LEFT-MOST (IL) OR RIGHT-MOST (IR) INDEX IS RESET.
C
C     ********  VARIABLE DICTIONARY  ********
C
C     I       : PATH INDEX
C     IDXNEW  : NEW INDEX FOR RIGHT OR LEFT-MOST PATH.
C     II      : DO LOOP INDEX.
C     IL      : INDEX OF LEFT-MOST ACTIVE PATH.
C     IR      : INDEX OF RIGHT-MOST ACTIVE PATH.
C     LASTDO  : DUMMY VARIABLE DENOTING LAST VALUE OF DO LOOP INDEX.
C     X       : X COORDINATE OF POINT TO BE TESTED.
C     Z       : Z COORDINATE OF POINT TO BE TESTED.
C
C     ***  END OF PROGRAM DESCRIPTION AND DICTIONARY  ***
C
C     PROGRAM DECLARATION STATEMENTS.
C     BLANK COMMON FOR LARGE ARRAYS, IO = INPUT-OUTPUT, PARAM = PARAMETERS.
C
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151),
     1    NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON / PARAM / N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4    DNOB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
C
C     TEST TO SEE IF THE X COORDINATE IS LESS THEN OR EQUAL TO THE BEAM
C     RADIUS, OR GREATER THEN OR EQUAL TO RBOUND.  TEST TO SEE IF THE Z
C     COORDINATE IS LESS THEN OR EQUAL TO 0 (ZERO) OR GREATER THEN OR
C     EQUAL TO ZBOUND.  FINALLY, TEST TO SEE IF Z IS LESS THAN OR EQUAL TO
C     THE THRUSTER LENGTH AND IF X IS LESS THEN OR EQUAL TO THE THRUSTER
C     RADIUS.  IF ANY OF THE ABOVE TESTS ARE TRUE, SET ISTAT(I) = N,
C     OTHERWISE RETURN TO SUBROUTINE CALC.
C
      IF (X .LE. RB)  GO TO 50
      IF (X .GE. RBOUND)  GO TO 50
      IF (Z .LE. 0.0)  GO TO 50
      IF (Z .GE. ZBOUND)  GO TO 50
      IF (Z .LE. THRLEN .AND. X .LE. RT)  GO TO 50
      RETURN
   50 ISTAT(I) = N
C
C     IF (Z,X) IS ON THE FIRST OR LAST ACTIVE PATH AND LIES OUTSIDE THE
```

```
C  DEFINED BOUNDARIES, RESET THE COORESPONDING INDEX TO THE INDEX OF
C  THE NEXT ACTIVE PATH.
C
C  FOR LEFT MOST PATH,
C  -------------------
C
      IF (I - IL)  200, 80, 120
   80 CONTINUE
      LASTDO = NUMION - IL
      DO 100 II = 1, LASTDO
         IDXNEW = I + II
         IF (ISTAT(IDXNEW))  210, 105, 100
  100 CONTINUE
  105 IL = IDXNEW
      RETURN
C
C  FOR RIGHT MOST PATH,
C  --------------------
C
  120 IF (I - IR)  160, 125, 200
  125 CONTINUE
      DO 140 II = 1, IR
         IDXNEW = I - II
         IF (ISTAT(IDXNEW))  210, 145, 140
  140 CONTINUE
  145 IR = IDXNEW
  160 RETURN
C
C  ERROR CONDITIONS.
C  -----------------
C
C  ***** STATEMENTS 200 THROUGH 210 ARE ERROR EXITS. *****
C
C      -  ERROR 610  - I,IL OR IR DEFINED INCORRECTLY, FATAL.
C
C      -  ERROR 612  - ISTAT(I) IMPROPERLY DEFINED, FATAL.
C
  200 WRITE (IOUT,610) I, IL, IR
      ISTAT(I) = 8888
      GO TO 250
  210 WRITE (IOUT,612) I, IDXNEW, ISTAT(I), ISTAT(IDXNEW), N,
     6                 NIP(I), NIP(IDXNEW)
      ISTAT(I) = 8888
  250 RETURN
C
C  ERROR CONDITION FORMATS, ERROR NUMBER IS FORMAT NUMBER.
C
  610 FORMAT (/,11X,23H****** ERROR 610 ******,/,/,11X,
     1        39HI, IL OR IR DEFINED INCORRECTLY (FATAL),/,11X,
     2        28HCALLED FROM SUBROUTINE BOUND,/,11X,3HI =,I5,
     3        6H  IL =,I5,6H  IR =,I5)
  612 FORMAT (/,11X,23H****** ERROR 612 ******,/,/,11X,
     1        34HISTAT() IMPROPERLY DEFINED (FATAL),/,11X,
     2        28HCALLED FROM SUBROUTINE BOUND,/,11X,
     3        3HI =,I5,10H  IDXNEW =,I5,12H  ISTAT(I) =,I5,
     4        17H  ISTAT(IDXNEW) =,I5,5H  N =,I5,10H  NIP(I) =,
     5        I5,15H  NIP(IDXNEW) =,I5)
       END
```

```
      SUBROUTINE WRIT(KE)
C--WRIT PRINTS INFORMATION ABOUT THE SIMULATION
C--KE=1: OUTPUT HEADING, INITIAL INFORMATION AND DATA
C--KE=2: OUTPUT INTERIM STATUS OF MAIN VARIABLES
C--KE=3: FINISH OF A PASS, RESULTS, START OF NEW PASS
C--KE=4: CREATE FILE OF PATH COORDINATES
C--KE=5: CREATE FILE OF POSITION - DENSITY TRIPLETS.
C
C  BLANK COMMON FOR LARGE ARRAYS
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151),
     1    NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON/PARAM/N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4    DNOB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
      DATA IPAG,LAB,NPAS /0,6HPLASIM,1  /
C
C--   FORMATS
C
   10 FORMAT( ////,43HO    THIS RUN MAY BE CHARACTERIZED BY INFO:,// )
   11 FORMAT(1H1,//,60X,A6,I3,//////  )
   13 FORMAT(////17X,33HP L A S M A   S I M U L A T I O N,///
     2       17X,43HA COMPUTER CODE TO DESCRIBE THE PROPAGATION,//
     3       17X,43HOF A CHARGE-EXCHANGE PLASMA IN THE VICINITY,//
     4       17X,40HOF AN ELECTRICALLY PROPELLED SPACECRAFT ,///
     5       17X,45HWRITTEN BY WILLIAM DEININGER AND DALE WINDER,//
     6       17X,33HFOR THE JET PROPULSION LABORATORY,//
     7       21X,27H( J P L  P. O. NO. 955322 ),///
     8       17X,41HHAROLD R. KAUFMAN, PRINCIPAL INVESTIGATOR ,//
     9       27X,21HDEPARTMENT OF PHYSICS,//
     1       25X,25HCOLORADO STATE UNIVERSITY,//
     2       26X,23HFORT COLLINS, CO  80523  ,//
     3       33X,9HFALL 1981)
   14 FORMAT(1H1,//60X,A6,I3,/////,10X,21HSCHEMATIC OF THRUSTER,///
     2       11X,3H= :,/,11X,3HA :,/,11X,3H: :,/,11X,3H: :, /,
     3       11X,3H: :,5X,6HTHRLEN,/,11X,3H: :,7X,1HV,/,
     4       11X,11H: ----------,43X,6HZBOUND,/,11X,1H:,6X,4HA  :,44X,1HV
     5       /,9X,6HRBOUND,3X,4H:  :,/11X,1H:,6X,1H:,2X,45(1H=),/,
     6       11X,1H:,6X,2HRT,5X,1HA,40X,1H:,/,11X,2(1H:,6X)2HRB,39X,1H:,/
     7       4(11X,1H:,6X,1H:,6X,1H:,40X,1H:,/),13X,1H+,13(4H- . ),/
     8       11X,5H(0,0),50X,1H:,/,5(66X,1H:,/),21X,45(1H=),/,
     9       2(21X,1H:,/),13X,9(1H-),/,6(13X,1H:,/),///)
   15 FORMAT(18X,28HINITIAL VALUES OF PARAMETERS,/,/,/,
     1       5X,22HFROM SECOND DATA CARD,,/,14X,6HNUMION,5X,5HNUMIT,
     2       7X,3HKEY,4X,6HICLWRT,4X,6HICLPLT,4X,6HNTOTST,4X,6HICLERR,/,
     3       10X,7I10,/,/,/,5X,21HFROM THIRD DATA CARD,,/,18X,2HRB,4X,
     4       6HRBOUND,8X,2HRT,4X,6HTHRLEN,5X,5HBMCUR,6X,4HUTIL,/,10X,
     5       6F10.3,/,/,/,8X,22HFROM FOURTH DATA CARD,,/,15X,5HTELIN,
     6       4X,6HTELOUT,4X,6HTTHNEU,4X,6HCEXSEC,4X,6HUMSION,/,10X,
     7       3F10.3,2F10.3,/,/,/,5X,21HFROM FIFTH DATA CARD,,/,14X,
     8       6HTIMEMU,4X,6HXVELMU,4X,6HZVELMU,/,10X,3F10.3,/,/,/,5X,
     9       22HCALCULATED QUANTITIES,,/,16X,4HTIME,4X,6HVELBOH,4X,
     1       6HZBOUND,/,10X,E10.3,2F10.3)
   21 FORMAT(1H1,12(5H -2- ),A6,I3//10X,27HINTERIM STATUS -- ITERATION,
     2       I4,3H OF,I4,11H ITERATIONS,
```

```
      3      /,/,3X,1HI,4X,1HN,3X,5HISTAT,4X,9HZION(I+1),6X,9HXION(I+1),
      4      6X,8HVELTZ(I),7X,8HVELTX(I),8X,6X,8X,7HDN(I,N),/)
   22 FORMAT(1X, I3, 1X, 2(I4, 2X), 4(E13.6, 2X), 15X, E13.6)
   31 FORMAT(/,/,27H RESULTS OF PASS--ITERATION,I4,3H OF,I4,9H ITERATIO,
      2      2HNS,/,3X,1HI,4X,1HN,3X,3HNIP,2X,5HISTAT,4X,9HZION(N+1),6X,
      3      9HXION(N+1),6X,8HVELTZ(I),7X,8HVELTX(I),8X,6HVELTOT,8X,
      4      7HDN(I,N),/)
   35 FORMAT(/,/,5X,39H************ BEGIN WRIT(5) ************,/)
   36 FORMAT(/,/,5X,39H************  END WRIT(5)  ************,/)
C--   WHAT KIND OF CALL IS IT
      GOTO  (1,2,3,4,5), KF
C--      1          1          1
C--INITIAL STATE--HEADING AND DATA
    1 IPAG=IPAG + 1
      WRITE(IOUT,11)LAB,IPAG
      WRITE(IOUT,13)
       IPAG = IPAG + 1
      WRITE(IOUT,14)LAB,IPAG
      IPAG = IPAG + 1
      WRITE(IOUT,11) LAB, IPAG
       WRITE(IOUT,10)
      WRITE(IOUT,IF1) (INFO(K),K=1,IW)
      WRITE(IOUT,15) NUMION, NUMIT, KEY, ICLWRT, ICLPLT, NTOTST, ICLERR,
     1               RB, RBOUND, RT, THRLEN, BMCUR, UTIL,
     2               TELIN, TELOUT, TTHNEU, CEXSEC, UMSION,
     3               TIMEMU, XVELMU, ZVELMU,
     4               TIME, VELBOH, ZBOUND
       RETURN
C--      2          2          2
C--THIS SECTION PRINTS THE INTERIM STATUS AT THE NTH ITERATION
    2 IPAG = IPAG + 1
      WRITE(IOUT,21) LAB,IPAG,N,NUMIT
      DO 28 I=1,NUMION
   28 WRITE(IOUT,22) I,N,ISTAT(I),ZION(I,N),XION(I,N),VELTZ(I),
     2   VELTX(I),DN(I,N)
       RETURN
C--      3          3          3
C THIS SECTION PRINTS RESULT OF A PASS AT NTH EXTRAPOLATION
    3 NITP = N + (NSTAG - 1) * (NUMIT + 1) - (10 * (NSTAG - 1))
      ITTOTN = (NTOTST * NUMIT + 1) - (NTOTST - 1) * 10
      WRITE(IOUT,31) NITP, ITTOTN
      RETURN
C--      4          4          4
C THIS SECTION CREATES FILE OF PATH COORDINATES
CWD  DEVICE CODES SHOULD BE CHANGED SO AS NOT TO INTERFER WITH WRIT(5)
    4 NMAX=IFIX(FLOAT(NUMION)/4.)
       REWIND IPATHS
      WRITE(IPATHS) NMAX,NUMION
      WRITE(IPATHS) (ISTAT(I),I=1,NMAX)
      DO 44 I=1,NMAX
      ISAT=ISTAT(I)
   44 WRITE(IPATHS) (ZION(I,NN),XION(I,NN),NN=1,ISAT)
      RETURN
C
C--      5          5          5
C
C  (WRIT(5) WRITTEN BY WILLIAM DEININGER)
```

```
C  THIS SECTION WRITES INFORMATION FROM THE FIRST (NUMIT - 9)
C  ITERATIONS IN CORE MEMORY TO AN EXTERNAL FILE.  THE INFORMATION
C  IS STORED ON THE EXTERNAL FILE IN "TRIPLETS"; EACH TRIPLET CONTAINS
C  ONE VALUE EACH FOR "XION()", "ZION()" AND "DN()".  THIS IS DONE IN
C  PATH MAJOR ORDER.  IN OTHER WORDS, ALL THE DESIRED RESULTS FOR ONE
C  PATH ARE OUTPUT BEFORE OUTPUTTING ANY RESULTS FOR NEIGHBORING PATHS.
C  FIRST THE PATH STATUS IS CHECKED TO MAKE SURE NO ERROR CONDITIONS
C  WERE SET DURING EXECUTION.  THEN THE INITIAL ITERATION INDEX IS SET
C  EQUAL TO ONE FOR THE FIRST STAGE AND 10 FOR ALL STAGES THERE AFTER.
C  THE FINAL ITERATION INDEX IS COMPUTED, FOR WHICH THE MAXIMUM VALUE
C  IS (NUMIT + 1) AND OCCURS IF THE PATH IS STILL ACTIVE.  IF THE PATH
C  IS ACTIVE (ISTAT = 0), THE NUMBER OF TRIPLETS BECOMES (NUMIT - 9).
C  IF ISTAT IS GREATER THEN ZERO AND BECAME GREATER THEN ZERO IN
C  THE CURRENT STAGE, THE NUMBER OF TRIPLETS BECOMES (ISTAT - 1).
C  IF ISTAT BECAME NON-ZERO IN A PREVIOUS STAGE WE CONSIDER THE NEXT
C  PATH.
C       AFTER CALCULATING THE NUMBER OF TRIPLETS, THE PATH NUMBER AND
C  NUMBER OF TRIPLETS ARE OUTPUT TO THE EXTERNAL FILE.  THEN THE
C  TRIPLETS ARE OUTPUT.  THE NEXT PATH IS THEN CONSIDERED, ETC.
C
C  -  ERROR 207  -  PATH STATUS IMPROPERLY DEFINED, FATAL.
C
      5 IF (NSTAG .EQ. 1)  REWIND IPATHS
        WRITE (IOUT,35)
        DO 100 IS = 1, NUMION
           IF (ISTAT(IS) - 8888)  25, 100, 95
     25       INITIT = 1
              IF (NSTAG .GT. 1)  INITIT = 10
              LASTIT = NIP(IS) - (NSTAG - 1) * (NUMIT - INITIT)
              IF (ISTAT(IS))  95, 50, 40
     40          IFVAR = (((NSTAG - 1) * NUMIT) + 1 - ((NSTAG - 2) *
       1          INITIT) * NSTGMU)
                  IF (NIP(IS) .LE. IFVAR)  GO TO 100
                     LASTIT = ISTAT(IS) - 1
                     GO TO 60
     50       LASTIT = LASTIT - 10
     60       NUMTRI = LASTIT
              MODNTR = (NUMTRI / 3) + 1
              WRITE (IPATHS)  IS, MODNTR
              DO 90 NS = 1, NUMTRI
                 IF (NS .EQ. 1)  GO TO 85
                 J = MOD (NS, 3)
                 IF (J .NE. 0)  GO TO 90
     85          WRITE (IPATHS)  XION(IS,NS), ZION(IS,NS), DN(IS,NS)
     90       CONTINUE
              GO TO 100
     95       WRITE (IOUT, 207)  IS, ISTAT(IS)
              ISTAT(IS) = 8888
              RETURN
    100 CONTINUE
        WRITE (IOUT,36)
        RETURN
    207 FORMAT (/,11X, 23H****** ERROR 207 ******,/,/,11X,
       1        38HPATH STATUS IMPROPERLY DEFINED (FATAL),/,11X,
       2        29HCALLED FROM SUBROUTINE READER,/,11X,
       3        6HISTAT(,I4,5H ) = ,I5)
        END
```

```
      FUNCTION DS (Z, X, SLOPEP, LR, I)
C
C  ************ BOUNDARY DISPLACEMENT ROUTINE ************
C
C  PROGRAM DESCRIPTION; PROGRAMMER - WILLIAM DEININGER, 8 - 26 - 81
C  REVISIONS; (INCLUDE DATE, INITIALS AND DESCRIBE CHANGE ** PLEASE **)
C
C     THIS FUNCTION SUBROUTINE FINDS THE PERPENDICULAR DISPLACEMENT FROM
C  THE FIRST OR LAST ACTIVE PATH TO THE BOUNDARY.  CALCD CONSTRUCTS A
C  PERPENDICULAR TO THE PATH FROM THE CURRENT POINT (Z,X) WITH SLOPE
C  "SLOPEP".  FUNCTION DS THEN EXTRAPOLATES THIS PERPENDICULAR OF SLOPE
C  "SLOPEP" TO THE LEFT OR RIGHT (DEPENDING ON WHETHER WE ARE CONSID -
C  ERING THE FIRST OR LAST ACTIVE PATH) AND FINDS THE Z AND X INTERCEPTS
C  (ZINT AND XINT) ALONG THE BOUNDARY LINE.  ZINT AND XINT ARE CHECKED
C  TO SEE IF THEY LIE ON OR BETWEEN THE BOUNDARY ENDPOINTS ON THE
C  BOUNDARY LINE.  IF THEY DO, THE DISPLACEMENT IS CALCULATED, IF THEY
C  DO NOT, ZINT AND XINT ARE CALCULATED ALONG THE NEXT BOUNDARY LINE
C  AND TESTED AGAIN.  THIS CONTINUES UNTIL "GOOD" INTERSECTION POINTS
C  ARE FOUND OR ALL BOUNDARIES HAVE BEEN CONSIDERED IN WHICH CASE AN
C  ERROR MESSAGE IS OUTPUT.  THE PERPENDICULAR DISPLACEMENT IS RETURNED
C  TO CALCD.
C
C    ********  VARIABLE DICTIONARY  ********
C
C  DS      : PERPENDICULAR DISPLACEMENT FROM CURRENT POINT TO BOUNDARY.
C  I       : PATH INDEX.
C  LR      : DETERMINES WHICH SIDE IS BEING CONSIDERED,
C              = 1   LOOKING TO THE LEFT.
C              = 2   LOOKING TO THE RIGHT.
C  X       : CURRENT X POSITION ON FIRST OR LAST ACTIVE PATH (XION(I,N)).
C  XINT    : X INTERSECTION POINT ON BOUNDARY LINE.
C  Z       : CURRENT Z POSITION ON FIRST OR LAST ACTIVE PATH (ZION(I,N)).
C  ZINT    : Z INTERSECTION POINT IN BOUNDARY LINE.
C
C  ***  END OF PROGRAM DESCRIPTION AND DICTIONARY  ***
C
C  PROGRAM DECLARATION STATEMENTS.
C  BLANK COMMON FOR LARGE ARRAYS, IO = INPUT-OUTPUT, PARAM = PARAMETERS.
C
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151),
     1    NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28),
     2    IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON / PARAM / N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3    TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,O,RB95N,
     4    DNOB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
C
C  DETERMINE WHETHER THE DISTANCE ON THE RIGHT OR THE LEFT IS DESIRED.
C
      IF (LR .EQ. 2)  GO TO 200
C
C  CALCULATIONS FOR LEFT.
C  ----------------------
C
C  TEST FOR INTERSECTIONS ALONG THE END OF THE THRUSTER BETWEEN THE
C  BEAM EDGE (THRLEN, RB) AND THE THRUSTER CORNER (THRLEN, RT).  (FIRST
```

```
C  THE INTERSECTIONS ARE FOUND ALONG THE BOUNDARY LINE AND THEN TESTED
C  TO SEE IF THEY LIE ON OR BETWEEN THE BOUNDARY ENDPOINTS ON THE
C  BOUNDARY LINE.)  (EQUATION OF BOUNDARY LINE:  Z = THRLEN)
C
  100 IF (SLOPEP .LT. -1.0E+10)  GO TO 110
      ZINT = THRLEN
      XINT = (THRLEN - Z) * SLOPEP + X
      IF (XINT .GE. PB .AND. XINT .LE. RT)  GO TO 300
C
C  TEST FOR INTERSECTIONS ALONG THE EDGE OF THE THRUSTER BETWEEN THE
C  THRUSTER CORNER (THRLEN, RT) AND THE SPACE CRAFT WALL (0, RT).
C  (EQUATION OF BOUNDARY LINE:  X = RT)
C
  110 IF (SLOPEP .EQ. 0.0)  GO TO 120
      ZINT = (RT - X) / SLOPEP + Z
      XINT = RT
      IF (ZINT .GE. 0.0 .AND. ZINT .LE. THRLEN)  GO TO 300
C
C  TEST FOR INTERSECTIONS ALONG THE SPACE CRAFT SURFACE BETWEEN (0, RT)
C  AND (0, RBOUND).  (EQUATION OF BOUNDARY LINE:  Z = 0.0)
C
  120 IF (SLOPEP .LT. -1.0E+10)  GO TO 130
      ZINT = 0.0
      XINT = X - Z * SLOPEP
      IF (XINT .GE. RT .AND. XINT .LE. RBOUND)  GO TO 300
C
C  TEST FOR INTERSECTIONS ALONG RBOUND BETWEEN THE SPACE CRAFT WALL
C  (0, RBOUND) AND (ZBOUND, RBOUND).  (EQUATION OF BOUNDARY LINE:
C  X = RBOUND)
C
  130 IF (SLOPEP .EQ. 0.0)  GO TO 400
      ZINT = (RBOUND - X) / SLOPEP + Z
      XINT = RBOUND
      IF (ZINT .GE. 0.0 .AND. ZINT .LE. ZBOUND)  GO TO 300
C
C  TEST FOR INTERSECTIONS ALONG THE BEAM EDGE BETWEEN THE THRUSTER
C  (THRLEN, RB) AND (ZBOUND, RB).  (EQUATION OF BOUNDARY LINE:  X = RB)
C
      ZINT = (RB - X) / SLOPEP + Z
      XINT = RB
      IF (ZINT .GE. THRLEN .AND. ZINT .LE. ZBOUND)  GO TO 300
      GO TO 400
C
C  CALCULATIONS FOR RIGHT.
C  ======================
C
C  TEST FOR INTERSECTIONS ALONG ZBOUND BETWEEN THE BEAM EDGE (ZBOUND,
C  RB) AND (ZBOUND, RBOUND).  THE Z COMPONENT OF THE TOTAL VELOCITY
C  CHECKED TO DETERMINE THE DIRECTION OF PATH PROPAGATION.  (EQUATION
C  OF BOUNDARY LINE:  Z = ZBOUND)
C
  200 IF (SLOPEP .LT. -1.0E+10)  GO TO 210
      ZINT = ZBOUND
      XINT = (ZBOUND - Z) * SLOPEP + X
      IF (XINT .GE. RB .AND. XINT .LE. RBOUND)  GO TO 300
  210 IF (VELTZ(I))  220, 400, 240
C
```

```
C  TEST FOR INTERSECTIONS ALONG RBOUND BETWEEN THE SPACE CRAFT SURFACE
C  (0, RBOUND) AND (ZBOUND, RBOUND).  (EQUATION OF BOUNDARY LINE:
C  X = RBOUND)
C
  220 ZINT = (RBOUND - X) / SLOPEP + Z
      XINT = RBOUND
      IF (ZINT .GE. 0.0 .AND. ZINT .LE. ZBOUND)  GO TO 300
      GO TO 400
C
C  TEST FOR INTERSECTIONS ALONG BEAM EDGE BETWEEN END OF THRUSTER
C  (THRLEN, RB) AND (ZBOUND, RB).  (EQUATION OF BOUNDARY LINE:  X = RB)
C
  240 ZINT = (RB - X) / SLOPEP + Z
      XINT = RB
      IF (ZINT .GE. THRLEN .AND. ZINT .LE. ZBOUND)  GO TO 300
      GO TO 400
C
C  CALCULATE THE DISTANCE TO THE BOUNDARY.
C
  300 DS = SQRT ((Z - ZINT) ** 2 + (X - XINT) ** 2)
      IF (DS .GT. 0.25)  GO TO 402
      RETURN
C
C  ********  ERROR CONDITIONS  ********
C
C  -  ERROR 410  -  NO "GOOD" INTERSECTION POINTS FOUND BETWEEN
C                   BOUNDARIES AND CURRENT PATH USING DS (FATAL).
C
C  -  ERROR 412  -  DS UNUSUALLY LARGE (NON-FATAL).
C
  400 WRITE (IOUT, 410) X, Z, SLOPEP, XINT, ZINT, I, VELTZ(I), LR
      GO TO 408
  402 IF (ICLERR .EQ. 1)  GO TO 409
      IF (LR .EQ. 1)  GO TO 409
      WRITE (IOUT, 412)  LR, I, DS, Z, X, SLOPEP, XINT, ZINT
      GO TO 409
  408 DS = 1.0E+20
  409 RETURN
C
C  ERROR CONDITION FORMATS.
C
  410 FORMAT (/,11X,23H****** ERROR 410 ******,/,/,11X,
     1        30HDS UNUSUALLY LARGE (NON-FATAL),/,11X,
     2        34HCALLED FROM FUNCTION SUBROUTINE DS,/,11X,
     3        3HX =,E9.3,5H  Z =,E9.3,10H  SLOPEP =,E9.3,8H  XINT =,E9.3,
     4        8H  ZINT =,E9.3,8H  VELTZ(,I5,4H ) =,E9.3,6H  LR =,I3)
  412 FORMAT (/,11X,23H****** ERROR 412 ******,/,/,11X,
     1        30HDS UNUSUALLY LARGE (NON-FATAL),/,11X,
     2        34HCALLED FROM FUNCTION SUBROUTINE DS,/,11X,
     3        4HLR =,I3,5H  I =,I5,6H  DS =,E9.3,5H  Z =,E9.3,5H  X =,
     4        E9.3,10H  SLOPEP =,E9.3,8H  XINT =,E9.3,8H  ZINT =,E9.3)
      END
```

```
      SUBROUTINE VRSPLT
C
C--VRSPLT USES VERSATEC PLOTTER TO PLOT ARRAYS X() AND Y()
C
CWD  THIS ROUTINE IS SITE DEPENDENT.  IT PLOTS THE CONTENTS OF ARRAYS
CWD  XION AND ZION FROM CORE MEMORY.
C
C  BLANK COMMON FOR LARGE ARRAYS
      COMMON ZION(41,151),XION(41,151),VZ(151),VX(151)
     1,NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28)
     1  ,IPATHS,IW,IF1(2),IF2(4),ICLERP
      COMMON/PARAM/N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     1 TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N
     1 ,DNOB,CEXSEC,TTHNEU,TIME,TTMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PINV2
      DIMENSION SAVZ(2),SAVX(2),DZ(151),DX(151)
      DATA ZAXLN,XAXLN,INC,LINTYP,ISYM /9.0,7.0,1,+0,1/
       DATA INTR/0/
CW USE INTR TO COUNT ENTRY NUMBER AND AVOID REINITIALIZING
C--FIRST ENTRY--SET UP THE SYSYEM, SCALE, AXES AND TITLE IF DESIRED
      INTR=INTR+1
       IF(KEY.EQ.0) GOTO 3
       IF(INTR.EQ.1) CALL PLOTS(0.,0.,0.)
C--SET ORIGIN OF PLOT
      CALL PLOT (1.,1.,-3)
      CALL SETMSG(1)
      SAVZ(1)=0.
      SAVX(1)=0.
      SAVZ(2)=ZBOUND/ZAXLN
      SAVX(2)=RBOUND/XAXLN
      ZMAX=ZAXLN*SAVZ(2)
CW WORD-SIZE DEPENDENT AREA; IW1, IW2 ARE FLAGS IN ITITL
       IW1=IW+ 1
       IW2=IW1 + IW/2
      CALL AXIS(0.,0.,ITITL(IW1),-40,ZAXLN,0.,SAVZ(1),SAVZ(2))
      CALL AXIS(0.,XAYLN,1H ,1,ZAXLN,0.,SAVZ(1),SAVZ(2))
      CALL AXIS(0.,0.,ITITL(IW2),40,XAXLN,90.,SAVX(1),SAVX(2))
      CALL SYMBOL(0.,8.0,0.14,ITITL(1),0.,80)
      DO 80 J=2,NUMION
      IF(ZION(J,1).GT.ZMAX) GO TO 82
   80 CONTINUE
   82 NUMPT=J-1
      IF(KEY.GT.0) NUMPT=NUMION+1
      DO 150 J=1,NUMPT
      NPTS=ISTAT(J)
      IF(ISTAT(J).EQ.0) NPTS=NIP(J)
      DO 100 NM=1,NPTS
      DZ(NM)=ZION(J,NM)
      DX(NM)=XION(J,NM)
  100 CONTINUE
      DO 120 I=1,2
      NPT=NPTS+I
      DZ(NPT)=SAVZ(I)
  120 DX(NPT)=SAVX(I)
      IF(J.EQ.NUMION+1) ISYM=0
      CALL LINE(DZ,DX,NPTS,INC,LINTYP,ISYM)
```

```
  190  CONTINUE
C--DRAW SCHEMATIC OF THRUSTER AND BEAM.
      DX(1)=RT
      DX(2)=RT
      DZ(1)=0.
      DX(3)=0.
      DZ(2)=THRLEN
      DZ(3)=THRLEN
      DZ(4)=THRLEN
      DX(4)=RB
      DX(5)=RB
      DZ(5)=ZBOUND
      NPTS=5
      DO 200 I=1,2
      NPT=NPTS+I
      DZ(NPT)=SAVZ(I)
  200 DX(NPT)=SAVX(I)
      CALL LINE(DZ,DX,NPTS,1,0,0)
CW FINISH THIS PLOT AND GO BACK FOR MORE
      CALL PLOT(0.,0.,-999)
      RETURN
CW-TERMINATE ALL PLOTTING--RELEASE OUTPUT TO VERSATEC PLOTTER
    3 IF (INTR .LT. 2) RETURN
      CALL PLOT(0.,0., +999)
      INTR = 0
      RETURN
      END
```

```
      SUBROUTINE VRSPL
C
C--VRSPL USES VERSATEC PLOTTER TO PLOT ARRAYS X() AND Y()
C
CWD  THIS ROUTINE IS SITE DEPENDENT.  IT PLOTS THE CONTENTS OF THE
CWD  ARRAYS XION AND ZION WHICH HAVE BEEN READ FROM THE EXTERNAL
CWD  FILE PATHS IN TERMS OF TRIPLETS.
C
C  BLANK COMMON FOR LARGE ARRAYS
      COMMON ZION(41,151),XION(41,151),VZ(151),VX(151)
     1,NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON / IO / IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28)
     1  ,IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON/PARAM/N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     1 TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N
     1 ,DNOB,CEXSEC,TTHNEU,TTME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5    NSTGMU,NTOTST,PIOV2
      DIMENSION SAVZ(2),SAVX(2),DZ(151),DX(151),D(151)
      DATA ZAXLN,XAXLN,INC,LINTYP,ISYM /8.0,6.5,1,+30,1/
       DATA INTR/0/
CW USE INTR TO COUNT ENTRY NUMBER AND AVOID REINITIALIZING
C--FIRST ENTRY--SET UP THE SYSTEM, SCALE, AXES AND TITLE IF DESIRED
CWD   11 - 23 - 81,   MUST REWIND FILE BEFORE READING
      REWIND IPATHS
      INTR=INTR+1
       IF(KEY.EQ.0) GOTO 3
       IF(INTR.EQ.1) CALL PLOTS(0.,0.,0.)
C--SET ORIGIN OF PLOT
      CALL PLOT (1.,1.,-3)
      SAVZ(1)=0.
      SAVX(1)=0.
      SAVZ(2)=0.92/ZAXLN
      SAVX(2)=RBOUND/XAXLN
      ZMAX=ZAXLN*SAVZ(2)
CW WORD-SIZE DEPENDENT AREA: IW1, IW2 ARE FLAGS IN ITITL
       IW1=IW+ 1
       IW2=IW1 + IW/2
      CALL AXIS(0.,0.,ITITL(IW1),-40,ZAXLN,0.,SAVZ(1),SAVZ(2))
      CALL AXIS(0.,XAXLN,1H ,1,ZAXLN,0.,SAVZ(1),SAVZ(2))
      CALL AXIS(0.,0.,ITITL(IW2),40,XAXLN,90.,SAVX(1),SAVX(2))
      CALL SYMBOL(1.,8.0,0.14,ITITL(1),0.,80)
CW--DC2981--MOD FOR READING, PLOTTING PATHS FILE
      WRITE(IOUT,21) NUMION,NUMIT
        NUM1 = NUMION +1
      DO 55 JS= 1, NTOTST
CWD  THE FOLLOWING CODE COUNTS THE NUMBER OF TRAJECTORIES WRITTEN
CWD  TO FILE PATHS.
      ICOUNT = 0
        IF (JS -1) 5,15,5
    5   IFVAR = ((JS-1) *NUMIT +1) - ((JS-2)*10)
        DO 7 I=1,NUMION
        IF(NIP(I)-IFVAR) 7,7,9
    7   CONTINUE
    9   INITOO = I
        DO 11 I = 1,NUMION
        II = NUM1 -I
        IF(NIP(II)-IFVAR) 11,11,12
```

```
      11    CONTINUE
      12    LASTDO = II
60          DO 14 IJ = INITDO, LASTDO
              IF (ISTAT(IJ) = 8888)  14, 13, 14
       13       IFVAR = (JS - NUMIT + 1) = ((JS - 1) * 10)
                IF (NIP(I) = IFVAR)  6, 6, 14
        6         ICOUNT = ICOUNT + 1
65     14   CONTINUE
            LASTDO = LASTDO - INITDO - ICOUNT
             GO TO 18
      15    INITDO = 1
            LASTDO = NUMION
70    18    CONTINUE
          DO 44 J=1, LASTDO
            READ(IPATHS) ION,NITER
            WRITE(IOUT,22) ION,NITER
      22  FORMAT(26H +++ VRSPL +++ ION NUMBER  ,I3,I5,11H ITERATIONS//)
75            INTEQ=5
      21  FORMAT(1HO,10X,13H+++ VRSPL +++,I5,6H IONS ,I5,11H ITERATIONS )
              DO 33 IT= 1,NITER
    CWD  12/7/81  VX,V7 TO DX,DZ  AND  D TO D(IT) TO HOLD DENSITIES
                READ(IPATHS) DX(IT),DZ(IT),D(IT)
80              IF(EOF(IPATHS) .NE. 0.0) GO TO 70
      33      CONTINUE
          DO 37 I=1,2
          NPT = NITER+I
          DZ(NPT)=SAV7(I)
85     37 DX(NPT)=SAVX(I)
          CALL LINE (DZ,DX,NITEP,INC,LINTYP,INTEQ)
       44 CONTINUE
       55 CONTINUE
    C--DRAW SCHEMATIC OF THRUSTER AND BEAM.
90     70 DX(1)=RT
          DX(2)=RT
          DZ(1)=0.
          DX(3)=0.
          DZ(2)=THRLEN
95        DZ(3)=THRLEN
          DZ(4)=THRLEN
          DX(4)=RB
          DX(5)=RB
          DZ(5)=ZBOUND
100       NPTS=5
          DO 200 I=1,2
          NPT=NPTS+I
          DZ(NPT)=SAVZ(I)
     200  DX(NPT)=SAVX(I)
105       CALL LINE(DZ,DX,NPTS,1,0,0)
    CW FINISH THIS PLOT AND GO BACK FOR MORE
          CALL PLOT(0.,0.,-999)
          RETURN
    CW-TERMINATE ALL PLOTTING--RELEASE OUTPUT TO VERSATEC PLOTTER
110     3 IF (INTR .LT. 2) RETURN
          CALL PLOT(0.,0., +999)
          INTR = 0
          RETURN
          END
```

```
      SUBROUTINE PLOTW  (X,Y,N,LB,LINE,LAB)
C
C--     SUB. PLOTW BY D. R. WINDER, PHYSICS DEPT., COLO. ST. UNIV.
C  PLOT ARRAYS Y VS X, EACH HAVING N POINTS, SELF SCALING.
C
C-- LB IS THE SYMBOL USED FOR THE CURRENT GRAPH--SUGGEST NUMERICAL ORDER
C--E.G.- FOR THE FIRST GRAPH, SET IN CALLING PROGRAM: LB=1H1
C--THEN FOR 2ND ONE, RESET IT: LB=1H2, ETC. ONLY ONE CHARACTER, PLEASE.
C--THE FIRST ENTRY IS CRITICAL--IT ESTABLISHES NUMBER OF POINTS, ALSO
C--  UPON FIRST ENTRY, MAX AND MIN VALUES ARE FOUND AND ARE USED LATER
C--IF LATER CALLS INVOLVE POINTS OUTSIDE THESE LIMITS, THEY WILL BE
C--TRUNCATED AND AN ERROR MESSAGE PRINTED ON THE PLOT FILE.
C--  TO SIGNAL : LAST GRAPH TO BE PLOTTED, PUT N=-0. ALL GRAPHS WILL
C--BE PLOTTED ON ONE SHEET VIA THE ARRAY LINE(103,60). IF ONLY ONE
C--GRAPH IS DESIRED, SET N=-N IN THE CALLING PROGRAM.
C--NOTE ON SCALING: THIS EXPECTS PRINTER WITH 10 CHARS/INCH, 6 LINES/IN
C--  LAB(16) CONTAINS TITLE (IN WORDS 1-8 ), X-AXIS LABEL (IN  9-12)
C--AND Y-AXIS LABEL (IN  12-16). THE LAST CALL (N=0) DETERMINES LABELS
C
C--  CALLING PROGRAM MUST SET UP LINE(103,60) AND LAB(16)
C
C--NOTE THAT FORMATS ASSUME IWD=103, LNG=60, IF OTHERWISE, ADJUST THEM
   20  FORMAT(8H1   PLOT, 22X,8A10,/,25X,8HX-AXIS: ,4A10,11H,  Y-AXIS: ,
     2    4A10                                                          )
   21  FORMAT(20X,1HI,10(10H....V....X),2H.I)
   22  FORMAT(//,7X,14HPLOT--ON ENTRY,I3,12H WITH SYMBOL,1X,A1,/9X,
     2    52HTHE RANGE OF VALUES EXCEEDED THAT SET ON FIRST ENTRY,/,9X,
     3    31HCURRENT RANGES ARE:  (ABSCISSA),14X,10H(ORDINATE),/,12X,
     4    13HFIRST ENTRY--,4X,4E11.3,/,12X,12HTHIS ENTRY--,5X,4E11.3,///)
   23  FORMAT(//7X,13HPLOT--  ENTRY,I3,8H SYMBOL ,A1,I5,14H POINTS, RANGE
     2    31HS OF ABSCISSA AND ORDINATE ARE:,/29X,4E11.3)
   24  FORMAT(20X,103A1)
   25  FORMAT(5X,1PE13.4,2X,103A1)
   26  FORMAT(13X,11(1X,1PE9.1),///)
C
      DIMENSION X(N),Y(N),LINE(103,60),LAB(16),ZX(11)
      DATA IBLNK,IBORD,IWD,LNG,NTR/1H ,1HI,103,58,0/
CW   7/27/81  REPLACED DATA ON I/O WITH NEXT LINE--RM PROBLEMS
      IOUT1=6
C
      NTR=NTR + 1
      IF(NTR.EQ.1)NN=IABS(N)
C--  MAX-MIN SECTION--FIRST ENTRY ONLY THE RANGE IS SET FOR X AND Y
      XS1=X(1)
      XL1=X(1)
      YS1=Y(1)
      YL1=Y(1)
      DO 31 J=2,NN
        XS1=AMIN1(XS1,X(J))
        XL1=AMAX1(XL1,X(J))
        YS1=AMIN1(YS1,Y(J))
   31   YL1=AMAX1(YL1,Y(J))
      IF(NTR.GT.1)GOTO 34
        XS=XS1
        XL=XL1
        YS=YS1
        YL=YL1
```

```
      GOTO 39
C--NOT THE FIRST ENTRY, SO CHECK RANGES
60   34 IF(XS1.LT.XS) GOTO 35
      IF(XL1.GT.XL) GOTO 35
      IF(YS1.LT.YS) GOTO 35
      IF(YL1.GT.YL) GOTO 35
      GOTO 39
65 C--OUT OF RANGE--WRITE MESSAGE AND SET A FLAG WITH  NTR
   35 WRITE(IOUT1,22) NTR,LB,XS,XL,YS,YL,XS1,XL1,YS1,YL1
      NTR= - NTR
      GOTO 55
C--WRITE MESSAGE ABOUT THIS (GOOD) ENTRY
70   39 WRITE(IOUT1,23) NTR,LB,N,XS1,XL1,YS1,YL1
      IF(NTR.NE.1) GOTO 55
C--DONE WITH RANGING.  NOW SCALING ON FIRST ENTRY ONLY
      XSCALE=(XL-XS)/(FLOAT(IWD-2))
      YSCALE=(YL-YS)/(FLOAT(LNG-1))
75    DO 70 J=1,11
   70   ZX(J)=XS + FLOAT(J-1)*XSCALE* 10.
C--BLANK THE PLOT ARRAY : LINE  ON THE FIRST ENTRY
      DO 33 J=1,IWD
         DO 33 K=1,LNG
80   33      LINE(J,K)=IBLNK
C--BORDERS--LEFT AND RIGHT
      DO 44 J=1,LNG
        LINE(1,J)=IBORD
   44   LINE(IWD,J)=IBORD
85   55 CONTINUE
C--  FILL IN ARRAY LINE
      DO 57 I=1,NN
      IX=(X(I)-XS)/XSCALE+1.5
        IF(X(I).LT.XS)IX=1
90      IF(X(I).GT.XL)IX=IWD
      IY=(Y(I)-YS)/YSCALE + .5
      IF(Y(I).LT.YS)IY=0
        IF(Y(I).GT.YL)IY=LNG -1
      IY=LNG - IY
95   57 LINE(IX,IY)=LB
C--IF NOT THE LAST GRAPH, GO BACK FOR MORE
      IF(N.GT.0)GOTO 91
C--MUST BE PLOTIME -- PRINT TITLE, AXES LABELS, AND TOP BORDER
      WRITE(IOUT1,20)LAB
100   WRITE(IOUT1,21)
C--PRINT Y-VALUES AND PLOT
      YVAL=YL + YSCALE
      DO 73 J=1,LNG,2
        YVAL=YVAL - YSCALE
105     WRITE(IOUT1,25) YVAL,(LINE(L,J),L=1,IWD)
      YVAL=YVAL - YSCALE
   73   WRITE(IOUT1,24)  (LINE(L,J+1),L=1,IWD)
      WRITE(IOUT1,21)
      WRITE(IOUT1,26)  (ZX(L),L=1,11)
110 C--CLEAN UP THIS MESS AND RETURN
   91 NTR=IABS(NTR)
C--IF THIS IS THE LAST GRAPH, RE-INITIALIZE
      IF(N.LT.1)NTR=0
      RETURN
115   END
```

```
      SUBROUTINE LNPLT
CW   7/24/81   LNPLT MODIFIED FOR PLASIM; NOW NO FORMAL PARAMETERS
C  LNPLT PREPARES ARRAYS X AND Y FOR PLOTTING VIA PLOTW
C  N IS NUMBER OF ENTRIES IN X AND Y ON FIRST ENTRY
C  N=0 SIGNALS LAST ENTRY. FOR ONLY 1 ENTRY, ENTER WITH -N
      DIMENSION LINE(103,60),LAB(16),X(151),Z(151)
CW    7/24/81  CHANGED CODE FOR PLASIM PLOTTING. ---STP =INCREMENTS
CW   7/24/81   BLANK COMMON AND LABELED COMMON FROM READER
      COMMON ZION(41,151),XION(41,151),VELTZ(151),VELTX(151)
     1,NIP(41),DN(41,151),DNI(42),ISTAT(41)
      COMMON/IO/ IN,IOUT,INFO(14),KEY,ICLPLT,ICLWRT,ITITL(28)
     2   ,IPATHS,IW,IF1(2),IF2(4),ICLERR
      COMMON/PARAM/ N,NUMION,NUMIT,RB,RBOUND,RT,TELOUT,BMCUR,UTIL,
     3     TELIN,THRLEN,UMSION,VELBOH,ZBOUND,IL,IR,PI,BK,Q,RB95N,
     4     DNOB,CEXSEC,TTHNEU,TIME,TIMEMU,XVELMU,ZVELMU,NSTAG,
     5     NSTGMU,NTOTST,PIOV2
      DATA NTR,IONSTP,ITRSTP/0,2,2/
      NTR=NTR+1
      IF(NTR.GT.1)GOTO 35
CW    7/24/81    PUT LABELS IN LAB FROM ITITL
      IW2=2*IW
      DO 33 J=1,IW2
   33 LAB(J)=ITITL(J)
   35 CONTINUE
CW   7/24/81   SET UP DUMMY X,Z FOR PLOTW USING XION,ZION
CW   7/24/81   IF OTHER PLOTS DESIRED, CHANGE NEXT LINES
      NIT=NUMIT+1
CW  7/28/81  AD HOC SET UP MAXMIN FOR PLOTW
      DO 39 ION=1,NUMION
        NITR=NIP(ION)-(NSTAG-1)*(NUMIT-10)
        X(1) = XION (ION,1)
        Z(1) = ZION (ION,1)
        X(NIT) = X(1)
        Z(NIT) = Z(1)
          DO 39 ITR=2,NITR
          X(1) = AMIN1(X(1),XION(ION,ITR))
          Z(1) = AMIN1(Z(1),ZION(ION,ITR))
          X(NIT)=AMAX1(X(NIT),XION(ION,ITR))
          Z(NIT)=AMAX1(Z(NIT),ZION(ION,ITR))
   39 CONTINUE
      LB=1H
      CALL PLOTW(Z,X,NIT,LB,LINE,LAB)
      DO 66 ION=2,NUMION,IONSTP
         DO 44 J=1,NIT
          X(J)=0.0
   44     Z(J)=0.0
      NITR=NIP(ION)
      LB=SHIFT(ION,+54)
      DO 55 ITR=1,NITR,ITRSTP
      X(ITR)=XION(ION,ITR)
   55 Z(ITR)=ZION(ION,ITR)
CW   7/24/81   IF LAST ENTRY TO PLOTW, TELL IT SO WITH N=0
      IF(ION.GE.NUMION)NIT=0
   66 CALL PLOTW(Z,X,NIT,LB,LINE,LAB)
CW   7/24/81   FINISHED WITH ONE PLOT, RESET FOR NEXT ONE
      NTR=0
      RETURN
```

```
      END
```

CARD NR.	SEVERITY	DETAILS	DIAGNOSIS OF PROBLEM
41	ANSI		HOLLERITH CONSTANT APPEARS OTHER THAN IN AN ARGUMENT LIST OF A CALL STATEMENT OR IN A DATA STATEMENT.