**DART: A SIMULATION CODE FOR A DIRECT ENERGY
CONVERTER FOR FUSION REACTORS**

Barry C. Howard
William L. Barr
Ralph W. Moir

May 13, 1974

Prepared for U.S. Atomic Energy Commission under contract No. W-7405-Eng-48

**LAWRENCE
LIVERMORE
LABORATORY**

University of California/Livermore

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

NOTICE

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

Printed in the United States of America
Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy \$ ____*; Microfiche \$0.95

<u>* Pages</u>	<u>NTIS Selling Price</u>
1-50	\$4.00
51-150	\$5.45
151-325	\$7.60
326-500	\$10.60
501-1000	\$13.60

LAWRENCE LIVERMORE LABORATORY
University of California, Livermore, California, 94550

UCRL - 51557

**DART: A SIMULATION CODE FOR A DIRECT ENERGY
CONVERTER FOR FUSION REACTORS**

Barry C. Howard
William L. Barr
Ralph W. Moir

MS. date: May 13, 1974

NOTICE

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

MAS 12K

24

Contents

Abstract	1
Introduction	1
Description of Simulation Method	2
Solving the Equation of Motion	2
Calculating Magnetic and Electric Forces	4
Establishing Particle Starting Conditions	13
Determining End Points and Efficiencies	16
Acknowledgment	25
References	26
Appendix A: Input for DART	27
Appendix B: Dictionary for CLICHE STORAGE	29
Appendix C: Descriptions of Subroutines	32
Appendix D: Memory Allocation for Potential Tables	35
Appendix E: Listings	36

DART: A SIMULATION CODE FOR A DIRECT ENERGY CONVERTER FOR FUSION REACTORS

Abstract

The DART code simulates the behavior of the charged particles which leak out of a mirror fusion reactor and are electrically recovered in a direct energy converter. The code has been used to design a laboratory-scale direct energy converter. A full-scale reactor converter is now being

designed. We describe the simulation technique, which includes space charge and secondary particle effects, and the method for calculating the efficiency with which the ion energy is converted to electricity. Output examples and listings of DART and several directly related codes are given.

Introduction

A direct energy conversion system is being designed at Lawrence Livermore Laboratory to improve the efficiency of an open-ended controlled thermonuclear reactor. The recovery system, which uses a scheme proposed by Post¹ and discussed further by others,^{2,3} will convert the kinetic energy of the escaping ions directly into electricity. First, the beam of escaping positive ions and electrons passes through a magnetic expander which spreads the beam into a fan, thereby both reducing the density of the beam and directing the ions radially into the collector system (Fig. 1). The electrons are removed from the beam by the separator magnetic field. The collector system uses spatially periodic electrostatic focusing to maintain and slow the collimated beam. Then, when the velocity of the ions has been sufficiently decreased, the focus field drives them into collector electrodes.

The DART code simulates the behavior of the ions in this direct converter system and is used primarily to optimize the design of the collector system. A laboratory-scale, direct energy converter which was designed using DART is now in operation.

In DART, the ion beam is approximated by a finite set of particles, with a distribution of starting conditions. The magnetic field of the expander is defined analytically, while the electric force field is derived from a potential function which has both analytic and tabular regions of definition. Both vacuum and space charge potentials are included.

Each ion trajectory is calculated until termination (usually collection on an electrode), and the efficiency of the energy conversion is then determined. After all particles have been run, the weighted average efficiency of the system is computed.

BLANK PAGE

$$\vec{r}' = \frac{q}{m} \vec{E}' + \frac{q}{m} (\vec{v}' \times \vec{B}'),$$

where the primed quantities are unnormalized and in MKS units.

Expressed in terms of vector components and assuming $E'_y = 0$, this becomes

$$\ddot{x}' = \frac{q}{m} E'_x + \frac{q}{m} (\dot{y}' B_z - \dot{z}' B_y)$$

$$\ddot{y}' = \frac{q}{m} (\dot{z}' B_x - \dot{x}' B_z)$$

$$\ddot{z}' = \frac{q}{m} E'_z + \frac{q}{m} (\dot{x}' B_y - \dot{y}' B_x).$$

Finally, we normalize the primed quantities by L , the length (in metres) of each rod-grid collector cup pair (Fig. 2), so that

$$\vec{r} = \vec{r}'/L \quad \text{and} \quad \vec{E} = L \vec{E}'.$$

The resulting equations are used by the code

$$\ddot{x} = \frac{q}{mL^2} E_x + \frac{q}{m} (\dot{y} B_z - \dot{z} B_y)$$

$$\ddot{y} = \frac{q}{m} (\dot{z} B_x - \dot{x} B_z)$$

$$\ddot{z} = \frac{q}{mL^2} E_z + \frac{q}{m} (\dot{x} B_y - \dot{y} B_x).$$

Fig. 2. Collector system notation.

The integration technique used to solve these equations requires converting to a first-order differential equation system.

Let $Y_1 = x$, $Y_2 = y$, $Y_3 = z$, $Y_4 = \dot{x}$, $Y_5 = \dot{y}$, $Y_6 = \dot{z}$, and $F(\vec{Y}(t)) = \dot{\vec{Y}}(t)$. The system of equations now becomes

$$F_1 = \dot{x} = Y_4$$

$$F_2 = \dot{y} = Y_5$$

$$F_3 = \dot{z} = Y_6$$

$$F_4 = \ddot{x} = \frac{q}{mL^2} E_x + \frac{q}{m} (\dot{y} B_z - \dot{z} B_y)$$

$$F_5 = \ddot{y} = \frac{q}{m} (\dot{z} B_x - \dot{x} B_z)$$

$$F_6 = \ddot{z} = \frac{q}{mL^2} E_z + \frac{q}{m} (\dot{x} B_y - \dot{y} B_x).$$

This system is solved using the Adams methods, which are contained in the GEAR package.⁴ The general form of the Adams method equation of order p is

$$\vec{Y}_n = \vec{Y}_{n-1} + h \sum_{j=0}^{p-1} \beta_j \dot{\vec{Y}}_{n-j}, \quad \beta_0 \neq 0 \quad (1)$$

where $\vec{Y}_k$ is an approximation to $\vec{Y}(t_k)$, $\dot{\vec{Y}}_k$ is an approximation to $\dot{\vec{Y}}(t_k) = F(\vec{Y}(t_k))$, and h is the step size, $h = t_{k+1} - t_k$. The fact that the order is p means that if the past values are exact, $\vec{Y}_n$ will differ from the exact solution, $\vec{Y}(t_n)$, by a truncation error which is $O(h^{p+1})$ for small h. The fact that $\beta_0 \neq 0$ means that Eq. (1) is an implicit equation for $\vec{Y}_n$ which must be solved at every step.

If Eq. (1) is rewritten in the form

$$\vec{G}(\vec{Y}_n) \equiv \vec{Y}_n - \vec{Y}_{n-1} + h\beta_0 \dot{\vec{Y}}_n$$

$$+ h \sum_{j=1}^{p-1} \beta_j \dot{\vec{Y}}_{n-j}$$

then the system $\vec{G}(\vec{Y}_n) = \vec{Y}_n$ is solved using the functional iteration

$$\vec{Y}_{n(m+1)} = \vec{G}(\vec{Y}_{n(m)}),$$

where (m) denotes the iteration number. The convergence of the iterates and the truncation error⁵ of the final iterate are both tested using an input error bound, EPS (see Appendix A). The Adams method varies both the order, p, and the step size, h, of Eq. (1) to achieve the optimum step size for the required error controls.

CALCULATING MAGNETIC AND ELECTRIC FORCES

Magnetic Field in the Expander

The magnetic field is defined analytically in cylindrical coordinates as $\vec{B} = (B_r, B_\theta, B_z)$ (Fig. 3). For $r_{\text{mirror}} \leq r < r_1$, the radial component, B_r , is hyperbolic with $B_r(r_{\text{mirror}}) = B_{\text{mirror}}$ and $B_r(r_1) = B_{r_1}$ (Fig. 4); i. e.,

$$B_r = \frac{a}{r - r_{\text{center}}}$$

where

$$a = B_{\text{mirror}} (r_{\text{mirror}} - r_{\text{center}})$$

and

$$r_{\text{center}} = \frac{B_{\text{mirror}} r_{\text{mirror}} - B_{r_1} r_1}{B_{\text{mirror}} - B_{r_1}}.$$

Fig. 3. Expander-collector system (top view).

Fig. 4. Sketch of B_r function.

The component B_θ is assumed to be zero, and B_z is chosen to satisfy Maxwell's divergence equation

$$\nabla \cdot \vec{B} = \frac{1}{r} \frac{\partial}{\partial r} r B_r + \frac{1}{r} \frac{\partial}{\partial \theta} B_\theta + \frac{\partial}{\partial z} B_z = 0,$$

where $B_\theta = 0$. The components are thus defined for the interval, $r_{\text{mirror}} \leq r < r_1$, as

$$B_r = B_{\text{mirror}} \frac{r_{\text{mirror}} - r_{\text{center}}}{r - r_{\text{center}}}$$

$$B_\theta = 0$$

$$B_z = z \frac{a r_{\text{center}}}{r(r - r_{\text{center}})^2}.$$

Maxwell's other equation,

$$\nabla \times \vec{B} = 0$$

is not satisfied, since

$$\begin{aligned} \nabla \times \vec{B} &= -\frac{\partial B_z}{\partial r} \\ &= z a r_{\text{center}} \left[\frac{3r - r_{\text{center}}}{r^2(r - r_{\text{center}})^3} \right] \neq 0, \end{aligned}$$

but this quantity is small for $r \gg r_{\text{center}}$.

For the interval, $r_1 \leq r \leq r_2$, we assume that B_r drops linearly from B_{r1} to zero, B_θ is zero, and B_z is chosen as before; i. e.,

$$B_r = br + c$$

$$B_\theta = 0$$

$$B_z = -z \frac{2br + c}{r}$$

where

$$b = -\frac{B_{r1}}{r_2 - r_1}, \quad c = \frac{r_2 B_{r1}}{r_2 - r_1}.$$

Again,

$$\nabla \times \vec{B} = -\frac{\partial B_z}{\partial r} \hat{z} = -\frac{z r_2 B_{r1}}{r^2(r_2 - r_1)} \neq 0,$$

but this quantity is small for $z/r^2 \ll 1$.

Since DART uses normalized, rectangular coordinates, a conversion to and from cylindrical coordinates must be used. First, $(x, y, z) \rightarrow (r, \theta, z)$ by

$$r = [(r_2 + 1 + x)^2 + y^2]^{1/2}$$

$$\theta = \sin^{-1}(y/r)$$

$$z = z$$

where 1 is the (normalized) length of the negative stage and $x < -1$. After computing (B_r, B_θ, B_z) , the conversion to (B_x, B_y, B_z) is

$$B_x = B_r \cos \theta$$

$$B_y = B_r \sin \theta$$

$$B_z = B_z$$

These values are then used by the integrator.

Electric Field in the Collector

The electrical field at a particle in the collector is defined as the negative gradient of the potential function, ϕ :

$$\vec{E} = -\nabla \phi.$$

The function ϕ is assumed to be independent of y [$\phi = \phi(x, z)$] and, by the principle of superposition, is the sum of

the vacuum potential, ϕ_v , and the space charge potential, ϕ_q :

$$\phi = \phi_v(x, z) + \phi_q(x, z).$$

The values of ϕ_v are defined analytically for points in the collector channel and are interpolated from tables for points in the negative-stage region and in a collector cup. The function ϕ_q is defined for points in the negative stage and collector channel regions, and its values are interpolated from a table. For points outside their regions of definition, the functions are set to zero.

DART reads the necessary potential tables from disk files before each run (see Appendix D for memory allocation). All table interpolations use the two-dimensional Lagrangian scheme documented in CIC Report E1.1-006,⁶ with the degree of the interpolating polynomial set at 2. Because the potential tables are unitless, the interpolated results must be converted to volts. The conversion factor, V_0 , is the change in vacuum potential at mid-channel ($z = 0$) per distance L in the x -direction.

Vacuum Potential in the Collector Channel

The values of ϕ_v for points in the collector channel region ($1 \leq x \leq XMAX$, $|z| \leq ZFIN$) are given by

$$\phi_v(x, z) = V_0 \left[x + \frac{c}{2\pi} \sin(2\pi x) \sinh(2\pi z) \right],$$

where V_0 and c are input constants.

Vacuum Potential in the Negative-Stage Region

The potential table for the negative stage region ($-1.916667 \leq x < 1$,

$|z| \leq Z_{FIN}$) is generated by the code POT2, which uses a relaxation technique⁷ to satisfy Laplace's equation

$$\nabla^2 \phi = \frac{d^2 \phi}{dx^2} + \frac{d^2 \phi}{dz^2} = 0. \quad (2)$$

The coordinate arrays for the table are normalized by L and defined as follows (Fig. 5):

$$x_i = i\Delta x, \quad i = -23, \dots, 12$$

where $\Delta x = 1/12$

$$z_k = k\Delta z, \quad k = -5, \dots, 5$$

where $\Delta z = Z_{\text{FIN}}/5$.

The table boundary values are held fixed throughout the relaxation process and are defined as follows:

$\underline{i}$	$\underline{k}$	$\phi_{\underline{i}, \underline{k}}$
-23	[-5, 5]	0
[-22, -12]	-5 and 5	0
[-11, -6]	-5 and 5	$\frac{a}{2\pi} \sin(2\pi x_i) \cosh(2\pi z_k)$
[-5, 0]	-5 and 5	0
[1, 6]	-5 and 5	$x_i + \frac{c_i}{2\pi} \sin(2\pi x_i) \sinh(2\pi z_k)$
{7, 11}	-5 and 5	$x_i + \frac{c_i}{2\pi} \sin(2\pi x_i) \sinh(2\pi z_k)$
12	[-5, 5]	$x_i + \frac{c_i}{2\pi} \sin(2\pi x_i) \sinh(2\pi z_k)$

The parameter, c' , which is used when $1 \leq i \leq 6$, differs from the parameter, c , used in the interval $7 \leq i \leq 12$. These parameters must differ in order for the beam to remain centered after it enters the focusing system. Usually $c' = c/2$.

The relaxation formula is derived by using central differences

$$\frac{d^2\phi}{dx^2} = \frac{\phi_{i+1,k} - 2\phi_{i,k} + \phi_{i-1,k}}{(\Delta x)^2}$$

Fig. 5. Potential table mesh for negative stage region.

$$\frac{d^2\phi}{dz^2} = \frac{\phi_{i,k+1} - 2\phi_{i,k} + \phi_{i,k-1}}{(\Delta z)^2}$$

Substituting these into Eq. (2) and solving for $\phi_{i,k}$, we get the relaxation formula

$$\phi_{i,k} = \frac{\phi_{i+1,k} + \phi_{i-1,k} + \left(\frac{\Delta x}{\Delta z}\right)^2 (\phi_{i,k+1} + \phi_{i,k-1})}{2 \left(1 + \left(\frac{\Delta x}{\Delta z}\right)^2\right)} \quad (3)$$

The potentials at the interior points of the table are generated by successive iterations using Eq. (3).

To increase the rate of convergence, each entry in the table interior is "over-relaxed" using

$$\phi_{i,k}^n = \phi_{i,k}^{n-1} + \alpha(\phi_{i,k} - \phi_{i,k}^{n-1}),$$

where $\phi_{i,k}$ is the result from Eq. (3), n is the iteration number, and $\alpha > 1$. The value $\alpha = 1.8$ was found to give the fastest convergence. The code POT2 prints out the table values at two fixed locations after each iteration. We check the convergence of the relaxation by observing if these two sequences of table values each tend toward a constant value.

The peak negative potential in the negative stage ($-12 \leq i \leq -6$) is specified by the user and is assumed to occur at ($i = -10, k = 0$). Using the linear relationship between this peak value and the parameter, a , two complete table generations are made to define the line, and a third pass uses the interpolated value of " a " for the desired peak value.

The coordinate arrays and relaxed table values are stored in a disk file, POTAB2, where they are used by DART. POT2 produces a plot showing the potential contours in the relaxed table (Fig. 6).

Fig. 6. Equipotential contours for negative stage table (x and z are given in units of L).

Vacuum Potential in the Collector Cup

The relaxation formula and convergence test used for the collector cup region ($1 \leq x \leq XMAX, |z| > ZFIN$) are identical to those used for the negative-stage table. The coordinate arrays for the cup table (Fig. 7) are

$$x_i = i\Delta x, \quad i = 0, \dots, 120$$

where $\Delta x = 1/120$

$$z_k = k\Delta z, \quad k = 0, \dots, 117$$

$$\text{where } \Delta z = \frac{COLHT - ZFIN}{116}$$

The cup table represents one rod-grid pair of collector cups and, in addition to the fixed potential at the boundary, has interior fixed potentials along the rod, grid, and divider fin. The relaxation code, POT1, which generates this table sets the (input) fixed potentials $V1, V2, V3$, and $V4$ along the fins and computes the fixed rod, grid, and channel boundary

Circled numbers are class numbers

Fig. 7. Potential table mesh for collector cup region.

potentials to match the analytic contours
of the channel using

$$\begin{aligned}\phi_{\text{rod}} &= VR = \phi(0.25, ZFIN) \\ &= V_0 \left(0.25 + \frac{c}{2\pi} \sinh(2\pi ZFIN) \right)\end{aligned}$$

$\phi(x, ZFIN)$

$$= V_0 \left(x + \frac{c}{2\pi} \sin(2\pi x) \sinh(2\pi ZFIN) \right)$$

$\phi_{\text{grid}} = VG = \phi(0.75, ZFIN)$

$$= V_0 \left(0.75 - \frac{c}{2\pi} \sinh(2\pi ZFIN) \right).$$

The coordinates of the rod and grid points in the cup table are constants in POT1. They must be regenerated any time the table mesh is changed. The coordinates are computed by choosing typical values for c and $ZFIN$, drawing the VR and VG potential contours, and then tracing these contours onto the table mesh. When different c and $ZFIN$ values are input to POT1, the resulting VR and VG contours are approximated by the constant coordinates. The accuracy of this approximation determines how well the contours of the relaxed potentials in the cup pair with the analytic contours in the channel. POT1 supplies a plot to check this match (Fig. 8).

Fig. 8. Equipotential contours for cup table (x and z are given in units of L).

The coordinate arrays and relaxed potentials are saved for DART in a disk file, POTAB1. Using the periodicity of the potential function, DART reduces the coordinates of a particle in any cup to the

dimensions of the cup table ($0 \leq x \leq 1$, $z > ZFIN$), performs the interpolation, and adjusts the result according to the original coordinates.

Space Charge Potential

The space charge potential, ϕ_q , values ($-1.916667 \leq x \leq XMAX$, $|z| \leq ZFIN$) are interpolated from a table which requires two steps to generate. First, the charge density table is produced by DART and second, the space charge table is made using these charge densities and a relaxation technique. Both tables cover the region of the negative stage and collector channel and have the following coordinate arrays (Fig. 9):

$$x_i = i\Delta x, i = -23, \dots, 132$$

where $\Delta x = 1/12$, and $XMAX = 11$

$$z_k = k\Delta z, k = -5, \dots, 5$$

where $\Delta z = ZFIN/5$.

To produce the charge density table, we let the total beam current per unit length along the collectors (or per circumferential length) be I , with the distribution, dI/dW , expressed in amperes per unit of energy per unit length.

$$I = \int_{W_{\min}}^{W_{\max}} \frac{dI}{dW} dW$$

Fig. 9. Mesh for charge density and space charge potential tables.

If we simulate the beam with N_S particles (line charges) at each of N_W energies, then the current, I_j , carried by the j^{th} particle of energy, W_m , is

$$I_j = \frac{1}{N_S} \int_{W_m - \frac{1}{2}\Delta W_m}^{W_m + \frac{1}{2}\Delta W_m} \frac{dI}{dW} dW$$

$$\approx \frac{1}{N_S} \frac{dI}{dW}(W_m) \Delta W_m,$$

where

$$\sum_{m=1}^{N_W-1} \Delta W_m = W_{\max} - W_{\min}.$$

To save computation time, we assume the "top hat" distribution

$$\frac{dI}{dW}(W_m) = \frac{I}{W_{\max} - W_{\min}},$$

$$m = 1, \dots, N_W.$$

This assumption is justified by the assertion that a more realistic distribution could be compensated for by adjusting the potential increment, V_0 , in the channel.⁸ With this assumption, we have

$$I_j = \frac{1}{N_S} \frac{I}{W_{\max} - W_{\min}} \frac{W_{\max} - W_{\min}}{(N_W - 1)}$$

$$= \frac{I}{N_S(N_W - 1)},$$

where we've chosen equal increments

$$\Delta W = \frac{W_{\max} - W_{\min}}{(N_W - 1)}.$$

If the power of the beam is P (W/m), and the mean energy is $\bar{W}$ (eV), then the beam current, I (A/m), is

$$I = P/\bar{W}.$$

Hence, we have

$$I_j = \frac{P/W}{N_S(N_W - 1)} \quad (4)$$

where I_j is independent of energy.

This current, I_j (A/m), is carried by particles of charge q_d (C/m) which are released at intervals of Δt (s); that is,

$$I_j = \frac{q_d}{\Delta t}$$

If all the N_S particles of energy W_m are launched at time zero, and again at intervals of Δt , then a "snapshot" picture would show the number of particles in each mesh zone of the table, i.e., the charge density. The same picture is produced by launching each of the N_S particles only once, taking snapshot pictures at intervals of Δt , and superimposing each of the pictures.⁹ Following each particle just once saves computation time, but gives only a steady-state, self-consistent behavior. In order to study self-consistent, time-dependent behavior such as flow instabilities, the particles must be followed simultaneously and Poisson's equation must be solved at each interval Δt . The enormous amount of computer time required does not allow us to do this.

The number of discrete particles must be large enough and the snapshot interval, Δt , small enough to obtain smooth density profiles. In order to keep Δt small without affecting the integration step size, h , DART linearly interpolates between successive points of the trajectory. Then at each multiple of Δt , the code calculates the interpolated location of the particle and distributes q_d to the four surrounding mesh points in the charge density table.

The technique shown below for weighting the four mesh points was developed by C. K. Birdsall.¹⁰

$$\rho_{i,k} = \rho[(x_{i+1} - x)(z_{k+1} - z)/(\Delta x \Delta z)]$$

$$\rho_{i+1,k} = \rho[(x - x_i)(z_{k+1} - z)/(\Delta x \Delta z)]$$

$$\rho_{i+1,k+1} = \rho[(x - x_i)(z - z_k)/(\Delta x \Delta z)]$$

$$\rho_{i,k+1} = \rho[(x_{i+1} - x)(z - z_k)/(\Delta x \Delta z)]$$

where

$$\rho = \frac{q_d}{(\Delta x \Delta z)}$$

is the charge density (C/m) contributed by the particle to the table at each multiple of Δt .

The charge density table is initialized to zero, and the charge density is then accumulated at each table mesh point. During these runs, DART stops each ion as it leaves the table region. This saves computer time, but eliminates the contribution made to the table by ions which reenter the channel from a cup. At the end of the run, the coordinate arrays and table values are stored in a disk file, QDTAB.

The code POT3 uses the charge densities and a relaxation technique to produce the space charge potential table. After the charge density file, QDTAB, has been

read, the potential table is initialized to zero. Then, with the table boundary values held fixed at zero, the interior values are generated from the charge density values by the relaxation technique.

The relaxation formula is derived from Poisson's equation

$$\nabla^2 \phi = \frac{d^2 \phi}{L^2 dx^2} + \frac{d^2 \phi}{L^2 dz^2} = - \frac{\rho}{L \epsilon_0},$$

where ρ is charge density (C/m) and $\epsilon_0 = 8.85 \times 10^{-12}$ C/m · V. Using central differences and solving for $\phi_{i,k}$ we get the relaxation formula

$$\phi_{i,k} = \frac{\frac{\rho_{i,k}(\Delta x)^2}{\epsilon_0} + \phi_{i+1,k} + \phi_{i-1,k} + \left(\frac{\Delta x}{\Delta z}\right)^2 (\phi_{i,k+1} + \phi_{i,k-1})}{2 \left(1 + \left(\frac{\Delta x}{\Delta z}\right)^2\right)}.$$

Over-relaxation is used ($\alpha = 1.8$) to speed convergence, and the coordinate arrays and relaxed table values are stored in a disk file, POTAB3. POT3 produces a contour plot of the relaxed space charge potentials (Fig. 10).

The following steps are involved in a space charge run:

- (a) Run DART using vacuum potentials only to generate a charge density table.
- (b) Run POT3 to produce a space charge potential table.
- (c) Run DART using the space charge potential table to generate a new charge density table.
- (d) Run POT3 to create a new space charge potential table.
- (e) Return to (c).

When the space charge potential values from successive passes through the loop (c) - (e) appear stable (usually two or

three passes suffice), a final run (f) is made using the space charge potentials. This last run does not generate a new charge density table, but completes all trajectories and computes weighted average efficiencies.

The program, DQC, acts as a controller for the codes DART and POT3 and executes the procedure (a) - (f) using a specified number of passes through the loop (c) - (e). This reduces the amount of user interaction required to make a space charge run.

Matching the Expander and Collector Geometries

Because the collector system is simulated in rectangular, not cylindrical, coordinates, the expander and collector structures are tangent only along the line (Fig. 3)

$$x = -1, \quad y = 0, \quad z = z.$$

Any particle crossing the expander-collector interface must do so at this line of tangency. This requirement is met in DART by shifting the y-coordinate of the particle back to zero each time the particle crosses the interface, while leaving the other coordinates unaltered. For convenience, DART does the shifting at $x = 0$ (not $x = -1$), and this forces the y-coordinate to be sufficiently small at the interface.

ESTABLISHING PARTICLE STARTING CONDITIONS

Several assumptions must be made in order to simulate the ion beam escaping from the mirror machine. The flux tube at the reactor mirror ($r = r_{\text{mirror}}$) is assumed to be circular. Also, because

Fig. 10. Equipotential contours for space charge table (x and z are given in units of L).

the direct converter is assumed to have infinite dimension in the y -direction, concern is given only to representing the extremes of the beam in the z -direction. This is done using three guiding center flux lines: two for the extrema, and one on the axis (Fig. 11a). The (circular) beam diameter at the mirror is given by

$$D_{\text{mirror}} = 2(d + a_{\text{mirror}})$$

where d is the distance between the guiding center lines, and a_{mirror} is the Larmor radius of a particle at the mirror.

To conserve computation time, the particles are started at $r = r_1$ by projecting the three guiding center orbits through the expander. If $R = B_{\text{mirror}}/B_{r_1}$

is the total expansion ratio, then the Larmor radius at $r = r_1$ is $a_{r_1} = \sqrt{R} \times a_{\text{mirror}}$, and the (elliptical) beam thickness is

$$D_{r_1} = 2(d + a_{r_1}) = 2(d + \sqrt{R} a_{\text{mirror}}).$$

By choosing $r_{\text{mirror}}/r_1 = 1/R$ so that $R_z = R(r_{\text{mirror}}/r_1) = 1$, we ensure that the total beam expansion is in the y -direction, $R = R_y$ (expander has constant height). Hence, d is a constant, independent of r .

Another assumption is made: 90% of the total (kinetic) energy of the particle is perpendicular to the x -axis at the mirror, or

Fig. 11. Projection of the three guiding center orbits through the expander.

$$(v_{\perp \text{mirror}})^2 = 0.9 v^2.$$

By the conservation of magnetic moment,

$$\frac{(v_{\perp r_1})^2}{B_{r_1}} = \frac{(v_{\perp \text{mirror}})^2}{B_{\text{mirror}}}$$

or,

$$(v_{\perp r_1})^2 = (v_{\perp \text{mirror}})^2 \left(\frac{B_{r_1}}{B_{\text{mirror}}} \right)$$

$$= 0.9 v^2 \left(\frac{1}{R} \right)$$

so,

$$v_{\perp r_1} = \frac{3v}{\sqrt{10R}}.$$

For each of the initial energies, W_m (keV), in the interval $[W_{\min}, W_{\max}]$, N_θ particles are started on each of the three guiding center orbits (Fig. 11b)

$$x_0 = r_1 - (r_2 + L)$$

$$y_0 = a_{r_1} \cos \theta$$

$$z_0 = b + a_{r_1} \sin \theta$$

where

$$a_{r_1} = \frac{mv_{\perp r_1}}{q B_{r_1}} = \frac{3v}{B_{r_1} \sqrt{10R} q/m}$$

$$v = \sqrt{2W_m/m},$$

q/m is the charge-to-mass ratio,
 $0 \leq \theta \leq 2\pi$, and $b = d, 0$, and $-d$.

By the conservation of energy,

$$\begin{aligned} v_x^2 &= v^2 - \left(v_{\perp r_1} \right)^2 = v^2 - 0.9 v^2 \left(\frac{1}{R} \right) \\ &= v^2 \left(1 - \frac{0.9}{R} \right). \end{aligned}$$

The velocity components corresponding to each of the N_θ starting positions at energy W_m are

$$\begin{aligned} v_x &= \dot{x}_0 = v \left(1 - \frac{0.9}{R} \right)^{1/2} \\ v_y &= \dot{y}_0 = \frac{3v}{\sqrt{10R}} \sin \theta \\ v_z &= \dot{z}_0 = - \frac{3v}{\sqrt{10R}} \cos \theta. \end{aligned}$$

The initial slope of the trajectory is defined as

$$\text{slope} = \arctan (v_z/v_x).$$

The total number of particles started at each energy, W_m , is $N_S = 3N_\theta$. Note that the starting positions and velocities are defined in MKS units; the code normalizes them by L before computing the trajectories.

DETERMINING END POINTS AND EFFICIENCIES

Conditions for Termination of Trajectory

Four conditions cause the termination of a trajectory calculation:

- If the integration routine is unable to compute a point using the smallest allowable time step ($H = HMIN$) so that the local relative error is permissible ($< EPS$),

- If the integration time exceeds the maximum set by the user ($T > TMAX$),
- If, when generating a charge density table, the particle leaves the table region, and finally, the most common cause of termination,
- If the trajectory is detected crossing a physical boundary.

DART first checks if the trajectory crossed a boundary in the expander or negative stage. If a particle returns (retrogrades) from the collector to the expander, an "impact" may occur with the top or bottom of the expander ($|z| > EXHT$). Also, a retrograde particle may fail to be reflected in the magnetic field of the expander, thereby intersecting the mirror ($r < r_{\text{mirror}}$). In this case, however, the trajectory calculation continues, and the particle is "reflected" by reversing the sign of $\dot{x}$. In the negative stage region of the collector ($-1 \leq x \leq 1$), impact occurs when the particle strikes the channel wall ($|z| > ZFIN$).

Impact in the collector region ($x > 1$) can happen either at the end of the collector channel ($x > XMAX$) or in a collector cup ($|z| > ZFIN$). In a cup, the impact areas are the cup fins, the back and sides of the cup, and the rod. The surface of the rod is defined as the analytic potential contour which is used to set the coordinates of the rod points in the cup table. DART translates the contour so that it is tangent to the channel boundary ($z = ZFIN$). The grid is similarly defined, but is considered transparent for impact purposes.

Calculations Following Termination

Unless the particle intersects the mirror, the time of each impact is computed

by performing linear extrapolation on the two integration points immediately preceding the hit. Starting at the last point before impact, small steps are taken along the extrapolated line until impact occurs again. The point just before this impact is used for the starting point for even smaller steps until a third impact is detected. The time of the step preceding this impact is defined as the impact time, and the corresponding position and velocity values are used as the final coordinates of the trajectory.

DART has two checks to insure the accuracy of the final coordinates. First, the extrapolated impact time may not exceed the time of the original impact. Second, the distances from the last integration point before impact to both the original impact point and the extrapolated final point are computed. If the latter distance is more than twice the former, the particle is thrown out.

The efficiency of collection is calculated for each ion as follows:

$$\eta_{\text{ion}} = T^N \frac{\phi(x_e, z_e)}{W_0} + \sum_{i=1}^N T^{i-1} (1 - T) \frac{\phi(x_i, z_i)}{W_0} \quad (5)$$

where

T is the transmission coefficient of the grids

N is the number of times the ion passed through grids

(x_e, z_e) is the final point of the trajectory

(x_i, z_i) is the point at which the ion passed through its i^{th} grid

W_0 is the initial (kinetic) energy of the ion.

The relative error in the total energy of the particle is also computed using

$$W_{\text{err}} = \left| \frac{W_e + \phi(x_e, z_e) - W_0}{W_0} \right|$$

where W_e is the final kinetic energy of the ion. If this error is greater than the limit set by the user ($W_{\text{err}} > \text{ERRMX}$), then the calculation of the trajectory is repeated using a smaller local error limit. DART prints the total number of these repeats at the end of the run. If the energy error check is satisfied, but $\eta_{\text{ion}} > 1$, then η_{ion} is set to unity.

To aid in the optimization of the collector cup design, the possible collection surfaces of the cup are divided into classes and each division is assigned a class number. After termination, DART classifies each ion by its end point. If the particle either impacted other than in a cup, or stopped before impact (e.g., if the integration time limit has been exceeded), then it is assigned a zero class number.

Secondary Particles

DART provides the option of computing, for each ion, secondary trajectories and their resulting (negative) efficiencies. If the option is used, DART starts a negative particle $[(q/m)_{\text{sec}} = -(q/m)_{\text{ion}}]$ at each point the ion crossed a grid and at the final point of the ion trajectory.

The initial position (x_i, z_i) of a secondary starting from a grid is found by a linear extrapolation back along the ion trajectory until a rising potential is found. This insures that the secondary will start on the edge of the grid. A secondary is begun at the final point (x_e, z_e) of an ion

trajectory only if the ion terminated in a collector cup ($1 \leq x \leq XMAX$, $|z| > ZFIN$). All secondaries are given the same initial energy and are aimed back along the trajectory of the incident ion so that

initial slope of secondary
= incident angle of ion + π .

After all the secondary particles of an ion have been followed, the efficiency lost by the collector system to the secondaries is calculated. DART first finds the secondary emission coefficient, γ , for each secondary using

$$\gamma = 0.1 + \frac{\log_{10} (W/2)}{1.13} \quad \text{for } W > 2$$

$$= 0.1 \quad \text{for } W \leq 2$$

where W is the kinetic energy (keV) of the incident ion at the point of emission. Then the secondary efficiency for the ion is found using

$$\eta_{\text{sec}} = T^N \gamma \frac{W_{\text{es}}}{W_o} + \sum_{i=1}^N T^{i-1} (1 - T) \gamma_i \frac{W_{\text{esi}}}{W_o}$$

where

T is the transmission coefficient of the grids

N is the number of times the ion passed through grids

γ is the secondary emission coefficient for the secondary starting at (x_e, z_e)

γ_i is the secondary emission coefficient for the secondary starting at (x_i, z_i)

W_{es} is the final kinetic energy of the secondary starting at (x_e, z_e)

W_{esi} is the final kinetic energy of the secondary starting at (x_i, z_i)

W_o is the initial kinetic energy of the ion.

If DART is unable to determine the initial position of a secondary, or if the secondary trajectory ends prematurely (e.g., if there are integration problems), then the secondary is excluded from η_{sec} .

Printing and Plotting Trajectories

DART has print and plot options which are useful for optimization of the collector system design. The print option allows the user to print the points of each trajectory at a frequency which he specifies. Each point is in MKS units and consists of the time, position and velocity components, potential energy, and total energy of the particle (Fig. 12).

The plot option saves and plots the final 500 points of each trajectory. These points are all normalized by L , are all in the collector region, and are plotted in groups of five (Fig. 13). Each ion is identified by its number and initial energy. If secondary particles are followed, the trace is a dotted line.

Calculation of Averages for Entire Distribution

Each ion is assigned a 3-digit status number which provides information about its trajectory. The digits, $S_1 S_2 S_3$, are defined as follows:

S_1 = the number of times the particle intersected the mirror at the entrance to the expander (and was reflected by changing the sign of its x-velocity component).

INPUT DATA FOR DART CODE

```

RMIROR      R1      R2      EXHT      BMIROR      BR1
0.7688      230.6000  232.6000  2.0000  15.0000  0.0500

BOUNC      ZFIN      COLHT      XMAX      XL      C      QM      VO
4.166670E-03 4.200000E-01 1.358785E+00 1.100000E+01 2.000000E+00 2.500000E+00 4.792400E+07 1.000000E+02

HBEG      HMIN      HMAX      TBEG      TMAX      EPS      FREPS      ERRMX
1.000000E-09 5.000000E-15 1.000000E-06 0. 1.000000E-02 5.000000E-04 1.000000E-01 1.000000E-01

SSDT      QD      TCUG
1.000000E-09 7.122400E-12 9.700000E-01

NTRAJ=      1      NOFSET=      0      IPRINT=      15      IPLCT=      0      ISC=      0      IGCD=      0      ISPAR=      0

ILIM1=      121      KLIM1=      118      ILIM2=      36      KLIM2=      11      ILIM3=      156      KLIM3=      11
    
```

RESULTS FOR TRAJECTORY NO. 1

AT START -

```

X=-4.000000E+00  Y= 7.07667E-02  Z= 2.82572E-01  VX= 6.18257E+06  VY= 2.93706E+05  VZ=-1.69571E+05
ENERGY= 4.00000E+02  SLOPE=-2.74204E-02
    
```

T	X	Y	Z	VX	VY	VZ	E(KINETIC)	E(TOTAL)
3.17374E-07	-2.03291E+00	5.08096E-02	2.18777E-01	6.22585E+06	-3.72020E+05	-2.01348E+05	4.06270E+02	3.98944E+02
3.22869E-07	-1.99869E+00	4.87388E-02	2.17664E-01	6.22808E+06	-3.80621E+05	-2.01190E+05	4.06626E+02	3.98944E+02
4.82979E-07	-9.98671E-01	-1.22027E-02	1.88167E-01	6.22437E+06	-3.60621E+05	-1.67443E+05	4.06014E+02	4.00766E+02
7.17970E-07	4.31564E-01	-2.40643E-02	1.32224E-01	5.88263E+06	-3.80621E+05	-4.64546E+05	3.64807E+02	4.02020E+02
9.93683E-07	1.59665E+00	-1.29006E-01	4.52378E-02	5.37239E+06	-3.80621E+05	2.04990E+05	3.03078E+02	4.02851E+02
1.34963E-06	3.74495E+00	-2.64487E-01	-3.17681E-02	4.49744E+06	-3.80621E+05	1.47051E+05	2.12769E+02	4.02874E+02
1.87473E-06	5.75345E+00	-4.64350E-01	-1.13740E-01	3.12398E+06	-3.80621E+05	4.05030E+05	1.05043E+02	4.02872E+02
2.56771E-06	7.33009E+00	-7.28113E-01	-2.44945E-01	7.35285E+05	-3.80621E+05	1.32280E+05	7.33468E+00	4.02877E+02
3.09840E-06	7.33134E+00	-9.30106E-01	7.73468E-01	4.55981E+05	-3.80621E+05	4.65656E+06	2.29909E+02	4.03012E+02
3.17820E-06	7.38261E+00	-9.60480E-01	1.17992E+00	5.45084E+05	-3.80621E+05	4.83768E+06	2.48780E+02	4.08153E+02
3.61743E-06	7.65165E+00	-1.12766E+00	2.22622E+00	5.61362E+05	-3.80621E+05	1.23280E+06	2.06555E+01	4.04430E+02

HIT

AT END -

```

X= 7.84705E+00  Y=-1.30087E+00  Z= 2.71743E+00  VX= 3.60237E+05  VY=-3.80621E+05  VZ= 1.01921E+06
TIME= 4.07250E-06  ENERGY(KINETIC)= 1.37033E+01  ENERGY(ERRCH)= 0.00926  ETA= 0.94679
    
```

Fig. 12. Printed integration steps.

SUMMARY OF PARTICLE EFFICIENCIES

NO.	X	Y	Z	ENERGY	SLCPE	ST	PRIMARY		SECONDARY		
							CL	ETA	NO.	ZST	ETA
1	-4.0000	0.0708	0.2826	400.0000	-0.0274	1	3	0.94679	0	0	0.

DISTRIBUTION 1

WEIGHTED AVERAGE EFFICIENCY*		COLLECTED	RETROGRADE	TOTAL	SECONDARY	NET	REFLECTED
FRACTION OF TOTAL PARTICLES*		0.94679	0.	0.94679	0.	0.94679	0.
		1.00000	0.	1.00000			0.

PARTICLE CLASS*	1	2	3	4	5	6	7	8	9	10
FRACTION IN CLASS*	0.	0.	1.00000	0.	0.	0.	0.	0.	0.	0.
AVERAGE ENERGY*	0.	0.	21.28256	0.	0.	0.	0.	0.	0.	0.

TIME PERCENTS*	034*63*02	TIME TOTAL(SEC)=	0.5097	TIME PER PARTICLE(SEC)=	0.5097	NO. REPEATS=	0
----------------	-----------	------------------	--------	-------------------------	--------	--------------	---

Fig. 12. (Continued).

Fig. 13. Plot of trajectories (x and z are given in units of L).

S_2 = the number of times the particle escaped from the collector back into the expander.

S_3 = 1 + the number of times the particle escaped from a collector cup.

Those ions whose trajectories were terminated prematurely (e.g., because of time-limit or integration problems) and those whose final coordinates fail the accuracy check are given status number zero and are excluded from all final averages.

If secondary particles are followed, they are also each assigned a status num-

ber. A zero status number is given to any secondary whose initial position could not be computed.

After all trajectories have been computed, the ions are sorted by status number into the following three groups:

Group	Status numbers
1) Collected	1 - 9
2) Retrograde	11 - 99
3) Reflected	111 - 999

For a given distribution, dI/dW , of beam current, the weights $(dI/dW_i)W_i$ are used

as follows to compute a weighted average efficiency for each status group:

$$\bar{\eta}_j = \frac{\text{power recovered}}{\text{power input}}$$

$$= \frac{\sum_{i,j} \left(\frac{dI}{dW_{i,j}} \right) W_{i,j} \eta_{i,j}}{\sum_i \left(\frac{dI}{dW_i} \right) W_i},$$

$j = 1, 2, 3. \quad (6)$

For each group, j , the numerator of Eq. (6) includes only those weights and efficiencies assigned to ions in that group. The W 's are the initial energies of the ions.

The fraction of the total number of particles contained in each status group is also computed. The weighted average efficiencies and population fractions of status groups 1 and 2 are summed to give the total numbers. The ions in these two groups are then sorted by class number, and the average final kinetic energy for each class is found. DART also computes the fraction of the total number of particles in each class.

The code calculates the weighted average, $\bar{\eta}_{\text{sec}}$, of the secondary efficiencies,

using the weights described in Eq. (6). (If secondaries are not followed, $\bar{\eta}_{\text{sec}} = 0$.) This average is subtracted from the total average ion efficiency, $\bar{\eta}_{\text{total}}$, to produce the net efficiency

$$\bar{\eta}_{\text{net}} = \bar{\eta}_{\text{total}} - \bar{\eta}_{\text{sec}}.$$

DART prints a summary table (Fig. 14) which lists for each ion its number; initial position, energy, and slope; status number; class number; efficiency; number of secondaries; number of secondaries thrown out; and secondary efficiency. This is followed by the average results for each status group and class number.

Finally, a plot is generated which shows the average efficiency versus energy (Fig. 15). The efficiencies of the particles at each initial energy are averaged in two ways. The first average sets the efficiencies of all status group 3 particles to zero, while the second does the same for status group 2 and 3 particles. The first averaging gives a (solid) curve in which the reflected particles are considered lost, and the second (dotted) curve results from considering both retrograde and reflected particles as lost.

INPUT DATA FOR DART CODE

RMIROR	R1	R2	EXHT	BMIROR	BR1		
0.7688	230.6000	232.6000	2.0000	15.0000	0.0500		
BOUND	ZFIN	COLHT	XMAX	XL	C	QM	VO
4.166670E-03	4.200000E-01	1.358785E+00	1.100000E+01	2.000000E+00	2.500000E+00	4.792400E+07	1.000000E+07
HBEG	HMIN	HMAX	TBEG	TMAX	EPS	FREPS	ERRMX
1.000000E-09	5.000000E-15	1.000000E-06	0.	1.000000E-02	5.000000E-04	1.000000E-01	1.000000E-01
SSDT	QD	TCOG					
1.000000E-09	7.122400E-12	9.700000E-01					
NTRAJ=	20	NOFSET=	180	IPRINT=	C	IPLOT=	0
				ISC=	0	IGCO=	C
						ISPAR=	1
ILIM1=	121	KLIM1=	118	ILIM2=	36	KLIM2=	11
				ILIM3=	156	KLIM3=	11

Fig. 14. Printed summary table.

SUMMARY OF PARTICLE EFFICIENCIES

NO.	X	Y	Z	ENERGY	SLOPE	ST	PRIMARY		NO.	ZST	ETA
							CL	ETA			
1	-4.0000	0.1007	0.3345	81C.2564	-0.0274	1	3	0.96788	2	1	0.03277
2	-4.0000	-0.2014	0.1630	81C.2564	0.0548	1	3	0.84561	2	0	0.04257
3	-4.0000	0.1007	-0.0145	81C.2564	-0.0274	1	4	0.97798	2	0	0.02977
4	-4.0000	0.1007	0.1745	81C.2564	-0.0274	1	4	0.97755	2	0	0.03053
5	-4.0000	-0.2014	0.0000	81C.2564	0.0548	12	1	1.00000	4	0	0.10152
6	-4.0000	0.1007	-0.1745	81C.2564	-0.0274	1	1	0.89992	4	0	0.07763
7	-4.0000	0.1007	0.0145	81C.2564	-0.0274	2	3	0.81592	7	0	0.13296
8	-4.0000	-0.2014	-0.1600	81C.2564	0.0548	1	1	0.91446	2	0	0.02240
9	-4.0000	0.1007	-0.3345	81C.2564	-0.0274	1	4	0.91548	2	0	0.03909
10	-4.0000	0.1020	0.3366	830.7692	-0.0274	2	4	0.96225	7	0	0.16350
11	-4.0000	-0.2040	0.1600	830.7692	0.0548	1	4	0.95952	2	0	0.02716
12	-4.0000	0.1020	-0.0166	83C.7692	-0.0274	2	0	0.	5	0	0.14342
13	-4.0000	0.1020	0.1766	83C.7692	-0.0274	2	4	0.85760	7	0	0.17360
14	-4.0000	-0.2040	0.0000	83C.7692	0.0548	11	10	0.48195	1	0	0.00191
15	-4.0000	0.1020	-0.1766	83C.7692	-0.0274	3	4	0.94534	10	0	0.20918
16	-4.0000	0.1020	0.0166	83C.7692	-0.0274	2	1	0.95676	7	0	0.12632
17	-4.0000	-0.2040	-0.1600	830.7692	0.0548	123	4	0.92794	6	0	0.13453
18	-4.0000	0.1020	-0.3366	83C.7692	-0.0274	1	4	0.89432	2	0	0.04248
19	-4.0000	0.1032	0.3388	851.2821	-0.0274	11	1	0.92919	2	0	0.03591
20	-4.0000	-0.2065	0.1600	851.2821	0.0548	1	1	0.93143	2	0	0.01691

DISTRIBUTION 1

WEIGHTED AVERAGE EFFICIENCY*	COLLECTED	RETRCGRAD	TOTAL	SECONDARY	NET	REFLECTED
FRACTION OF TOTAL PARTICLES*	0.68918	0.12152	0.81070	0.07935	0.73134	0.04680
	0.80000	0.15000	0.95000			0.05000

PARTICLE CLASS#	1	2	3	4	5	6	7	8	9	10
FRACTION IN CLASS#	0.30000	0.	0.15000	0.40000	0.	0.	0.	0.	0.	0.05000
AVERAGE ENERGY#	50.82978	0.	100.09279	52.62741	0.	0.	0.	0.	0.	430.37815

TIME PERCENTS# 092#07#00 TIME TOTAL(SEC)= 21.8768 TIME PER PARTICLE(SEC)= 1.0938 NO. REPEATS= 0

Fig. 14. (Continued).

Fig. 15. Plot of average efficiency vs energy.

Acknowledgment

The guidance given by A. Hindmarsh was very helpful in using the GEAR package. We also acknowledge the insight we gained from earlier trajectory codes developed by R. F. Post and R. P. Freis.

References

1. R. F. Post, Mirror Systems: Fuel Cycles, Loss Reduction and Energy Recovery, Lawrence Livermore Laboratory, Rept. UCRL-71753 (1969).
2. R. W. Moir, W. L. Barr, R. P. Freis, and R. F. Post, "Plasma Physics and Controlled Nuclear Fusion Research," in Proc. Fourth Conf. on Plasma Physics and Controlled Nuclear Fusion Research, University of Wisconsin, Madison, 1971, (III IAEA, Vienna, 1971), p. 1431.
3. R. P. Freis, Efficiency of a Direct Conversion Collector for a Fusion Reactor, Lawrence Livermore Laboratory, Rept. UCRL-74029 (1972).
4. A. C. Hindmarsh, GEAR: Ordinary Differential Equation System Solver, Lawrence Livermore Laboratory, Rept. UCID-30001, Rev. 1 (1972).
5. A. C. Hindmarsh, "The Control of Errors in the GEAR Package for Ordinary Differential Equations" in Construction of Mathematical Software, F. N. Fritsch, Ed., Lawrence Livermore Laboratory, Rept. UCID-30050, Part 3 (1972).
6. B. Bartolini, Interpolation - A Function of Two Variables, Lawrence Livermore Laboratory, CIC Rept. E1.1-006 (1966).
7. P. T. Kirstein, G. S. Kino and W. E. Waters, Space Charge Flow (McGraw-Hill Book Co., New York, 1967), p. 426.
8. R. W. Moir, W. L. Barr, and T. N. Haratani, Optimization of Collector Efficiency for Direct Energy Converters for Fusion Reactors, Lawrence Livermore Laboratory, Rept. UCRL-51321 (1973).
9. Reference 7, p. 383.
10. C. K. Birdsall and D. Fuss, J. Comput. Phys. 3, 494, (1969).

Appendix A Input for DART

(Filename = DARTDATA)

<u>Card #</u>	<u>Field Format</u>
1	RMIROR, R1, R2, EXHT, BMIROR, BR1 (F10.0) RMIROR = radial distance to mirror of expander (metres) R1 = radial distance to start of linear B_r -field in expander (metres) R2 = radial distance to negative stage (metres) EXHT = half-height of expander (metres) BMIROR = magnitude of radial B-field (B_r) at RMIROR (Wb/m^2) BR1 = magnitude of radial B-field (B_r) at R1 (Wb/m^2)
2	BOUND, ZFIN, COLHT, XMAX, XL, C, QM, V0 (F10.0) BOUND = half-width of collector cup fin (units of L) ZFIN = half-height of collector channel (units of L) COLHT = half-height of collector (ZFIN + cup depth) (units of L) XMAX = length of collector (units of L) XL = normalization constant, L (metres) C = periodic focusing constant QM = charge-to-mass ratio of particles (C/kg) V0 = normalization constant for potential energy in collector channel – the change in vacuum potential at mid-channel ($z = 0$), per distance L in the x-direction (keV)
3	HBEG, HMIN, HMAX, TBEG, TMAX, EPS, FREPS, ERRMX ... (F10.0) HBEG = initial step size in integration (s) HMIN = minimum allowed step size (s) HMAX = maximum allowed step size (s) TBEG = start time (s) TMAX = maximum allowed trajectory time (s) EPS = relative error test constant for checking accuracy of integration FREPS = fraction by which EPS multiplied (reduced) to re-do any trajectory with error > ERRMX ERRMX = maximum error, relative to initial energy, allowed in the final total energy of each ion
4	SSDT, QD, TCOG (F10.0) SSDT = increment of time between distributions of charge, QD, to charge density table (s) QD = charge on each particle, for charge density purposes (C/m) TCOG = transmission coefficient of the grids

Card #Field Format

5

NTRAJ, NOFSET, IPRINT, IPLOT, ISC, IGCD, ISPAR (IE)

NTRAJ = number of particles to be followed (must be less than
MXNP, a code parameter)

NOFSET = number of particles in standard distribution of starting
conditions to skip before selecting NTRAJ particles

IPRINT = print frequency; time, position, velocity, and energy
of particle printed every IPRINT time steps; IPRINT
= $\emptyset$ means no printing

IPLOT = plot sentinel; IPLOT = $\emptyset$ means no trajectory plots;
IPLOT = 1 means plot particle collection only; IPLOT
= 2 means plot entire trajectory of each particle in
collector

ISC = space charge sentinel; ISC = $\emptyset$ means no space charge used
in integration; ISC = 1 means space charge used (requires space
charge potential table disk file - POTAB3)

IGCD = charge density sentinel; IGCD = $\emptyset$ means no charge
density table generated; IGCD = 1 means charge QD
is distributed to charge density table - QT - every
SSDT seconds, and trajectories are halted at first
exit from density table region

ISPAR = secondary particle sentinel; ISPAR = $\emptyset$ means no
secondaries; ISPAR $\neq \emptyset$ means a secondary is released
at each point an ion crosses a grid and at the trajectory
end points. DART follows the secondaries and computes
their (negative) efficiencies.

Appendix B

Dictionary for CLICHE STORAGE

CLICHE STORAGE allows communication among all DART subroutines.

<u>NAME</u>	<u>DEFINITION</u>
<u>Parameters :</u>	
MXNP	Maximum number of particles to be followed.
N	Size of differential equation system (=6).
ILIM1, KLIM1	Size of cup potential table is $ILIM1 \times KLIM1$.
ILIM2, KLIM2	Size of negative stage potential table is $ILIM2 \times KLIM2$.
ILIM3, KLIM3	Size of space charge potential table is $ILIM3 \times KLIM3$.
INQ, IPQ	Limits for x-axis in space charge table: $INQ \leq I \leq IPQ$.
KNQ, KPQ	Limits for z-axis in space charge table: $KNQ \leq K \leq KPQ$.
LPT1, LPT2, LPT3	Lengths of potential table disk files, POTAB1, POTAB2, POTAB3.
XMIN	Lower limit for x in space charge table: $XMIN \leq X \leq XMAX$.
MAXDER	Maximum order of Adams methods to be used.
LMAX	Used for dimensioning (=MAXDER + 1).
MXSSP	Maximum number of secondary particles followed per ion.
<u>Arrays:</u>	
Y	An N by LMAX array of the dependent variables and their derivatives.
YS	An N by 2 array used to save the past two integration values.
YEX	An array of length N containing the final point of a trajectory.
TAB1	An array of length LPT1 containing the disk file, POTAB1.
TAB2	An array of length LPT2 containing the disk file, POTAB2.
TAB3	An array of length LPT3 containing the disk file, POTAB3.
TAB4	An array of length LPT3 containing the disk file, QDTAB.
SSP	An N by MXSSP array of starting coordinates for secondary particles.
QX	An array of length ILIM3 containing the x-coordinates for the charge density table.
QZ	An array of length KLIM3 containing the z-coordinates for the charge density table.
QT	An ILIM3 by KLIM3 array containing the charge density table.
TABLE	An array of length $LPT1 + LPT2 + LPT3$ containing the disk files POTAB1, POTAB2, and POTAB3.
<u>Variables:</u>	
RCENTER	Contains r_{center} , a radial distance in the expander.
RMIRROR	Contains r_{mirror} , the radial distance to expander mirror (input).

<u>NAME</u>	<u>DEFINITION</u>
R1	Contains r_1 , a radial distance in the expander (input).
R2	Contains r_2 , the radial distance to the negative stage (input).
XMAX	The length of the collector system (input).
XL	Contains L, the normalization constant: length of a rod-grid collector cup pair (input).
XLI	$1/XL$.
BOUND	Half-width of collector cup fin (input).
EXHT	Half-height of expander (input).
ZFIN	Half-height of collector channel (input).
COLHT	Half-height of collector structure (ZFIN + cup depth) (input).
BMIROR	Contains B_{mirror} , the magnitude of the radial B-field (B_r) at RMIROR (input).
BR1	Contains B_{r1} , the magnitude of B_r at R1 (input).
QM	Contains q/m , the charge-to-mass ratio of the ions (input).
QML	Contains q/mL^2 .
C	Contains c, the periodic focusing strength constant (input).
V ϕ	Contains V_0 , the normalization constant for potential (input).
XOLD, ZOLD	The x- and z-coordinates of preceding point.
T	Contains t, the independent variable.
TMAX	Maximum value of T allowed for each particle (input).
TOLD	The value of T at the preceding point.
TEX	The value of T at the final point of a trajectory.
H	Contains h, the current integration step size.
HBEG	The initial value of H (input).
HMIN	The minimum value of H allowed (input).
HMAX	The maximum value of H allowed (input).
HS1	The value of H used for point two steps preceding.
HS2	The value of H used for preceding point.
EPS	Relative error test constant for integration (input).
KFLAG	Completion code for integration step.
JSTART	Integration flag.
IPS	Status number of a particle.
NOSTEP	Number of integration steps taken since last print.
IPRINT	Integration step print frequency (input).
IPLOT	Trajectory plot sentinel (input).
ZCONT, CCONT	The z (ZFIN) and c values used to generate the analytic contour from which the rod and grid points are set in the cup table.
VR	Normalized (by V ϕ) potential value on the rod in cup table.
DELX, DELZ	Contain Δx , Δz for coordinate arrays of cup table.
ISC	Space charge sentinel (input).
IGCD	Charge density sentinel (input).

<u>NAME</u>	<u>DEFINITION</u>
RHOCON	Contains $q_d/(\Delta x \Delta z)^2$, a charge density distribution constant, where $\Delta x, \Delta z$ are the coordinate increments for the space charge table.
SSDT	Contains Δt , the time increment between charge density distributions (input).
SST	The value of T at the preceding charge density distribution.
IYZERO	Sentinel used for bringing y-coordinate of particle back to zero, at each crossing of $x = 0$: IYZERO = 0 x-coordinate of previous point was negative; IYZERO = 1 x-coordinate of previous point was positive.
EVCF	Contains $2 \cdot QM \cdot 1000$., the energy (keV) to velocity (m/s) conversion factor.
EVCFI	Contains $1/EVCF$, the velocity to energy conversion factor.
ISPAR	Secondary particle sentinel (input).
NSSPG	The number of secondary particles started from grids.
<u>Plot Storage</u>	
XPLOT	A 500 by 5 array containing the x-coordinates of the points of the preceding 5 trajectories.
ZPLOT	Same as XPLOT, but for z-coordinates.
NOFF	An array of length 5 containing the number of points in each column of the XPLOT and ZPLOT arrays.
KPLOT	The number of trajectories (columns) saved in the XPLOT and ZPLOT arrays.

Appendix C

Descriptions of Subroutines

The communication between DART subroutines is shown graphically in Fig. C-1. Each subroutine is briefly described below.

MAIN

The MAIN subroutine reads the input cards and then generates the distribution of starting conditions for the particles. After each trajectory is calculated, the efficiency of the energy conversion is found. After the entire distribution of particles has been followed, the weighted average efficiency is computed.

DRIVESA

This subroutine calculates trajectories by calling subroutine STIFFA for the integration values at each time step. Before incrementing the integration time and continuing, the current values are checked for several conditions (including termination).

STIFFA

For each call, STIFFA uses the Adams methods to perform one step of the integration of the initial value problem for a system of ordinary differential equations. It also calculates the optimum step size and order of the next integration step.

COSETA

This subroutine supplies coefficient values to subroutine STIFFA.

FUNCT

Given the current point, $Y(t)$, FUNCT computes the time derivative, dY/dt . The electromagnetic forces are first calculated as a function of the current particle position.

PHI

This subroutine computes the potential energy, either vacuum or space charge, of a particle as a function of the current position of the particle.

LAG2

Using a two-dimensional interpolation scheme, this subroutine returns a potential value from a specified potential table as a function of the position of the particle.

QDENT

The charge density table is generated by QDENT. Linear interpolation is used between trajectory points to weight the mesh points of the table at small increments in time.

IMP

By comparing the current particle position with the boundary locations of the direct converter, IMP determines if impact has occurred. In the case of a hit, linear extrapolation is used to determine the time of impact.

PARPLT

This subroutine superimposes plots of the particle trajectories on a facsimile of the collector structure. The trajectories are plotted in groups of five.

Fig. C1. DART subroutine communication.

Appendix D

Memory Allocation for Potential Tables

In subroutine LAG2,

L1 = offset for XTAB array of table NT

L2 = offset for ZTAB array of table NT

L3 = offset for V array of table NT

L4 = dimension of XTAB array (ILIM) of table NT

L5 = dimension of ZTAB array (KLIM) of table NT

IP, KP, I3 and K3 are pointers to XTAB and ZTAB arrays

Given the coordinates, (TABLE (IP + I), TABLE (KP + K)), the corresponding element of the V array is referenced with a single subscript

$$\text{TABLE}(\text{L3} + ((\text{KP} - \text{L2}) + \text{K} - 1) * \text{L4} + (\text{IP} - \text{L1}) + \text{I}) = \text{TABLE}(\text{L6} + \text{K} * \text{L4} + \text{I})$$

where

$$\text{L6} = \text{L3} + (\text{KP} - \text{L2} - 1) * \text{L4} + \text{IP} - \text{L1}.$$

Appendix E

Listings


```

C*****
C      PROGRAM NAME:  DART
C      ORIGINATED:   13 AUGUST 1971
C      VERSION/DATE:  REACTOR/16 JANUARY 1974
C      PHYSICISTS:   WILLIAM BARR AND RALPH MOIR
C      PROGRAMMER:   BARRY C. HOWARD
C      PURPOSE:      TO COMPUTE PARTICLE TRAJECTORIES IN A
C                   DIRECT CONVERTER AND FIND THE AVERAGE
C                   EFFICIENCY WITH WHICH THE PARTICLES ARE
C                   COLLECTED
C      REFERENCE:    B.C. HOWARD, "DART: A SIMULATION CODE
C                   FOR A DIRECT ENERGY CONVERTER FOR FUSION
C                   REACTORS", SEPT. 23, 1973 (DRAFT)
C*****

```

PROGRAM DARTSE (DARTDATA,TAPE2=DARTDATA,DARTOUTO,TAPE3=DARTOUTO)

```

IN1 FORMAT(8F10.0)
IN2 FORMAT(16I5)
IN3 FORMAT(5F10.0)
OUT1 FORMAT(8F13.4)
OUT2 FORMAT(8E13.6)
OUT3 FORMAT(/5X,6HNTRAJ=,15,5X,7HNOFSET=,15,5X,7HIPRINT=,15,5X,
* 6HIPLOT=,15,5X,4HISC=,15,5X,5HIGCD=,15,5X,6HISPAR=,15)
OUT4 FORMAT(/5X,6HILIM1=,15,5X,6HKLIM1=,15,5X,6HILIM2=,15,5X,6HKLIM2=,
* 15,5X,6HILIM3=,15,5X,6HKLIM3=,15)
OUT5 FORMAT(1H1//44X,26HRESULTS FOR TRAJECTORY NO.,13//10HAT START -)
OUT6 FORMAT(5X,2HX=,E12.5,5X,2HY=,E12.5,5X,2HZ=,E12.5,5X,3HVX=,E12.5,
* 5X,3HUY=,E12.5,5X,3HVZ=,E12.5/5X,7HENERGY=,E12.5,5X,
* 6HSLOPE=,E12.5)
OUT7 FORMAT(/8HAT END -/5X,2HX=,E12.5,5X,2HY=,E12.5,5X,2HZ=,E12.5,5X,
* 3HVX=,E12.5,5X,3HUY=,E12.5,5X,3HVZ=,E12.5/5X,5HTIME=,E12.5,
* 5X,16HENERGY(KINETIC)=,E12.5,5X,14HENERGY(ERROR)=,F8.5,
* 5X,4HETA=,F8.5)
OUT8 FORMAT(113,5F10.4,215,F10.5,110,15,F10.5)
OUT9 FORMAT(/44X,9HCOLLECTED,5X,10HRETROGRADE,5X,5HTOTAL 6X,
* 9HSECONDARY,7X,3HNET,6X,9HREFLECTED/10X,16HWEIGHTED AVERAGE
* 12H EFFICIENCY::2F15.5 2(F10.5 F15.5)/10X,11HFRACTION OF
* 17H TOTAL PARTICLES: 2F15.5 F10.5 25X F15.5)
OUT10 FORMAT(/5X,14HTIME PERCENTS: X,A10,5X,16HTIME TOTAL(SEC)=,F9.4,
* 5X,23HTIME PER PARTICLE(SEC)=,F7.4,5X,12HNO. REPEATS=,15)
OUT11 FORMAT(/8HAT END -/5X,2HX=,E12.5,5X,2HY=,E12.5,5X,2HZ=,E12.5,5X,
* 3HVX=,E12.5,5X,3HUY=,E12.5,5X,3HVZ=,E12.5/5X,5HTIME=,E12.5,
* 5X,16HENERGY(KINETIC)=,E12.5,5X,4HETA=,F8.5,5X,4HIPS=,13)
OUT12 FORMAT(/5X,15HPARTICLE CLASS:,10I10/2X,18HFRACTION IN CLASS:,
* 10F10.5/5X,15HAVERAGE ENERGY:,10F10.5)
OUT13 FORMAT(/15X,12HDISTRIBUTION,12)
HDG0 FORMAT(5X,24HINPUT DATA FOR DART CODE)
HDG1 FORMAT(/7X,6HRMIROR,11X,2HR1,11X,2HR2,9X,4HEXHT,7X,6HBMIROR,10X,
* 3HBR1)
HDG2 FORMAT(/8X,5HBOUND,9X,4HZF1N,8X,5HCOLHT,9X,4HXMAY,11X,2HXL,12X,
* 1HC,11X,2HON,11X,2HVO)
HDG3 FORMAT(/9X,4HHBEG,9X,4HHMIN,9X,4HHMAX,9X,4HTBEG,9X,4HTMAX,
* 10X,3HEPS,8X,5HFREPS,8X,5HERRMX)
HDG4 FORMAT(/9X,4HSSOT,11X,2HOD,9X,4HTCOG)
HDG5 FORMAT(/12X,1HT,12X,1HX,12X,1HY,12X,1HZ,11X,2HVX,11X,2HUY,11X,
* 2HVZ,3X,10HE(KINETIC),5X,8HE(TOTAL))
HDG7 FORMAT(1H1//43X,32HSUMMARY OF PARTICLE EFFICIENCIES/71X,7HPRIMARY,
* 16X,9HSECONDARY/10X,3HNO.,9X,1HX,9X,1HY,9X,1HZ,4X,6HENERGY,
* 5X,5HSLOPE,3X,2HST,3X,2HCL,7X,3HETA,7X,3HNO.,2X,3HZST,7X,3HETA)

```

HDG8 FORMAT(///48X,18HSECONDARY PARTICLE//10HAT START -)

```
CLICHE STORAGE
OPTIMIZE
PARAMETER (MXNP=360)
PARAMETER (N=6,ILIM1=121,KLIM1=118,ILIM2=36,KLIM2=11,ILIM3=156,
* KLIM3=11,INQ=-23,IPQ=132,KNQ=-5,KPQ=5,LPT1=ILIM1+KLIM1+
* ILIM1*KLIM1,LPT2=ILIM2+KLIM2+ILIM2*KLIM2,LPT3=ILIM3+
* KLIM3+ILIM3*KLIM3,XMIN=-1.916667,MAXDER=6,
* LMAX=MAXDER+1,MXSSP=25)
COMMON Y(N,LMAX),YS(N,2),YEX(N),TAB1(LPT1),TAB2(LPT2),TAB3(LPT3),
* TAB4(LPT3),SSP(N,MXSSP)
DIMENSION QX((INQ,IPQ)),QZ((KNQ,KPQ)),QT((INQ,IPQ),(KNQ,KPQ)),
* TABLE(LPT1+LPT2+LPT3)
EQUIVALENCE (QX,TAB4),(QZ,TAB4(ILIM3+1)),(QT,TAB4(ILIM3+KLIM3+1)),
* (TAB1,TABLE)
COMMON RCENTR,RMIROR,R1,R2,XMAX,XL,XL1,BOUND,EXHT,ZFIN,COLHT,
* BMIROR,BR1,QM,QML,C,VO,XOLD,ZOLD,T,TMAX,TOLD,TEX,H,HBEG,
* HMIN,HMAX,HS1,HS2,EPS,KFLAG,JSTART,IPS,NOSTEP,IPRINT,
* IPLOT,ZCONT,CCONT,VR,DELX,DELZ,ISC,IGCD,RHOCN,SSDT,SSST,
* IYZERO,EVCF,EVCFI,ISPAR,NSSPG
COMMON /PLTBLK/ XPLOT(500,5),ZPLOT(500,5),NOPP(5),KPLOT
ENDCLICHE
```

```
USE STORAGE
COMMON /MAINPP/ WIN(MXNP),I!,JJ
COMMON /PHIARG/ XPHI,ZPHI
GENERIC PHI
```

C PARAMETER DEFINITION:
C NK = THE NUMBER OF PARTICLE ENDPOINT CLASSES
C NENL = THE MAXIMUM NUMBER OF DISCRETE INITIAL ENERGIES

```
PARAMETER (TWOPI=6.2831853,NK=10,NENL=50)
DIMENSION XP(MXNP),YP(MXNP),ZP(MXNP),SLOPE(MXNP),
* IPSAV(MXNP),ETA(MXNP),NSSAV(MXNP),ETASEC(MXNP),
* KLAS(MXNP),DIDW(MXNP),NSZS(MXNP)
DIMENSION SUMDIF(3),ETAVG(3),P1PS(3),EAVK(NK),POPK(NK),NPOL(NENL),
* SMETA1(NENL),SMETA2(NENL),ENLEV(NENL),LOCEL(NENL)
EQUIVALENCE (NPOL,XPLOT(501)),(SMETA1,XPLOT(NENL+501)),
* (SMETA2,XPLOT(2*NENL+501)),(ENLEV,XPLOT(3*NENL+501)),
* (LOCEL,XPLOT(4*NENL+501)),(DIDW,XPLOT(5*NENL+501))
```

```
CALL BODROP(4RDART,5R+DART,0)
CALL TICHEK(I,J,K,L)
```

C READ INPUT CARDS

```
RIT 2,IN1,RMIROR,R1,R2,EXHT,BMIROR,BR1
RIT 2,IN1,BOUND,ZFIN,COLHT,XMAX,XL,C,QM,VO
RIT 2,IN1,HBEG,HMIN,HMAX,TBEG,TMAX,EPS,FREPS,ERRMX
RIT 2,IN1,SSDT,QD,TCOG
RIT 2,IN2,NTRAJ,NOFSET,IPRINT,IPLST,ISC,IGCD,ISPAR
```

C GENERATE STANDARD DISTRIBUTION OF STARTING CONDITIONS

```
RCON=SQRTF(BR1/BMIROR)
EVCF=2.*QM*1000.
WMIN=400.
WMAX=1200.
```

```

NW=40
WINC=(WMAX-WMIN)/(NW-1)
NTHETA=3
THINC=TWOPI/NTHETA
X=R1-(R2+XL)
WM=WMIN
D=.16
CONST=RCON*.9486834/(BR1*QM)
DO 15 I=1,NXNP,3=NTHETA
Z=D+D
VIN=SQRT(EVCF*W4)
AR1=CONST*VIN
DO 10 J=0,3*NTHETA-1
IF(XMODF(J,NTHETA)) 5,,5
THETA=.5*THINC
Z=Z-D
5 XP(I+J)=X
  YP(I+J)=AR1*COSF(THETA)
  ZP(I+J)=Z+AR1*SINF(THETA)
  SLOPE(I+J)=THETA
  WIN(I+J)=WM
10 THETA=THETA+THINC
15 WM=WM+WINC

C   SKIP OVER FIRST NOFSET PARTICLES
    IF(NOFSET) 22,
    DO 20 I=1,NTRAJ
    XP(I)=XP(I+NOFSET)
    YP(I)=YP(I+NOFSET)
    ZP(I)=ZP(I+NOFSET)
    WIN(I)=WIN(I+NOFSET)
20 SLOPE(I)=SLOPE(I+NOFSET)

C   READ POTENTIAL TABLES
22 CALL DEVICE(4HOPEN,6RPOTAB1)
   CALL ASSIGN(21,6RPOTAB1)
   BUFFER IN (21,1) (TAB1(1),TAB1(LPT1),0)
24 IF(UNIT,21) 24,,
   CALL DEVICE(5HCLOSE,6RPOTAB1)

   CALL DEVICE(4HOPEN,6RPOTAB2)
   CALL ASSIGN(22,6RPOTAB2)
   BUFFER IN (22,1) (TAB2(1),TAB2(LPT2),0)
26 IF(UNIT,22) 26,,
   CALL DEVICE(5HCLOSE,6RPOTAB2)

   IF(ISC) 30,
   CALL DEVICE(4HOPEN,6RPOTAB3)
   CALL ASSIGN(23,6RPOTAB3)
   BUFFER IN (23,1) (TAB3(1),TAB3(LPT3),0)
28 IF(UNIT,23) 28,,
   CALL DEVICE(5HCLOSE,6RPOTAB3)
30 CONTINUE

C   LIST INPUT CARDS
    WOT 3,HOG0
    WOT 3,HOG1

```

```

WOT 3,OUT1,RMIRROR,R1,R2,EXHT,BMIRROR,BR1
WOT 3,HOG2
WOT 3,OUT2,BOUND,ZFIN,COLHT,XMAX,XL,C,QM,VO
WOT 3,HOG3
WOT 3,OUT2,HBEQ,HMIN,HMAX,TBEG,TMAX,EPS,FREPS,ERRMX
WOT 3,HOG4
WOT 3,OUT2,SSDT,QD,TCOG
WOT 3,OUT3,NTRAJ,NOFSET,IPRINT,IPLLOT,ISC,IGCD,ISPAR
WOT 3,OUT4,ILIM1,KLIM1,ILIM2,KLIM2,ILIM3,KLIM3

```

C INITIALIZE CONSTANTS

```

XL1=1./XL
RMIRROR=RMIRROR*XL1
R1=R1*XL1
R2=R2*XL1
EXHT=EXHT*XL1
QML=QM*XL1*XL1
RCENTR=(BMIRROR-RMIRROR-BR1*R1)/(BMIRROR-BR1)
VO=VO*1000.
EVCFF=1./EVCF
EPS1=EPS
NREDO=0

```

C SET ROD AND GRID POTENTIALS IN CUP TABLE

```

DELX=1./(ILIM1-1)
DELZ=(COLHT-ZFIN)/(KLIM1-2)
X=.25
ZCONT=.380166
CCONT=2.5
VR=X*(CCONT/TWOPI)*SINF(TWOPI*X)*SINH(TWOPI*ZCONT)

```

C SET PLOT PARAMETERS

```

CALL KEEP80(1)
CALL CRTID(4HDART,1,1)
IF(IPLLOT) 34,
DO 32 I=1,5
32 NOPP(I)=0
KPLLOT=1

```

C DEFINE CHARGE DENSITY COORDINATE ARRAYS AND ZERO THE TABLE

```

34 IF(IGCD) ,40,
DOQ=QD/VO
DOX=1./12.
DOZ=ZFIN/5.
RHOCEN=QD/(DOX*DOX*DOZ*DOZ)
DO 36 I=1,NQ,IPO
36 QX(I)=I*DOX
DO 37 K=KNQ,KPQ
37 QZ(K)=K*DOZ
DO 38 I=ILIM3+KLIM3+1,LPT3
38 TAB4(I)=0.
40 CONTINUE

```

C ION TRAJECTORY LOOP

```

DO 500 I1=1,NTRAJ

```

```

IGCD=XABSF(IGCD)
EPS=EPS1
THETA=SLOPE(11)
50 IPS=1
   NSTEP=0
   T=TBEG
   H=HBEG
   JSTART=0
   NSSPG=0
   IYZERO=0
   IF(XP(11)) 60,60,
   IYZERO=1
60 CONTINUE

C   SET INITIAL VELOCITY AND SLOPE
VIN=SQRTF(EVCF*WIN(11))
XDOTP=VIN*SQRTF(1.-9*RCON*RCON)
YDOTP=RCON*.9486834*VIN*SINF(THETA)
ZDOTP=(RCON*.9486834*VIN)*COSF(THETA)
SLOPE(11)=ATAN2(ZDOTP,XDOTP)
WOT 3,OUTS,11
WOT 3,OUTS,XP(11),YP(11),XDOTP,YDOTP,ZDOTP,WIN(11),
* SLOPE(11)
WOT 3,HDS5

C   NORMALIZE INITIAL POSITION AND VELOCITY
Y(1)=YP(11)*XLI
Y(2)=Y(11)*XLI
Y(3)=ZP(11)*XLI
Y(4)=XDOTP*XLI
Y(5)=YDOTP*XLI
Y(6)=ZDOTP*XLI

C   SET DUMMY PAST POINT
YS(1,1)=XOLD=Y(1)+Y(4)*H
YS(2,1)=Y(2)+Y(5)*H
YS(3,1)=ZOLD=Y(3)+Y(6)*H
YS(4,1)=Y(4)
YS(5,1)=Y(5)
YS(6,1)=Y(6)
HS1=H

C   CALCULATE ION TRAJECTORY
CALL DRIVESA

C   COMPUTE FINAL POTENTIAL ENERGY OF ION
XPHI=YEX(1)
ZPHI=YEX(3)
PE=.001*PHI(1)
IF(ISC) 90,
PE=PE+.001*PHI(2)

C   COMPUTE EFFICIENCY OF ION COLLECTION
90 ETA(11)=0.

```

```

      IF(YEX(1)-1.) 100,
      SUM=(TCOG**NSSPG)*PE
      TEMP=1.-TCOG
      DO 95 I=1,NSSPG
      XPHI=SSP(I,1)*XLI
      ZPHI=SSP(3,I)*XLI
      PEG=.001*PHI(1)
      SUM=SUM+TEMP*PEG
95    TEMP=TEMP*TCOG
      ETA(1)=SUM/WIN(1)

C      FIND AND PRINT FINAL COORDINATES AND RELATIVE ERROR IN TOTAL
C      ION ENERGY

100   DO 120 I=1,N
120   YEX(I)=YEX(I)*XL
      WEND=(YEX(4)*YEX(4)+YEX(5)*YEX(5)+YEX(6)*YEX(6))*EVCFI
      WERR=ABSF(WEND+PE-WIN(1))/WIN(1)
      WOT 3,OUT7,(YEX(I),I=1,N),TEX,WEND,WERR,ETA(1)
      IPSAV(I)=IPS
      IF(IPS) ,200,

C      IF EXCESSIVE ENERGY ERROR, REDO TRAJECTORY

      IF(IGCD) ,125,
      IF(WERR-ERRMX) 200,,
      NREDO=NREDO+1
      GO TO 200

125   IF(WERR-ERRMX) ,130,130
      IF(ETA(1)-1.) 140,140,
      ETA(1)=1.
      GO TO 140
130   NREDO=NREDO+1
      EPS=EPS*FREPS
      IF(IPLT) ,50,
      NOPP(KPLOT)=0
      GO TO 50

C      CLASSIFY ION BY ENDPOINT

      ARITHREG XEND,ZEND
140   XEND=YEX(1)*XLI
      ZEND=YEX(3)*XLI
      IF(XEND-1.) 142,,
      IF(ABSF(ZEND)+.0001-ZFIN) ,143,143
142   KLAS(1)=0
      GO TO 200
143   XEND=XEND-INTF(XEND)
      IF(XEND-.5) 145,,
      IF(ZEND) ,150,
      XEND=XEND-.5
      ZEND=-ZEND
      GO TO 180
145   IF(ZEND) ,180,
      XEND=XEND+.5
      ZEND=-ZEND

C      ENDPOINT IN GRID CUP

```

```

150 XEND=XEND-.75
    IF(ZEND+.0001-ZFIN-92.*DELZ) ,160,160
152 IF(XEND)
    KLAS(11)=4,155
    GO TO 200
155 KLAS(11)=1
    GO TO 200
160 IF(ZEND+.0001-COLHT) ,170,170
    IF(ABSF(XEND)-27.*DELX) ,152,152
165 KLAS(11)=3
    GO TO 200
170 IF(ABSF(XEND)-24.*DELX) 165,,
    KLAS(11)=2
    GO TO 200

C      ENDPPOINT IN ROD CUP

180 IF(ZEND-ZFIN-30.*DELZ) ,195
    ARG=(TWOPI/CCONT)*(VR-XEND)/SINF(TWOPI*XEND)
    ZEP=LOGF(ARG+SQRTF(ARG*ARG+1.))/TWOPI
    ZEP=ZEP+(ZFIN-ZCONT)
    IF(ZEND+.0001-ZEP) 185,,
    KLAS(11)=10
    GO TO 200
185 IF(XEND-.25) ,190
    KLAS(11)=9
    GO TO 200
190 KLAS(11)=7
    GO TO 200
195 IF(ZEND+.0001-COLHT) 185,,
    KLAS(11)=8
    RELEASE XEND,ZEND

C      PLOT TRAJECTORIES

200 IF(IPL0T) ,250,
    IF(ISPAR) ,210,
    IF(IPS) 240,220,240
210 IF(KPLOT-5) ,220,
    IF(11-NTRAJ) 240.,
220 CALL PARPLT
    DO 230 I=1,5
230 NGPP(I)=0
    KPLOT=0
240 KPLOT=KPLOT+1

C      SECONDARY PARTICLES - INITIALIZATION

250 IF(ISPAR) ,500,
    IF(IPS) ,500,
    JGCD=IGCD
    IGCD=0
    JSPAR=ISPAR
    ISPAR=0
    QM=-QM
    QML=-QML
    EPS=EPS*.1
    NSZS(11)=0
    ETASEC(11)=0.

```

C START A SECONDARY FROM EACH POINT THAT ION PASSED THRU A GRID, AND
C ANOTHER FROM ION TRAJECTORY END POINT

```

Nssp=Nsspg
IF(YEX(1)-XL) 310,310
IF(ABS(YEX(3))*XLI+.0001-ZFIN) 310,,
Nssp=Nssp+1
DO 300 I=1,N
300 SSP(I,Nssp)=YEX(I)
310 NSSAV(I)=Nssp
IF(Nssp) 320,,320
JJ=1
GO TO 375

```

C SECONDARY TRAJECTORY LOOP

```

320 DO 400 JJ=1,Nssp
IFS=1
NSTEP=0
T=TBEQ
H=HMIN
JSTART=0
IYZERO=1

```

C IF INITIAL X COORD. IS ZERO, STARTING CONDITIONS COULD NOT BE
C COMPUTED, SO THROW OUT SECONDARY

```

IF(SSP(1,JJ)) 325,325
NSZS(I)=NSZS(I)+1
KPL0T=KPL0T-1
GO TO 375

```

C SET INITIAL SLOPE OF SECONDARY TO NEGATIVE OF INCIDENT ION SLOPE

```

325 W=.1
S=SSP(6,JJ)/SSP(4,JJ)
WXP=W/(1.+S*S)
WYP=0.
WZP=W-WXP
XD0TP=SQRTF(EVCF*WXP)
YD0TP=SQRTF(EVCF*WYP)
ZD0TP=SQRTF(EVCF*WZP)
XD0TP=SIGNF(XD0TP,-SSP(4,JJ))
YD0TP=SIGNF(YD0TP,-SSP(5,JJ))
ZD0TP=SIGNF(ZD0TP,-SSP(6,JJ))
THETA=ATAN2(ZD0TP,XD0TP)
WOT 3,HDS
WOT 3,OUT6,(SSP(1,JJ),I=1,3),XD0TP,YD0TP,ZD0TP,W,THETA
WOT 3,HDS

```

C NORMALIZE INITIAL POSITION AND VELOCITY

```

Y(1)=SSP(1,JJ)*XLI
Y(2)=SSP(2,JJ)*XLI
Y(3)=SSP(3,JJ)*XLI
Y(4)=XD0TP*XLI
Y(5)=YD0TP*XLI
Y(6)=ZD0TP*XLI

```



```

C      SET DUMMY FAST POINT
      YS(1,1)=XOLD=Y(1)+Y(4)*H
      YS(2,1)=Y(2)+Y(5)*H
      YS(3,1)=ZOLD=Y(3)+Y(6)*H
      YS(4,1)=Y(4)
      YS(5,1)=Y(5)
      YS(6,1)=Y(6)
      HS1=H

C      CALCULATE SECONDARY TRAJECTORY
      CALL DRIVESA

C      COMPUTE SECONDARY EMISSION COEFFICIENT USING KINETIC ENERGY
      OF INCIDENT ION
      WION=(SSP(4,JJ)*SSP(4,JJ)+SSP(5,JJ)*SSP(5,JJ)+SSP(6,JJ)*SSP(6,JJ))
      *EVCFI
      IF(WION-2.) , ,340
      SEC=.1
      GO TO 345
340 SEC=.1+.4342945*LOGF(.5*WION)*.8849557
      SEC=MIN1F(SEC,2.)

C      COMPUTE (NEGATIVE) EFFICIENCY OF SECONDARY PARTICLES
345 DO 350 I=1,N
350 YEX(I)=YEX(I)*XL
      WEND=(YEX(4)*YEX(4)+YEX(5)*YEX(5)+YEX(6)*YEX(6))*EVCFI
      IF(JJ-NSSPG) , ,355
      ETA2=(TCOG**(JJ-1))*(1.-TCOG)*SEC*WEND/WIN(I)
      GO TO 360
355 ETA2=(TCOG**NSSPG)*SEC*WEND/WIN(I)

C      ACCUMULATE SECONDARY EFFICIENCIES UNLESS STATUS IS ZERO
360 IF(IPS) , ,365
      ETASEC(I)=ETASEC(I)+ETA2
      GO TO 370
365 NSZS(I)=NSZS(I)+1
370 WOT 3,OUT11,(YEX(I),I=1,N),TEX,WEND,ETA2,IPS

C      PLOT TRAJECTORIES
375 IF(IPLT) , ,400
      IF(KPLOT-5) , ,380
      IF(JJ-NSSPG) , ,390
380 CALL PARPLT
      DO 385 I=1,5
385 NOPP(I)=0
      KPLOT=0
390 KPLOT=KPLOT+1
400 CONTINUE

C      RETURN ION PARAMETERS TO ORIGINAL VALUES
      QM=-QM
      QML=-QML
      JGCD=JGCD

```

```

      ISPAR=JSPAR
500  CONTINUE

C    PRINT SUMMARY TABLE
      IF(IGCD) 700,,700
      WOT 3,HG7
      WOT 3,OUT8,{1,XP(1),YP(1),ZP(1),WIN(1),SLOPE(1),IPSAV(1),KLAS(1),
      *      ETA(1),NSSAV(1),NSZS(1),ETASEC(1)},I=1,NTRAJ}

C    ION DISTRIBUTION LOOP - FIRST PASS IS "FLAT TOP"
      DO 600 J=1,1

C    SET DISTRIBUTION WEIGHTS
      IF(J-2) ,505,
      DO 503 I=1,NTRAJ
503  DIDW(I)=1.
      GO TO 510
505  DO 508 I=1,NTRAJ
508  DIDW(I)=1./WIN(I)

C    CLEAR ACCUMULATORS
510  SUMWT1=0.
      SUMWT2=0.
      SUM=0.
      DO 515 I=1,3
      PIPS(I)=0.
515  SUMDIF(I)=0.
      DO 520 I=1,NK
      POPK(I)=0.
520  EAVK(I)=0.

C    ASSIGN STATUS GROUP NUMBERS AND ACCUMULATE
      DO 550 I=1,NTRAJ
      IF(IPSAV(I)) ,550,
      IF(!IPSAV(I)-10) ,525,
      IPS=1
      GO TO 535
525  IF(IPSAV(I)-110) ,530,
      IPS=2
      GO TO 535
530  IPS=3

535  SUMDIF(IPS)=SUMDIF(IPS)+DIDW(I)*WIN(I)*ETA(I)
      SUM=SUM+DIDW(I)*WIN(I)*ETASEC(I)
      SUMWT1=SUMWT1+DIDW(I)*WIN(I)
      PIPS(IPS)=PIPS(IPS)+DIDW(I)
      SUMWT2=SUMWT2+DIDW(I)
      IF(IPS-3) ,550,
      IF(KLAS(I)) ,550,
      EAVK(KLAS(I))=EAVK(KLAS(I))+DIDW(I)*(WIN(I)-WIN(I)*ETA(I))
      POPK(KLAS(I))=POPK(KLAS(I))+DIDW(I)
550  CONTINUE

C    COMPUTE AND PRINT THE WEIGHTED AVERAGE EFFICIENCIES AND
C    FRACTIONAL COUNT FOR EACH STATUS GROUP

```

```

      IF(SUMWT1) ,750,
      CONST1=1./SUMWT1
      CONST2=1./SUMWT2
      ETATOT=TPIPS=0.
      DO 375 I=1,3
      EAVG(I)=SUMDIF(I)*CONST1
      PIPS(I)=PIPS(I)*CONST2
      IF(I-3) ,580,
      ETATOT=ETATOT+EAVG(I)
575  TPIPS=TPIPS+PIPS(I)
580  EAVGS=SUM*CONST1
      ETANET=ETATOT-EAVGS
      WGT 3,OUT13,J
      WGT 3,OUT9,ETAVG(1),ETAVG(2),ETATOT,ETAVGS,ETANET,ETAVG(3),
      *      PIPS(1),PIPS(2),TPIPS,PIPS(3)

C      COMPUTE AND PRINT THE MEAN KINETIC ION ENERGY
C      AND FRACTIONAL COUNT FOR EACH ENDPOINT CLASS

      DO 590 I=1,NK
      IF(POPK(I)) ,590,
      EAVK(I)=EAVK(I)/POPK(I)
      POPK(I)=POPK(I)*CONST2
590  CONTINUE
      WGT 3,OUT12,(I,I=1,NK),(POPK(I),I=1,NK),(EAVK(I),I=1,NK)
600  CONTINUE

C      COMPUTE AVERAGE EFFICIENCY FOR EACH DISCRETE INITIAL ENERGY

      NEL=0
      DO 610 JJ=1,NENL
      NPEL(JJ)=0
      SMETA1(JJ)=0.
610  SMETA2(JJ)=0.

      DO 650 JJ=1,NTRAJ
      IF(IPS AV(JJ)) ,650,
      DO 620 KK=1,NEL
      IF(WIN(JJ)-ENLEV(LOCEL(KK))) 625,640,
620  CONTINUE
      GO TO 535
625  DO 630 LL=NEL,KK,-1
630  LOCEL(LL+1)=LOCEL(LL)
635  NEL=NEL+1
      ENLEV(NEL)=WIN(JJ)
      LOCEL(KK)=NEL
640  LEL=LOCEL(KK)
      NPEL(LEL)=NPEL(LEL)+1
      IF(IPS AV(JJ)-110) ,650,
      SMETA1(LEL)=SMETA1(LEL)+ETA(JJ)
      IF(IPS AV(JJ)-10) ,650,
      SMETA2(LEL)=SMETA2(LEL)+ETA(JJ)
650  CONTINUE

C      PLOT ENERGY VERSUS EFFICIENCY

      DO 660 JJ=1,NEL
      LEL=LOCEL(JJ)
      XPLOT(JJ,1)=ENLEV(LEL)

```

```

ZPLOT(JJ,1)=ZPLOT(JJ,2)=0.
IF(NPEL(LEL)) ,660
CONST1=1./NPEL(LEL)
ZPLOT(JJ,1)=SMETA1(LEL)*CONST1
ZPLOT(JJ,2)=SMETA2(LEL)*CONST1
660 CONTINUE
J=NEL
WMAX=XPLOT(NEL,1)

CALL MAPG(0.,WMAX,0.,1.)
CALL SETCH(39.,1.,1,0,1,0,0)
CALL CRTBCD(6HENERGY)
CALL SETCH(1.,37.,1,0,1,1,0)
CALL CRTBCD(10HEFFICIENCY)
CALL SETCH(26.,10.,1,0,1,0,1)
CALL CRTBCD(33HDOTTED LINE - WITHOUT RETROGRADES)
CALL SETCH(28.,8.)
CALL CRTBCD(29HSOLID LINE - WITH RETROGRADES)
CALL TRACE(XPLOT(1,1),ZPLOT(1,1),J)
CALL TRACE(XPLOT(1,1),ZPLOT(1,2),J,4)
GO TO 750

C      CREATE DISK FILE AND SAVE CHARGE DENSITY TABLE

700 LNG=LPT3+10
CALL DEVICE(4HOPEN,5RQDTAB)
CALL DEVICE(7HDESTROY,5RQDTAB)
CALL DEVICE(5HCLOSE,5RQDTAB)
CALL DEVICE(6HCREATE,5RQDTAB,LNG,IERR)
CALL ASSIGN(24,5RQDTAB)
BUFFER OUT (24,1) (TAB4(1),TAB4(LPT3),0)
725 IF(UNIT,24) 725,
CALL DEVICE(6HCLOSER,5RQDTAB)

C      PRINT ELAPSED TIME AND NO. OF REQUIRED REPEATS

750 CALL TICKEK(I,J,K,L)
TS=J*1.E-6
TPP=TS/NTRAJ
WOT 3,OUT10,1,TS,TPP,NREDO
CALL EXIT
END

```

SUBROUTINE DRIVESA

```

C*****
C* DRIVESA IS DRIVER SUBROUTINE FOR STIFFA.
C* THE GEARA VERSION OF THE GEAR PACKAGE IS FOR THE ADAMS METHODS
C* WITH FUNCTIONAL ITERATION ONLY.
C* THE VALUES AT EACH TIME STEP ARE RETURNED FROM STIFFA, AND SEVERAL
C* CONDITIONS ARE CHECKED BEFORE CONTINUING.
C* OUTPUT VARIABLES:
C*   TEX - TIME OF FINAL POINT
C*   YEX - ARRAY OF POSITION AND VELOCITIES OF FINAL POINT
C*****
OUT1 FORMAT(2X,3H IT)
OUT2 FORMAT(2X,36H REQUIRED ACCURACY CANNOT BE OBTAINED)
OUT3 FORMAT(2X,19H TIME LIMIT EXCEEDED)
OUT4 FORMAT(2X,27H PARTICLE REFLECTED WITH WX=,F10.4)
OUT5 FORMAT(/,31H KFLAG = -1 FROM STIFFA AT T = ,E16.8/
1 38H ERROR TEST FAILED WITH ABS(H) = HMIN/)
OUT6 FORMAT(/,31H KFLAG = -2 FROM STIFFA AT T = ,E16.8,5H H = ,E16.8/
1 52H THE REQUESTED ERROR IS SMALLER THAN CAN BE HANDLED/)
OUT7 FORMAT(/,31H KFLAG = -3 FROM STIFFA AT T = ,E16.8/
1 45H CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED/)
OUT8 FORMAT(9E13.5)
OUT9 FORMAT(2X,18H FINAL POINT IS BAD)

USE STORAGE
GENERIC PHI
COMMON /PHIARG/ XPHI,ZPHI
COMMON /DRIMP/ NC

10 CALL STIFFA
   IF(KFLAG) 150,,

C   BRING Y COORDINATE BACK TO ZERO AT EACH CROSSING OF X=0.

   IF(1YZERO) ,20,
   IF(Y(1)) ,40,40
   1YZERO=0
   GO TO 30
20 IF(Y(1)) 40,40,
   1YZERO=1
30 Y(2)=0.

C   PRINT CALCULATED VALUES

40 IF(IPRINT) ,80,
   NOSTEP=NOSTEP+1
   IF(NOSTEP-IPRINT) 80,,80
   NOSTEP=0
   XPHI=Y(1)
   ZPHI=Y(3)
   PE=.001*PHI(1)
   IF(ISC) 50,
   PE=PE+.001*PHI(2)
50 DO 60 I=1,N
60 Y(I)=Y(I)*XL
   E=(Y(4)*Y(4)+Y(5)*Y(5)+Y(6)*Y(6))*EVCFI
   TE=PE+E
   WOT 3,OUT8,T,(Y(I),I=1,N),E,TE

```

```

      DO 70 I=1,N
      Y(I)=Y(I)*XLI
C     SAVE POINTS FOR PLOTTING
      60 IF(IPLDT) ,110
      IF(Y(I)-XMIN) 110,
      IF(NOPP(KPLECT)-500) 100,,
      DO 90 I=1,400
      XPLOT(I,KPLOT)=XPLOT(I+100,KPLOT)
      90 ZPLOT(I,KPLOT)=ZPLOT(I+100,KPLOT)
      NOPP(KPLOT)=400
      100 NOPP(KPLOT)=NOPP(KPLECT)+1
      XPLOT(NOPP(KPLOT),KPLOT)=Y(I)
      ZPLOT(NOPP(KPLOT),KPLOT)=Y(3)
C     GENERATE CHARGE DENSITY TABLE
      110 IF(IGCD) ,150,
      IF(Y(I)-XMIN) 130,,
      IF(Y(I)-XMAX) ,130
      IF(ABS(Y(3))-ZFIN) ,130
      IF(IGCD) 120,120,
      CALL QDENT
      IGCD=-1
      GO TO 150

      120 IF(T-(SST+SSDT)) 150,,
      SST=SST+SSDT
      CALL QDENT
      GO TO 150

      130 IF(IGCD) ,150,150
C     PARTICLE HAS LEFT CHARGE DENSITY TABLE
      TEX=TOLD
      DO 140 I=1,N
      140 YEX(I)=YS(I,2)
      RETURN
C     CHECK FOR PARTICLE IMPACT
      150 CONTINUE
      CALL IMP
      IF(NC) 200,,160
      GO TO (180,240,250,260),1-KFLAG
C     PARTICLE HAS IMPACTED - COMPUTE AND CHECK FINAL COORDINATES
      160 WOT 3,OUT1
      FRAC=(TEX-TOLD)/HS1
      DO 170 I=1,N
      170 YEX(I)=YS(I,2)+FRAC*(YS(I,2)-YS(I,1))
      DHIT=SQRT((YS(I,2)-Y(1))**2+(YS(3,2)-Y(3))**2)
      DFIN=SQRT((YS(I,2)-YEX(1))**2+(YS(3,2)-YEX(3))**2)
      IF(2.*DHIT-DFIN) ,175,175
      WOT 3,OUT9
      IPS=0
      175 RETURN

```

```

C      CHECK FOR TIME LIMIT EXCEEDED

180  IF (T-TMAX) , 220, 220
    DO 190 I=1, N
    YS(I,1)=YS(I,2)
190  YS(I,2)=Y(I)
    XOLD=Y(I)
    ZOLD=Y(3)
    HS1=HS2
    GO TO 10

C      REFLECT IMPACTED PARTICLE

200  WXP=(YS(4,2)*YS(4,2)*XL*XL)*EVCFI
    WOT 3, OUT4, WXP
    YS(4,2)=-YS(4,2)
    DO 210 I=1, N
210  Y(I)=YS(I,2)
    XOLD=YS(I,1)
    ZOLD=YS(3,1)
    H=HS1=HBEG
    T=TOLD
    JSTART=0
    GO TO 10

C      ESTIMATE VALUE AT T=TMAX

220  WOT 3, OUT3
    IPS=0
    TEX=TMAX
    RATIO=(TMAX-TOLD)/HS2
    DO 230 I=1, N
230  YEX(I)=YS(I,2)+RATIO*(Y(I)-YS(I,2))
    RETURN

C      ERROR RETURNS

240  WOT 3, OUT5, T
    GO TO 270

250  WOT 3, OUT6, T, H
    GO TO 270

260  WOT 3, OUT7, T
270  IPS=0
    TEX=T
    DO 280 I=1, N
280  YEX(I)=Y(I)
    RETURN
    END

```

```

SUBROUTINE STIFFA
C*****
C* STIFFA PERFORMS ONE STEP OF THE INTEGRATION OF AN INITIAL VALUE
C* PROBLEM FOR A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS.
C* STIFFA IS A SHORTENED VERSION OF STIFF WITH MF = 10 ASSUMED.
C* COMMUNICATION WITH STIFFA IS DONE WITH THE FOLLOWING VARIABLES.
C*
C* Y      AN NO BY LMAX ARRAY CONTAINING THE DEPENDENT VARIABLES
C*        AND THEIR SCALED DERIVATIVES. LMAX IS CURRENTLY 13 FOR
C*        THE ADAMS METHODS. LMAX = 1
C*        IS MAXDER, THE MAXIMUM ORDER USED. SEE SUBROUTINE COSETA.
C*        Y(I,J+1) CONTAINS THE J-TH DERIVATIVE OF Y(I), SCALED BY
C*        H**J/FACTORIAL(J). ONLY Y(I), 1.LE. I.LE. N, NEED BE
C*        SET BY THE CALLING PROGRAM ON THE FIRST ENTRY.
C*        IF IT IS DESIRED TO INTERPOLATE TO NON-MESH POINTS,
C*        THE Y ARRAY CAN BE USED. IF THE CURRENT STEP SIZE
C*        IS H AND THE VALUE AT T + E IS NEEDED, FORM
C*        S = E/H, AND THEN COMPUTE
C*
C*          NO
C*          Y(I)(T+E) = SUM Y(I,J+1)*S**J.
C*          J=0
C*
C*        THE Y ARRAY SHOULD NOT BE ALTERED BY THE CALLING PROGRAM.
C*        WHEN REFERENCING Y AS A 2-DIMENSIONAL ARRAY, USE A
C*        COLUMN LENGTH OF NO, AS THIS IS THE VALUE USED IN STIFFA.
C* N      THE NUMBER OF FIRST ORDER DIFFERENTIAL EQUATIONS. N
C*        MAY BE DECREASED ON LATER CALLS IF THE NUMBER OF
C*        ACTIVE EQUATIONS REDUCES, BUT IT MUST NOT BE
C*        INCREASED WITHOUT CALLING WITH JSTART = 0.
C* NO     A CONSTANT INTEGER GE N, USED FOR DIMENSIONING PURPOSES.
C*        NO MUST NOT BE CHANGED WITHOUT SETTING JSTART = 0.
C* T      THE INDEPENDENT VARIABLE. T IS UPDATED ON EACH STEP TAKEN.
C* H      THE STEP SIZE TO BE ATTEMPTED ON THE NEXT STEP.
C*        H MAY BE ADJUSTED UP OR DOWN BY THE ROUTINE
C*        IN ORDER TO ACHIEVE AN ECONOMICAL INTEGRATION.
C*        HOWEVER, IF THE H PROVIDED BY THE USER DOES
C*        NOT CAUSE A LARGER ERROR THAN REQUESTED, IT
C*        WILL BE USED. TO SAVE COMPUTER TIME, THE USER IS
C*        ADVISED TO USE A FAIRLY SMALL STEP FOR THE FIRST
C*        CALL. IT WILL BE AUTOMATICALLY INCREASED LATER.
C*        H CAN BE EITHER POSITIVE OR NEGATIVE, BUT ITS SIGN
C*        MUST REMAIN CONSTANT THROUGHOUT THE PROBLEM.
C* HMIN   THE MINIMUM ABSOLUTE VALUE OF THE STEP SIZE THAT WILL BE
C*        USED FOR THE PROBLEM. ON STARTING THIS MUST BE MUCH
C*        SMALLER THAN THE AVERAGE ABS(H) EXPECTED, SINCE
C*        A FIRST ORDER METHOD IS USED INITIALLY.
C* HMAX   THE MAXIMUM ABSOLUTE VALUE OF THE STEP SIZE THAT WILL
C*        BE USED FOR THE PROBLEM.
C* EPS    THE RELATIVE ERROR TEST CONSTANT. SINGLE STEP ERROR
C*        ESTIMATES DIVIDED BY YMAX(I) MUST BE LESS THAN THIS
C*        IN THE EUCLIDEAN NORM. THE STEP AND/OR ORDER IS
C*        ADJUSTED TO ACHIEVE THIS.
C* MF     THE METHOD FLAG, NOT USED AS MF = 10 IS ASSUMED.
C* YMAX   AN ARRAY OF N LOCATIONS WHICH CONTAINS THE MAXIMUM
C*        ABSOLUTE VALUE OF EACH Y SEEN SO FAR.
C* ERROR  AN ARRAY OF N ELEMENTS PROPORTIONAL TO THE ESTIMATED
C*        ONE STEP ERROR IN EACH COMPONENT.
C* KFLAG  A COMPLETION CODE WITH THE FOLLOWING MEANINGS..
C*        -1 THE STEP WAS SUCCESSFUL
C*        -1 THE REQUESTED ERROR COULD NOT BE ACHIEVED
C*        WITH ABS(H) = HMIN.

```



```

C*          -2 THE REQUESTED ERROR IS SMALLER THAN CAN
C*            BE HANDLED FOR THIS PROBLEM.
C*          -3 CORRECTOR CONVERGENCE COULD NOT BE
C*            ACHIEVED FOR ABS(H) .GT. HMIN.
C*          ON A RETURN WITH KFLAG NEGATIVE, THE VALUES OF T AND
C*          THE Y ARRAY ARE AS OF THE BEGINNING OF THE LAST
C*          STEP, AND H IS THE LAST STEP SIZE ATTEMPTED.
C*    JSTART AN INTEGER USED ON INPUT AND OUTPUT.
C*          ON INPUT, IT HAS THE FOLLOWING VALUES AND MEANINGS..
C*          0 PERFORM THE FIRST STEP. THIS VALUE ENABLES
C*            THE SUBROUTINE TO INITIALIZE ITSELF.
C*          .GT.C TAKE A NEW STEP CONTINUING FROM THE LAST.
C*            ASSUMES THE LAST STEP WAS SUCCESSFUL AND
C*            USER HAS NOT CHANGED ANY PARAMETERS.
C*          .LT.O TAKE THE NEXT STEP WITH A NEW VALUE OF
C*            H AND/OR EPS. THIS MAY BE
C*            EITHER IN REDOING A STEP THAT FAILED, OR
C*            IN CONTINUING FROM A SUCCESSFUL STEP.
C*          ON EXIT, JSTART IS SET TO NO, THE CURRENT ORDER OF THE
C*          METHOD. THIS IS ALSO THE ORDER OF THE MAXIMUM DERIVATIVE
C*          AVAILABLE IN THE Y ARRAY. AFTER A SUCCESSFUL STEP,
C*          JSTART NEED NOT BE RESET FOR THE NEXT CALL.
C*    FSAVE A BLOCK OF AT LEAST 2*NO LOCATIONS FOR TEMPORARY STORAGE.
C*
C*    THE PARAMETERS WHICH MUST BE INPUT BY THE USER ARE..
C*    N, NO, T, Y, H, HMIN, HMAX, EPS, JSTART.
C*
C*    ADDITIONAL SUBROUTINES REQUIRED ARE..
C*    COSETA(METH,NO,EL,TQ,MAXDER) SETS METHOD COEFFICIENTS.
C*    IT IS PROVIDED IN THE PACKAGE.
C*    DIFFUN(N,T,Y,YDOT) COMPUTES DY/DT GIVEN T AND Y.
C*
C*****
C*    USE STORAGE
C*    PARAMETER (NO=N)
C*    COMMON /STCOS/ EL(LMAX),TQ(4),NQ
C*    COMMON /STFUN/ NOFF,NSAVE,FSAVE(2*N)
C*    DIMENSION YMAX(N),ERROR(N)
C*    KFLAG = 0
C*    TOLD = T
C*    IF (JSTART.GT.O) GO TO 200
C*    IF (JSTART.NE.O) GO TO 150
C*****
C*    ON THE FIRST CALL, THE ORDER IS SET TO 1 AND THE INITIAL
C*    DERIVATIVES ARE CALCULATED. YMAX IS INITIALIZED USING THE INITIAL
C*    Y AND YDOT. IF BOTH ARE INITIALLY ZERO IN ANY COMPONENT, THE DEFAULT
C*    VALUE IS 1. RMAX IS THE MAXIMUM RATIO BY WHICH H CAN BE INCREASED
C*    IN A SINGLE STEP. IT IS INITIALLY 1.E4 TO COMPENSATE FOR THE SMALL
C*    INITIAL H, BUT THEN IS NORMALLY EQUAL TO 10. IF A FAILURE
C*    OCCURS (IN CORRECTOR CONVERGENCE OR ERROR TEST), RMAX IS SET AT 2
C*    FOR THE NEXT INCREASE.
C*****
C*    NOFF=0
C*    NSAVE=1
C*    CALL FUNCT
C*    DO 110 I = 1,N
C*      Y(I,2)=FSAVE(I)*H
C*      AYI = ABS(Y(I,1))
C*      IF (AYI.EQ.O.) AYI = ABS(Y(I,2))

```

```

      IF (AY1.EQ.0.) AY1 = 1.
110   YMAX(1) = AY1
      XOLD=Y(1)
      ZOLD=Y(3)
      NQ = 1
      L = 2
      RMAX = 1.E4
      CRATE = 1.
      HOLD = H
C*****
C*      COSETA IS CALLED
C* TO SET THE COEFFICIENTS OF THE METHOD. IF THE CALLER HAS CHANGED
C* EPS, THE CONSTANTS E, EDN, EUP, AND BND MUST BE RESET.
C* E IS A COMPARISON FOR ERRORS OF THE CURRENT ORDER NQ. EUP IS
C* TO TEST FOR INCREASING THE ORDER, EDN FOR DECREASING THE ORDER.
C* BND IS USED TO TEST FOR CONVERGENCE OF THE CORRECTOR ITERATES.
C* IF THE CALLER HAS CHANGED H, Y MUST BE RESCALED.
C* IF H HAS BEEN CHANGED, IDOUB IS RESET TO L + 1 TO PREVENT
C* FURTHER CHANGES IN H FOR THAT MANY STEPS.
C*****
      IDOUB = L + 1
      IRET = 1
130   CALL COSETA
140   EDM = (TQ(1)*EPS)**2
      E = (TQ(2)*EPS)**2
      EUP = (TQ(3)*EPS)**2
      BND = (TQ(4)*EPS)**2
      GO TO ( 160, 170, 200 ) IRET
150   IF(EPS.EQ.EPSOLD) GO TO 160
      IRET=1
      GO TO 140
160   EPSOLD = EPS
      IF (H.EQ.HOLD) GO TO 200
      RH = H/HOLD
      H = HOLD
170   RH = AMAX1(RH,HMIN/ABS(H))
      RH = AMIN1(RH,HMAX/ABS(H),RMAX)
      RR=1.
      DO 180 J = 2,L
      RR=RR*RH
      DO 180 I = 1,N
180   Y(I,J)=Y(I,J)*RR
      H = H*RH
      IDOUB = L + 1
      IF (T.NE.TOLD) GO TO 690
C*****
C* THIS SECTION COMPUTES THE PREDICTED VALUES BY EFFECTIVELY
C* MULTIPLYING THE Y ARRAY BY THE PASCAL TRIANGLE MATRIX.
C*****
200   T = T + H
      DO 210 J1 = 1,NQ
      DO 210 J2 = J1,NQ
      J = NQ - J2 + J1
      DO 210 I = 1,N
210   Y(I,J) = Y(I,J) + Y(I,J+1)
C*****
C* UP TO 3 CORRECTOR ITERATIONS ARE TAKEN. CONVERGENCE IS TESTED
C* BY REQUIRING CHANGES TO BE LESS THAN BND, WHICH IS DEPENDENT ON
C* EPS, IN EUCLIDEAN NORM. THE SUM OF THE CORRECTIONS IS ACCUMULATED
C* IN THE VECTOR ERROR(1). IT IS APPROXIMATELY EQUAL TO THE L-TH

```

```

C= DERIVATIVE OF Y MULTIPLIED BY H**L/(FACTORIAL(L-1)*EL(L)), AND IS
C= THUS PROPORTIONAL TO THE ACTUAL ERRORS TO THE LOWEST POWER OF
C= H PRESENT (H**L).
C= THE Y ARRAY IS NOT ALTERED IN THE CORRECTION LOOP. THE UPDATED
C= Y VECTOR IS STORED TEMPORARILY IN FSAVE. THE NORM OF THE
C= ITERATE DIFFERENCE IS STORED IN D.
C=====
220 DO 230 I = 1,N
230 ERROR(I) = 0.
    M = 0
    NOFF=N
    NSAVE=2
    CALL FUNCT
C=====
C= TO PERFORM THE FUNCTIONAL ITERATION, UPDATE Y DIRECTLY FROM
C= THE RESULT OF THE ' ' DIFFUN CALL.
C=====
340 D = 0
    DO 350 I = 1,N
        R = H*FSAVE(I+NO) - Y(I,2)
        D = D + ((R-ERROR(I))/YMAX(I))**2
        FSAVE(I) = Y(I) + EL(I)*R
    350 ERROR(I) = R
C=====
C= TEST FOR CONVERGENCE. IF M.GT.0, AN ESTIMATE OF THE CONVERGENCE
C= RATE CONSTANT IS STORED IN CRATE, AND THIS IS USED IN THE TEST.
C=====
410 IF (M.NE.0) CRATE = AMAX1(.9*CRATE,D/D1)
    IF ((D*AMIN1(1.,2.*CRATE)).LE.BND) GO TO 450
    D1 = D
    M = M + 1
    IF (M.EQ.3) GO TO 420
    NOFF=N
    NSAVE=3
    CALL FUNCT
    GO TO 340
C=====
C= THE CORRECTOR ITERATION FAILED TO CONVERGE IN 3 TRIES.
C= THE Y ARRAY IS RETRACTED TO ITS VALUES
C= BEFORE PREDICTION, AND M IS REDUCED, IF POSSIBLE. IF NOT, A
C= NO-CONVERGENCE EXIT IS TAKEN.
C=====
420 IF (ABS(H).LE.(HMIN=1.00001)) GO TO 880
    T=TOLD
    RMAX = 2.
    DO 430 J1 = 1,NQ
        DO 430 J2 = J1,NQ
            J = NQ - J2 + J1
            DO 430 I = 1,N
                Y(I,J) = Y(I,J) - Y(I,J+1)
    430 RH = .25
    GO TO 170
C=====
C= THE CORRECTOR HAS CONVERGED.
C= THE ERROR TEST IS MADE AND CONTROL PASSES TO
C= STATEMENT 500 IF IT FAILS.
C=====
450 D = 0.
    DO 460 I = 1,N
        D = D + (ERROR(I)/YMAX(I))**2
    460

```

```

      IF (D.GT.E) GO TO 500
C*****
C= AFTER A SUCCESSFUL STEP, UPDATE THE Y ARRAY AND YMAX.
C= CONSIDER CHANGING H IF IDOUB = 1. OTHERWISE DECREASE IDOUB BY 1.
C= IF IDOUB IS THEN 1 AND NO.LT. MAXDER, THEN ERROR IS SAVED FOR
C= USE IN A POSSIBLE ORDER INCREASE ON THE NEXT STEP.
C= IF A CHANGE IN H IS CONSIDERED, AN INCREASE OR DECREASE IN ORDER
C= BY ONE IS CONSIDERED ALSO. A CHANGE IN H IS MADE ONLY IF IT IS BY A
C= FACTOR OF AT LEAST 1.1. IF NOT, IDOUB IS SET TO 10 TO PREVENT
C= TESTING FOR THAT MANY STEPS.
C*****
      KFLAG = 0
      HS2=H
      DO 470 J = 1,L
        DO 470 I = 1,N
470      Y(I,J) = Y(I,J) + EL(J)*ERROR(I)
      DO 480 I = 1,N
480      YMAX(I) = AMAX1(YMAX(I),ABS(Y(I)))
      IF (IDOUB.EQ.1) GO TO 520
      IDOUB = IDOUB - 1
      IF (IDOUB.GT.1) GO TO 700
      IF (NO.EQ.MAXDER) GO TO 700
      DO 490 I = 1,N
490      Y(I,LMAX) = ERROR(I)
      GO TO 700
C*****
C= THE ERROR TEST FAILED. KFLAG KEEPS TRACK OF MULTIPLE FAILURES.
C= RESTORE T AND THE Y ARRAY TO THEIR PREVIOUS VALUES, AND PREPARE
C= TO TRY THE STEP AGAIN. COMPUTE THE OPTIMUM STEP SIZE FOR THIS OR
C= ONE LOWER ORDER.
C*****
      500 IF (ABS(H).LE.(HMIN*1.00001)) GO TO 660
      KFLAG=KFLAG+1
      IF (KFLAG.EQ.-7) GO TO 670
      T = TOLD
      DO 510 J1 = 1,NO
        DO 510 J2 = J1,NO
          J = NO - J2 + J1
          DO 510 I = 1,N
510      Y(I,J) = Y(I,J) - Y(I,J+1)
      RMAX = 2.
      IF (KFLAG.LE.-3) GO TO 540
      PR3 = 1.E+20
      GO TO 540
C*****
C= REGARDLESS OF THE SUCCESS OR FAILURE OF THE STEP, FACTORS
C= PR1, PR2, AND PR3 ARE COMPUTED, BY WHICH H COULD BE DIVIDED
C= AT ORDER NO - 1 ORDER NO OR ORDER NO + 1, RESPECTIVELY.
C= IN THE CASE OF FAILURE, PR3 = 1.E20 TO AVOID AN ORDER INCREASE.
C= THE SMALLEST OF THESE IS DETERMINED AND THE NEW ORDER CHOSEN
C= ACCORDINGLY. IF THE ORDER IS TO BE INCREASED, WE COMPUTE ONE
C= ADDITIONAL SCALED DERIVATIVE.
C*****
      520 PR3 = 1.E+20
      IF (NO.EQ.MAXDER) GO TO 540
      D1 = 0.
      DO 530 I = 1,N
530      D1 = D1 + ((ERROR(I) - Y(I,LMAX))/YMAX(I))*2
      ENQ3 = 0.5/FLOAT(L+1)
      PR3 = ((D1/EUF)*ENQ3)*1.4 + 1.4E-6

```

```

540 ENQ2 = 0.5/LOAT(L)
    PR2 = ((D/E)*ENQ2)*1.2 + 1.2E-6
    PR1 = 1.2*20
    IF (NC.EQ.1) GO TO 560
    D = 0.0
    DO 550 I = 1,N
        D = D + (Y(I,L)/YMAX(I))*2
    ENQ1 = 0.5/LOAT(NQ)
    PR1 = ((D/EDN)*ENQ1)*1.3 + 1.3E-6
560 IF (PR2.LE.PR3) GO TO 570
    IF (PR3.LT.PR1) GO TO 590
570 IF (PR2.GT.PR1) GO TO 580
    NEWQ = NQ
    RH = 1./PR2
    GO TO 620
580 NEWQ = NQ - 1
    RH = 1./PR1
    GO TO 620
590 NEWQ = L
    RH = 1./PR3
    IF (RH.LT.1.1) GO TO 610
    DO 600 I = 1,N
        GO TO 630
    Y(I,NEWQ+1) = ERROR(I)*EL(L)/LOAT(L)
610 IDOUB = 10
    GO TO 700
620 IF ((KFLAG.EQ.0).AND.(RH.LT.1.1)) GO TO 610
C*****
C* IF THERE IS A CHANGE OF ORDER, RESET NQ, L, AND THE COEFFICIENTS.
C* IN ANY CASE H IS RESET ACCORDING TO RH AND THE Y ARRAY IS RESCALED.
C* THEN EXIT FROM 690 IF THE STEP WAS OK, OR REDO THE STEP OTHERWISE.
C*****
    IF (NEWQ.EQ.NQ) GO TO 170
630 NQ = NEWQ
    L = NQ + 1
    IRET = 2
    GO TO 130
C*****
C* CONTROL REACHES THIS SECTION IF 3 OR MORE FAILURES HAVE OCCURED.
C* IT IS ASSUMED THAT THE DERIVATIVES THAT HAVE ACCUMULATED IN THE
C* Y ARRAY HAVE ERRORS OF THE WRONG ORDER. HENCE THE FIRST
C* DERIVATIVE IS RECOMPUTED, AND THE ORDER IS SET TO 1. THEN
C* H IS REDUCED BY A FACTOR OF 10, AND THE STEP IS RETRIED.
C* AFTER A TOTAL OF 7 FAILURES, AN EXIT IS TAKEN WITH KFLAG = -2.
C*****
640 RH = .1
    RH = AMAX1(HMIN/ABS(H),RH)
    H = H*RH
    NOFF=0
    NSAVE=1
    CALL FUNCT
    DO 650 I = 1,N
        Y(I,2) = H*FSAVE(I)
650 IDOUB = 10
    IF (NQ.EQ.1) GO TO 200
    NQ = 1
    L = 2
    IRET = 3
    GO TO 130
C*****

```

```

C* ALL RETURNS ARE MADE THROUGH THIS SECTION.  H IS SAVED IN HOLD
C* TO ALLOW THE CALLER TO CHANGE H ON THE NEXT STEP.
C*****
660 KFLAG = -1
   GO TO 683
670 KFLAG = -2
   GO TO 683
680 KFLAG = -3
683 DO 685 I=1,N
685 Y(I)=FSAVE(I)
   GO TO 700
690 RMAX = 10.
700 HOLD = H
   JSTART = NQ
   RETURN
C*****=***** END OF STIFFA *****
      END

```

```

SUBROUTINE COSETA
C=====
C= COSETA IS CALLED BY STIFFA AND SETS COEFFICIENTS FOR USE THERE.
C= METH IS ASSUMED TO BE 1 HERE.
C= THE VECTOR EL, OF LENGTH NQ + 1, DETERMINES THE BASIC METHOD.
C= THE VECTOR TQ, OF LENGTH 3, IS INVOLVED IN ADJUSTING THE STEP SIZE
C= IN RELATION TO TRUNCATION ERROR. ITS VALUES ARE GIVEN BY THE
C= PERTST ARRAY.
C= THE VECTORS EL AND TQ DEPEND ON NQ.
C= THE MAXIMUM ORDER, MAXDER, MUST BE .LE. 12 FOR ADAMS METHODS.
C= LMAX = MAXDER + 1 IS THE NUMBER OF COLUMNS IN THE Y ARRAY.
C= THE COEFFICIENTS IN PERTST NEED BE GIVEN TO ONLY ABOUT
C= ONE PERCENT ACCURACY. THE ORDER IN WHICH THE GROUPS APPEAR BELOW
C= IS... COEFFICIENTS FOR ORDER NQ - 1, COEFFICIENTS FOR ORDER NQ,
C= COEFFICIENTS FOR ORDER NQ + 1.
C=====
      USE STORAGE
      COMMON /STCOS/ EL(LMAX),TQ(4),NQ
      DIMENSION PERTST(12,3)
      DATA (EL(2)=1.)
      DATA (PERTST = 1.,1.,2.,1.,.3158,.07407,.01391,.002182,
1      .0002945,.00003492,.000003692,.0000003524,
1      2.,12.,24.,37.89,53.33,70.08,87.97,106.9,
1      126.7,147.4,168.8,191.0,
1      12.0,24.0,37.89,53.33,70.08,87.97,106.9,
1      126.7,147.4,168.8,191.0,1.)
      GO TO (101,102,103,104,105,106,107,108,109,110,111,112),NQ
C=====
C* THE FOLLOWING COEFFICIENTS SHOULD BE DEFINED TO
C* MACHINE ACCURACY. FOR EACH ORDER NQ, THEY CAN BE CALCULATED
C* FROM THE GENERATING POLYNOMIAL,
C*  $L(T) = EL(1) + EL(2)*T + \dots + EL(NQ+1)*T**NQ$ .
C* FOR THE IMPLICIT ADAMS METHODS, L(T) IS GIVEN BY
C*  $DL/DT = (T+1)*(T+2)* \dots *(T+NQ-1)/K$ ,  $L(-1) = 0$ ,
C* WHERE  $K = FACTORIAL(NQ-1)$ .
C=====
101 EL(1) = 1.0
   GO TO 900
102 EL(1) = 0.5
   EL(3) = 0.5
   GO TO 900
103 EL(1) = 4.1666666666667E-01
   EL(3) = 0.75
   EL(4) = 1.6666666666667E-01
   GO TO 900
104 EL(1) = 0.375
   EL(3) = 9.1656666666667E-01
   EL(4) = 3.3333333333333E-01
   EL(5) = 4.1666666666667E-02
   GO TO 900
105 EL(1) = 3.4861111111111E-01
   EL(3) = 1.0416366666667
   EL(4) = 4.8611111111111E-01
   EL(5) = 1.0416666666667E-01
   EL(6) = 8.3333333333333E-03
   GO TO 900
106 EL(1) = 3.2986111111111E-01
   EL(3) = 1.1416666666667
   EL(4) = 0.625
   EL(5) = 1.7708333333333E-01

```

```

EL(6) = 0.025
EL(7) = 1.3888888888889E-03
GO TO 900
107 EL(1) = 3.1559193121693E-01
    EL(3) = 1.225
    EL(4) = 7.5185185185185E-01
    EL(5) = 2.5520833333333E-01
    EL(6) = 4.8611111111111E-02
    EL(7) = 4.8611111111111E-03
    EL(8) = 1.9841269841270E-04
GO TO 900
108 EL(1) = 3.0422453703704E-01
    EL(3) = 1.2964285714286
    EL(4) = 8.6851851851852E-01
    EL(5) = 3.3377777777778E-01
    EL(6) = 7.7777777777778E-02
    EL(7) = 1.0548148148148E-02
    EL(8) = 7.9355079355079E-04
    EL(9) = 2.4801087301087E-05
GO TO 900
109 EL(1) = 2.9488800044092E-01
    EL(3) = 1.3589285714286
    EL(4) = 9.755423280423E-01
    EL(5) = 0.4171875
    EL(6) = 1.1135416666667E-01
    EL(7) = 0.01875
    EL(8) = 1.9345238095238E-03
    EL(9) = 1.1160714285714E-04
    EL(10) = 2.7557319223986E-06
GO TO 900
110 EL(1) = 2.8697544642857E-01
    EL(3) = 1.4144841269841
    EL(4) = 1.0772156084656
    EL(5) = 4.9856701940035E-01
    EL(6) = 0.1484375
    EL(7) = 2.9060570987654E-02
    EL(8) = 3.7202380952381E-03
    EL(9) = 2.9968584656085E-04
    EL(10) = 1.3778659611993E-05
    EL(11) = 2.7557319223986E-07
GO TO 900
111 EL(1) = 2.8018959644394E-01
    EL(3) = 1.4644841269841
    EL(4) = 1.1715145502646
    EL(5) = 5.7935819003527E-01
    EL(6) = 1.8832286155203E-01
    EL(7) = 4.1430362654321E-02
    EL(8) = 6.2111441798942E-03
    EL(9) = 6.2520667989418E-04
    EL(10) = 4.0417491523513E-05
    EL(11) = 1.5156525573192E-06
    EL(12) = 2.5052108365442E-08
GO TO 900
112 EL(1) = 2.7426554003150E-01
    EL(3) = 1.5099386724387
    EL(4) = 1.2602711640212
    EL(5) = 6.5923418209877E-01
    EL(6) = 2.3045800264550E-01
    EL(7) = 5.5697246105232E-02
    EL(8) = 9.4394841269841E-03

```


EL(9) = 1.1192749669312E-03
EL(10) = 9.0939153439153E-05
EL(11) = 4.8225308641875E-06
EL(12) = 1.5031265031285E-07
EL(13) = 2.0876756967868E-09

C*

900 DO 910 K=1,3
910 TQ(K)=PERTST(NQ,K)
TQ(4)= .01*TQ(2)
RETURN

C***** END OF COSETA *****
END

SUBROUTINE FUNCT

```

C*****
C= GIVEN THE CURRENT POINT, Y(T), FUNCT COMPUTES THE TIME DERIVATIVE,
C* DY/DT = F(Y,T). TO DO SO, THE MAGNETIC (BX,BY,BZ) AND ELECTRIC
C* (EX,EZ) FORCES ARE COMPUTED AS A FUNCTION OF POSITION OF THE
C* PARTICLE (XNOW,YNOW,ZNOW).
C* INPUT VARIABLES:
C* NOFF - DETERMINES WHERE DERIVATIVES STORED: NOFF=0 MEANS FIRST
C* COLUMN OF FSAVE ARRAY; NOFF=N MEANS SECOND COLUMN.
C* NSAVE - DETERMINES WHERE CURRENT POINT FETCHED FROM: NSAVE=1 OR
C* 2 MEANS USE Y ARRAY; NSAVE=3 MEANS USE FSAVE ARRAY.
C* Y - SEE SUBROUTINE STIFFA
C* FSAVE - INPUT/OUTPUT ARRAY
C*****
      USE STORAGE
      GENERIC PHI
      COMMON /PHIARG/ XPHI,ZPHI
      COMMON /STFUN/ NOFF,NSAVE,FSAVE(2*N)
      PARAMETER (T=OP1=6.2831853)

      IF(NSAVE-3) ,20,
      XNOW=Y(1)
      YNOW=Y(2)
      ZNOW=Y(3)
      XDTN=Y(4)
      YDTN=Y(5)
      ZDTN=Y(6)
      GO TO 30

20 XNOW=FSAVE(1)
   YNOW=FSAVE(2)
   ZNOW=FSAVE(3)
   XDTN=FSAVE(4)
   YDTN=FSAVE(5)
   ZDTN=FSAVE(6)
30 CONTINUE

      IF(XNOW-1.) ,100,100
      IF(XNOW+1.) ,125,125

C   PARTICLE IN EXPANDER - MAGNETIC FIELD - CALCULATE BX,BY,BZ

      R=SQRT((R2+1.+XNOW)*(R2+1.+XNOW)+YNOW*YNOW)
      THETA=ASINF(YNOW/R)
      IF(R-R1) ,50,50

C   PARTICLE BETWEEN RMIRROR AND R1

      A=BMIRROR*(RMIRROR-RCENTR)
      BR=A/(R-RCENTR)
      BZ=ZNOW*(A+RCENTR)/(R*(R-RCENTR)*(R-RCENTR))
      GO TO 60

C   PARTICLE BETWEEN R1 AND R2

50 BB=-BR1/(R2-R1)
   CC=-R2*BB
   BR=BB*R+CC
   BZ=-ZNOW*(2.*BB*R+CC)/R

```

```

60 BX=BR*COSF(THETA)
   BY=BR*SINF(THETA)
   IF(XNOW-XMIN) 70,
   IF(ABSF(ZNOW)-ZFIN) 130,,
70 EX=EZ=0.
   GO TO 200

C   PARTICLE IN COLLECTOR - ELECTRIC FIELD - CALCULATE EX,EZ

100 IF(ABSF(ZNOW)-ZFIN) ,,125

C   PARTICLE IN CHANNEL - E ANALYTIC

   BX=BY=BZ=0.
   EX=-VO*(1.+C*COSF(TWOPI*XNOW)*SINH(TWOPI*ZNOW))
   EZ=-VO*C*SINF(TWOPI*XNOW)*COSH(TWOPI*ZNOW)
   GO TO 150

C   FIND E IN VACUUM

125 BX=BY=BZ=0.
130 XPHI=XNOW
   ZPHI=ZNOW
   TEMP=PHI(1)
   XPHI=XOLD
   IF(XNOW-XOLD) 135,,135
   XPHI=XNOW-1.E-5
135 EX=-(TEMP-PHI(1))/(XNOW-XPHI)
   XPHI=XNOW
   ZPHI=ZOLD
   IF(ZNOW-ZOLD) 140,,140
   ZPHI=ZNOW-1.E-5
140 EZ=-(TEMP-PHI(1))/(ZNOW-ZPHI)

C   FIND E DUE TO SPACE CHARGE

150 IF(ISC) 200,
   IF(ABSF(ZNOW)-ZFIN) ,,200
   XPHI=XNOW
   ZPHI=ZNOW
   TEMP=PHI(2)
   XPHI=XOLD
   IF(XNOW-XOLD) 155,,155
   XPHI=XNOW-1.E-5
155 EX=EX-(TEMP-PHI(2))/(XNOW-XPHI)
   XPHI=XNOW
   ZPHI=ZOLD
   IF(ZNOW-ZOLD) 160,,160
   ZPHI=ZNOW-1.E-5
160 EZ=EZ-(TEMP-PHI(2))/(ZNOW-ZPHI)

C   COMPUTE DERIVATIVES USING BX,BY,BZ AND EX,EZ

200 FSAVE(NOFF+1)=XDTN
   FSAVE(NOFF+2)=YDTN
   FSAVE(NOFF+3)=ZDTN
   FSAVE(NOFF+4)=QML*EX+QM*(YDTN*BZ-ZDTN*BY)
   FSAVE(NOFF+5)=QM*(ZDTN*BX-XDTN*BZ)
   FSAVE(NOFF+6)=QML*EZ+QM*(XDTN*BY-YDTN*BX)
   RETURN

```

END

```

      GENERIC PHI
      FUNCTION PHI(IFLAG)
C*****
C* PHI COMPUTES THE POTENTIAL ENERGY (VOLTS) OF A PARTICLE IN A VACUUM,
C* OR DUE TO SPACE CHARGE, AS A FUNCTION OF ITS POSITION.
C* INPUT VARIABLES:
C*   X - X-COORDINATE OF PARTICLE
C*   Z - Z-COORDINATE OF PARTICLE
C*   IFLAG - IF IFLAG=1, COMPUTE VACUUM POTENTIAL
C*           IF IFLAG=2, COMPUTE SPACE CHARGE POTENTIAL
C*****
      USE STORAGE
      COMMON /PHIARG/ X,Z
      COMMON /PHILAG/ XT,ZT
      GENERIC LAG2
      REAL LAG2
      PARAMETER (TWOP1=6.2831853)

      GO TO (10,100),IFLAG

C    FIND POTENTIAL OF PARTICLE IN VACUUM

10  IF(X-XMIN) 15,
    IF(X-1.) 60,20,20
15  PHI=0.
    RETURN

20  IF(ABSF(Z)-ZFIN) 80,80,

C    PARTICLE IN TABLE-1 REGION -
C    REDUCE COORDINATES TO CUP TABLE BOUNDARIES

      XNX=INTF(X+BOUND)
      XRM=(X+BOUND)-XNX
      IF(Z) 30,,
      XT=XRM
      ZT=Z
      GO TO 50

30  IF(XRM-.5) 35,35
      XT=(XRM+.5)-INTF(XRM+.5)
      XNX=XNX-.5
      GO TO 40
35  XT=XRM-.5
      XNX=XNX+.5
40  ZT=-Z

50  PHI=VO*(LAG2(1)+XNX)
    RETURN

C    PARTICLE IN TABLE-2 REGION

60  XT=X
    ZT=Z
    PHI=VO*LAG2(2)
    RETURN

C    PARTICLE IN ANALYTIC REGION

```

```

80 IF(X-XMAX) 20,90,
   X=XMAX
90 PHI=VO*(X+(C/TWOPI))*SINF(TWOPI*X)*SINF(TWOPI*Z)
   RETURN
C    FIND SPACE CHARGE POTENTIAL OF PARTICLE
100 IF(X-XMIN) 110,
    IF(ABSF(Z)-ZFIN) 120,120,
110 PHI=0.
    RETURN
C    PARTICLE IN TABLE-3 REGION
120 XT=X
    ZT=Z
    PHI=VO*LAG2(3)
    RETURN
END

```

```

GENERIC LAG2
REAL LAG2
FUNCTION LAG2(NT)
C*****
C* LAG2 IS A MODIFICATION OF CIC REPORT E1.1-006, AND PERFORMS TWO-
C* DIMENSIONAL INTERPOLATION ON A SPECIFIED POTENTIAL TABLE. THE
C* DEGREE OF THE INTERPOLATING POLYNOMIAL IS SET USING THE VARIABLE
C* NE, AND IS NOW 2.
C* INPUT VARIABLES:
C* XT - X-COORDINATE OF POINT
C* ZT - Z-COORDINATE OF POINT
C* NT - SPECIFIES WHICH TABLE TO BE USED FOR INTERPOLATION (NT=1
C* MEANS POTAB1, NT=2 MEANS POTAB2, NT=3 MEANS POTAB3)
C* THE TABLES POTAB1, POTAB2, AND POTAB3 MUST BE SUPPLIED.
C* THE ARRAYS I3 AND K3 CONTAIN THE MOST RECENT VALUES OF THE POINTERS
C* TO THE COORDINATE ARRAYS (XTAB,ZTAB) FOR EACH POTENTIAL TABLE.
C*****

USE STORAGE
PARAMETER (NE=2,ND=NE+1,NEX=ND/2+1)
COMMON /PHILAG/ XT,ZT
DIMENSION I3(3),K3(3)
DATA (I3=1,LPT1+1,LPT1+LPT2+1),(K3=ILIM1+1,L1+1+ILIM2+1,LPT1+LPT2+
* ILIM3+1)

C SET POINTERS TO SPECIFIED TABLE

IF(NT-2) ,3,6
L1=0
L2=ILIM1
L3=ILIM1+KLIM1
L4=ILIM1
L5=KLIM1
GO TO 8

3 L1=LPT1
L2=LPT1+ILIM2
L3=LPT1+ILIM2+KLIM2
L4=ILIM2
L5=KLIM2
GO TO 8

6 L1=LPT1+LPT2
L2=LPT1+LPT2+ILIM3
L3=LPT1+LPT2+ILIM3+KLIM3
L4=ILIM3
L5=KLIM3
8 CONTINUE
IT3=I3(NT)
KT3=K3(NT)

C LOCATE X-COORDINATE IN X ARRAY - SET IT3

IF(XT-TABLE(L1+1)) ,10,10
XT=TABLE(L1+1)
GO TO 15
10 IF(XT-TABLE(L2)) 15,15,
XT=TABLE(L2)
15 IF(XT-TABLE(IT3)) , ,30

```

```

    ARITHREG I,J,AT
    J=L1+1
    AT=XT
    DO 20 I=IT3-1,J,-1
    IF(AT-TABLE(I))',,25
20  CONTINUE
25  IT3=I3(NT)=I+1
    RELEASE I,J,AT
    GO TO 45

30  DO 35 I=IT3+1,L2
    IF(XT-TABLE(I)) 40,40,
35  CONTINUE
40  IT3=I3(NT)=I

C    LOCATE Z-COORDINATE IN Z ARRAY - SET KT3

45  IF(ZT-TABLE(L2+1)) ,50,50
    ZT=TABLE(L2+1)
    GO TO 55
50  IF(ZT-TABLE(L3)) 55,55,
    ZT=TABLE(L3)

55  IF(ZT-TABLE(KT3)) ,70
    ARITHREG K,L,AT
    L=L2+1
    AT=ZT
    DO 60 K=KT3-1,L,-1
    IF(AT-TABLE(K))',,65
60  CONTINUE
65  KT3=K3(NT)=K+1
    RELEASE K,L,AT
    GO TO 85

70  DO 75 K=KT3+1,L3
    IF(ZT-TABLE(K)) 80,80,
75  CONTINUE
80  KT3=K3(NT)=K

C    SET INTERPOLATION LIMITS IP AND KP

85  IP=IT3-NEX
    IF(IP-L1) 90,90
    IP=L1
    GO TO 100
90  IF(IP-(L2-ND)) 100,100,
    IP=L2-ND

100  KP=KT3-NEX
    IF(KP-L2) ,110,110
    KP=L2
    GO TO 120
110  IF(KP-(L3-ND)) 120,120,
    KP=L3-ND

C    PERFORM 2-D INTERPOLATION

120  PTOT=0
    L6=L3+(KP-L2-1)*L4+IP-L1
    ARITHREG PTE

```

```

DO 150 I=1,ND
DO 150 K=1,ND
PIE=1.

ARITHREQ T1,T2
T1=TABLE(IP+I) $ T2=TABLE(KP+K)
DO 140 L=1,ND
IF(L-I) PIE=PIE*(XT-TABLE(IP+L))/(T1-TABLE(IP+L))
IF(L-K) PIE=PIE*(ZT-TABLE(KP+L))/(T2-TABLE(KP+L))
140 CONTINUE
RELEASE T1,T2

150 PTOT=PTOT+PIE*TABLE(L6+K*L4+1)
RELEASE PIE
LAG2=PTOT
200 RETURN
END

```



```

SUBROUTINE QDENT
C*****
C* QDENT GENERATES A CHARGE DENSITY TABLE (QT) WHICH IS USED BY THE
C* POT3 CODE. THE TABLE INCLUDES THE COLLECTOR CHANNEL AND NEGATIVE
C* STAGE REGION. AFTER THE INTEGRATION TIME HAS PROCEEDED AT
C* LEAST ONE SNAPSHOT INTERVAL (SSDT), LINEAR INTERPOLATION IS PER-
C* FORMED BETWEEN THE PREVIOUS CHARGE DENSITY POINT AND THE CURRENT
C* ONE. CHARGE IS THEN DISTRIBUTED BY BIRDSALL'S SCHEME AT EACH
C* SNAPSHOT TIME ALONG THE LINE.
C* INPUT VARIABLES:
C* Y - CURRENT X- AND Z-COORDINATES IN Y(1), Y(3)
C* I3 AND K3 CONTAIN LAST VALUES OF POINTERS TO COORDINATE ARRAYS QX,QZ.
C*****
      USE STORAGE
      DATA (I3=INQ),(K3=KNQ)

C      IF FIRST POINT, NO INTERPOLATION PERFORMED

      IF(IGCD) 5,5,
      SST=T
      X=Y(1)
      Z=Y(3)
      GO TO 15

C      PERFORM LINEAR INTERPOLATION

      5 CONST=1./(T-TQDD)
      SLOPEX=(Y(1)-XQDD)*CONST
      XINC=SLOPEX*SSDT
      SLOPEZ=(Y(3)-ZQDD)*CONST
      ZINC=SLOPEZ*SSDT
      X=XQDD+SLOPEX*(SST-TQDD)
      Z=ZQDD+SLOPEZ*(SST-TQDD)

C      LOCATE X-COORDINATE IN X ARRAY - SET I3

      15 IF(X-QX(I3)) ,30,30
      DO 20 I=I3-1,INQ,-1
      IF(X-QX(I)) ,25,25
      20 CONTINUE
      25 I3=I
      GO TO 60
      30 DO 40 I=I3+1,IPQ-1
      IF(X-QX(I)) 50,,
      40 CONTINUE
      50 I3=I-1

C      LOCATE Z-COORDINATE IN Z ARRAY - SET K3

      60 IF(Z-QZ(K3)) ,90,90
      DO 70 K=K3-1,KNQ,-1
      IF(Z-QZ(K)) ,80,80
      70 CONTINUE
      80 K3=K
      GO TO 120
      90 DO 100 K=K3+1,KPQ-1
      IF(Z-QZ(K)) 110,,
      100 CONTINUE
      110 K3=K-1

```

```

C    ASSIGN PARTICLE CHARGE TO SURROUNDING MESH POINTS
120 QT(I3,K3)=QT(I3,K3)+RHOCON*(QX(I3+1)-X)*(QZ(K3+1)-Z)
    QT(I3+1,K3)=QT(I3+1,K3)+RHOCON*(X-QX(I3))*(QZ(K3+1)-Z)
    QT(I3+1,K3+1)=QT(I3+1,K3+1)+RHOCON*(X-QX(I3))*(Z-QZ(K3))
    QT(I3,K3+1)=QT(I3,K3+1)+RHOCON*(QX(I3+1)-X)*(Z-QZ(K3))
C    GET NEXT SNAPSHOT TIME AND POSITION
    IF(T-(SST+SSDT)) 200,,
    SST=SST+SSDT
    X=X+XINC
    Z=Z+ZINC
    GO TO 15
C    UPDATE VARIABLES
200 TQQD=T
    XQQD=Y(1)
    ZQQD=Y(3)
    RETURN
    END

```

```

SUBROUTINE IMP
C=====
C* IMP COMPARES COORDINATES OF CURRENT PARTICLE POSITION WITH LOCATION
C* OF BOUNDARIES OF DIRECT CONVERTER. IF A 'HIT' HAS OCCURRED, THEN
C* A LINEAR EXTRAPOLATION IS PERFORMED, USING THE PREVIOUS TWO
C* POINTS. AN ACCURATE 'IMPACT TIME' IS FOUND BY TAKING SUCCESSIVELY
C* SMALLER TIME STEPS ALONG THE LINE UNTIL HIT OCCURS AGAIN.
C* IMP ALSO CHECKS FOR THE ION PASSING THRU A GRID, AND IF
C* ISPAR.NE. 0, IMP SAVES COORDINATES FOR STARTING SECONDARIES
C* INPUT VARIABLES:
C* Y - CURRENT X- AND Z-COORDINATES IN Y(1), Y(3)
C* OUTPUT VARIABLES:
C* TEX - IMPACT TIME
C* NC - NC=0 MEANS NO IMPACT, NC>0 MEANS IMPACT, NC<0 MEANS
C* PARTICLE STRUCK MIRROR
C* IPS - CURRENT PARTICLE STATUS - SEE DART DESCRIPTION
C=====

USE STORAGE
COMMON /DRIMP/ NC
COMMON /PHI
COMMON /PHIARG/ XPHI,ZPHI
PARAMETER (TWOP1=6.2831853)

NC=0
2 X=Y(1)
2 Y=Y(3)
5 IF(X+1.) ,50,50

C CHECK FOR HIT IN EXPANDER

IF(XOLD+1.) 10,,
IPS=IPS+10
GO TO MISS

10 R=SQRT((R2+1.+X)*(R2+1.+X)+Y(2)*Y(2))
IF(R-RMIRROR) ,,20
IPS=IPS+100
NC=-1
GO TO 300
20 IF(ABSF(Z)-EXHT) MISS,HIT,HIT

C CHECK FOR HIT IN NEGATIVE STAGE

50 IF(X-1.) ,100
IF(ABSF(Z)-ZFIN) MISS,HIT,HIT

C CHECK FOR HIT IN COLLECTOR

100 IF(X-XMAX) ,HIT,HIT
AZ=ABSF(Z)
AZOLD=ABSF(ZOLD)
IF(AZ-COLHT) HIT,HIT
IF(AZ-ZFIN) ,110,110
IF(AZOLD-ZFIN) MISS,,
IPS=IPS+1

C CHECK FOR HIT ON FIN

110 NX=XP=2.*X

```

```

BD2=2.*BOUND
XMX=XP-NX
IF(XMX-BD2) 120,120,
IF(XMX+BD2-1.) ,120,120
NX0=2.*XOLD
IF(NX-NX0) ,128,

IF(Z) 115,,
IF(ABSF((X-INTF(X))-.5)-.1) 125,120,120
115 IF(ABSF((X-INTF(X))-.5)-.1) 120,125,125
120 IF(AZ-ZFIN-18.*DELZ) MISS,HIT,HIT
125 IF(AZ-ZFIN-18.*DELZ) MISS,HIT,HIT

C CHECK FOR HIT ON ROD

128 XX=XMX/2.
ARG=(TWOPI/CCONT)*(VR-XX)/SINF(TWOPI*XX)
ZEP=LOGF(ARG*SQRT(ARG*ARG+1.))/TWOPI
ZEP=ZEP+(ZFIN-ZCONT)
IF(NX.INT.1) ,130,
IF(Z) 133,136,136
130 IF(Z) 136,,
133 IF(AZ-ZEP) MISS,,
IF(AZ-ZFIN-30.*DELZ) HIT,HIT,MISS

C CHECK FOR ION PASSING THRU GRID

136 IF(QM) 142,142,
IF(AZ-ZFIN-30.*DELZ) ,,142
XX=MODF(XOLD,.5)
ARG=(TWOPI/CCONT)*(VR-XX)/SINF(TWOPI*XX)
ZOLDEP=LOGF(ARG*SQRT(ARG*ARG+1.))/TWOPI
ZOLDEP=ZOLDEP+(ZFIN-ZCONT)
IF((AZ-ZEP)*(AZOLD-ZOLDEP)) ,142,142
IF(NSSPG+1-MXSSP) ,142,
NSSPG=NSSPG+1
IF(1SPAR) ,142,

C IF ION PASSED THRU GRID AND SECONDARIES ARE INCLUDED, THEN
C EXTRAPOLATE BACK ALONG TRAJECTORY UNTIL RISING POTENTIAL FOUND

XPHI=YS(1,1)
ZPHI=YS(3,1)
VNEW=PHI(1)
XINC=YS(1,1)-XOLD
ZINC=YS(3,1)-ZOLD
DO 137 I=1,20
VOLD=VNEW
XPHI=XPHI+XINC
ZPHI=ZPHI+ZINC
VNEW=PHI(1)
IF(VNEW-VOLD) ,,138
137 CONTINUE
SSP(1,NSSPG)=0.
GO TO 142

C SAVE COORDINATES FOR SECONDARY STARTING CONDITIONS

138 SSP(1,NSSPG)=XPHI*XL
SSP(2,NSSPG)=0.

```

```

      SSP(3,NSSPG)=ZPHI*XL
      DO 139 I=4,N
139    SSP(I,NSSPG)=YS(I,1)*XL
142  IF(Z) 145,150,150
C      CHECK FOR HIT IN GRID CUP
145  XX=X-INTF(X)*.5
      ZZ=-Z-ZFIN
      XO=XOLD-INTF(XOLD)*.5
      ZO=ZOLD-ZFIN
      GO TO 155
150  XX=X-INTF(X)
      ZZ=Z-ZFIN
      XO=XOLD-INTF(XOLD)
      ZO=ZOLD-ZFIN
155  IF(ZZ-92.*DELZ),158,158
      IF(XX-63.5*DELX),,MISS
      IF(ZZ-49.*DELZ) MISS,
158  IF(ZZ-52.*DELZ) HIT,HIT,MISS
      IF((ABSF(ABSF(XX-.75)-24.*DELX)-(ABSF(XO-.75)-24.*DELX)) HIT,HIT,MISS
      IF((ABSF(XX-.75)-24.*DELX)*(ABSF(XO-.75)-24.*DELX)) HIT,MISS,MISS
C      JUMP HERE IF IMPACT DETECTED
      HIT NC=NC+1
      GO TO (160,160,200),NC
C      FIRST IMPACT - SET UP LINEAR EXTRAPOLATION
160  CONST=1./HS1
      SLOPEX=(XOLD-YS(1,1))*CONST
      SLOPEZ=(ZOLD-YS(3,1))*CONST
      X=XOLD
      Z=ZOLD
      TIME=TOLD
      TINC=.1*HS2
      GO TO 250
C      SECOND IMPACT - CUT STEP SIZE
180  TIME=TIME-TINC
      X=X-SLOPEX*TINC
      Z=Z-SLOPEZ*TINC
      TINC=.01*TINC
      GO TO 250
C      THIRD IMPACT - SAVE FINAL TIME
200  TEX=TIME-TINC
      GO TO 300
C      JUMP HERE IF NO IMPACT DETECTED
      MISS IF(NC) ,300,
C      CHECK IF EXTRAPOLATED LINE MISSED IMPACT SURFACE
      CONTINUE STEPPING ALONG LINE IF OK
250  IF(TIME-T) 260,260,

```

```
      TEX=T  
      GO TO 300  
260    TIME=TIME+TINC  
      X=X+SLOPEX*TINC  
      Z=Z+SLOPEZ*TINC  
      GO TO 5  
300    RETURN  
      END
```

```

SUBROUTINE PARPLT
C*****
C* PARPLT PLOTS THE PARTICLE TRAJECTORIES SUPER-IMPOSED ON THE COLLECTOR
C* OF A DIRECT CONVERTER. THE TRAJECTORIES ARE PLOTTED IN GROUPS OF
C* FIVE, WITH EACH NUMBER AND INITIAL ENERGY IDENTIFIED.
C* INPUT VARIABLES:
C* WIN - ARRAY OF INITIAL ENERGIES
C* II - CURRENT ION NUMBER
C* JJ - CURRENT SECONDARY NUMBER
C* IPL0T - IPL0T=1 MEANS PLOT COLLECTIONS ONLY; IPL0T=2 MEANS PLOT
C* ENTIRE TRAJECTORIES.
C* KPL0T - NUMBER OF TRAJECTORIES IN THIS GROUP (USUALLY 5)
C* THE ARRAYS XPL0T, ZPL0T CONTAINING POINTS TO PLOT, AND NOPP CONTAIN-
C* ING THE NUMBER OF POINTS ARE REQUIRED.
C*****
CRT1 FORMAT(X,13)
CRT2 FORMAT(X,13,F16.4)

USE STORAGE
COMMON /MAINPP/ WIN(MXNP),II,JJ

PARAMETER (NGP=20)
DIMENSION GRIDX(NGP),GRIDY(NGP),GRX(NGP),GRY(NGP)
DATA (GRIDX=.045833,.067325,.088816,.110307,.131798,.153289,
* .174781,.196272,.217763,.239254,.260746,.282237,.303728,.325219,
* .346711,.368202,.389693,.411184,.432675,.454167)
DATA (GRIDY=.593683,.533699,.491841,.460628,.436642,.416030,
* .403772,.393138,.385731,.381304,.379739,.381035,.385301,.392779,
* .403685,.419286,.440061,.468048,.506667,.563407)

C FIND MAXIMUM X-COORDINATE OF ENDPOINTS IN GROUP

XMA=XPL0T(NOPP(1),1)
DO 3 K=2,KPL0T
3 XMA=MAX1F(XMA,XPL0T(NOPP(K),K))
XMA=MIN1F(XMA,INTF(XMA)+1.)
IF(IPL0T-2).5,
XMX=XMA
XMN=XMX-3.
GO TO 15

C DEFINE MAPPING

5 XMN=-2.
10 XMX=XMN+3.
15 YMX=1.4
YMN=-YMX
CALL MAPS(XMN,XMX,YMN,YMX,.128069,.985211,.2,1.)

C PLOT COLLECTOR CUP FINS

Y1=ZFIN+18.*DELZ
Y2=YMX
Y3=-ZFIN
Y4=-YMX
IF(XMX-INTF(XMX)).20,
Y1=-Y1 $ Y2=-Y2
Y3=-Y3 $ Y4=-Y4
20 XYG=XMX-.5

```

```

CALL SETCRT(XXG,Y1,1)
DO 30 I=1,5
CALL LINE(XXG,Y1,XXG,Y2)
CALL LINE(XXG,Y3,XXG,Y4)
Y1=-Y1 $ Y2=-Y2
Y3=-Y3 $ Y4=-Y4
30 XXG=XXG-.5

C   PLOT ROD AND GRID CONTOURS
DO 40 I=1,NGP
40 GRY(I)=GR1DY(I)+(ZF1N-.380166)
IF(XMX-INTF(XMX)) ,60,
DO 50 I=1,NGP
50 GRY(I)=-GRY(I)

60 XXG=XMX-.5
DO 90 I=1,6
DO 70 J=1,NGP
70 GRX(J)=XXG+.5-GR1DX(J)
CALL POINTS(GRX,GRY,NGP)
DO 75 J=1,NGP
75 GRX(J)=GR1DX(J)+XXG
DO 80 J=1,NGP
80 GRY(J)=-GRY(J)
CALL TRACE(GRX,GRY,NGP)
90 XXG=XXG-.5

C   PLOT TRAJECTORIES WITH NUMBERS AT ENDPOINTS
C   SOLID TRACE FOR IONS, AND DOTTED TRACE FOR SECONDARIES
DO 130 K=1,KPLOT
IF(QM) ,115
IF(JJ-5) ,105
IF(K-1) ,110
105 CALL TRACEP(XPLOT(1,K),ZPLOT(1,K),NOPP(K),10)
NN=JJ-KPLOT+K
GO TO 125
110 NN=11
GO TO 120
115 NN=11-KPLOT+K
120 CALL TRACE(XPLOT(1,K),ZPLOT(1,K),NOPP(K))
125 CALL SETLCH(XPLOT(NOPP(K),K),ZPLOT(NOPP(K),K))
CALL CONVI(NN,KK,D,3)
CALL CRTBCD(KK,1)
130 CONTINUE

C   PLOT INITIAL ION ENERGIES
NN=11-KPLOT+1
IF(QM) ,140
NN=11
140 CALL SETCH(40.,10.,1,0,0)
CALL CRTBCD(19HNO ENERGY)
CALL SETCH(39.,9.,1,0,0)
WOT 100,CRT2,(K,WIN(K),K=NN,11)

C   LOOP IF ANOTHER FRAME REQUIRED
CALL FRAME

```



```
IF(IPL0T-1) ,150,  
IF(XMX-XMA) ,150,150  
XMN=XMX  
GO TO 10  
150 RETURN  
END
```

```

PROGRAM POT1TEST (POT1DATA,TAPE2=PC11DATA,POT1OUT0,TAPES=POT1OUT0)
OPTIMIZE
C*****
C POT1 GENERATES A TABLE OF NORMALIZED (V/V0) POTENTIAL VALUES
C WHICH REPRESENTS A ROD-GRID PAIR OF CUPS IN THE COLLECTOR OF
C A DIRECT CONVERTER. GIVEN THE FIXED BOUNDARY VALUES, POTENTIALS
C FOR THE INTERIOR POINTS ARE GENERATED BY RELAXATION. THE
C COORDINATE ARRAYS (XTAB,ZTAB) AND POTENTIAL TABLE ARE WRITTEN
C INTO A DISK FILE (POTAB1), WHERE THEY ARE USED BY THE DART CODE.
C INPUT VARIABLES:
C ZFIN - HALF-HEIGHT OF COLLECTOR CHANNEL (UNITS OF L)
C COLHT - HALF-HEIGHT OF COLLECTOR (ZFIN+CUP DEPTH)
C (UNITS OF L)
C C - DETERMINES FOCUSING STRENGTH OF COLLECTOR
C ALPHA - RELAXATION COEFFICIENT (ALPHA>1 MEANS OVER-
C RELAXATION)
C ITERNM - NUMBER OF RELAXATION ITERATIONS
C V1,V2,V3,V4 - FIXED BOUNDARY VALUES
C*****
IN1 FORMAT(4F10.0,15)
IN2 FORMAT(4F10.0)
OUT1 FORMAT(//11HPOT1 INPUT:/5X,5HZFIN=F10.6,5X,6HCOLHT=F10.6,5X,
* 2HC=F10.4,5X,6HALPHA=F10.4,5X,7HITERNM=15)
OUT2 FORMAT(5X,3HV1=F10.4,5X,3HV2=F10.4,5X,3HV3=F10.4,5X,3HV4=,
* F10.4)
OUT3 FORMAT(18,5X,2F15.7)
OUT4 FORMAT(11H1//4HXTAB,6X,10F10.6/(10X,10F10.6))
OUT5 FORMAT(//4HZTAB,6X,10F10.6/(10X,10F10.6))
OUT6 FORMAT(//15,5X,10F10.6/(10X,10F10.6))
OUT7 FORMAT(//5X,14HTIME PERCENTS: X,A10,5X,16HTIME TOTAL(SEC)=,F9.4)
H001 FORMAT(//13HITERATION NO. 6X,8HV(30,50),6X,8HV(91,15))
H002 FORMAT(//4X,1HK,7X,5HTABLE)
ERR1 FORMAT(//26HUNABLE TO CREATE DISK FILE)

PARAMETER (THOP1=6.2831853,IN=0,IP=120,KN=0,KP=117,ILIM=121,
* KLIM=118,NGP=78,NRP=78,MXIT=400)
DIMENSION XTAB((IN,IP)),ZTAB((KN,KP)),V((IN,IP),(KN,KP)),
* ARAY(ILIM+KLIM+1,ILIM+KLIM),LGP(2*NGP),LRP(2*NRP),PLEV(2),
* SAV1(MXIT),SAV2(MXIT)
EQUIVALENCE (ARAY,XTAB),(ARAY(ILIM+1),ZTAB),(ARAY(ILIM+KLIM+1),V)
DATA (LRP=27,0,28,0,29,0,30,0,31,0,32,0,33,0,34,0,35,0,24,1,25,1,
* 26,1,36,1,37,1,38,1,22,2,23,2,39,2,40,2,41,2,20,3,21,3,42,3,18,4,
* 19,4,43,4,17,5,44,5,16,6,45,6,15,7,46,7,14,8,47,8,14,9,48,9,13,
* 10,49,10,12,11,50,11,11,12,51,12,11,13,51,13,10,14,52,14,10,15,
* 53,15,9,16,53,16,9,17,53,17,8,18,53,18,9,19,54,19,7,20,54,20,
* DATA (LRP(117)=7,21,54,21,7,22,54,22,7,23,55,23,6,24,55,24,6,25,
* 55,25,6,26,55,26,6,27,55,27,6,28,55,28,6,29,55,29,6,30,55,30)

CALL OODROP(4RPOT1,5R+POT1,0)
CALL TICHEK(1,J,K,L)
CALL KEEP80(1)
CALL CRT1D14HPOT1,1,1)

C READ CONSTANTS
RIT 2,IN1,ZFIN,COLHT,C,ALPHA,ITERNM
RIT 2,IN2,V1,V2,V3,V4
WOT 3,OUT1,ZFIN,COLHT,C,ALPHA,ITERNM
WOT 3,OUT2,V1,V2,V3,V4

```

C CALCULATE ROD AND GRID POTENTIALS

```

DELX=1./(ILIM-1)
DELZ=(COLHT-ZFIN)/(KLIM-2)
X=.25
Z=ZFIN
TERM=(C/TWOPI)*SINF(TWOPI*X)*SINH(TWOPI*Z)
VR=X+TERM
VG=X+.5-TERM

```

C SET INITIAL TABLE VALUES

```

TERM=(V2-V1)/60.
DO 10 I=IN,60
DO 5 K=KN,35
5 V(I,K)=VR
DO 10 K=36,KP
10 V(I,K)=V1+I*TERM

DO 20 I=61,IP
DO 15 K=KN,35
15 V(I,K)=VG
DO 20 K=36,KP
20 V(I,K)=.6

```

C DEFINE COORDINATE ARRAYS

```

DO 30 I=IN,IP
30 XTAB(I)=I*DELX
DO 40 K=KN,KP
40 ZTAB(K)=K*DELZ+ZFIN

```

C SET FIXED BOUNDARY VALUES V1,V2,V3,V4

```

DO 50 K=KN,KP-1
50 V(IN,K)=V(IN+1,K)=V1
V(IN,KP)=V1

TERM=(V2-V1)/60.
DO 60 I=IN+1,60
60 V(I,KP-1)=V(I,KP)=V1+I*TERM
V(61,KP)=V2
DO 65 I=62,64
DO 65 J=49,52
65 V(I,J)=V2

DO 70 K=18,KP-1
70 V(60,K)=V(61,K)=V2

TERM=(V3-V2)/5.
DO 72 I=62,65
72 V(I,KP)=V(I,KP-1)=V2+(I-61)*TERM
TERM=(V4-V3)/5.
DO 73 I=116,119
73 V(I,KP)=V(I,KP-1)=V3+(I-115)*TERM

V(66,KP)=V(115,KP)=V3
DO 75 K=92,KP-1
75 V(66,K)=V(67,K)=V(114,K)=V(115,K)=V3

```

```

      DO 80 I=67,114
80  V(I,KP)=V(I,KP-1)=V3

      DO 90 K=KN,KP
80  V(IP,K)=V4

C     SET FIXED ROD AND GRID VALUES VR,VG
      DO 100 I=1,2*NRP,2
      LRP(I)=121-LRP(I)
100  LRP(I+1)=LRP(I+1)

      DO 110 I=1,2*NRP,2
110  V(LRP(I),LRP(I+1))=VR
      DO 120 I=1,2*NGP,2
120  V(LGP(I),LGP(I+1))=VG

C     SET CHANNEL BOUNDARY VALUES
      Z=ZF1N
      CONST=(C/TWOPI)*SINH(TWOPI*Z)
      DO 130 I=IN,60
      X=(I-.5)*DELX
      TERM=CONST*SINF(TWOPI*X)
      V(I,KN)=X+TERM
130  V(I+60,KN)=X+.5*TERM

C     RELAXATION-ITERATION LOOP
C     EACH POINT V(I,K) IS CHECKED AGAINST FIXED POINT LOCATIONS BEFORE
C     RELAXING. VALUES AT TWO LOCATIONS ARE SAVED AFTER EACH
C     ITERATION, AND LATER PRINTED TO CHECK FOR CONVERGENCE.
      DDSQ=(DELX*DELX)/(DELZ*DELZ)
      CONST=1./[2.*(1.+DDSQ)]
      WOT 3,HDS1
      DO 250 L=1,ITERM
      DO 200 I=IN+2,IP-1
      DO 200 K=KN+1,KP-2
      IF(I-80) 170,1
      IF(I-61) 180,
      IF(K-18) 180,200,200

C     POINT IN ROD CUP
170  J=1
175  J=IQ8SCH(LRP,J,2*NRP,2,1)
      IF(J) 190,
      IF(K-LRP(J+1)) 200,
      IF(J+1-2*NRP) 190,
      J=J+2
      GO TO 175

C     POINT IN GRID CUP
180  IF(K-49) 185,181,191,
      IF(K-52) 181,191,
      IF(I-66) 182,
      IF(I-67) 182,
      IF(I-114) 182,
      IF(I-115) 190,182,190

```

```

181 IF(I-64) 200,200,190
182 IF(K-92) 190,200,200

185 J=1
186 J=IQBSCH(LGP,J,2*NGP,2,1)
   IF(J) 190
   IF(K-LGP(J+1)) 200,
   IF(J+1-2*NGP) 190,
   J=J+2
   GO TO 185

C   RELAX POINT
190 TERM=(V(I+1,K)+V(I-1,K)+DDSQ=(V(I,K+1)+V(I,K-1)))*CONST
   V(I,K)=V(I,K)+ALPHA*(TERM-V(I,K))
200 CONTINUE
   SAV1(L)=V(30,50)
   SAV2(L)=V(91,15)
250 CONTINUE
   WOT 3,OUT3,(L,SAV1(L),SAV2(L),L=1,ITERM)

C   PRINT AND BUFFER OUT RESULTS
   WOT 3,OUT4,(XTAB(I),I=IN,IP)
   WOT 3,OUT5,(ZTAB(K),K=KN,KP)
   WOT 3,H092
   DO 300 K=KN,KP
   WOT 3,OUT6,K,(V(I,K),I=IN,IP)
300 CONTINUE

   LDF=ILIM+KLIM+ILIM*KLIM+10
   CALL DEVICE(4HOPEN,6RPOTAB1)
   CALL DEVICE(7HDESTROY,6RPOTAB1)
   CALL DEVICE(6HCREATE,6RPOTAB1,LDF,IERR)
   CALL ASSIGN(21,6RPOTAB1)
   IF(IERR) 320,
   WOT 3,ERR1
   GO TO 400
320 BUFFER OUT (21,1) (ARAY(1),ARAY(ILIM+KLIM+ILIM*KLIM),0)
330 IF(UNIT,21) 330,
   CALL DEVICE(6HCLOSER,6RPOTAB1)

C   PLOT EQUIPOTENTIAL CONTOURS IN TABLE (RELAXED)
   CALL MAPS(0.,1.4,0.,1.4)
   PLEV(1)=0.
   PLEV(2)=2
   CALL RCONTR(0,PLEV,0,V(IN,KN),ILIM,XTAB(IN),1,ILIM,1,
   * ZTAB(KN),1,KLIM,1)

C   PLOT EQUIPOTENTIAL CONTOURS IN CHANNEL (ANALYTIC)
   NZ=XMINOF(XINTF(ZFIN/DELZ)-1,KLIM)
   CONST=C/TWOP1
   Z=ZFIN-(NZ+1)*DELZ
   XIN=(IN-1-.5)*DELX
   DO 350 K=NZ,1,-1
   ZTAB(K)=Z+Z*DELZ
   TERM=CONST*SINH(Z/TWOP1)
   X=XIN

```

```

DO 340 I=IN,60
X=X+DELX
V(I,K)=X+TERM*SINF(TWOPI*X)
340 V(I+60,K)=X+.5-TERM*SINF(TWOPI*X)
350 CONTINUE
CALL RCONTR(0,PLEV,0,V(IN,1),ILIM,XTAB(IN),1,ILIM,1,
* ZTAB(1),1,NZ,1)
400 CALL TICHEK(I,J,K,L)
TS=J*1.E-6
WOT 3,OUT7,1,TS
CALL EXIT
END

```

```

PROGRAM POT2TEST (POT2DATA, TAPE2=POT2DATA, POT2OUTO, TAPE3=POT2OUTO)
OPTIMIZE
C *****
POT2 GENERATES A TABLE OF NORMALIZED (BY V0) POTENTIAL VALUES
WHICH REPRESENTS THE NEGATIVE STAGE AND TWO FOCUSING CUPS OF
A DIRECT CONVERTER. GIVEN THE FIXED BOUNDARY VALUES, POTENTIALS
FOR THE INTERIOR POINTS ARE GENERATED BY RELAXATION. THE
COORDINATE ARRAYS (XTAB, ZTAB) AND POTENTIAL TABLE ARE WRITTEN
INTO A DISK FILE (POTAB2), WHERE THEY ARE USED BY THE DART CODE.
C INPUT VARIABLES:
C C - DETERMINES FOCUSING STRENGTH OF SECOND CUP
C CPRIME - DETERMINES FOCUSING STRENGTH OF FIRST CUP
C ZFIN - HALF-HEIGHT OF COLLECTOR CHANNEL
C PKPOT - PEAK (NEGATIVE) POTENTIAL IN NEGATIVE STAGE, WHICH
C IS ASSUMED TO OCCUR AT (-10,0)
C ALPHA - RELAXATION COEFFICIENT
C ITERNM - NUMBER OF RELAXATION ITERATIONS
C USING THE LINEAR RELATION OF THE VARIABLE A (WHICH DETERMINES THE
C STRENGTH OF NEGATIVE STAGE LENS) AND THE PEAK NEGATIVE POTENTIAL,
C TWO PASSES ARE MADE TO DEFINE THE LINE, AND A THIRD PASS USES THE
C INTERPOLATED VALUE OF A WHICH GIVES THE INPUT VALUE PKPOT.
C *****
IN1 FORMAT(SF10.0,15)
OUT1 FORMAT(/11HPOT2 INPUT:/5X,3HA =,F10.4,5X,3HC =,F10.4,5X,4HCP =,
* F10.4,5X,6HZFIN =,F10.6,5X,7HPKPOT =,F10.4/5X,7HALPHA =,
* F10.4,5X,8HITERNM =,15//)
OUT2 FORMAT(15,F15.7)
OUT3 FORMAT(1H1//4HXTAB,6X,10F10.6/(10X,10F10.6))
OUT4 FORMAT(//4HZTAB,6X,10F10.6/(10X,10F10.6))
OUT5 FORMAT(//15,5X,10F10.6/(10X,10F10.6))
OUT6 FORMAT(//5X,14HTIME PERCENTS: ,X,A10,5X,16HTIME TOTAL(SEC)=,F9.4)
HDG1 FORMAT(//4X,1HK,7X,5HTABLE)
ERR1 FORMAT(//26HUNABLE TO CREATE DISK FILE)

* PARAMETER (TWOPI=6.2831853, IN=-23, IP=12, KN=-5, KP=5, ILIM=36,
* KLIM=11)
* DIMENSION XTAB((IN,IP)), ZTAB((KN,KP)), V((IN,IP),(KN,KP)),
* ARAY((ILIM+KLIM+ILIM+KLIM))
* EQUIVALENCE (ARAY,XTAB),(ARAY((ILIM+1),ZTAB),(ARAY((ILIM+KLIM+1),V)
* DIMENSION PLEV(2)

CALL GDDROP(4RPOT2,5R+POT2,0)
CALL TICHEK(I,J,K,L)

C READ CONSTANTS

RIT 2, IN1, C, CPRIME, ZFIN, PKPOT, ALPHA, ITERNM

C 3-PASS LOOP TO GIVE PEAK POTENTIAL=PKPOT
DO 300 IPASS=1,3
GO TO (10,20,30),IPASS
10 A= 3
GO TO 40
20 V1=V(-10,0)
A= 4
GO TO 40
30 V2=V(-10,0)
A= 3+.1*(PKPOT-V1)/(V2-V1)

```

```

WOT 3,OUT1,A,C,CPRIME,ZFIN,PKPOT,ALPHA,ITERNM
C   SET INITIAL TABLE VALUES
40 DELX=1./12.
   DELZ=ZFIN/5.
   DDSQ=(DELX*DELX)/(DELZ*DELZ)

   DO 50 I=IN,0
   DO 50 K=KN,KP
50  V(I,K)=0.
   DO 70 I=1,IP
   DO 70 K=KN,KP
70  V(I,K)=1*DELX

C   DEFINE COORDINATE ARRAYS
   DO 80 I=IN,IP
80  XTAB(I)=I*DELX
   DO 90 K=KN,KP
90  ZTAB(K)=K*DELZ

C   SET NEGATIVE STAGE BOUNDARY
   Z=KP*DELZ
   DO 100 I=-12,-6
   X=1*DELX
   TEMP=(A/TWOPI)*SINF(TWOPI*X)*COSH(TWOPI*Z)
100  V(I,KP)=TEMP
   V(I,KN)=TEMP

C   SET FIRST FOCUSING STAGE BOUNDARY
   DO 120 I=1,6
   X=1*DELX
   TEMP=(CPRIME/TWOPI)*SINF(TWOPI*X)*SINH(TWOPI*Z)
120  V(I,KP)=X+TEMP
   V(I,KN)=X-TEMP

C   SET SECOND FOCUSING STAGE BOUNDARY
   DO 140 I=7,IP
   X=1*DELX
   TEMP=(C/TWOPI)*SINF(TWOPI*X)*SINH(TWOPI*Z)
140  V(I,KP)=X+TEMP
   V(I,KN)=X-TEMP

   X=IP*DELX
   V(IP,0)=X
   DO 150 K=1,KP-1
   Z=K*DELZ
   TEMP=(C/TWOPI)*SINF(TWOPI*X)*SINH(TWOPI*Z)
150  V(IP,K)=X+TEMP
   V(IP,-K)=X-TEMP

C   RELAXATION-ITERATION LOOP
C   ON THIRD PASS, PEAK POTENTIAL VALUE IS WRITTEN AFTER EACH
C   ITERATION
   DENOM=2.*(1.+DDSQ)

```



```

      DO 300 J=1,ITERNM
      DO 200 I=IN+1,IP-1
      DO 300 K=KN+1,KP-1
      TEMP=(V(I+1,K)+V(I-1,K)+DDSQ*(V(I,K+1)+V(I,K-1)))/DENOM
200  V(I,K)=V(I,K)+ALPHA*(TEMP-V(I,K))
      IF(IPASS-3) 300,300
      WOT 3,OUT2,J,V(-10,0)
300  CONTINUE

C      PRINT AND BUFFER OUT RESULTS

      WOT 3,OUT3,(XTAB(I),I=IN,IP)
      WOT 3,OUT4,(ZTAB(K),K=KN,KP)
      WOT 3,HDB1
      DO 320 K=KN,KP
      WOT 3,OUT5,K,(V(I,K),I=IN,IP)
320  CONTINUE

      LNG=ILIM*KLIM+ILIM+KLIM+10
      CALL DEVICE(4HOPEN,6RPOTAB2)
      CALL DEVICE(7HDESTROY,6RPOTAB2)
      CALL DEVICE(6HCREATE,6RPOTAB2,LNG,1ERR)
      CALL ASSIGN(22,6RPOTAB2)
      IF(1ERR) 330,
      WOT 3,ERR1
      GO TO 350

330  BUFFER OUT (22,1) (ARRAY(1),ARRAY(ILIM+KLIM+ILIM+KLIM),0)
340  IF(UNIT,22) 340,
      CALL DEVICE(6HCLOSE,6RPOTAB2)

C      PLOT EQUIPOTENTIAL CONTOURS

      CALL KEEP80(1)
      CALL CRTID(4HPOT2,1,1)

      CALL MAPS(-2.0,1.0,-1.2,1.2,.11328,1.,.290624,1.)
      PLEV(1)=0.
      PLEV(2)=.05
      CALL RCONTR(0,PLEV,0,V(IN,KN),ILIM,XTAB(IN),1,-IN,1,
      * ZTAB(KN),1,KLIM,1)
      * PLEV(2)=.2
      CALL RCONTR(0,PLEV,0,V(IN,KN),ILIM,XTAB(IN),-IN,ILIM,1,
      * ZTAB(KN),1,KLIM,1)
350  CALL TICHEK(1,J,K,L)
      TS=J*.E-6
      WOT 3,OUT6,1,TS
      CALL EXIT
      END

```

```

PROGRAM POT3TEST (POT3DATA,TAPE2=POT3DATA,POT3OUT0,TAPE3=POT3OUT0)
OPTIMIZE
C*****
C POT3 GENERATES A TABLE OF NORMALIZED (BY V0) POTENTIAL VALUES
C WHICH REPRESENTS THE SPACE CHARGE POTENTIAL IN THE COLLECTOR
C CHANNEL OF A DIRECT CONVERTER. THE BOUNDARIES ARE FIXED AT ZERO,
C AND THE INTERIOR VALUES ARE CALCULATED USING A DART-GENERATED
C CHARGE DENSITY TABLE AND RELAXATION. THE COORDINATE ARRAYS
C (XTAB,ZTAB) AND POTENTIAL TABLE ARE WRITTEN INTO A DISK FILE
C (POTAB3), WHERE THEY ARE USED BY THE DART CODE.
C INPUT VARIABLES:
C   ALPHA - RELAXATION COEFFICIENT
C   ITERNM - NUMBER OF RELAXATION ITERATIONS
C   THE FILE (QDQTAB) CONTAINING CHARGE DENSITY VALUES IS REQUIRED.
C*****
IN1 FORMAT(F10.0,I5)
OUT1 FORMAT(/11HPOT3 INPUT:/5X,6HALPHA=,F10.4,5X,7HITERNM=,I5)
OUT2 FORMAT(18,5X,2F15.7)
OUT3 FORMAT(1H1//4HXTAB,6X,10F10.6/(10X,10F10.6))
OUT4 FORMAT(//4HZTAB,6X,10F10.6/(10X,10F10.6))
OUT5 FORMAT(15,5X,10F10.6/(10X,10F10.6))
OUT6 FORMAT(//5X,14HTIME PERCENTS: ,X,A10,5X,16HTIME TOTAL(SEC)=,F9.4)
HDG1 FORMAT(//13HITERATION NO. ,7X,7HV(50,3),5X,9HV(100,-3))
HDG2 FORMAT(//4X,1HK,7X,5HTABLE)

PARAMETER (IN=-23,IP=132,KN=-5,KP=5,ILIM=156,KLIM=11)
DIMENSION XTAB((IN,IP)),ZTAB((KN,KP)),O((IN,IP),(KN,KP)),
*          V((IN,IP),(KN,KP)),ARRAY(ILIM+KLIM+1,ILIM*KLIM),
*          T(ILIM*KLIM),PLEV(2)
EQUIVALENCE (ARRAY,XTAB),(ARRAY(ILIM+1),ZTAB),(ARRAY(ILIM+KLIM+1),V),
*          (T,Q)

CALL 00OROP(4RPOT3,5R+POT3,0)
CALL TICHEK(1,J,K,L)
CALL KEEPSO(1)
CALL CRTID(4HPOT3,1,1)

C READ CONSTANTS AND BUFFER IN CHARGE DENSITY TABLE

RIT 2,IN1,ALPHA,ITERNM
WOT 3,OUT1,ALPHA,ITERNM

CALL DEVICE(4HOPEN,5RQDTAB)
CALL ASSIGN(24,5RQDTAB)
BUFFER IN (24,1) (ARRAY(1),ARRAY(ILIM+KLIM),0)
20 IF(UNIT,24) 20,
BUFFER IN (24,1)'(T(1),T(ILIM*KLIM),ILIM+KLIM)
30 IF(UNIT,24) 30,
CALL DEVICE(5HCL0SE,5RQDTAB)

C SET INITIAL VALUES

DELX=XTAB(IN+1)-XTAB(IN)
DELZ=ZTAB(KN+1)-ZTAB(KN)
DO 50 I=IN,IP
DO 50 K=KN,KP
50 V(I,K)=0.

C RELAXATION-ITERATION LOOP

```

```

C      THE POTENTIALS AT TWO POINTS ARE PRINTED AFTER EACH ITERATION
C      TO TEST FOR CONVERGENCE

EPS0=8.854E-12
CONST=(DELX*DELX)/EPS0
DDSQ=(DELX*DELX)/(DELZ*DELZ)
DEN=2.*(1.+DDSQ)
WOT 3, HDG1
DO 200 J=1, ITERNM
DO 100 I=IN+1, IP-1
DO 100 K=KN+1, KP-1
TERM=(Q(I,K)*CONST+V(I+1,K)+V(I-1,K)+DDSQ*(V(I,K+1)+V(I,K-1)))/DEN
100 V(I,K)=V(I,K)+ALPHA*(TERM-V(I,K))
WOT 3, OUT2, J, V(50,3), V(100,-3)
200 CONTINUE

C      PRINT AND BUFFER OUT RESULTS

WOT 3, OUT3, (XTAB(I), I=IN, IP)
WOT 3, OUT4, (ZTAB(K), K=KN, KP)
WOT 3, HDG2
DO 300 K=KN, KP
WOT 3, OUT5, K, (V(I,K), I=IN, IP)
300 CONTINUE

LNG=ILIM+KLIM+ILIM+KLIM+10
CALL DEVICE(4HOPEN, 6RPOTAB3)
CALL DEVICE(7HDESTROY, 6RPOTAB3)
CALL DEVICE(6HCREATE, 6RPOTAB3, LNG, IERR)
CALL ASSIGN(23, 6RPOTAB3)
BUFFER OUT (23, 1) (ARAY(1), ARAY(ILIM+KLIM+ILIM+KLIM), 0)
325 IF (UNIT 23) 325,
CALL DEVICE(6HCLOSE, 6RPOTAB3)

C      PLOT EQUIPOTENTIAL CONTOURS

XMIN=-2.
XMAX=11.
XINC=3.
XMX=XMIN+XINC
IMN=1
IMX=INC+36
IMAX=156
PLEV(1)=0.
PLEV(2)=.01

350 CALL MAPS(XMIN, XMX, -1.4, 1.4, .11328, 1., .172395, 1.)
CALL RCONTR(0, PLEV, 0, V(IN,KN), ILIM, XTAB(IN), IMN, IMX, 1,
* ZTAB(KN), 1, KLIM, 1)

IF(XMX-XMAX) , 360, 360
XMIN=XMX
XMX=XMX+XINC
IMN=IMX
IMX=XMINDF(IMAX, IMX+INC)
CALL FRAME
GO TO 350

360 CALL TICHEK(I, J, K, L)
TS=J*1.E-6
WOT 3, OUT6, I, TS
CALL EXIT
END

```

```

*****
C*      PROGRAM NAME:   DQC
C*      PROGRAM TITLE:  DART SPACE CHARGE RUN CONTROLLER
C*      ORIGINATED:     3 AUGUST 1973
C*      VERSION/DATE:    1/6 MAY 74
C*      PROGRAMMER:      BARRY C. HOWARD
C*      PURPOSE:         TO GENERATE A SPACE CHARGE POTENTIAL TABLE
C*                       AND USE IT IN A FINAL DART RUN. DART, POT3
C*                       AND MICROPUS ARE CALLED AS CONTROLLEES
C*      REFERENCE:       B. C. HOWARD, "DART: A SIMULATION CODE FOR A
C*                       DIRECT ENERGY CONVERTER FOR FUSION REACTORS"
C*                       SEPT. 23, 1973 (DRAFT)
C*      EXECUTE LINE:    DQC N'S I / T V
C*      WHERE            N      IS THE NUMBER OF TABLE ITERATIONS BEFORE
C*                           THE FINAL DART RUN
C*                           S      IS THE SYMBOL FOR INCLUDING SECONDARIES IN
C*                           THE FINAL RUN (OPTIONAL)
C*                           I      IS THE ITERATION NO. TO BE USED TO RESTART
C*                           *DART AFTER A MACHINE CRASH (OPTIONAL)
C*      FILES NEEDED:    DART, DARTDATA, POTAB1, POTAB2, POT3, POT3DATA
*****

```

```

OPTIMIZE
DIMENSION IBETA(5), IBETA1(3), IBETA2(3), IBETA3(3), IBETA4(4),
* IBETA5(10), IBETA6(3), MTTY(3), MSG(3)
DATA (IBETA1=4RDART, 0, 0), (IBETA2=4RPOT3, 0, 0), (IBETA3=8RMICROPUS, 0,
* 0), (IBETA4=8RDARTOUT0, 0, 8RPOT3OUT0, 0),
* (IBETA5=42H:3 360 0 0 0 0 1 1 0 0_@"$),
* (IBETA6=5R+DART, 0, 0),
* (IALDQ=4077700415454004457B),
* (MTTY=28HITERATION COMPLETED@"$),
* (MSG=26HTYPE NO. OF ITERATIONS @"$),
* (MASK=777777777777777B)

```

```

CALL CHANGE(4R+DQC)
IBETA5(9)=10H0 0@_@
CALL GOB(2001B, IER, 300B, IBETA)
IF(IER) 5,
CALL GOB(1400B, IER, 3, MSG)
CALL GOB(2000B, IER, 300B, IBETA)
5 MESS=IBETA(1)
CALL BCDINT(MESS, NIT, 8, 2)
MESS=IBETA(2)
IF(MESS-1RS) 10, 10
IBETA5(9)=IBETA5(9).UN.(1.SHL.24)
MESS=IBETA(3)
10 IF(MESS-400204B) 20,
CALL BCDINT(MESS, I, 9, 1)
15 CALL GOB(2210B, IER, 0, IBETA6)
IF(I-NIT) 40, 40, 120

```

```

$$$CLEAR SECONDARIES FLAG
$$$GET TTY MESSAGE
$$$IF NONE, PROMPT USER
$$$TRY AGAIN
$$$SET NO. OF ITERATIONS
$$$CHECK FOR SECONDARIES
$$$TURN ON SECONDARIES FLAG
$$$CHECK FOR EOM
$$$STORE LOOP PASS NO. IN I
$$$RESTART *DART
$$$RESUME WAIT FOR MESSAGE

```

C ITERATION LOOP

```

20 J=1
DO 100 I=1, NIT
CALL GOB(2210B, IER, 0, IBETA1)
40 CALL GOB(1700B, IER, 500B, IBETA)
IF(IBETA-IALDQ) ERR, ERR
CALL GOB(2210B, IER, 0, IBETA2)
$$$START DART
$$$GET DART MESSAGE
$$$CHECK FOR "ALL DONE"
$$$START POT3

```

```

CALL GOB(1700B,IER,500B,IBETA)      $$$GET POT3 MESSAGE
IF(1BETA-1ALDO) ERR,ERR              $$$CHECK FOR "ALL DONE"
CALL GOB(0202B,IER,1,1BETA4)         $$$DESTROY DART,POT3 OUTPUT

C   ALTER DARTDATA FILE AFTER FIRST RUN AND BEFORE LAST RUN

IF(1-1) 50,
IF(1-NIT) 80,,80
J=6
50 CONTINUE
CALL GOB(2200B,IER,0,1BETA3)          $$$INITIALIZE MICROBUS
CALL GOB(1500B,IER,2,20HDARTDATA ALTER,0"s) $$$START MICROBUS
CALL GOB(1700B,IER,100B,IBETA)        $$$GET "1"
CALL GOB(1500B,IER,2,11H# R 50_0"s)   $$$GET "1"
CALL GOB(1700B,IER,100B,IBETA)        $$$GET "1"
CALL GOB(1500B,IER,5,1BETA5(J))       $$$REPLACE LNE 5 OF DARTDATA
CALL GOB(1700B,IER,100B,IBETA)        $$$GET "1"
CALL GOB(1500B,IER,1,6H0_0"s)         $$$CAUSE FILE UPDATE
CALL GOB(1700B,IER,100B,IBETA)        $$$GET "THE ALTER IS . . ."
CALL GOB(1500B,IER,1,9HEND0_0"s)      $$$TERMINATE MICROBUS
CALL GOB(1700B,IER,500B,IBETA)
IF(1BETA-1ALDO) ERR,ERR               $$$CHECK FOR "ALL DONE"
80 CALL CONVI(1,K,0,2)                 $$$SEND COMPLETION INFO
MTTY(2)=(MTTY(2).INT.MASK).UN.K       $$$ TO TTY
CALL GOB(1400B,IER,3,MTTY)
100 CONTINUE

CALL GOB(2210B,IER,0,1BETA1)          $$$START DART - FINAL RUN
120 CALL GOB(1700B,IER,500B,IBETA)
IF(1BETA-1ALDO) 200,                  $$$CHECK FOR "ALL DONE"
ERR CALL GOB(1400B,IER,5,IBETA)       $$$WRITE ERROR TO TTY
200 CALL EXIT
END

```