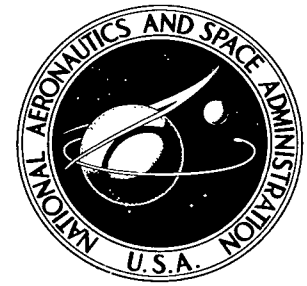
NASA TECHNICAL NOTE

NASA TN D-4043



MONTE CARLO CODE FOR SOLUTION OF PLANAR ELECTRON-DIODE PROBLEMS INCLUDING ELECTRON-NEUTRAL ELASTIC COLLISIONS

by Paul Swigert and Charles M. Goldstein

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • AUGUST 1967

NASA TN D-4043

MONTE CARLO CODE FOR SOLUTION OF PLANAR ELECTRON-DIODE PROBLEMS INCLUDING ELECTRON-NEUTRAL ELASTIC COLLISIONS

By Paul Swigert and Charles M. Goldstein

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – CFSTI price $3.00

CONTENTS

MONTE CARLO CODE FOR SOLUTION OF PLANAR ELECTRON-DIODE PROBLEMS INCLUDING ELECTRON-NEUTRAL ELASTIC COLLISIONS

by Paul Swigert and Charles M. Goldstein

Lewis Research Center

SUMMARY

A Monte Carlo electron transport code for the self-consistent potential solutions of one-dimensional planar electron-diode problems including electron-neutral elastic collisions capable of employing differential scattering cross sections is presented. An analytical description of a class of problems for which this code was written and the methods and techniques for solution of these problems are also presented. The code is given including instructions for the user, flow charts, and listings of all FORTRAN IV programs. Also included is material on curve and surface fits, convergence of a second-order differential equation, machine language routines, and a sample problem.

INTRODUCTION

The solution of steady-state electron transport problems in the presence of a low density, scattering background gas and a nonuniform electric field has not as yet been achieved by the usual analytical methods. In the field of neutron transport problems, where the usual methods of analysis also fail, resort is made to the Monte Carlo method to obtain particular solutions. This report describes a Monte Carlo code ENEC (electron-neutral elastic collisions) for the self-consistent potential field solution of a class of electron-diode problems including the effects of electron-neutral elastic collisions.

A preliminary version of this code, restricted to hard-sphere electron-neutral collisions was published in appendix 1 of reference 1. The present report, however, is completely self-contained and presents many improvements. In addition, the restriction to hard-sphere collisions has been removed in order to treat energy- and angle-dependent differential scattering cross sections.

An analytical description of the class of problems for which this code was written is given, and then the code itself, ENEC, is presented. The mathematical symbols used in

the analysis are defined in appendix A. One-dimensional spline curve fits and two-dimensional spline surface fits are discussed in appendix B. A convergence experiment is described, the results obtained are discussed, and some conclusions are drawn in appendix C. Presented in appendix D is an explanation of an improved square root routine. Appendix E contains machine language routines used in ENEC, which were not programed by the authors, but are available in the Lewis 7094 Library. The output of a sample problem is given in appendix F.

ANALYSIS

DESCRIPTION OF PROBLEM

The geometric configuration of one-dimensional field, flux, and electrodes is depicted in figure 1. The one-dimensional problem is treated wherein the emitter and collector

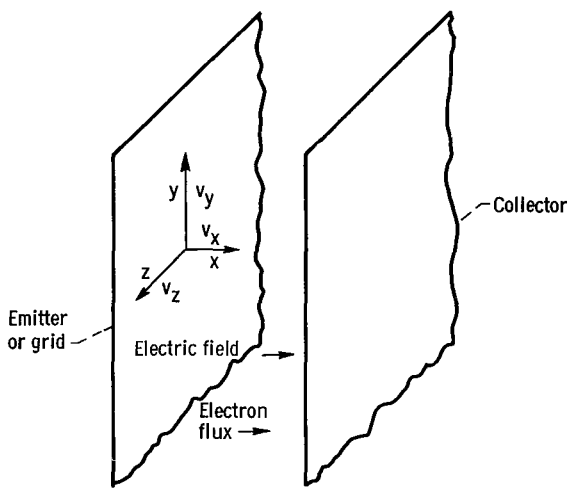


Figure 1. - Configuration of one-dimensional field, flux, and electrodes.

are assumed to be infinite parallel planes. The electric field and x-direction are normal to the electrode surfaces.

The class of problems treated here is considerably more complicated than the neutron transport problems because of the nonlinearity introduced by the potential field, the existences of curvilinear rather than rectilinear trajectories, and the spatial as well as energy variation of the mean free path.

The potential distribution is obtained as a solution to Poisson's equation

$$\frac{d^2\psi(\hat{x})}{d\hat{x}^2} = \frac{1}{\epsilon}\rho(\hat{x}) \tag{1}$$

by Picard iteration. (All symbols are defined in appendix A.) For every assumed or computed potential distribution $\psi(\hat{x})$, the electron charge density $\rho(\hat{x})$ is obtained by a Monte Carlo calculation. The question of convergence is discussed in the section CONVERGENCE OF SOLUTION and in appendix C.

The code assumes that the electrons are thermionically emitted with a half-Maxwellian velocity distribution:

$$f(\hat{u},\hat{v},\hat{w})d\hat{u}\,d\hat{v}\,d\hat{w} = 2\left(\frac{m}{2\pi kT_e}\right)^{3/2} e^{-(\hat{u}^2+\hat{v}^2+\hat{w}^2)m/2kT_e}\,d\hat{u}\,d\hat{v}\,d\hat{w}$$

$$\left.\begin{aligned} 0 \le \hat{u} \le \infty \\ -\infty \le \hat{v} \le \infty \\ -\infty \le \hat{w} \le \infty \end{aligned}\right\} \tag{2}$$

Only minor modifications are necessary, however, to treat monoenergetic beam emission (see ref. 1).

SOLUTION OF DIFFERENTIAL EQUATION

In dimensionless variables, equation (1) becomes

$$\frac{d^2\varphi(x)}{dx^2} = C \cdot n(x) \tag{3}$$

where $n(x)$ is the dimensionless electron density distribution, $\varphi(x)$ is the dimensionless potential distribution, and the space charge parameter C is given by

$$C = \frac{8}{\epsilon}\left(\frac{\pi}{2kT}\right)^{3/2} m^{1/2} e J_o L^2 \tag{4}$$

As previously mentioned, equation (3) is solved by Picard iteration. The particular method, however, was that of Clenshaw and Norton (ref. 2). This method is appropriate here because it allows full use to be made of the many desirable features of Chebyshev expansions.

An expansion in Chebyshev polynomials enables one to minimize the number of data points at which n(x) must be evaluated for a given average error. This is particularly important in the problem because n(x) is obtained from a Monte Carlo calculation, as described in the next section.

In reference 1, the Chebyshev expansion $n(x) = \Sigma a_n T_n(x)$ was obtained and then transformed to a power series expansion in x before equation (3) was integrated. For any reasonably high degree polynomial, however, the coefficients of the power series expansion are badly conditioned; that is, they become so large that all precision is soon lost due to machine truncation errors. On the other hand, the coefficients a_k of the Chebyshev series expansion are well behaved. It can be shown that $|a_k| \leq 2M$ where M is the maximum value of n(x). Since the Chebyshev polynomials $T_k(x)$ are such that $|T_k(x)| \leq 1$, the error of truncating the series after a finite number of terms is readily apparent from the magnitudes of the first coefficients neglected.

The power of the Clenshaw-Norton method is in its ability to employ only Chebyshev expansions and to eliminate the need for transforming to a power series expansion with the attendant loss of accuracy. Another desirable feature is that the solution (in this case $\varphi(x)$) is also given in terms of a Chebyshev expansion instead of the usual tabulation of numerical values.

CONVERGENCE OF SOLUTION

The convergence of interest here is that of the sequence $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)$ derived from the Picard iteration of equation (3). If n(x) were a given function (not a stochastic function), then, under the physical constraints imposed on the electron density, a solution exists to the initial value problem. This knowledge gives no information, however, on the rate of convergence or the effect of stochastic variations of n(x) on the sequence $\{\varphi_k(x)\}$. The effect of stochastic fluctuations on the convergence is discussed in appendix C.

In practice, after about three iterations (depending, of course, on initial distribution $\varphi_0(x)$) the potential distribution "settled down." Thereafter (succeeding iterations), the distribution fluctuated within a relatively small region of the function space defined by the solutions of equation (3). Successive iterations were then treated as independent trials. Their sample mean was accepted as the solution.

MONTE CARLO EVALUATION OF ELECTRON DENSITY AND CURRENT

The general concepts of Monte Carlo calculations for electron-diode problems are discussed in reference 1. Basic to the procedure is to sample a great number of electrons emitted according to a given velocity distribution and obtain their averaged contribution to the electron density distribution and current. From these averages, an estimate of the desired diode characteristics is obtained.

Initial Velocity Components

At the beginning of each electron history, the initial velocity components are chosen in accord with a half-Maxwellian velocity distribution at the emitter. It is necessary, at this point, to mention that, in the sampling process, the histories of electron fluxes and not elements of electron density are being traced. The elements of electron density are not spatially invariant entities while the electron fluxes are. It is for this reason that the initial velocity components are not chosen from equation (2) but from the velocity distribution of electron flux, which in dimensionless cylindrical coordinates is

$$g(u, V)du\ dV = 4uVe^{-(u^2+V^2)}\ du\ dV \tag{5}$$

where u is the dimensionless velocity component parallel to the x-direction, and V is the transverse component. The marginal distribution functions of the random variables u and V are

$$G_u(t)dt = G_V(t)dt = 2te^{-t^2}\ dt \tag{6}$$

Hence, the initial velocity components are chosen by the equations

$$\left.\begin{aligned} u^2 &= -\ln R_u \\ V^2 &= -\ln R_V \end{aligned}\right\} \tag{7}$$

where R_u and R_V are, ideally, random numbers uniformly distributed over the range from 0 to 1 (see ref. 1, p. 23). Computer programs for choosing randomly from the range 0 to 1 result in, at best, sequences of pseudorandom numbers. These numbers, for the most part, however, are sufficiently random for Monte Carlo applications. For further information on pseudorandom numbers see references 3 and 4.

Distance to Collision

Given an energy-independent mean free path λ, the probability that an electron will travel a distance l without colliding is $e^{-l/\lambda}$. Hence, the distance to collision l_c is obtained from the equation

$$l_c = -\lambda \ln R_l \tag{8}$$

where R_l is again a random number between 0 and 1 (see ref. 1, p. 12).

In this code, however, energy-dependent free path lengths $\lambda(E)$ are considered as follows: The interelectrode space is subdivided into a series of virtual cells separated by imaginary planes parallel to the electrodes. An average energy $\overline{E}$ is defined as the average kinetic energy the test electron would have in a cell assuming no collisions. On the basis of this average kinetic energy, a mean free path $\overline{\lambda} = \lambda(\overline{E})$ is obtained and employed in equation (8) to obtain the distance to collision l_c in the cell. If l_c is greater than the distance along the electron trajectory to the cell boundary, the process is repeated in the new cell (unless an electrode is reached). If l_c is less than this distance, a simple search routine is employed to determine the correct x_c corresponding to the point of collision.

Angle of Scatter

ENEC considers only electron-neutral elastic collisions; hence, the simplifying assumptions of infinite-mass (stationary) target particles is employed. Since the scattering is anisotropic, however, the angle of scatter is a function of the angle of incidence. The polar angle of incidence $\theta_0 = \tan^{-1}(V/u)$ is known at each collision. Because of the symmetry of the configuration, the azimuthal angle of incidence may be arbitrarily assumed to be $\varphi = 0$.

It is, of course, assumed that the differential cross section $\sigma(\theta, E)$ is known or can be approximated. The methods used to smooth, fit, and tabulate the data for input to this code are discussed in the first section of ENEC CODE (also see appendix B).

The cumulative distribution function of the random variable θ can be defined, knowing $\sigma(\theta, E)$, as

$$P[\theta < \Theta] = \int_0^{\Theta} \frac{\sigma(\theta', E)}{\sigma(E)} \sin \theta' \, d\theta' \tag{9}$$

where

$$\sigma(E) = \int_0^{\pi} \sigma(\theta, E) \sin\theta \, d\theta \tag{10}$$

In the present case, it was decided to sample $\cos\theta$ instead of θ. From equation (9), the cumulative distribution function of $\cos\theta$ may be obtained by a simple transformation:

$$P[\cos\theta < t] = \int_{-1}^{t} \frac{\sigma(\cos^{-1}t', E)}{\sigma(E)} \, dt'$$

As further explained in the section Angular distribution of scatter in ENEC CODE, the random variable $\cos\theta$ is tabulated as a function of a uniform distribution R between -1 and 1 by solving the equation

$$R = \int_{-1}^{\cos\theta} \frac{\sigma(\cos^{-1}t, E)}{\sigma(E)} \, dt \tag{11}$$

for $\cos\theta$ (see eq. (A16) of ref. 1 and the accompanying discussion). The azimuthal angle of scatter φ is chosen from a uniform distribution over the range from 0 to 2π.

With the scattering angles after collision about the incident direction denoted by primes, the transformation back to the coordinate system of figure 1 gives the following expression for the cosine of the polar scattering angle:

$$\cos\theta = \cos\theta_o \cos\theta' - \sin\theta_o \cos\varphi' \sqrt{1 - \cos^2\theta'} \tag{12}$$

Sampling from Electron Histories

Density. - The locations of the data points x_k, where the electron density is sampled, are specified by the CHEBY subroutine. The contribution of a test electron (unit of flux) to the density at a given x_k is

$$\frac{1}{\pi^{1/2} u(x_k)} \tag{13}$$

for each passage past x_k, where

$$u(x) = \sqrt{u_o^2 + \varphi(x) - \varphi(x_o)} \tag{14}$$

x_o is the position of the last event (collision or emission), and u_o is the initial velocity of this trajectory at x_o. The $\pi^{1/2}$ results from the nondimensionalyzing factor employed for the velocity (see definitions of u and V in appendix A); note that equation (13), averaged over the initial distribution g(u,V) (eq. (5)), gives n(0) = 1, as assumed.

The sample density at x_k at the end of one iteration (N_o histories) is given by

$$n(x_k) = \frac{1}{\pi^{1/2} N_o} \sum_i \frac{1}{u_i(x_k)} \tag{15}$$

where the sum over i (flux passages past x_k) may be greater than, equal to, or less than N_o because of collisions and/or reflections from the potential field.

After each iteration, the density distribution n(x) is obtained by fitting a Chebyshev expansion to the sampled values of $n(x_n)$.

Current to collector. - The ratio of current density to the collector to the emitted current density J/J_o is computed at the end of each iteration from

$$\frac{J}{J_o} = \frac{N_c}{N_o} \tag{16}$$

where N_c is the number of test electron fluxes reaching the collector.

Collisions and flux passages. - The total number of collisions is tallied in subroutines XIC and XICTP for each iteration, and the sample means are presented in the output (see sample problem, appendix F).

The total number of flux passages at each data point is similarly tallied and presented. These data have proved helpful from both a heuristic point of view and for debugging.

General Programing Features

While the program details are described in the section ENEC CODE, clarification of certain features may help to connect the analytical description with the code logically.

Equations (7) and (8), which define the random variables u^2, v^2, and l_c, are not used in the program functions. Instead, the function -ln R was tabulated for 1024 equidistant values of R ($0 < R < 1$). The application of a table look-up is five times as fast as evaluating -ln R on the Lewis IBM 7094 II. While the coarse graining of the initial

velocity distribution is not deleterious in the present class of problems, care must be taken if this method is used to treat, for instance, inelastic effects, because of the truncation of the Maxwellian tail. For the same reason of improving computing efficiency, $\overline{\lambda}$ was tabulated as a function of $\overline{E}$, and $\cos\theta$ was tabulated as a function of R and E.

Instead of standardizing on one formula, several different quadrature formulas have been employed for increased computational speed. The types of formulas employed were dictated, in part, by the desirability of tabulating the potential distribution $\varphi(x)$ at predetermined sets of mesh points before each iteration, instead of evaluating the potential each time from its Chebyshev expansion.

VELOCITY AND ENERGY DISTRIBUTION HISTOGRAMS

After obtaining a solution for $\varphi(x)$, the program was rerun for the express purpose of sampling the distribution functions. Rerunning did not involve any further iterations, but simply followed enough electron histories to obtain sufficient statistics.

The histograms were obtained by first accumulating a sample of 250 histories of the random variable. This sample was then ordered, and from the resulting empirical distribution the boundaries of 10 cells were chosen on the basis of equal probability. Then as many additional histories were tallied in the deciles as computer time and prudence would allow. In any event, this procedure precluded choosing cells wherein the subsequent sample was too small.

The original histograms obtained by sampling had to be modified. Whereas the velocity distribution of the electron flux emitted at $x = 0$ is sampled, the histogram shows the velocity distribution of the electrons. The same is true for the energy distribution. More explicitly, a sample may be taken from a flux distribution function $g(u, V, x)$; but this distribution function is related to the density distribution function $f(u, V, x)$ by

$$g(u, V, x) = Auf(u, V, x) \tag{17}$$

where A is a normalization factor. If, at given x, u is independent of the other velocity components, marginal distributions of u become

$$g(u, x) = Auf(u, x) \tag{18}$$

or

$$f(u, x) \propto \frac{g(u, x)}{u} \tag{19}$$

Hence, to obtain the histogram of $f(u,x)$, it is only necessary to divide the ordinate in each cell of $g(u,x)$ by u, evaluated at the center of the cell, and to normalize.

If, at given x, the kinetic energy is independent of polar and azimuthal angles,

$$g(E,x) = A\int_0^{2\pi} d\varphi \int_0^{\pi/2} uf(E,\theta,\varphi,x)\sin\theta\, d\theta \tag{20}$$

$$= A\int_0^{2\pi} d\varphi \int_0^{\pi/2} E^{1/2}\cos\theta\, f(E,\theta,\varphi,x)\sin\theta\, d\theta$$

$$= A'E^{1/2}f(E,x) \tag{21}$$

or

$$f(E,x) \propto \frac{g(E,x)}{E^{1/2}} \tag{22}$$

The procedure for obtaining the histogram of $f(E,x)$ is then directly analogous to that for $f(u,x)$.

ENEC CODE

GENERAL FEATURES

Cross Sections

The user must supply either a functional expression or data for the differential scattering cross section $\sigma(\theta, E)$. In the case of data, a functional expression is first obtained by the method of spline interpolation, as described in appendix B.

Geometry, Subdivisions, and Functional Tabulations

ENEC is programed for a one-dimensional geometry as depicted in figure 2. Four sets of subdivisions are employed:

Set a: 1024 Equally spaced subdivisions

Set b: NS equally spaced cells

Set c: Nonequidistant subdivisions defined by the abscissas of the three-point Gaussian quadrature formula (see Quadrature Formulas) in each cell

Set d: Nonequidistant subdivisions defined by N1 arguments (called XD) of the Chebyshev curve fit (see Quadrature Formulas)

The potential distribution $\varphi(x)$ is tabulated at the 1025 equally spaced boundaries of set a; $\varphi(x)$ is also tabulated on points defining sets c and d. These tabulated values of $\varphi(x)$ are employed in the trajectory calculations (see Quadrature Formulas).

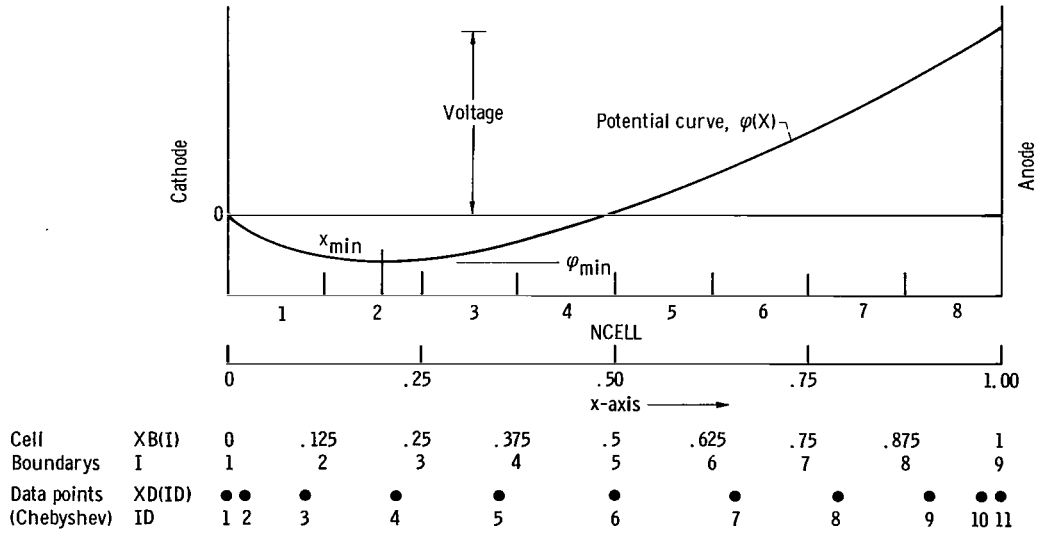


Figure 2. - Electrode geometry and subdivisions.

The cells, set b, are employed to obtain a spatial average of the electron kinetic energy (see section Distance to Collision). Note that set c is really a set of subsets of set b (one subset per cell). In addition, the potential minimum (PHIMIN) and its location (XMIN) are tabulated for each cell. The boundaries of set b are tabulated in the array XB. Sets c and d are also depicted in figure 2 for NS = 8 and N1 = 11.

Trajectories

In figure 3, the possible trajectories in a cell are shown for an electron moving toward the collector in a retarding potential field (fig. 3(a)). In figure 3(b), the electron is just passing into the cell and has an initial location XO, the cell boundary, while in figure 3(c), the trajectories begin at the location of the last collision. In figures 3(b-3), 3(b-5), and 3(c-3) to 3(c-6), the square of the u-component of velocity USQ is such that a turning point XTP occurs in the cell.

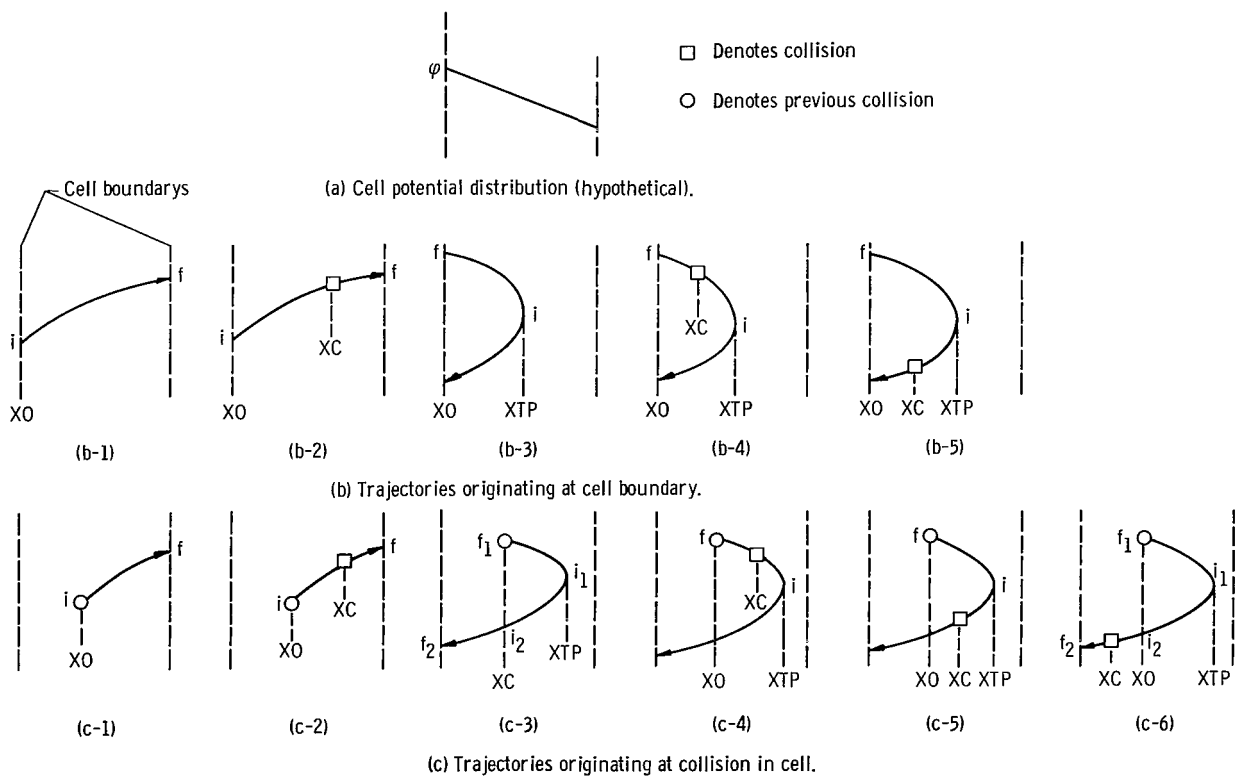


Figure 3. - Possible trajectories in a cell for electrons moving toward the collector. Potential, φ; initial cell boundary, XO; collision coordinate, XC; turning point coordinate, XTP; initial and final coordinates for numerical integrations, i and f, respectively.

Quadrature Formulas

As mentioned in the ANALYSIS, several different quadrature formulas are employed to compute distances along the trajectories in order to reduce computing time. To compute distances along a trajectory, the integrand of equation (23) must be evaluated at every argument of the quadrature formula employed:

$$l = \int \sqrt{1 + \frac{v^2}{u^2 - \varphi(x_o) + \varphi(x)}}\, dx \qquad (23)$$

Hence, computing time can be reduced by minimizing the number of arguments needed for a given accuracy and by having $\varphi(x)$ tabulated at all such arguments.

Gauss's Quadrature Formula (ref. 5, p. 150) is one of the most efficient known for the numerical integration of a well-behaved function. The abscissas, however, are irrational numbers (in general). This fact together with the desireability of using only tabulated values of $\varphi(x)$ precludes the extensive use of Gauss's formula here. This formula (QUAD) is used in ENEC, therefore, only for trajectories of the type depicted in figure 3(b-1) and 3(b-2).

For trajectories such as those depicted in figure 3(c-1), Simpson's Rule (ref. 5, p. 137) (QUADS) is employed between the limits of integration i and f over an equidistant subset of set a (see section Geometry, Subdivisions, and Functional Tabulation). This includes trajectories such as those depicted in figure 3(c-3) between the limits i_2 and f_2.

Simpson's Rule is not appropriate, however, for trajectories wherein a turning point exists (or would exist in the absence of collisions), for then the integrand (eq. (23)) becomes infinite at the turning point (XTP). It can be shown that, in the neighborhood of a turning point, the denominator in equation (23) goes to zero as $(x - XTP)^{1/2}$. Hence, a Newton-Cotes type $x^{1/2}$-weighted quadrature formula (QUADTP) was derived (see ref. 6), which did not require the integrand to be evaluated at $x = 0$.

Tabulated Distributions

Exponential distribution. - The exponential distribution is used to obtain the initial velocity components and the distance to collision, (eqs. (7) and (8)). Instead of evaluating -ALOG(R) for every random number R generated, -ALOG(R) is tabulated in array VEL for 1024 equidistant values ($0 < R < 1$).

Free path length. - The energy-dependent mean free path $\lambda(E)$ is obtained from the total collision cross section $\sigma(E)$ (eq. (10)), by the formula $\lambda(E) = 1.01/\sigma(E)$. Then $\lambda(E)$ is tabulated in array MFP over the required range of E.

Angular distribution of scatter. - The cosine of the scattering angle ($\cos\theta$) is obtained at a mesh of points (R, E) by numerical solution of equation (11) A surface fit of $\cos\theta = f(R, E)$ is then obtained (see appendix B), and subsequently $\cos\theta$ is tabulated in the two-dimensional array DISTB(R, E).

Initial Velocities

The square of the initial velocity components USQ (equal to USQO at x = 0) and VSQ is exponentially distributed (see eq. (7)). Use of the array VEL (see section Exponential distribution) is made in the following way:

(1) Choose a random number R.

(2) Let I = [1024.* R + 1.5], where the brackets imply a truncation to the nearest integer.

(3) Then USQO = VEL(I).

(4) Repeat steps (1), (2), and (3) for VSQ.

Elastic Collisions

Distance to collision. - After determination of the average electron kinetic energy $\overline{E}$ in a given cell, or between the last collision and cell boundary (see section Distance to Collision), a distance to collision FPATH is chosen in the following way:

(1) An integer I associated with $\overline{E}$ is obtained. (The value of this integer I depends on how $\lambda(E)$ is tabulated as a function of E in array MFP.)

(2) Choose a random number R.

(3) Let J = [1024.* R + 1.5].

(4) Then FPATH = MFP(I)* VEL(J).

Location of collision. - If the distance to collision (along the trajectory) is less than the distance to a cell boundary (along a trajectory in the absence of collisions) as, for example, depicted in figures 2(b-2) and 2(c-2), the collision location XC must be determined. A simple binary search that uses either Simpson's Rule (XIC) or a $x^{-1/2}$-weighted Newton-Cotes formula (XICTP) is employed to obtain XC. In the neighborhood of a turning point, XICTP is used.

Angle of scatter. - After locating XC, the angles of scatter relative to the electron velocity vector before scatter are obtained in the following way:

(1) A random number R is chosen.

(2) The azimuthal angle φ' is given by $2\pi R$.

(3) The total kinetic energy E(XC) of the electron at XC is computed.

(4) Another random number R is chosen.

(5) Integers I and J are associated with R and E(XC), respectively.

(6) Cos θ' is given by DISTB(I,J).

The cosine of the scattering (polar) angle (cos θ) in the original system of coordinates is then obtained from equation (12) where

$$\cos \theta = \frac{u(XC)}{\sqrt{E(XC)}}$$

$$\sin \theta_o = \frac{V}{\sqrt{E(XC)}}$$

and u(XC) and V are the velocity components before collision.

Generation of Electron Histories

During the complete trajectory of the electron from the emitter to the collector, or back to the emitter, the following items are tallied:

(1) The contribution to the electron density (see section Density) for each passage past an argument XD of the Chebyshev curve fit (see Solution of Differential Equation)

(2) The number of electrons NTHRU reaching the collector

(3) The total number of collisions in one iteration

Random Number Generation and Selection

The method employed at Lewis for generation of the pseudorandom number R on the unit interval is of the "congruential multiplicative" type (ref. 3). Most computing installations have their own library routine for this purpose, but for completeness, and possibly for those users who would prefer not to change the calling sequence, the basic machine language (MAP) program RANDOM used in ENEC is given in appendix E.

Solution of Differential Equation

After each iteration, the CHEBY routine is called to obtain a Chebyshev polynomial approximation (ref. 7) for the electron density distribution DEN(X). Then DEN(X) is employed in the program CLN2S to obtain a new potential distribution by the method of Clenshaw-Norton (see CONVERGENCE OF SOLUTION). The iterations are continued after convergence (see VELOCITY AND ENERGY DISTRIBUTION HISTOGRAMS) for purposes of averaging.

Averaging Iterations

As discussed in the section VELOCITY AND ENERGY DISTRIBUTION HISTOGRAMS, convergence is only achieved in a stocastic sense dependent on the random error associated with the Monte Carlo evaluation of the density (see appendix C). Hence, in CLN2S, as soon as the variation in Chebyshev coefficients for $\varphi(x)$ falls within a precalculated range for two successive iterations, "convergence" is assumed. At this point KI more iterations are performed, the resulting densities at each XD are averaged, and a final evaluation is made to obtain $\varphi(x)$. In addition, the KI sample collector potentials and currents are averaged and their standard deviations are computed.

DIRECTIONS FOR ENEC USERS

Preparation of Input Tables

Three tables must be constructed and punched on cards in a binary format before ENEC may be used. These tables contain information needed by ENEC to compute the initial velocity components of the electrons, the distance to a collision (free path length), and the angle of scatter after a collision (see Tabulated Distributions). Listings and descriptions of the programs that construct these tables (CVEL, MFP, ARGON, and ARGINV) are given in the section Auxiliary FORTRAN IV Program Descriptions. The SPLINE curve and surface fitting subroutines (SPLINE and SPLIN2) needed for interpolation, and the machine language subroutine for punching binary formatted cards (BCDUMP) are given in appendixes B and E, respectively. Samples of input and output for programs CVEL, MFP, ARGON, and ARGINV are given in appendix F.

Exponential distribution. - To obtain the table of the exponential distribution (see section on Exponential Distribution) on punched cards, load and execute the programs shown in figure 4. The deck of output cards is to be loaded with ENEC, as shown in figure 5.

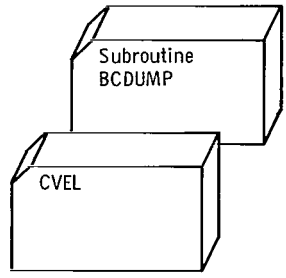


Figure 4. - Deck configuration for exponential distribution table.

Free path lengths. - To obtain the table of $\lambda(E)$ (see section Free Path Length, p. 14), load and execute the programs shown in figure 6. The input consists of known values of $\sigma(E^{1/2})$ for the gas under consideration. The deck of output cards is to be loaded with ENEC.

Angular distribution of scatter. - Two steps are involved in obtaining the table of the angular distribution noted in the section Angular Distribution of Scatter. The first step is to execute the deck configuration of figure 7(a). The input to this deck is a table of known values for the differential cross section for the gas under consideration $\sigma(\theta, E)$ (see section Angle of Scatter). The output from this deck (values on the surface defined by eq. (11)) is then input to the deck configuration shown in figure 7(b). This second deck configuration produces the table of angular distribution on punched cards that is to be loaded with ENEC.

Input Data - Problem Preparation

Input to ENEC consists of two parts. In the first part, three tables are constructed and punched on cards by the deck configurations of the preceding section. The binary formatted cards punched by BCDUMP are read by subroutine BCREAD given in appendix E.

The second part of the input to ENEC consists of variables that define the particular diode configuration to be studied. This input is formatted by the FORTRAN NAMELIST statement.

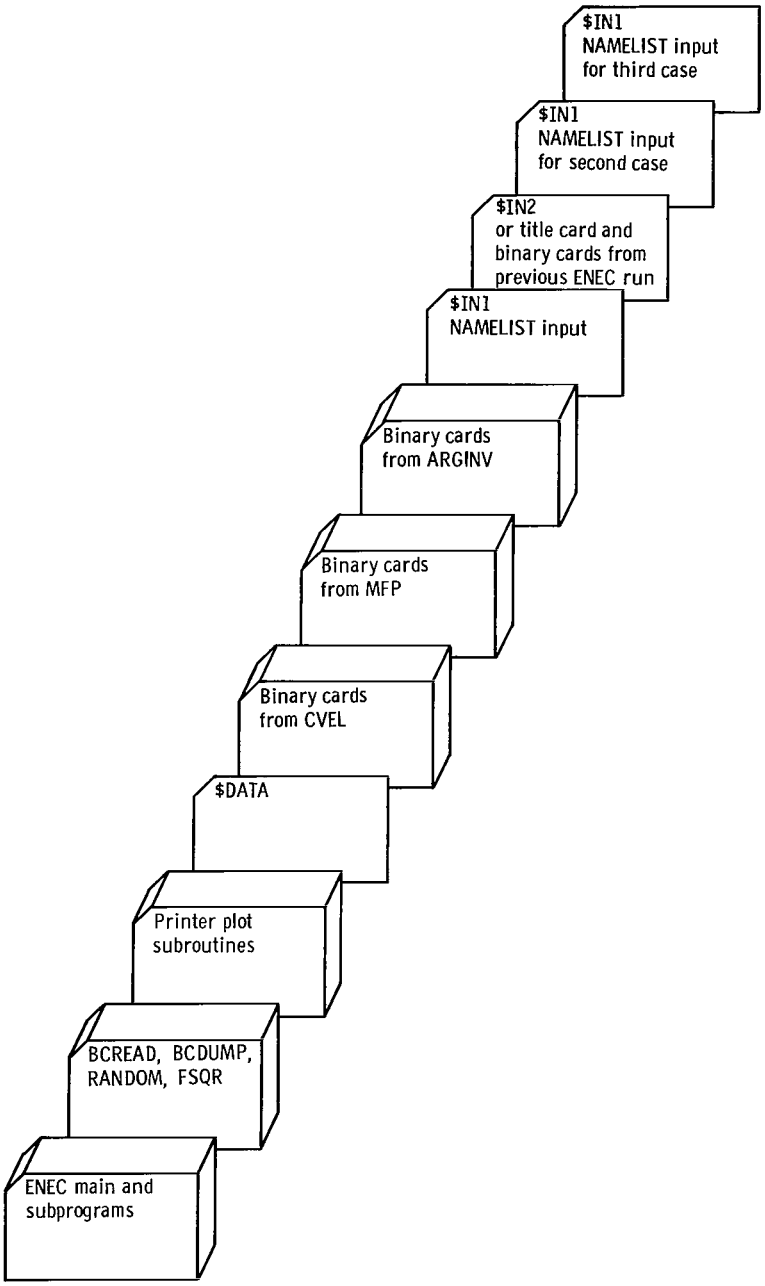


Figure 5. - ENEC deck configuration and input setup.

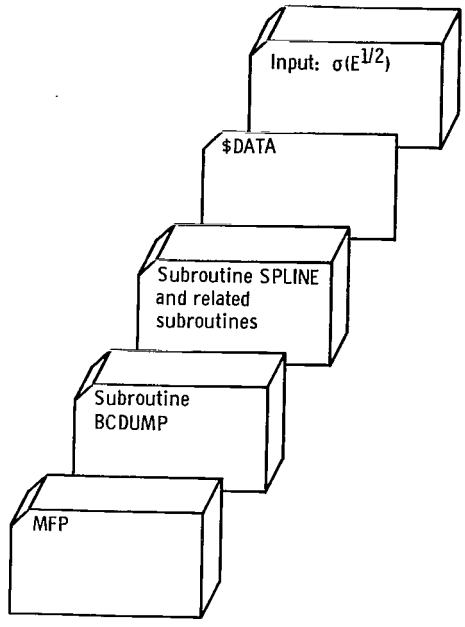
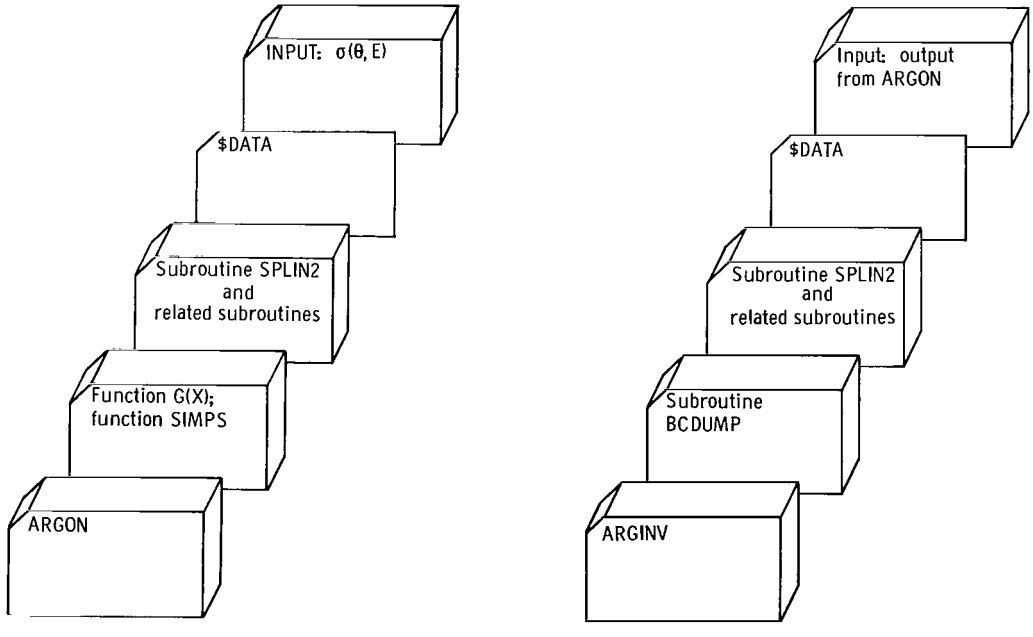


Figure 6. - Deck configuration for table of energy-dependent mean free path.



(a) First step.

(b) Second step.

Figure 7. - Deck configuration for obtaining angular distribution table.

The following outline gives the order (see fig. 5), format, and description of the input data of ENEC:

(1) BCREAD (VEL(1), VEL(1024))

The table of values computed by program CVEL

(2) BCREAD (MFP(1), MFP(126))

The table of values computed by program MFP

(3) BCREAD (DISTB(1, 1), DISTB(64, 64))

The two-dimensional table of values computed by programs ARGON and ARGINV

(4) NAMELIST/IN1/NO, N1, KI, ALPHA, CONST, NFLAG, BC, KODE, MODE, ERROR, NS, KFLAG, TEMPK, LFLAG

NO	Number of particles to be processed for each iteration
N1	Number of Chebyshev data points to be used; N1 $\leq$ 17 and odd
KI	Number of iterations after convergence in CLN2S to gather statistics on various parameters; KI $\leq$ 20
ALPHA	Ratio of mean free path to electrode spacing when KFLAG = 1; P_0L when KFLAG = 2
CONST	Constant C in Poisson's equation
NFLAG	Control on reading of AI array, NFLAG = 1; AI is read by NAMELIST/IN2/. NFLAG = 2; AI is read by BCREAD.
BC	Boundary condition on differential equation, $\varphi'(0)$ = BC when MODE = 1, and $\varphi(1)$ = BC when MODE = 2
KODE	Control on output from CLN2S (see section ENEC output). KODE = 1; CLN2S plots $\varphi(x)$ and writes Chebyshev coefficients for each iteration during convergence. KODE = 2; no output from CLN2S. KODE = 3; no plots, but Chebyshev coefficients are written for each iteration during convergence.
MODE	Control on boundary condition BC. MODE = 1, $\varphi'(0)$ = BC; MODE = 2, $\varphi(1)$ = BC.
ERROR	Convergence is assumed in CLN2S when the maximum difference between the corresponding coefficients AI for two successive iterations is less than ERROR.
NS	Number of cells; must be 1, 2, 4, 8, or 16
KFLAG	Control on the use of the MFP table. KFLAG = 1; constant ALPHA assumed. KGLAG = 2; ALPHA dependent on MFP table.
TEMPK	Temperature constant
LFLAG	Control on the calculation of the scattering angle; LFLAG = 1, isotropic scattering angle assumed; LFLAG = 2, scattering distribution used, DISTB.

(5) The initial coefficients for the Chebyshev fit of $\varphi(x)$ are read at this point. NFLAG controls the format of this input.

For NFLAG = 1, the input format is:

NAMELIST/IN2/AI

For this case the data read into AI are usually the coefficients of a Chebyshev fit for a straight line satisfying the boundary condition BC.

For NFLAG = 2, the input format is:

FORMAT (12A6)

BCREAD (AI(1), AI(20))

BCREAD (DA(1), DA(20))

BCREAD (B(1), B(20))

The first card of this input contains 72 Hollerith characters describing the ENEC run producing the AI data.

The next three binary cards contain the coefficients of the Chebyshev fit of $\varphi(x)$, $\varphi'(x)$, and $\varphi''(x)$ produced by a previous ENEC run. Although the coefficients for $\varphi'(x)$ and $\varphi''(x)$ are not needed, they are read in to keep all coefficients generated by a particular ENEC run together.

(6) For multiple runs, the NAMELIST statement /IN1/ is read until the input is exhausted.

ENEC Deck Configuration

The deck and input setup for execution of ENEC is shown in figure 5 (p. 18). Deck RANDOM is the random number generator given in appendix E, and deck FSQR is a fast square root subprogram designed specifically for Monte Carlo work. A discussion of square root subroutines is given in appendix D, while the square root subroutine itself is presented in appendix E. The printer plot subroutines are used to give plots of the functions $\varphi(x)$, $\varphi'(x)$, and $\varphi''(x)$ computed by ENEC. Reference 8 presents the necessary subroutines. The user may, if he wishes, remove all calls to PLOTF from ENEC thus eliminating the need for the plot subroutines. The input has been described in the previous section and must be placed in the order shown in figure 5.

ENEC Output

The output generated by ENEC consists of printed listings and punched cards. The printed output lists the input read by NAMELIST/IN1/ and the initial coefficients of the Chebyshev fit of $\varphi(x)$ along with the following information computed by the code:

(1) The N1 Chebyshev data points are listed with the mean and standard deviations of the number of particles crossing each data point.

(2) The mean and standard deviations of the N1 coefficients of the Chebyshev fit of $\varphi(x)$, $\varphi'(x)$, and $\varphi''(x)$ are listed.

(3) The mean and standard deviations of $\varphi(x)$, $\varphi'(x)$, and $\varphi''(x)$ are listed at the Chebyshev data points.

(4) The mean and standard deviations of the current, the voltage $[\varphi(1)]$, $\varphi'(0)$, and KNTR (where KNTR is the number of collisions of the particles per iteration) are listed.

(5) Values of φ_{min} and x_{min} for the mean $\varphi(x)$ are listed.

(6) Values of $\varphi(x)$, $\varphi'(x)$, and $n(x)\,[\varphi''(x)/C]$ are listed at 21 equally spaced points over the range from 0 to 1.

(7) Printer plots (ref. 8) are given for the three functions $\varphi(x)$, $\varphi'(x)$, and $\varphi''(x)$ over the range from 0 to 1.

Depending on the input variable KODE, output may be obtained from subroutine CLN2S. For each iteration during convergence in CLN2S, the user may have printed either the coefficients of the Chebyshev fit of $\varphi(x)$, $\varphi'(x)$, and $\varphi''(x)$ and printer plots of these functions, or just the coefficients of these functions. Thus, the user may "see" the convergence process take place in CLN2S.

The punched output consists of three binary cards, punched by BCDUMP, containing the mean coefficients of the Chebyshev fits of $\varphi(x)$, $\varphi'(x)$, and $\varphi''(x)$. The information contained on these cards is useful for input to other ENEC runs and for further investigation into the properties of a particular diode by other computer programs. For an example of the printed output, see appendix F.

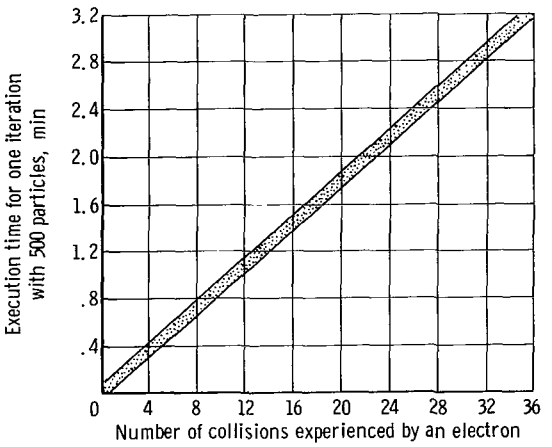


Figure 8. - Execution time.

Execution Time

An estimate of the execution time for ENEC is difficult to predict because the number of parameters used to define the diode and Monte Carlo solution is large. Figure 8 does show, however, that the execution time for one iteration does vary linearly with respect to the number of collisions that the electrons encounter. Experience with ENEC has shown that an average run usually takes about 13 iterations: 3 for convergence and 10 to gather statistics. The number of collisions an electron will make is, again, difficult to determine. Generally, the number of collisions will increase when the voltage is increased or when the mean free path is decreased. The digital computer used to run ENEC was an IBM 7094 II-7044 direct couple system. Therefore, the execution times are based on this system.

PROGRAM DETAILS

ENEC Labeled COMMON

A description of all FORTRAN variables appearing in labeled COMMON in ENEC is given in table I. The COMMON label is listed with the variables in the order and with the dimension information used in ENEC. The cross reference between ENEC programs and labeled COMMON are shown in table II.

TABLE I. - DESCRIPTION OF FORTRAN VARIABLES APPEARING IN ENEC COMMON

COMMON label	FORTRAN variable	Description
/CMAIN/	NO	Number of particles to be processed for one iteration
	NI	Number of Chebyshev data points
	ALPHA	Ratio of mean free path to electrode spacing or the product P_0L
	CONST	Constant in Poisson's equation
	NS	Number of cells
	AI(20)	Array of coefficients of Chebyshev fit to $\varphi(x)$
	VEL(1024)	Array of values of -ln R, where R is equally spaced between 0 and 1 (see section Exponential distribution)
	NK	Counter of iterations after convergence
	MFP(126)	Array of values of $\lambda(E)$ (see section Exponential distribution)
	KFLAG	Control used in choosing mean free path; KFLAG = 1 for constant ALPHA; KFLAG = 2 for ALPHA dependent on MFP array
	THETA	TEMPK/11600.0
	DISTB(64, 64)	Gas scattering angle distribution (see section Angular distribution of scatter)
	LFLAG	Control used in choosing scattering angle; LFLAG = 1 for an isotropic scattering angle; LFLAG = 2 for scattering angle chosen from DISTB
/CITER/	N(20)	Counter for the number of particles passing each Chebyshev data point
	Y(20)	The tally count at each Chebyshev data point; this array contains the new $\varphi''(x)$ after ITER and determines the new Chebyshev fit
	KNTR	Counter for the number of collisions for each iteration
	VSQ	V^2
	FPATH	Mean free path
	IBO	Index of cell boundary behind particle
	IBF	Index of cell boundary ahead of particle
	SIGMA	Control indicating direction of particle; SIGMA = 1 if the particle is traveling to the right; SIGMA = -1 if the particle is traveling to the left

TABLE I. - Concluded. DESCRIPTION OF FORTRAN VARIABLES APPEARING IN ENEC COMMON

COMMON label	FORTRAN variable	Description
/CITER/	ID	Index of next Chebyshev data point ahead of particle
	CRRNT	Ratio of number of particles processed to number escaping to the right of the diode per iteration
/CMNPHI/	NDEL	1024/NS; number of equally spaced data points in each cell
	DELX	1./1024; distance between equally spaced data points
	XB(20)	Cell boundaries
	TPHIQ(3,20)	Functional values of $\varphi(x)$ at three Gaussian data points for each cell
	IMIN(20)	Index of location of φ_{min} for each cell
	PHIMIN(20)	φ_{min} for each cell
	XMIN(20)	Location of φ_{min} for each cell
	TPHID(20)	Functional values of $\varphi(x)$ at the Chebyshev data points
	TPHIX(1026)	Functional values of $\varphi(x)$ at the equally spaced data points
/CCLN2S/	DA(20)	Array of coefficients of Chebyshev fit to $\varphi'(x)$
/CCHEBY/	B(20)	Array of coefficients of Chebyshev fit to $\varphi''(x)$
	XD(20)	Chebyshev data points
/CCELL/	NCELL	Cell number particle is in
	XO	Position of particle
	XF	Position of cell boundary ahead of particle
	IO	Index of particle position
	IF	Index of position of cell boundary ahead of particle
	USQO	u_o^2
/CXITP/	ITP	Index of position of turning point
	XTP	Position of turning point
/CXICTP/	IC	Index of position of collision
	XC	Position of collision
/CPATH/	EV	Energy of particle

TABLE II. - LABELED COMMON AND ENEC PROGRAM CROSS REFERENCE

ENEC program	COMMON label								
	/CMAIN/	/CITER/	/CMNPHI/	/CCLN2S/	/CCHEBY/	/CCELL/	/CXITP/	/CXICTP/	/CPATH/
MAIN	×	×	×	×	×				
ITER	×	×	×			×			
MINPHI	×		×		×				
CELL		×	×		×	×	×	×	
STOSS	×	×	×			×		×	×
PATH	×	×	×			×			×
XIC		×	×					×	
XICTP		×	×				×		
XITP			×			×	×		
QUAD	×	×	×			×			
QUADTP		×	×			×			
QUADS		×	×			×			
CLN2S	×			×	×				
CHEBY	×	×			×				
PHI	×								
DPHI	×			×					
DENS	×				×				
DISCR1									
DISCR2									
PLOTF									
PLOTYX									
SORTYX									
SCALEY									

ENEC FORTRAN IV Program Descriptions, Flow Charts, and Listings

Table III presents a brief description of all FORTRAN programs used in the ENEC code. The MAP coded programs (FSQR, RANDOM, BCREAD, and BCDUMP) are presented in appendix E. Printer plot programs used by PLOTYX are given in reference 8. A directed graph, of the programs given in table III is shown in figure 9. This figure depicts the overall interrelation of the ENEC FORTRAN programs. Listings and flow charts (figs. 10 to 36) for the FORTRAN programs are presented in the following sections.

TABLE III. - DESCRIPTION OF ENEC FORTRAN IV PROGRAMS

Program	Description
MAIN PROGRAM	Controls the flow of the Monte Carlo solution of planar electron-diode problems; reads all input and writes most output
CLN2S	Solves second-order ordinary differential equations of the form $y''(x) = f(x,y,y')$ $0 \leq x \leq 1$ by the method of Clenshaw-Norton using Picard iteration
ITER	Initializes variables for the start of an iteration; chooses initial conditions of an electron entering the diode; determines the cell and cell boundaries that the particle is entering; counts the number of particles passing through the diode; normalizes the functional values of $\varphi''(x)$ at the Chebyshev data points
MINPHI	Computes the cell boundaries and the functional values of $\varphi(x)$ at the necessary data points; locates $\varphi_{min}(x)$ along with the corresponding data point and index for each cell
CELL	Follows the path of a particle through a cell
STOSS	Determines conditions of a particle after collision such as direction and velocity
PATH	Computes the energy and mean free path of a particle
XIC	Determines the position of a collision when a turning point is not involved
XICTP	Determines the position of a collision when a turning point is involved
XITP	Determines the position of a turning point
QUAD	Computes the distance of path across a cell by applying a three-point Gaussian quadrature
QUADTP	Computes the distance of path between two data points where one of the data points is a turning point; a three-point open-ended quadrature is used near the turning point, while Simpson's Rule is used otherwise
QUADS	Computes the distance of path between two data points by applying Simpson's Rule; neither data point is a turning point
CHEBY	Initially computes the Chebyshev data points for a given N1 and subsequently obtains coefficients for the Chebyshev fit of $\varphi''(x)$
PHI	Computes the functional values of $\varphi(x)$ from the Chebyshev fit
DPHI	Computes the functional values of $\varphi'(x)$ from the Chebyshev fit
DENS	Computes the functional values of $\varphi''(x)$ from the Chebyshev fit
DISCR1	Averages and computes the standard deviation of values stored in a two-dimensional array
DISCR2	Averages and computes the standard deviation of values stored in a one-dimensional array
PLOTF	Plots functions described in the calling vector on a single page
PLOTYX	Plots values stored in arrays on a single page
SORTYX	Sorts arrays to be plotted
SCALEY	Scales the ordinate array so that the plot will be contained on a single page

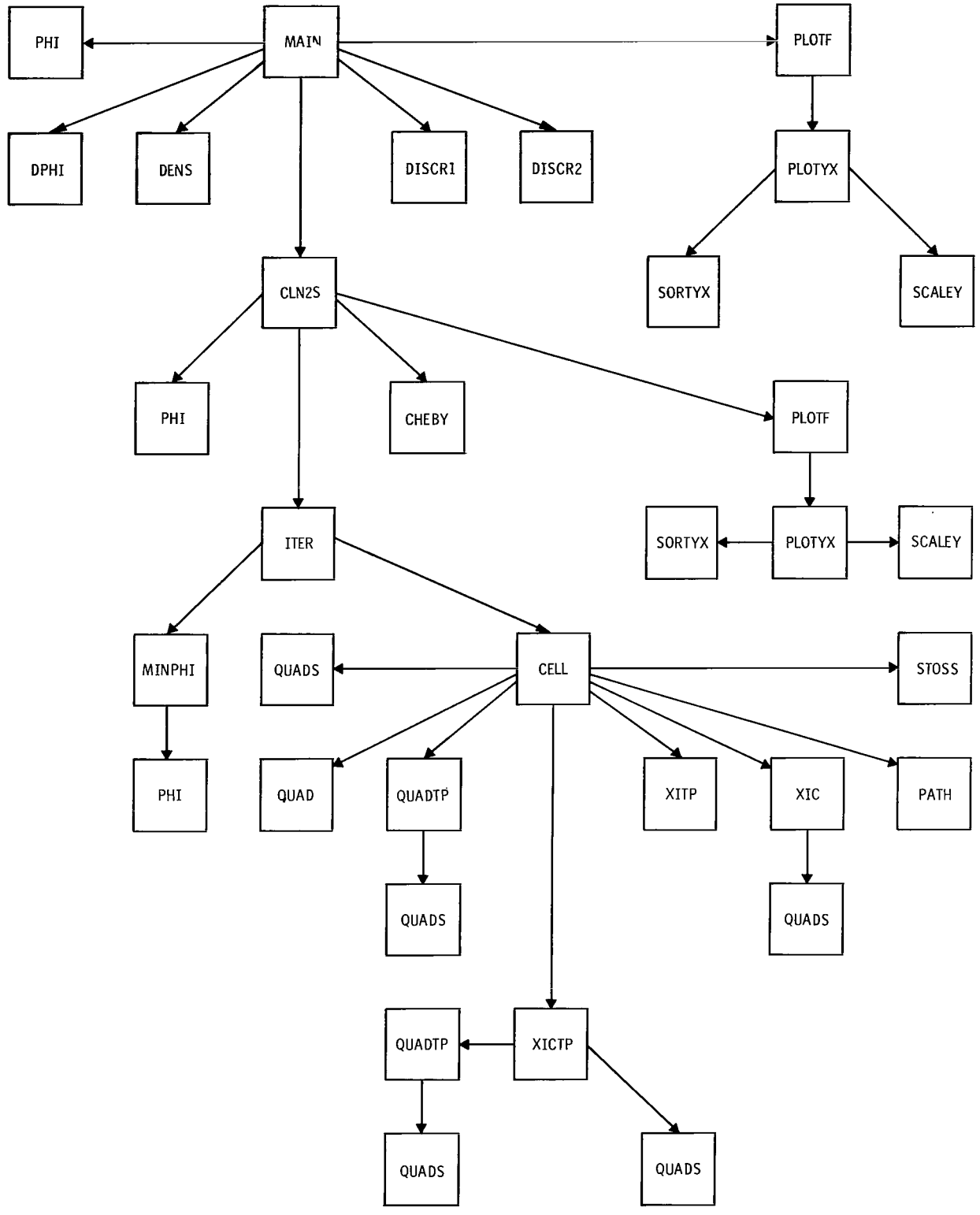


Figure 9. Directed graph of FORTRAN programs used in ENEC code.

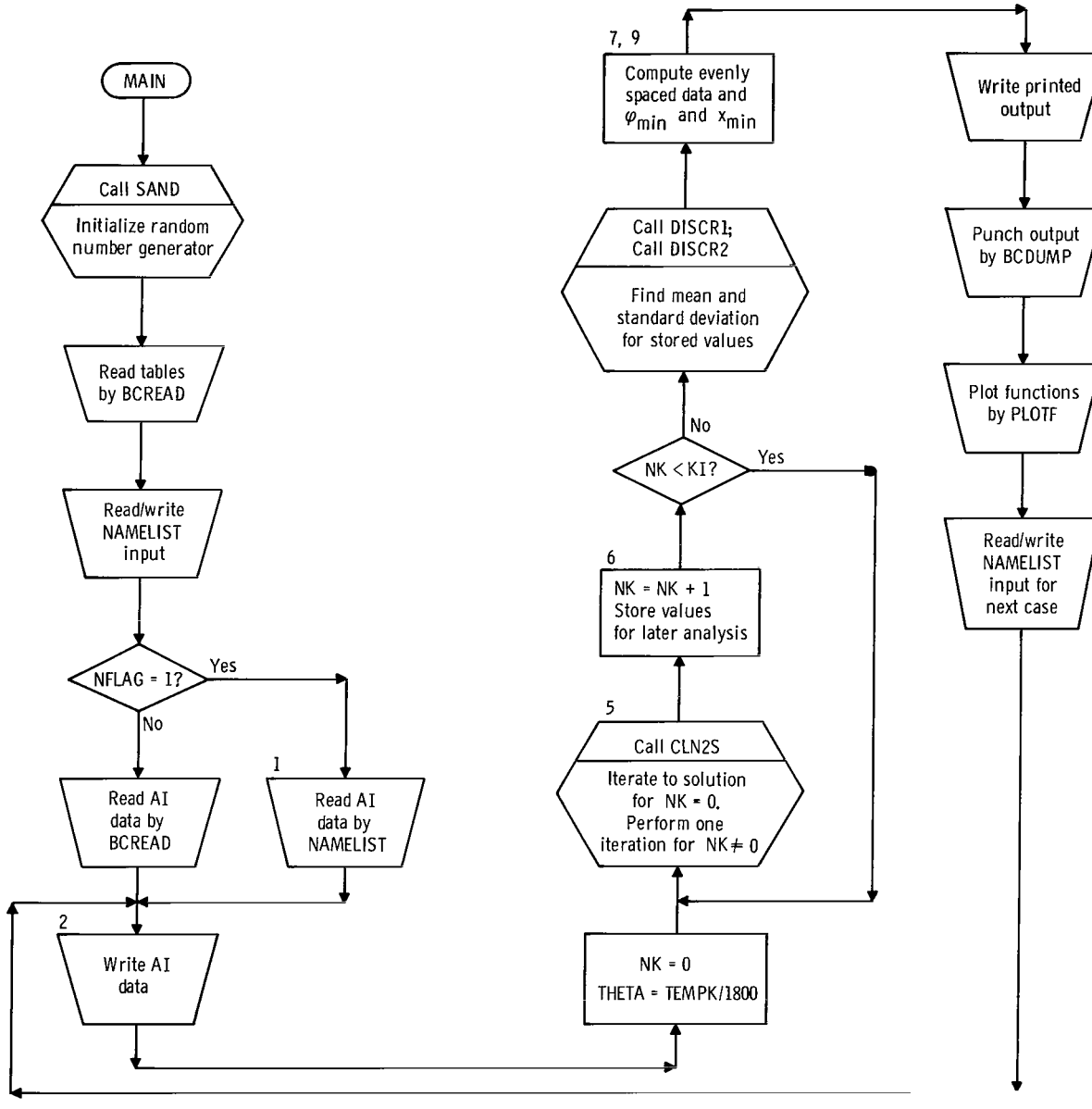


Figure 10. - Flow chart for program MAIN.

```
$IBFTC MAIN

      COMMON /CMAIN/ NO,N1,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON /CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     *XMIN(20),TPHID(20),TPHIX(1026)
      COMMON /CCLN2S/ DA(20)
      COMMON /CCHEBY/ B(20),XD(20)
      DIMENSION MAI(20,20),MDA(20,20),MB(20,20),STDAI(20),STDDA(20),
     * STDB(20),MY(20,20),MDY(20,20),MDDY(20,20),STDY(20),STDDY(20),
     * STDDDY(20),DY(20),DDY(20),CURRNT(20),H(12),HH(12),X1(21),Y1(21),
     * DY1(21),DDY1(21),FN(20,20),FKNTR(20),FN1(20),STDN(20)
      DATA H/4*6H       ,6HAI DAT,6HA READ,6H BY NA,6HMELIST,4*6H
      DATA HH/4*6H       ,6HAI DAT,6HA FROM,6H PRECE,6HEDING ,6HRUN    ,
     *3*6H       /
      DATA M/1025/
      REAL MEANC,MEANV,MAI,MDA,MB,MDPHIO,MFP,MY,MDY,MDDY
      EXTERNAL PHI,DPHI,DENS
      NAMELIST /IN1/NO,N1,KI,ALPHA,CONST,NFLAG,BC,KODE,MODE,ERROR,NS,
     *KFLAG,TEMPK,LFLAG
      NAMELIST /IN2/ AI
      CALL SAND(RO)
C     READ IN INITIAL DATA
      CALL BCREAD(VEL(1),VEL(1024))
      CALL BCREAD(MFP(1),MFP(126))
      CALL BCREAD(DISTB(1,1),DISTB(64,64))
      READ (5,IN1)
      WRITE (6,202)
      WRITE (6,IN1)
      IF(NFLAG.EQ.1)  GO TO 1
      READ (5,101) H
      CALL BCREAD(AI(1),AI(20))
      CALL BCREAD(DA(1),DA(20))
      CALL BCREAD(B(1),B(20))
      GO TO 2
    1 READ (5,IN2)
    2 WRITE (6,101) H
      WRITE (6,201) (AI(I),I=1,20)
      NK = 0
      THETA = TEMPK/11600.0
    5 CALL CLN2S(KODE,MODE,BC,ERROR)
      NK = NK + 1
      DO 6 I=1,N1
      FN(NK,I) = N(I)
      MY(NK,I) = PHI(XD(I))
      MDY(NK,I) = DPHI(XD(I))
      MDDY(NK,I) = DENS(XD(I))
      MAI(NK,I) = AI(I)
      MDA(NK,I) = DA(I)
```

```
    6 MB(NK,I) = B(I)
      FKNTR(NK) = KNTR
      CURRNT(NK) = CRRNT
      IF(NK.LT.KI)  GO TO 5
      CALL DISCR1(NK,N1,MY,Y,STDY)
      CALL DISCR1(NK,N1,MDY,DY,STDDY)
      CALL DISCR1(NK,N1,MDDY,DDY,STDDDY)
      CALL DISCR1(NK,N1,MAI,AI,STDAI)
      CALL DISCR1(NK,N1,MDA,DA,STDDA)
      CALL DISCR1(NK,N1,MB,B,STDB)
      CALL DISCR1(NK,N1,FN,FN1,STDN)
      CALL DISCR2(NK,FKNTR,FKNTR1,STDKTR)
      CALL DISCR2(NK,CURRNT,MEANC,STDC)
      MEANV = Y(N1)
      STDV = STDY(N1)
      MDPHIO = DY(1)
      STDPHI = STDDY(1)
      DO 7 I=1,21
      X1(I) = FLOAT(I-1)/20.0
      Y1(I) = PHI(X1(I))
      DY1(I) = DPHI(X1(I))
    7 DDY1(I) = DENS(X1(I))/CONST
      PHIM = 0.0
      XM = 0.0
      DO 9 I=1,M
      X = DELX*FLOAT(I-1)
      PHII = PHI(X)
      IF(PHII.GT.PHIM)  GO TO 9
      PHIM = PHII
      XM = X
    9 CONTINUE
      WRITE (6,205) (I,XD(I),FN1(I),STDN(I),I=1,N1)
      WRITE (6,203) (I,AI(I),STDAI(I),DA(I),STDDA(I),B(I),STDB(I),
     *I=1,N1)
      WRITE (6,207) (I,Y(I),STDY(I),DY(I),STDDY(I),DDY(I),STDDDY(I),I=1,
     *N1)
      WRITE (6,204) MEANC,STDC,MEANV,STDV,MDPHIO,STDPHI,FKNTR1,STDKTR
      WRITE (6,209) PHIM,XM
      WRITE (6,208) (X1(I),Y1(I),DY1(I),DDY1(I),I=1,21)
      CALL BCDUMP(AI(1),AI(20))
      CALL BCDUMP(DA(1),DA(20))
      CALL BCDUMP(B(1),B(20))
      CALL PLOTF(21,0.0,1.0,PHI)
      CALL PLOTF(21,0.0,1.0,DPHI)
      CALL PLOTF(21,0.0,1.0,DENS)
      READ (5,IN1)
      WRITE (6,202)
      WRITE (6,IN1)
      DO 8 I=1,12
    8 H(I) = HH(I)
      GO TO 2
  101 FORMAT (12A6)
  201 FORMAT (4HKAI=,/,(6E20.8))
  202 FORMAT (1H1)
  203 FORMAT (1HK,13X,7HMEAN AI,13X,7HSTD. AI,13X,7HMEAN DA,13X,7HSTD. D
     *A,13X,6HMEAN B,14X,6HSTD. B,/,1HK,/,(1H ,I2,6F20.8))
```

```
204 FORMAT (1HK,30X,4HMEAN,16X,4HSTD.,//,13X,7HCURRENT,2F20.8,//,13X,4
   *HVOLT,3X,2F20.8,//,13X,5HDPHIO,2X,2F20.8,//,13X,4HKNTR,3X,2F20.3)
205 FORMAT (1H1,16X,2HXD,20X,6HMEAN N,19X,6HSTD. N,///,(1H ,I2,F20.8,2
   *F20.3))
206 FORMAT (1HK,2H K,10X,1HY,18X,2HDY,17X,3HDDY,/,(1H ,I2,3F20.8))
207 FORMAT (1HK,14X,6HMEAN Y,14X,6HSTD. Y,13X,7HMEAN DY,13X,7HSTD. DY,
   *12X,8HMEAN DDY,12X,8HSTD. DDY,/,1HK,/,(1H ,I2,6F20.8))
208 FORMAT (20H1EQUALLY SPACED DATA,///,10X,1HX,19X,6HPHI(X),14X,7HDPH
   *I(X),13X,4HN(X),///,(4F20.8))
209 FORMAT (1HK,12X,7HPHIMIN=,F15.8,//,13X,5HXMIN=,F17.8)
    END
```

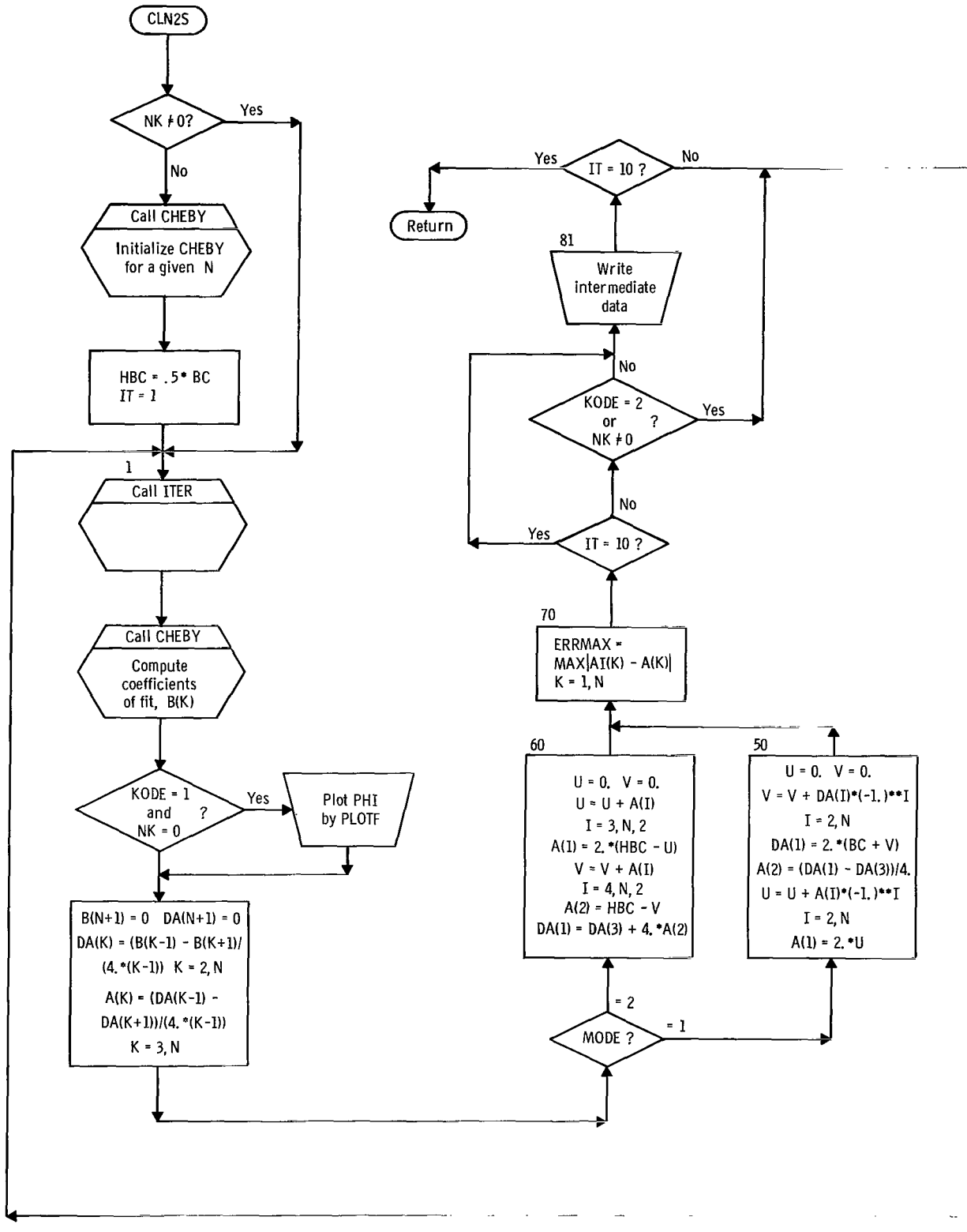
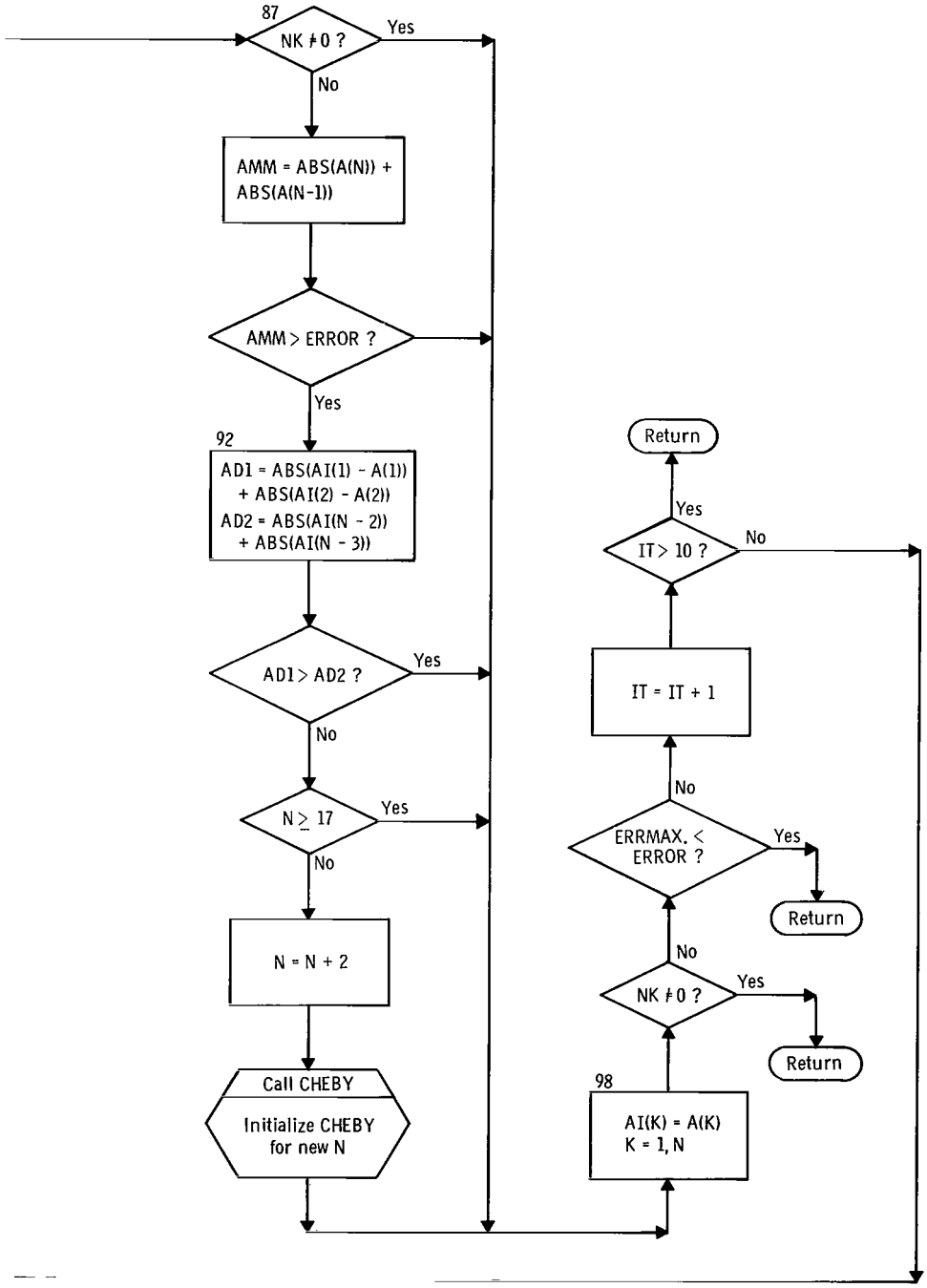


Figure 11. - Flow chart for



subroutine CLN2S.

```
$IBFTC CLN2S

      SUBROUTINE CLN2S(KODE,MODE,BC,ERROR)
      COMMON /CCLN2S/ DA(20)
      COMMON /CCHEBY/ B(20),XD(20)
      COMMON /CMAIN/ NO,N,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      DIMENSION A(20)
      DATA NIT/10/
      EXTERNAL PHI
      IF(NK.NE.0)  GO TO 1
      CALL CHEBY(1)
      HBC=.5*BC
    1 DO 99 IT=1,NIT
      CALL ITER
      CALL CHEBY(2)
      IF(KODE.EQ.1.AND.NK.EQ.0)  CALL PLOTF(21,0.0,1.0,PHI)
      B(N+1)=0
      DA(N+1)=0
      DO 40 K=2,N
      R=K-1
   40 DA(K)=(B(K-1)-B(K+1))/(4.*R)
      DO 41 K=3,N
      R=K-1
   41 A(K)=(DA(K-1)-DA(K+1))/(4.*R)
      GO TO (50,60),MODE
   50 U=0
      V=0
      DO 51 I=2,N
   51 V = V + DA(I)*(-1.)**I
      DA(1)= 2.*(BC+V)
      A(2)=(DA(1)-DA(3))/4.
      DO 52 I=2,N
   52 U = U + A(I)*(-1.)**I
      A(1)= 2.*U
      GO TO 70
   60 U=0
      V=0
      DO 61 I=3,N,2
   61 U=U + A(I)
      A(1)=2.*(HBC-U)
      DO 62 I=4,N,2
   62 V=V + A(I)
      A(2) = HBC - V
      DA(1)=DA(3) + 4.*A(2)
   70 ERRMAX = 0
      DO 80 K=1,N
      ERR=ABS(AI(K)-A(K))
   80 IF(ERR.GT.ERRMAX) ERRMAX=ERR
      IF (IT.EQ.NIT) GO TO 81
      IF(KODE.EQ.2.OR.NK.NE.0)  GO TO 87
   81 WRITE (6,82)
```

```
 82 FORMAT(1HL,2H K,9X,2HAI,19X,1HB,19X,1HA,19X,2HDA/1HJ)
    WRITE (6,84) (I,AI(I),B(I),A(I),DA(I),I=1,N)
 84 FORMAT(1H ,I2,4F20.8)
    WRITE (6,85) ERRMAX,IT
 85 FORMAT (8HKERRMAX=,E18.8,5X,3HIT=,I4)
    IF (IT.EQ.NIT) RETURN
 87 IF(NK.NE.0)  GO TO 98
    AMM = ABS(A(N)) + ABS(A(N-1))
    IF(AMM    .GT.ERROR) GO TO 92
    GO TO 98
 92 AD1 = ABS(AI(1)-A(1)) + ABS(AI(2)-A(2))
    AD2 = ABS(AI(N-2)) + ABS(AI(N-3))
    IF(AD1.GT.AD2) GO TO 98
    IF(N.GE.17)  GO TO 98
    N=N+2
    CALL CHEBY(1)
 98 DO 100 K=1,N
100 AI(K) = A(K)
    IF(NK.NE.0)  RETURN
    IF(ERRMAX.LT.ERROR)  RETURN
 99 CONTINUE
    RETURN
    END
```

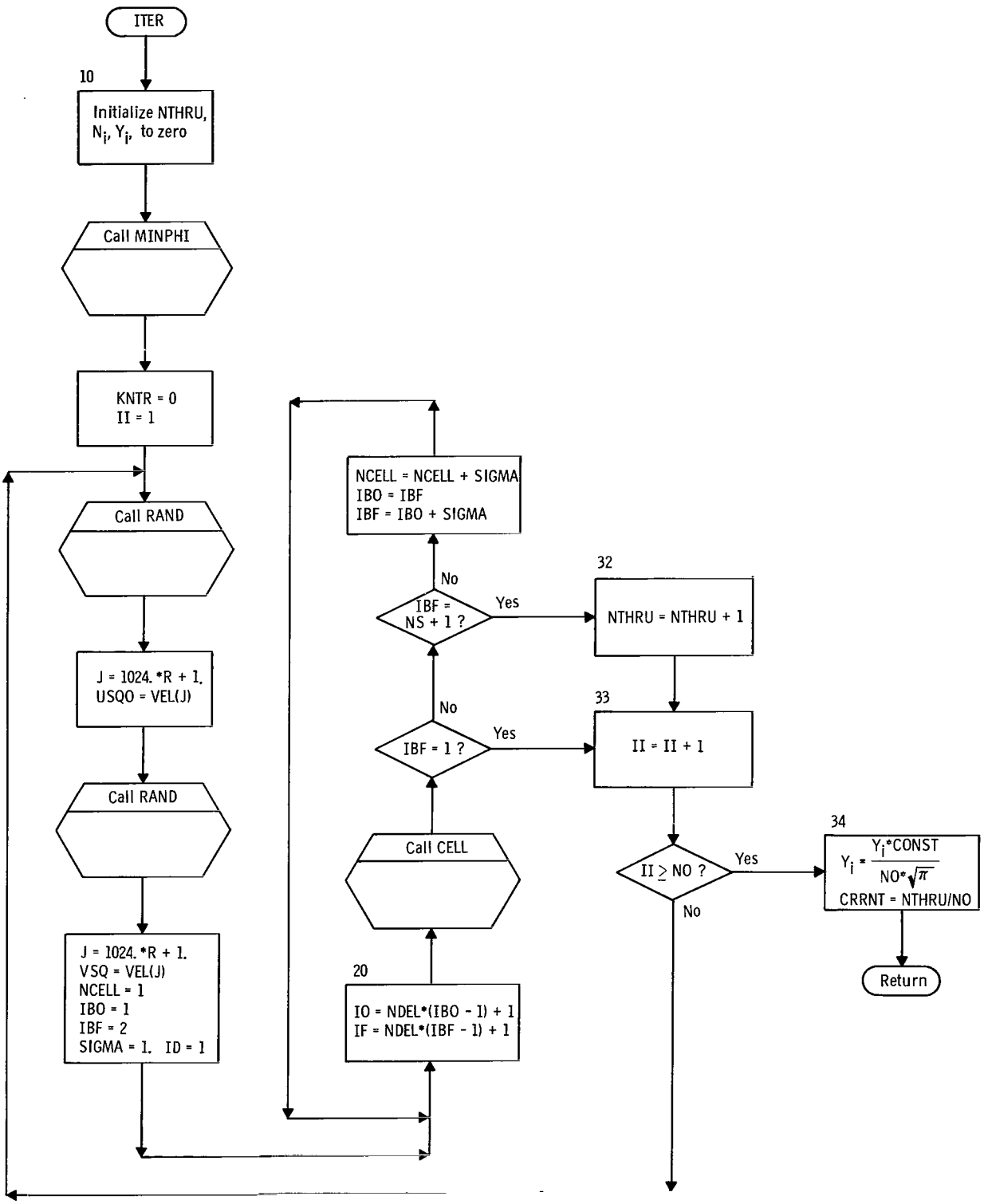


Figure 12. - Flow chart for subroutine ITER.

```
$IBFTC ITER

      SUBROUTINE ITER
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON /CMAIN/ NO,N1,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
       COMMON /CCELL/NCELL,XO,XF,IO,IF,USQO
      DATA SQRTPI/1.77245385/
      NTHRU = 0
      DO 10 I=1,20
      N(I)=0
   10 Y(I) = 0.0
      CALL MINPHI
      KNTR=0
      DO 33 II=1,NO
      CALL RAND(R)
      J = IFIX(1024.*R) + 1
      USQO = VEL(J)
      CALL RAND(R)
      J = IFIX(1024.*R) + 1
      VSQ = VEL(J)
      NCELL = 1
      IBO = 1
      IBF = 2
      SIGMA = 1.0
      ID = 1
   20 IO = NDEL*(IBO-1)+1
      IF = NDEL*(IBF-1)+1
      CALL CELL
      IF(IBF.EQ.1)  GO TO 33
      IF(IBF.EQ.NS+1)  GO TO 32
      NCELL = NCELL + IFIX(SIGMA)
      IBO = IBF
      IBF = IBO + IFIX(SIGMA)
      GO TO 20
   32 NTHRU = NTHRU + 1
   33 CONTINUE
      DO 34 I=1,N1
   34 Y(I) = (Y(I)/(FLOAT(NO)*SQRTPI))*CONST
      CRRNT = FLOAT(NTHRU)/FLOAT(NO)
      RETURN
      END
```

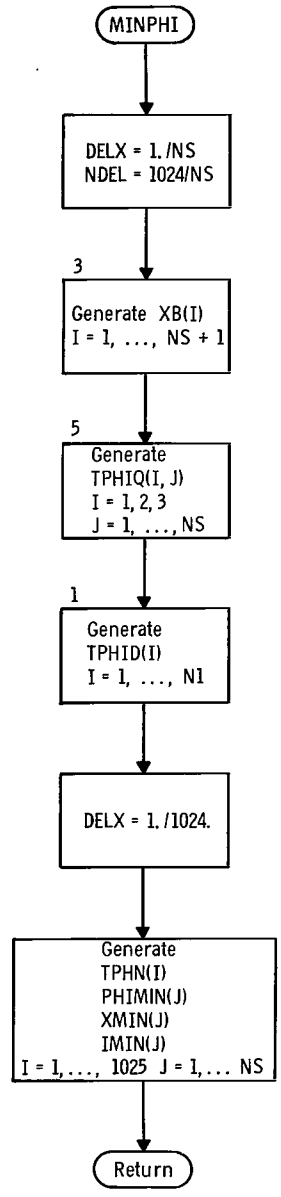


Figure 13. - Flow chart for subroutine MINPHI.

```
$IBFTC MINPHI

      SUBROUTINE MINPHI
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      COMMON /CMAIN/ NO,N1,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      COMMON /CCHEBY/ B(20),XD(20)
      DATA N/1024/,FN/1024.0/
      DATA AA/7.74596692E-1/
      DELX = 1.0/FLOAT(NS)
      NDEL = N/NS
      DO 3 I=1,NS
    3 XB(I) = DELX*FLOAT(I-1)
      XB(NS+1) = 1.0
      DO 5 J=1,NS
      XX = (XB(J+1)-XB(J))*0.5
      TPHIQ(1,J) = PHI(XB(J)+XX*(1.0-AA))
      TPHIQ(2,J) = PHI((XB(J)+XB(J+1))*0.5)
    5 TPHIQ(3,J) = PHI(XB(J)+XX*(1.0+AA))
      DO 1  I=1,N1
    1 TPHID(I) = PHI(XD(I))
      PHIMIN(1) = 1.0E+30
      DELX = 1.0/FN
      M = N+1
      J = 1
      DO 2 I=1,M
      XX=DELX*FLOAT(I-1)
      IF(XX.LE.XB(J+1))  GO TO 11
      J = J+1
      PHIMIN(J) = TPHIX(I-1)
      IMIN(J) = I-1
      XMIN(J) = DELX*FLOAT(I-2)
   11 TPHIX(I) = PHI(XX)
      IF(PHIMIN(J).LT.TPHIX(I))  GO TO 2
      PHIMIN(J) = TPHIX(I)
      IMIN(J) = I
      XMIN(J) = XX
    2 CONTINUE
      RETURN
      END
```

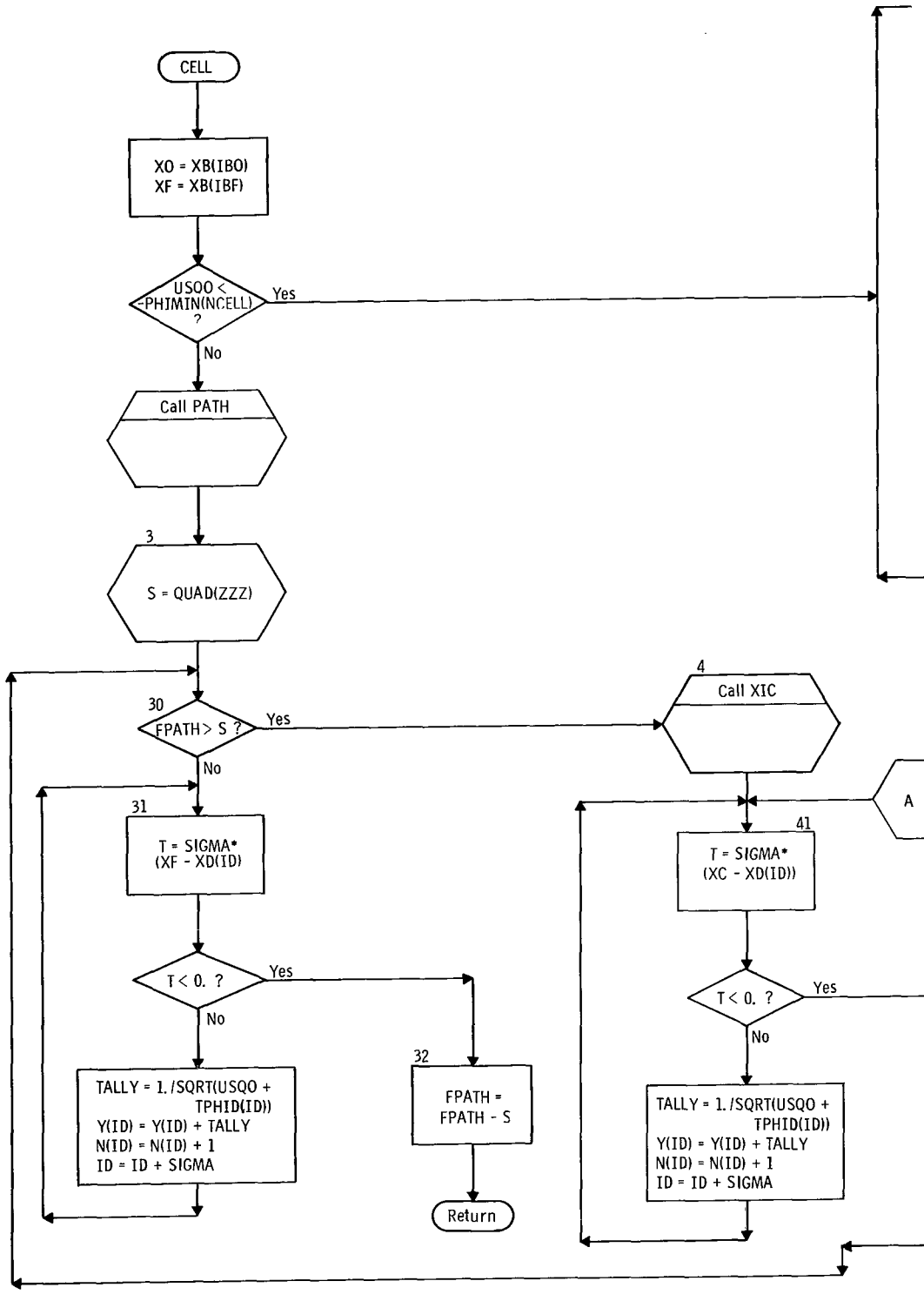
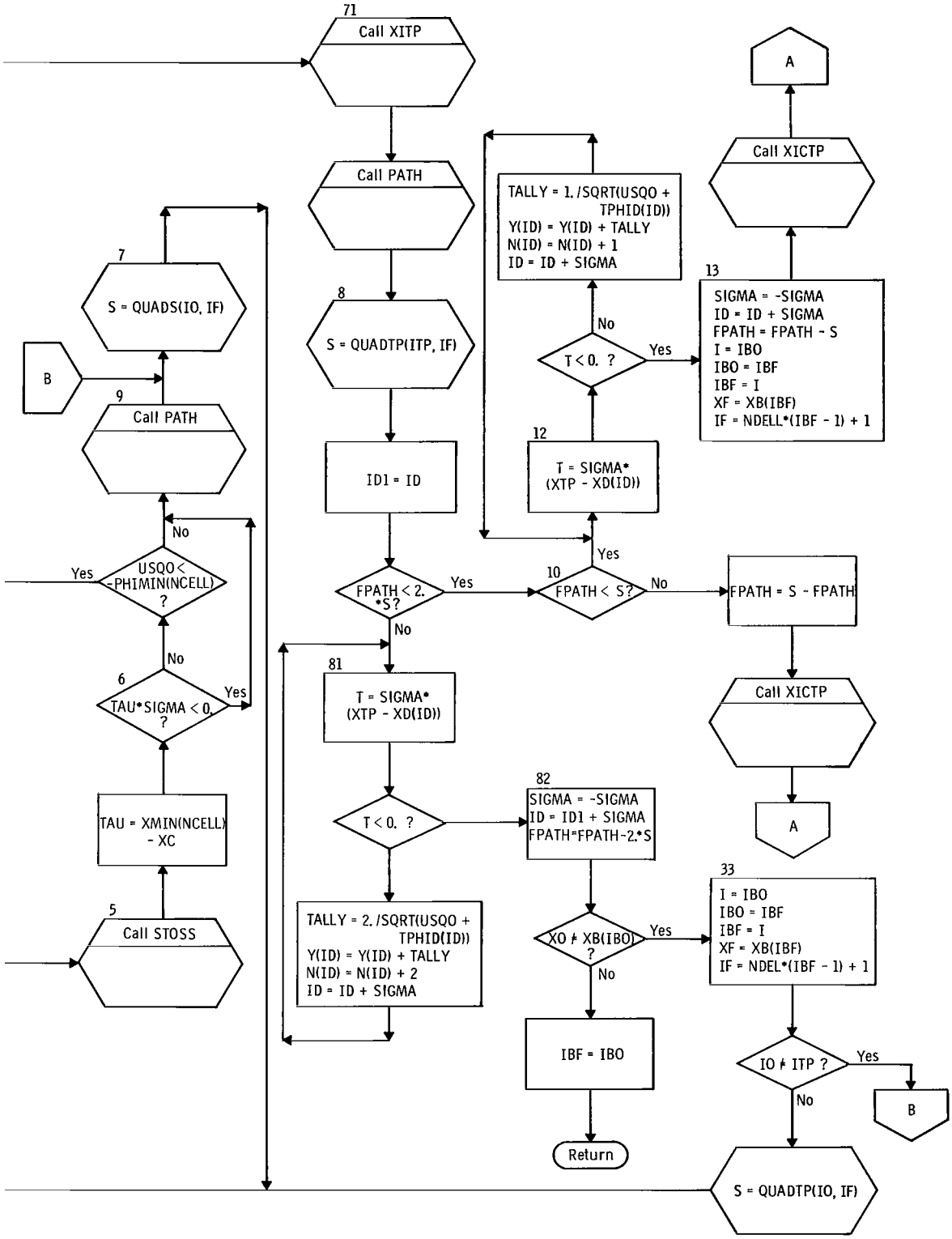


Figure 14. - Flow chart for



subroutine CELL.

```
$IBFTC CELL

      SUBROUTINE CELL
      COMMON /CCELL/ NCELL,XO,XF,IO,IF,USQO
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      COMMON /CXITP/ ITP,XTP
      COMMON /CXICTP/ IC,XC
      COMMON /CCHEBY/ B(20),XD(20)
C     ENTER CELL
      XO = XB(IBO)
      XF = XB(IBF)
      IF(USQO .LT.-PHIMIN(NCELL))  GO TO 71
      CALL PATH(IO,IF)
    3 S = QUAD(ZZZZZZ)
   30 IF(FPATH.LT.S)  GO TO 4
   31 T = SIGMA*(XF-XD(ID))
      IF(T.LT.0.0)  GO TO 32
      TALLY = 1.0/SQRT(USQO+TPHID(ID))
      Y(ID) = Y(ID) + TALLY
      N(ID) = N(ID) + 1
      ID = ID + IFIX(SIGMA)
      GO TO 31
   32 FPATH = FPATH - S
      RETURN
    4 CALL XIC(IO,IF)
   41 T = SIGMA*(XC-XD(ID))
      IF(T.LT.0.0)  GO TO 5
      TALLY = 1.0/SQRT(USQO+TPHID(ID))
      Y(ID) = Y(ID) + TALLY
      N(ID) = N(ID) + 1
      ID = ID + IFIX(SIGMA)
      GO TO 41
    5 CALL STOSS
      TAU = XMIN(NCELL)-XC
    6 IF(TAU*SIGMA.LT.0.0)  GO TO 9
      IF(USQO.LT.-PHIMIN(NCELL))  GO TO 71
    9 CALL PATH(IO,IF)
    7 S = QUADS(IO,IF)
      GO TO 30
   71 CALL XITP(IO)
      CALL PATH(IO,ITP)
    8 S = QUADTP(ITP,IO)
      ID1 = ID
      IF(FPATH.LT.2.0*S)  GO TO 10
   81 T = SIGMA*(XTP-XD(ID))
      IF(T.LT.0.0)  GO TO 82
      TALLY = 2.0/SQRT(USQO+TPHID(ID))
      Y(ID) = Y(ID) + TALLY
      N(ID) = N(ID) + 2
      ID = ID + IFIX(SIGMA)
```

```
      GO TO 81
   82 SIGMA = -SIGMA
      ID = IDL + IFIX(SIGMA)
      FPATH = FPATH - 2.0*S
      IF(XO.NE.XB(IBO))  GO TO 33
      IBF = IBO
      RETURN
   33 I = IBO
      IBO = IBF
      IBF = I
      XF = XB(IBF)
      IF = NDEL*(IBF-1) + 1
      IF(IO.NE.ITP)  GO TO 7
      S = QUADTP(IO,IF)
      GO TO 30
   10 IF(FPATH.GT.S)  GO TO 12
      FPATH = S - FPATH
      CALL XICTP(ITP,IO)
      GO TO 41
   12 T = SIGMA*(XTP-XO(ID))
      IF(T.LT.0.0)  GO TO 13
      TALLY = 1.0/SQRT(USQO+TPHID(ID))
      Y(ID) = Y(ID) + TALLY
      N(ID) = N(ID) + 1
      ID = ID + IFIX(SIGMA)
      GO TO 12
   13 SIGMA = -SIGMA
      ID = ID + IFIX(SIGMA)
      FPATH = FPATH - S
      I = IBO
      IBO = IBF
      IBF = I
      XF = XB(IBF)
      IF = NDEL*(IBF-1) + 1
      CALL XICTP(ITP,IO)
      GO TO 41
      END
```

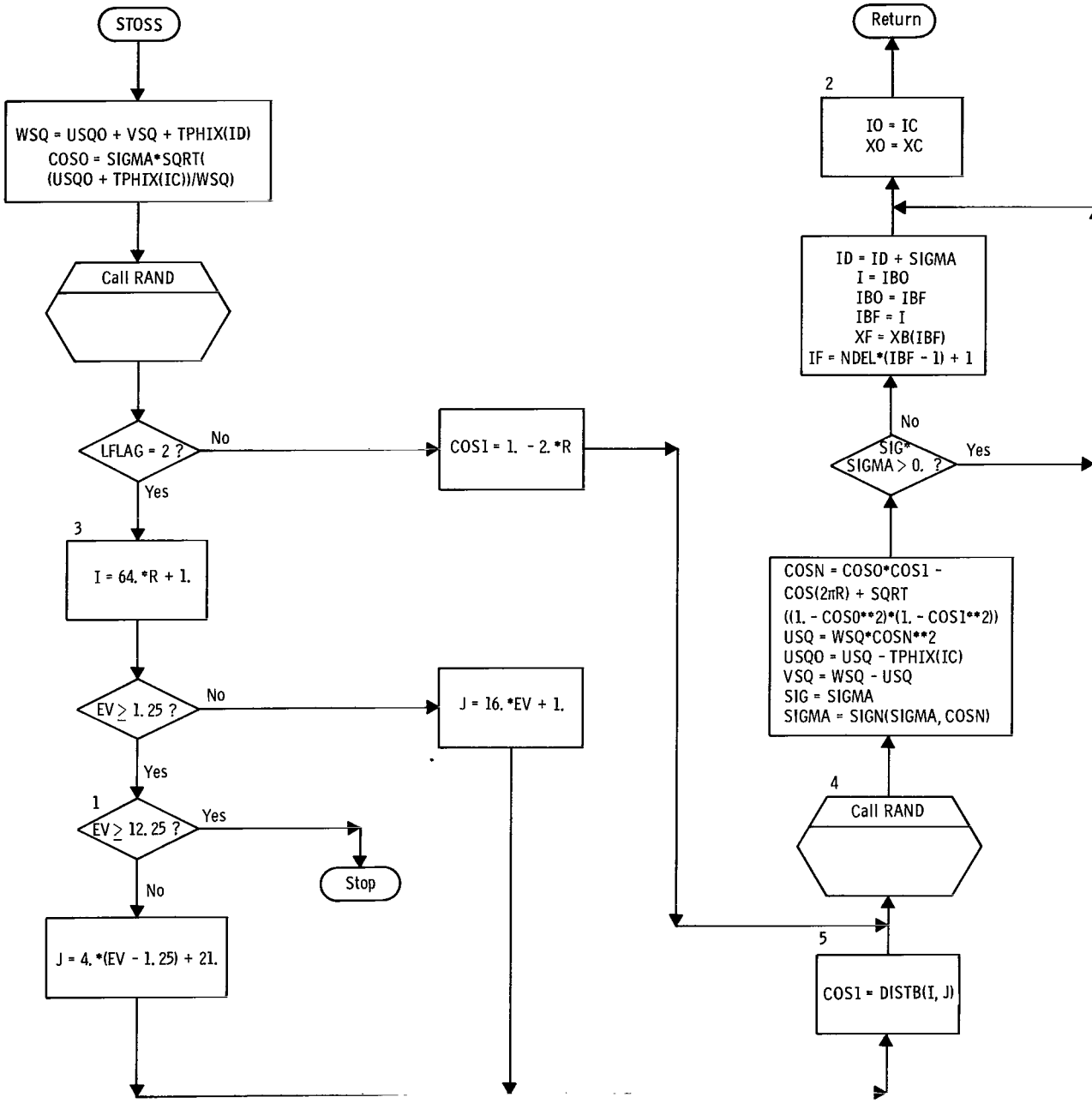


Figure 15. - Flow chart for subroutine STOSS.

```
$IBFTC STOSS

      SUBROUTINE  STOSS
      COMMON /CMAIN/ NO,NI,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      COMMON /CCELL/ NCELL,XO,XF,IO,IF,USQO
      COMMON /CXICTP/ IC,XC
      COMMON /CPATH/ EV
      DATA TWOPI/6.2831853/
      WSQ=USQO+VSQ+TPHIX(IC)
      COSO = SIGMA*SQRT((USQO+TPHIX(IC))/WSQ)
      CALL RAND(R)
      IF(LFLAG.EQ.2)  GO TO 3
      COS1 = 1.0-2.0*R
      GO TO 4
    3 I = IFIX(64.0*R)+1
      IF(EV.GE.1.25)  GO TO 1
      J = IFIX(16.0*EV)+1
      GO TO 5
    1 IF(EV.GE.12.25)  STOP
      J = IFIX(4.0*(EV-1.25))+21
    5 COS1 = DISTB(I,J)
    4 CALL RAND(R)
      COSN = COSO*COS1-COS(TWOPI*R)*SQRT((1.0-COSO**2)*(1.0-COS1**2))
      USQ = WSQ*COSN**2
      USQO = USQ-TPHIX(IC)
      VSQ = WSQ-USQ
      SIG = SIGMA
      SIGMA = SIGN(SIGMA,COSN)
      IF(SIG*SIGMA.GT.0.0)  GO TO 2
      ID = ID + IFIX(SIGMA)
      I = IBO
      IBO = IBF
      IBF = I
      XF = XB(IBF)
      IF = NDEL*(IBF-1) + 1
    2 IO = IC
      XO = XC
      RETURN
      END
```

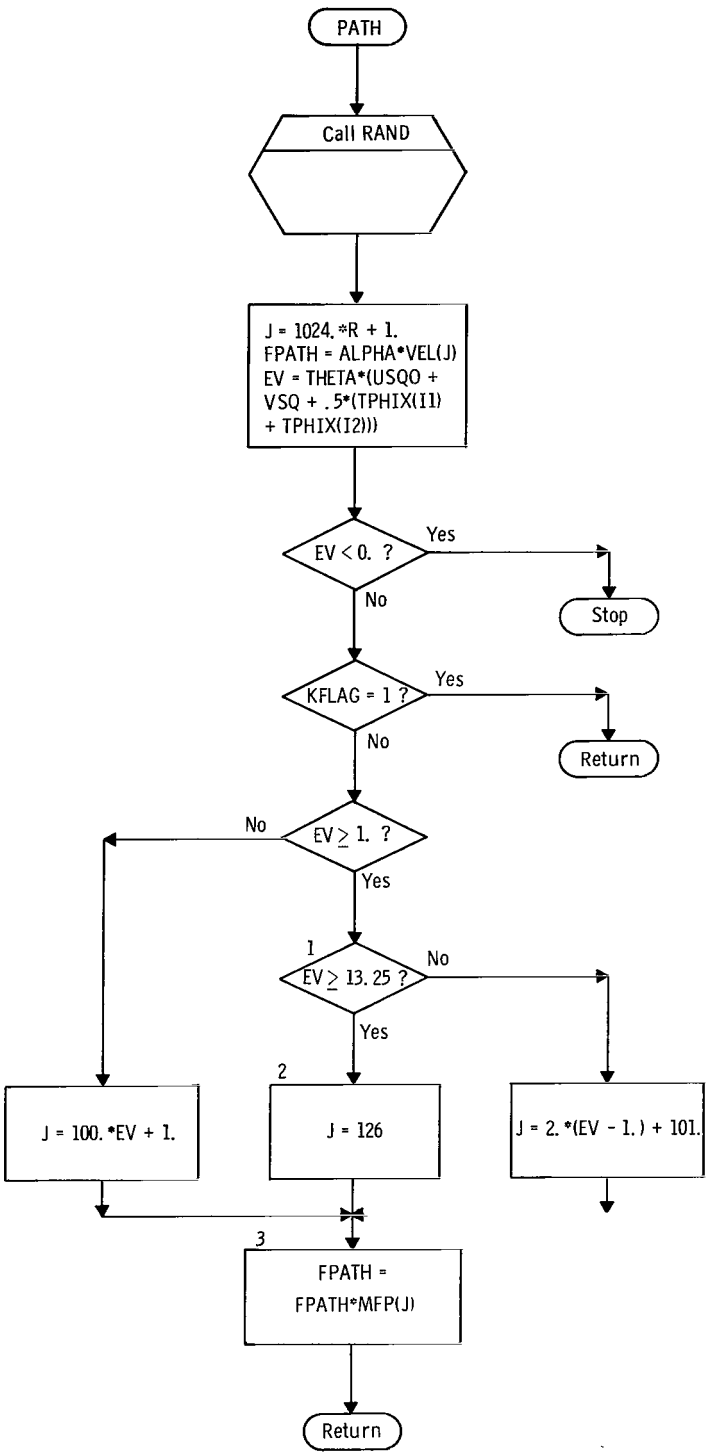


Figure 16. - Flow chart for subroutine PATH.

```
$IBFTC PATH

      SUBROUTINE PATH(I1,I2)
      COMMON /CPATH/ EV
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON /CMAIN/ NO,N1,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      COMMON /CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      COMMON /CCELL/ NCELL,XO,XF,IO,IF,USQO
      REAL MFP
      CALL RAND(R)
      J = IFIX(1024.*R)+1
      FPATH = ALPHA*VEL(J)
      EV = THETA*(USQO+VSQ+0.5*(TPHIX(I1)+TPHIX(I2)))
      IF(EV.LT.0.0)  STOP
      IF(KFLAG.EQ.1)  RETURN
      IF(EV.GE.1.0)  GO TO 1
      J = IFIX(100.0*EV)+1
      GO TO 3
    1 IF(EV.GE.13.25)  GO TO 2
      J = IFIX(2.0*(EV-1.0))+101
      GO TO 3
    2 J = 126
    3 FPATH = FPATH*MFP(J)
      RETURN
      END
```

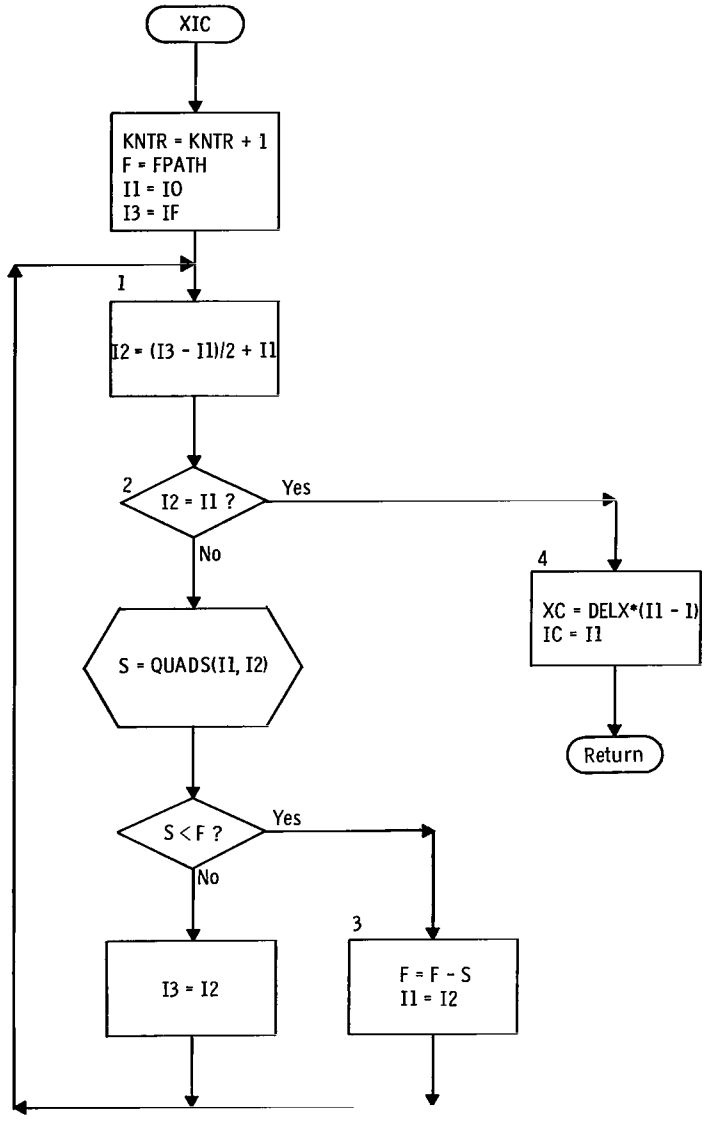


Figure 17. - Flow chart for subroutine XIC.

```
$IBFTC XIC

      SUBROUTINE XIC(IO,IF)
      COMMON /CXICTP/ IC,XC
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      KNTR = KNTR + 1
      F = FPATH
      I1 = IO
      I3 = IF
    1 I2 = (I3-I1)/2 + I1
    2 IF(I2.EQ.I1)  GO TO 4
      S = QUADS(I1,I2)
      IF(S.LT.F)  GO TO 3
      I3 = I2
      GO TO 1
    3 F = F - S
      I1 = I2
      GO TO 1
    4 XC = DELX*FLOAT(I1-1)
      IC = I1
      RETURN
      END
```

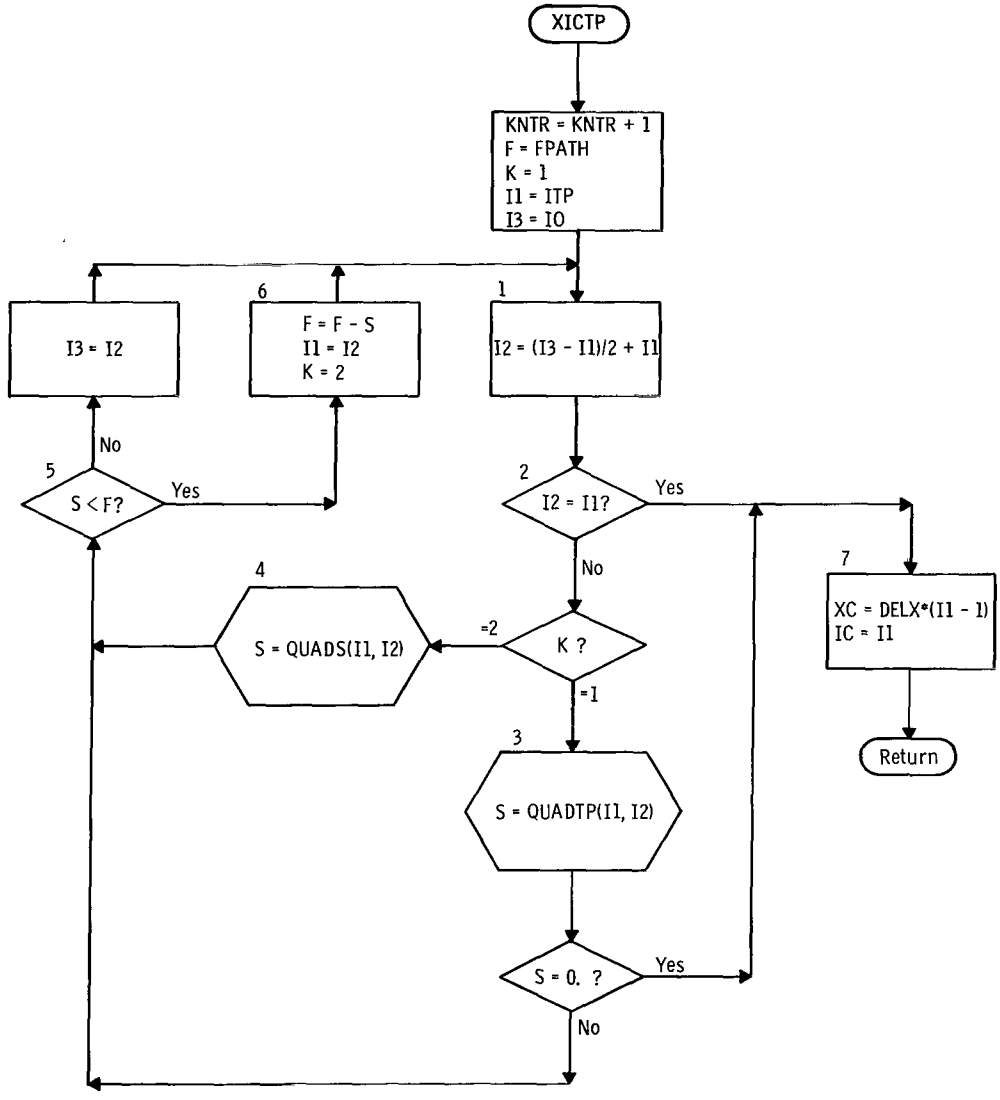


Figure 18. - Flow chart for subroutine XICTP.

```
$IBFTC XICTP

      SUBROUTINE XICTP(ITP,IO)
      COMMON /CXICTP/ IC,XC
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      KNTR = KNTR + 1
      F = FPATH
      K = 1
      I1 = ITP
      I3 = IO
    1 I2 = (I3-I1)/2 + I1
    2 IF(I2.EQ.I1)  GO TO 7
      GO TO (3,4),K
    3 S = QUADTP(I1,I2)
      IF(S.EQ.0.0)  GO TO 7
      GO TO 5
    4 S = QUADS(I1,I2)
    5 IF(S.LT.F)  GO TO 6
      I3 = I2
      GO TO 1
    6 F = F - S
      I1 = I2
      K = 2
      GO TO 1
    7 XC = DELX*FLOAT(I1-1)
      IC = I1
      RETURN
      END
```

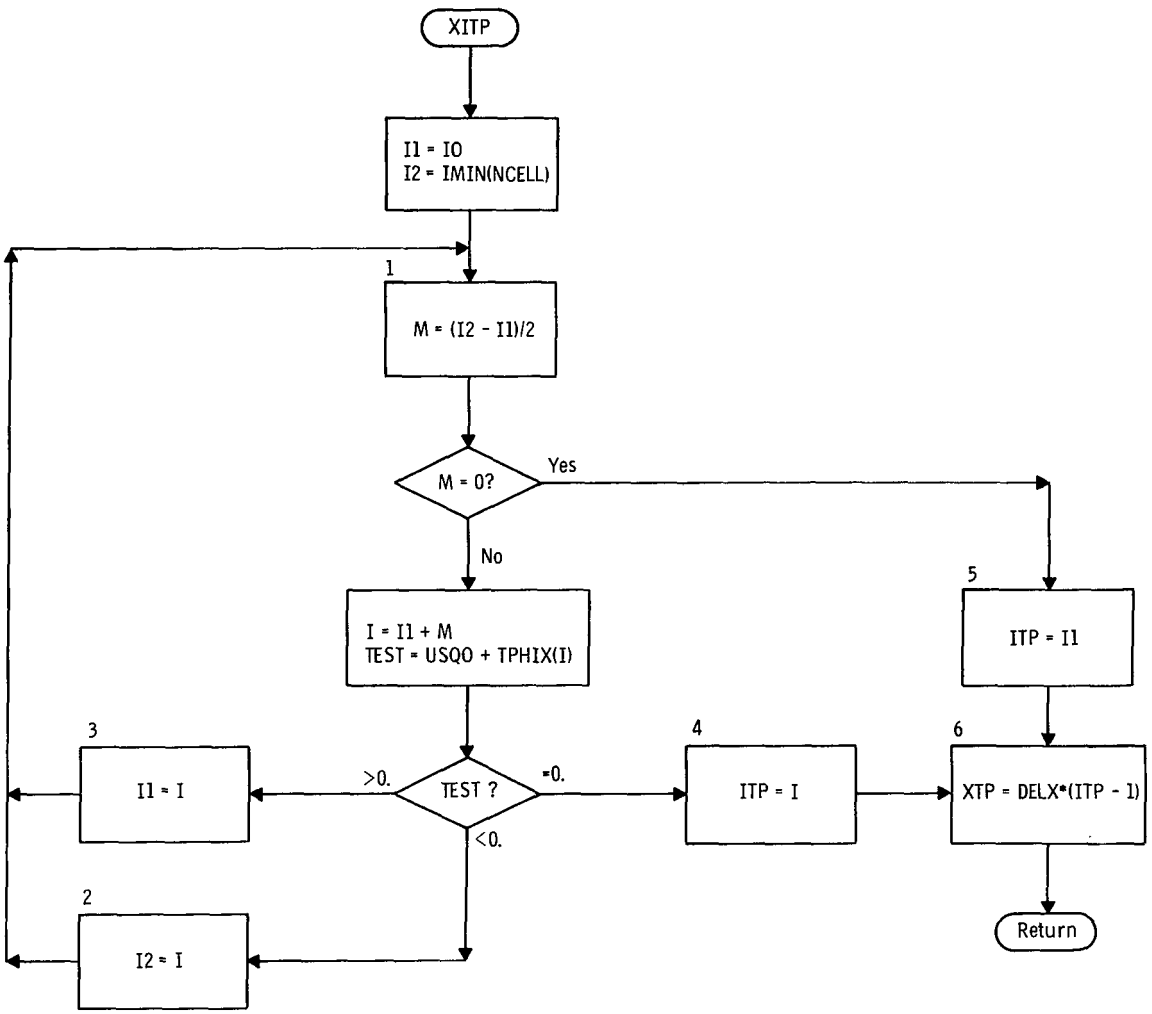


Figure 19. - Flow chart for subroutine XITP.

```
$IBFTC XITP

      SUBROUTINE XITP(IO)
      COMMON /CXITP/ ITP,XTP
      COMMON /CCELL/ NCELL,X0,XF,ZZZZZZ,IF,USQO
      COMMON/CMNPHI/NDEL,DELX,X8(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      I1 = IO
      I2 = IMIN(NCELL)
    1 M=(I2-I1)/2
      IF(M.EQ.0) GO TO 5
      I=I1+M
      TEST=USQO+TPHIX(I)
      IF(TEST)2,4,3
    2 I2=I
      GO TO 1
    3 I1=I
      GO TO 1
    4 ITP=I
      GO TO 6
    5 ITP=I1
    6 XTP = DELX*FLOAT(ITP-1)
      RETURN
      END
```

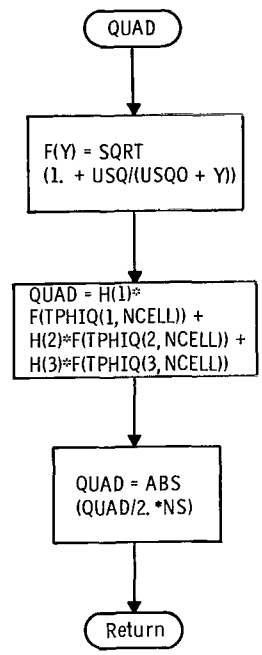


Figure 20. - Flow chart for subroutine QUAD.

```
$IBFTC QUAD

      FUNCTION QUAD(ZZZZZZ)
      COMMON /CMAIN/ NO,N1,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON /CCELL/ NCELL,XO,XF,IO,IF,USQO
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      DIMENSION H(2)
      DATA H/5.55555556E-1,8.88888889E-1/
      F(Y) = SQRT(1.0+VSQ/(USQO+Y))
      QUAD = H(1)*F(TPHIQ(1,NCELL)) + H(2)*F(TPHIQ(2,NCELL)) +
     * H(1)*F(TPHIQ(3,NCELL))
      QUAD = ABS(QUAD/(2.0*FLOAT(NS)))
      RETURN
      END
```

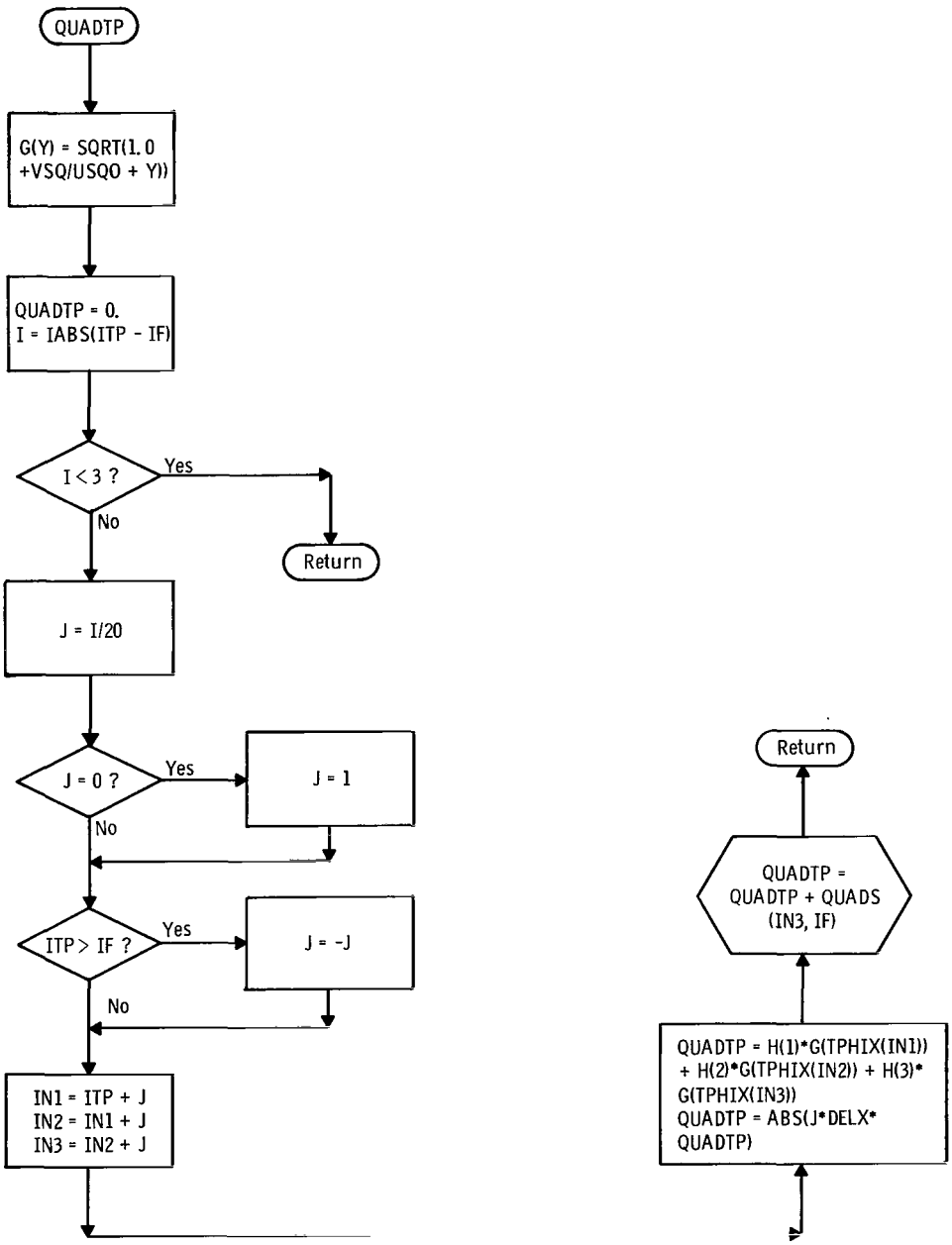


Figure 21. - Flow chart for subroutine QUADTP.

```
$IBFTC QUADTP

      FUNCTION QUADTP(ITP,IF)
C     OPEN ENDED INTEGRATION
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON /CCELL/ NCELL,XO,XF,IO,ZZZZZZ,USQO
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
      DIMENSION H(3)
      DATA H/4.84974226,-3.91918359,2.4/
      G(Y) = SQRT(1.0+VSQ/(USQO+Y))
      QUADTP = 0
      I = IABS(ITP-IF)
      IF(I.LT.3)  RETURN
      J = I/20
      IF(J.EQ.0)  J = 1
      IF(ITP.GT.IF)  J = -J
      IN1 = ITP + J
      IN2 = IN1 + J
      IN3 = IN2 + J
      QUADTP = H(1)*G(TPHIX(IN1)) + H(2)*G(TPHIX(IN2)) +
     * H(3)*G(TPHIX(IN3))
      QUADTP = ABS(FLOAT(J)*DELX*QUADTP)
      QUADTP = QUADTP + QUADS(IN3,IF)
      RETURN
      END
```

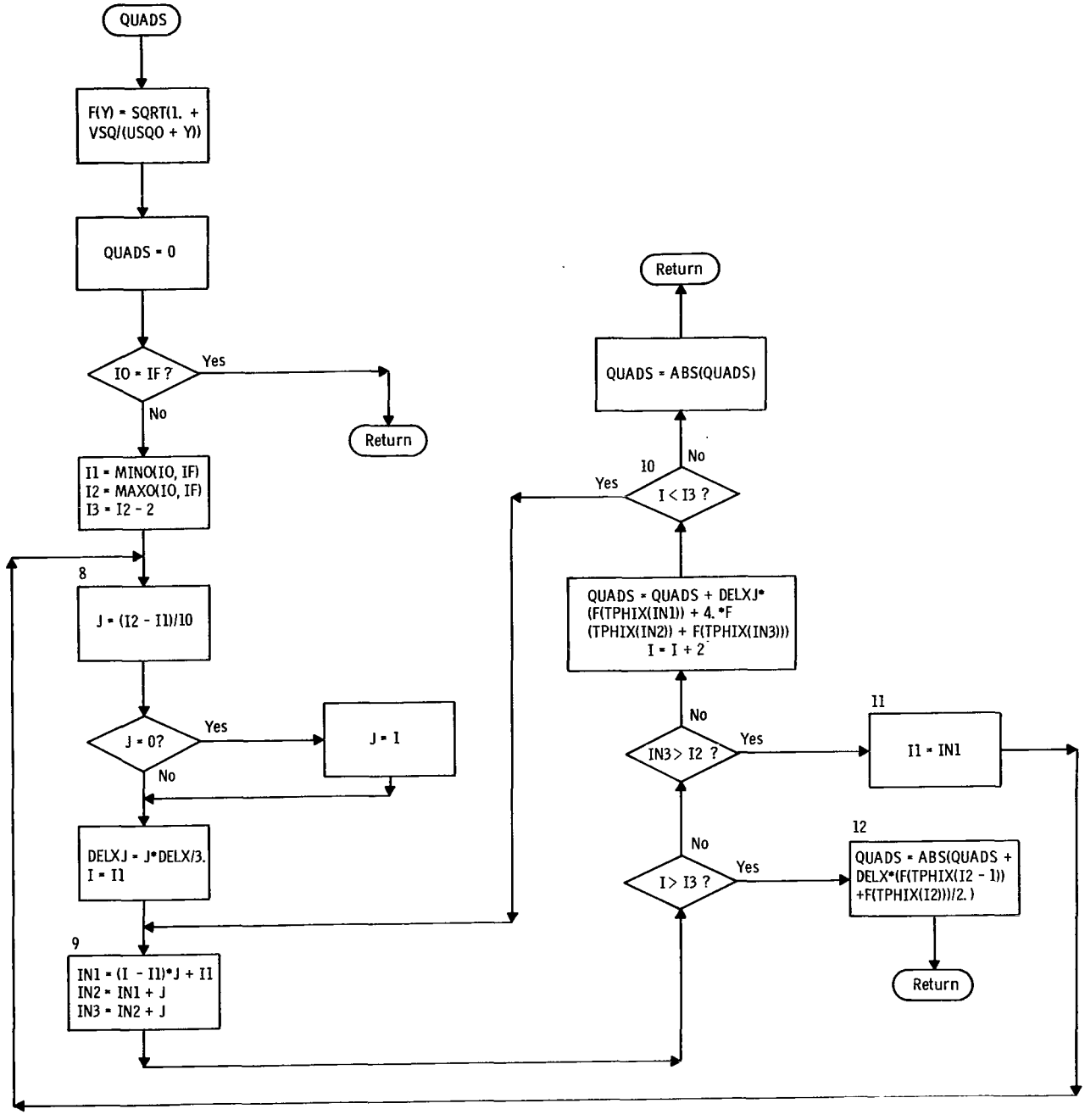


Figure 22. - Flow chart for subroutine QUADS.

```
$IBFTC QUADS

      FUNCTION QUADS(IO,IF)
      COMMON /CITER/N(20),Y(20),KNTR,VSQ,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      COMMON /CCELL/ NCELL,XO,XF,ZZZZZZ(2),USQO
      COMMON/CMNPHI/NDEL,DELX,XB(20),TPHIQ(3,20),IMIN(20),PHIMIN(20),
     * XMIN(20),TPHID(20),TPHIX(1026)
C     SIMPSON'S RULE
      F(Y) = SQRT(1.0+VSQ/(USQO+Y))
      QUADS = 0
      IF(IO.EQ.IF)  RETURN
      I1 = MINO(IO,IF)
      I2 = MAXO(IO,IF)
      I3 = I2-2
    8 J = (I2-I1)/10
      IF(J.EQ.0)  J=1
      DELXJ = FLOAT(J)*DELX/3.0
    9 DO 10 I=I1,I3,2
      IN1 = (I-I1)*J+I1
      IN2 = IN1+J
      IN3 = IN2+J
      IF(I.GT.I3)  GO TO 12
      IF(IN3.GT.I2)  GO TO 11
   10 QUADS = QUADS + DELXJ*(F(TPHIX(IN1))+4.0*F(TPHIX(IN2))+
     *F(TPHIX(IN3)))
      QUADS = ABS(QUADS)
      RETURN
   11 I1 = IN1
      GO TO 8
   12 QUADS = ABS(QUADS+DELX*(F(TPHIX(I2-1))+F(TPHIX(I2)))/2.0)
      RETURN
      END
```

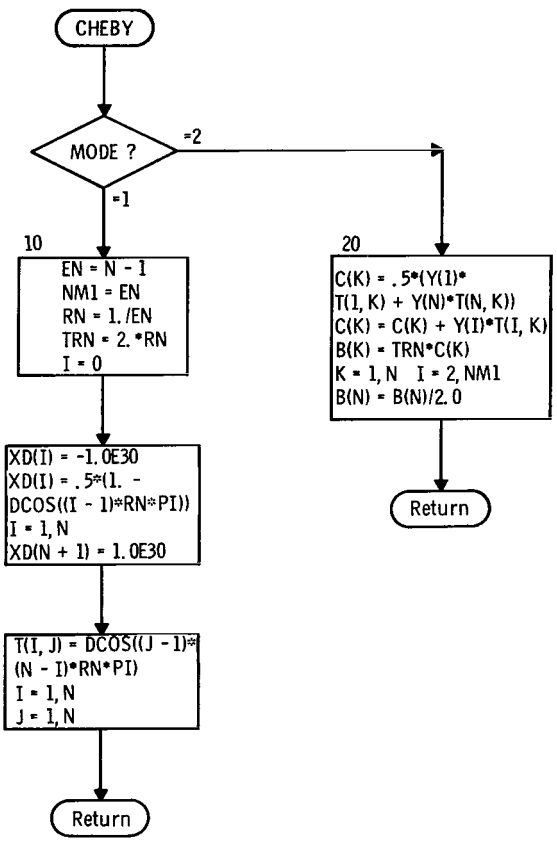


Figure 23. - Flow chart for surboutine CHEBY.

```
$IBFTC CHEBY

      SUBROUTINE CHEBY(MODE)
      COMMON /CCHEBY/ B(20),XD(20)
      COMMON /CMAIN/ NO,N,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,ZZZZZ,DISTB(64,64),LFLAG
      COMMON /CITER/ZZZ(20),Y(20),KNTR,VSO,FPATH,IBO,IBF,SIGMA,ID,CRRNT
      DIMENSION T(20,20),C(20)
      DOUBLE PRECISION PI,THETA
      DATA  PI/3.1415926535897932/
      GO TO (10,20),MODE
   10 EN=N-1
      NM1=EN
      RN=1./EN
      TRN = 2.*RN
      I = 0
      XD(I) = -1.0E+30
      DO 12 I=1,N
      A=I-1
      THETA=  A*RN*PI
   12 XD(I) = .5*(1.-DCOS(THETA))
      XD(N+1) = 1.0E+30
      DO 15 I=1,N
      DO 14 J=1,N
      R=J-1
      S=N-I
      THETA =  R*S*RN*PI
   14 T(I,J)= DCOS(THETA)
   15 CONTINUE
      RETURN
   20 CONTINUE
      DO 26 K=1,N
      C(K)=.5*(Y(1)*T(1,K) + Y(N)*T(N,K))
      DO 24 I=2,NM1
   24 C(K)= C(K)+ Y(I)* T(I,K)
   26 B(K) = TRN*C(K)
      B(N) = B(N)/2.0
      RETURN
      END
```

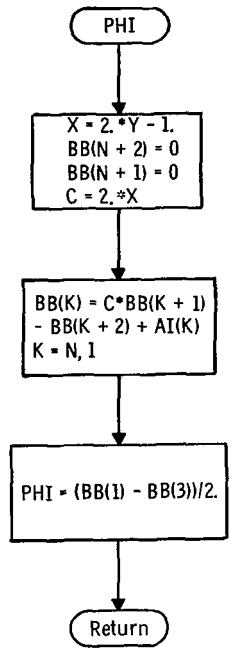


Figure 24. - Flow chart for subroutine PHI.

```
$IBFTC PHI

      FUNCTION PHI(Y)
      COMMON /CMAIN/ NO,N,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      DIMENSION BB(22)
      X=2.*Y-1.
      BB(N+2) = 0
      BB(N+1) = 0
      C=2.* X
      DO 10 J=1,N
      K = N+1-J
   10 BB(K) = C*BB(K+1)-BB(K+2) + AI(K)
      PHI = (BB(1)-BB(3))/2.0
      RETURN
      END
```

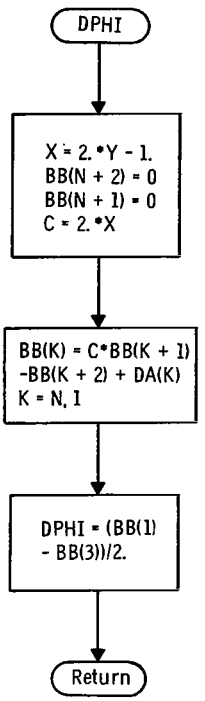


Figure 25. - Flow chart for subroutine DPHI.

```
$IBFTC DPHI

      FUNCTION DPHI(Y)
      COMMON /CCLN2S/ DA(20)
      COMMON /CMAIN/ NO,N,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      DIMENSION BB(22)
      X=2.*Y-1.
      BB(N+2) = 0
      BB(N+1) = 0
      C=2.* X
      DO 10 J=1,N
      K = N+1-J
   10 BB(K) = C*BB(K+1)-BB(K+2) + DA(K)
      DPHI = (BB(1)-BB(3))/2.0
      RETURN
      END
```

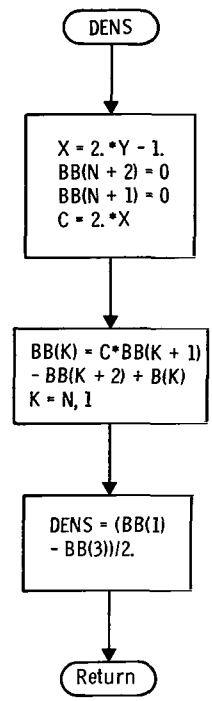


Figure 26. - Flow chart for subroutine DENS.

```
$IBFTC DENS

      FUNCTION DENS(Y)
      COMMON /CCHEBY/ B(20),XD(20)
      COMMON /CMAIN/ NO,N,ALPHA,CONST,NS,AI(20),VEL(1024),NK,MFP(126),
     * KFLAG,THETA,DISTB(64,64),LFLAG
      DIMENSION BB(22)
      X=2.*Y-1.
      BB(N+2) = 0
      BB(N+1) = 0
      C=2.* X
      DO 10 J=1,N
      K = N+1-J
   10 BB(K) = C*BB(K+1)-BB(K+2) + B(K)
      DENS = (BB(1)-BB(3))/2.0
      RETURN
      END
```

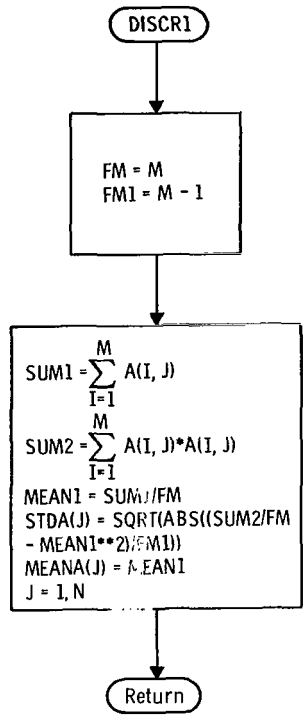


Figure 27. - Flow chart for subroutine DISCRI.

```
$IBFTC DISCR1

      SUBROUTINE DISCR1(M,N,A,MEANA,STDA)
      REAL MEANA
      DOUBLE PRECISION FM,MEAN1,SUM1,SUM2
      DIMENSION A(20,20),MEANA(20),STDA(20)
      FM = M
      FM1 = M-1
      DO 2 J=1,N
      SUM1 = 0.0D0
      SUM2 = 0.0D0
      DO 1 I=1,M
      SUM1 = SUM1 + A(I,J)
    1 SUM2 = SUM2 + A(I,J)*A(I,J)
      MEAN1 = SUM1/FM
      STDA(J) = SQRT(ABS((SUM2/FM-MEAN1*MEAN1)/FM1))
    2 MEANA(J) = MEAN1
      RETURN
      END
```

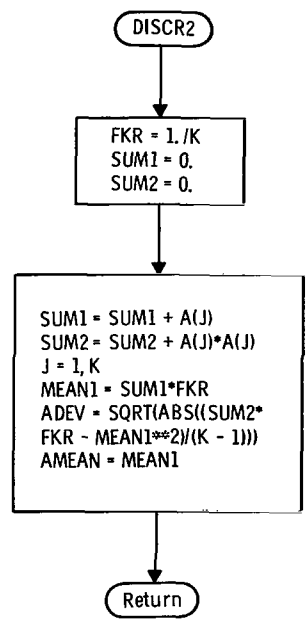


Figure 28. - Flow chart for subroutine DISCR2.

```
$IBFTC DISCR2

      SUBROUTINE DISCR2(K,A,AMEAN,ADEV)
      DIMENSION A(20)
      DOUBLE PRECISION FKR,MEAN1,SUM1,SUM2
      FKR = 1.0D0/DBLE(FLOAT(K))
      SUM1 = 0.
      SUM2 = 0.
      DO 13 J=1,K
      SUM1 = SUM1 + A(J)
   13 SUM2 = SUM2 + A(J) *A(J)
      MEAN1 =SUM1* FKR
      ADEV = SQRT(ABS((SUM2*FKR-MEAN1*MEAN1)/FLOAT(K-1)))
      AMEAN = MEAN1
      RETURN
      END
```

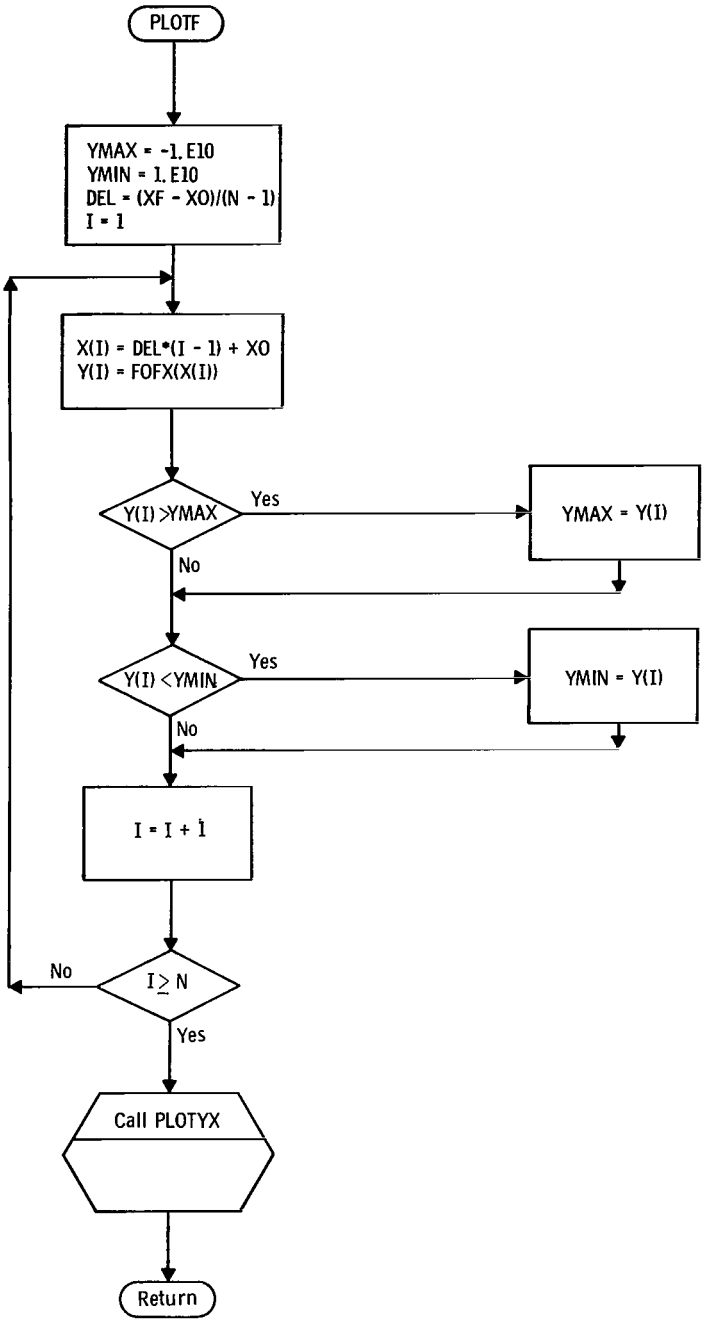


Figure 29. - Flow chart for subroutine PLOTF.

```
$IBFTC PLOTF

      SUBROUTINE PLOTF(N,XO,XF,FOFX)
C  SUBROUTINE FOR SINGLE-PAGE PLOTTING OF FUNCTION FOFX(X) FROM XO TO XF
C     USING N POINTS (N ODD.AND.LE.101)
C
C  CALLS SUBROUTINE PLOTYX
C
C
      DIMENSION X(101),Y(101)
      YMAX = -1.E10
      YMIN =  1.E10
      DEL=(XF-XO)/FLOAT(N-1)
      DO 10  I=1,N
      X(I) = DEL*FLOAT(I-1) + XO
      Y(I) = FOFX(X(I))
      IF(Y(I).GT.YMAX) YMAX = Y(I)
   10 IF(Y(I).LT.YMIN) YMIN = Y(I)
      CALL PLOTYX( N ,Y,X,YMAX,YMIN,XO,XF)
      RETURN
      END
```

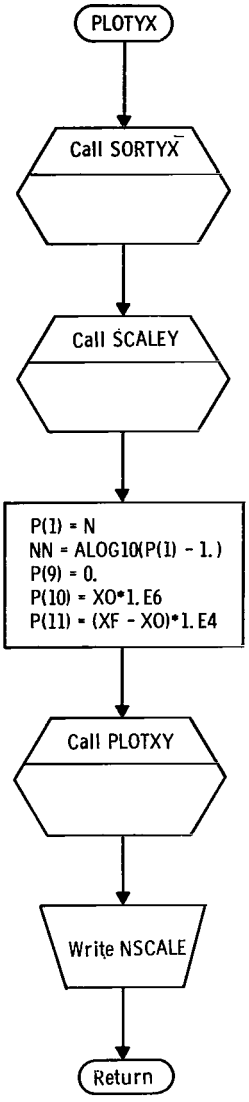


Figure 30. - Flow chart for subroutine PLOTYX.

```
$IBFTC PLOTYX

      SUBROUTINE PLOTYX(N,Y,X,YMAX,YMIN,XO,XF)
C
C  SUBROUTINE FOR N-POINT, SINGLE-PAGE PLOTTING OF ARRAYS X(N),Y(N)
C
C  N MUST BE ODD.AND.LE.101
C
C  CALLS SUBROUTINE SORTYX AND SCALEY
C
      DIMENSION Y(N),X(N),P(11)
      DATA P/11*0./
      CALL SORTYX(N,Y,X)
      CALL SCALEY(N,Y,YMAX,YMIN,P(6),P(7),P(8),NSCALE)
      P(1) = N
      NN=ALOG10(P(1)-1.)
      P(9) =0
      P(10) = 1.E6*XO
      P(11) = (XF-XO) * 1.E4
      CALL PLOTXY(Y,X,118,P)
      WRITE(6,10) NSCALE
   10 FORMAT(2HPL,20X, 35HSCALE FACTOR FOR ORDINATES IS 10**(,I3,1H))
      RETURN
      END
```

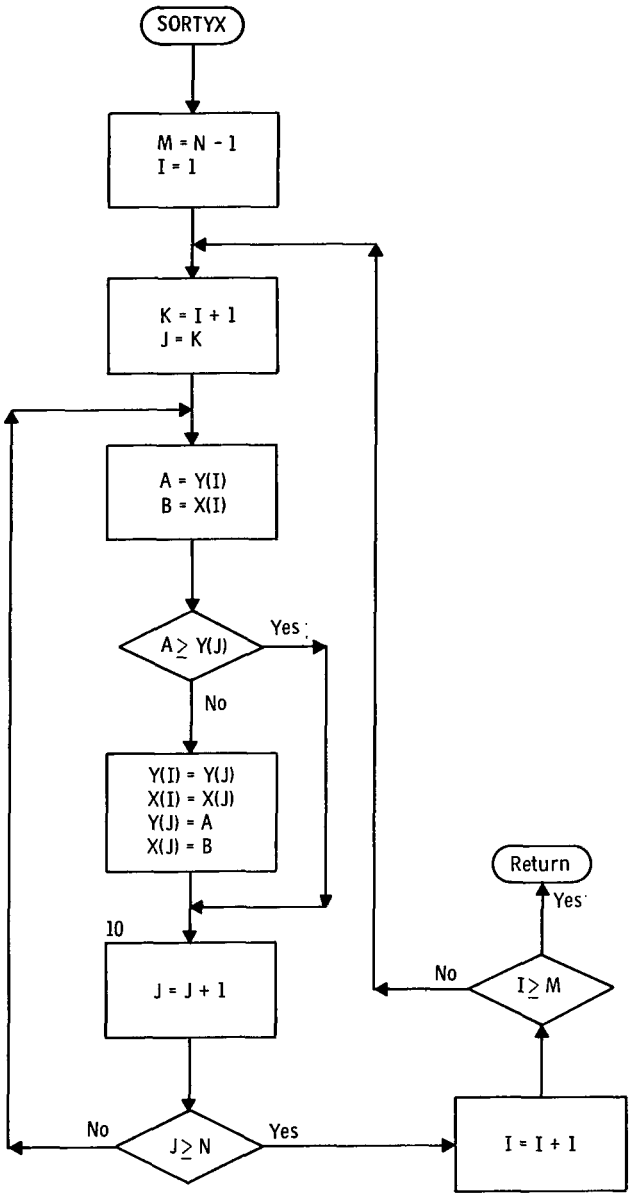


Figure 31. - Flow chart for subroutine SORTYX.

```
$IBFTC SORTYX

      SUBROUTINE SORTYX(N,Y,X)
      DIMENSION Y(N),X(N)
      M=N-1
      DO 10  I = 1,M
      K = I+1
      DO 10  J = K,N
      A = Y(I)
      B = X(I)
      IF(A.GE.Y(J)) GO TO 10
      Y(I) = Y(J)
      X(I) = X(J)
      Y(J) = A
      X(J) = B
   10 CONTINUE
      RETURN
      END
```

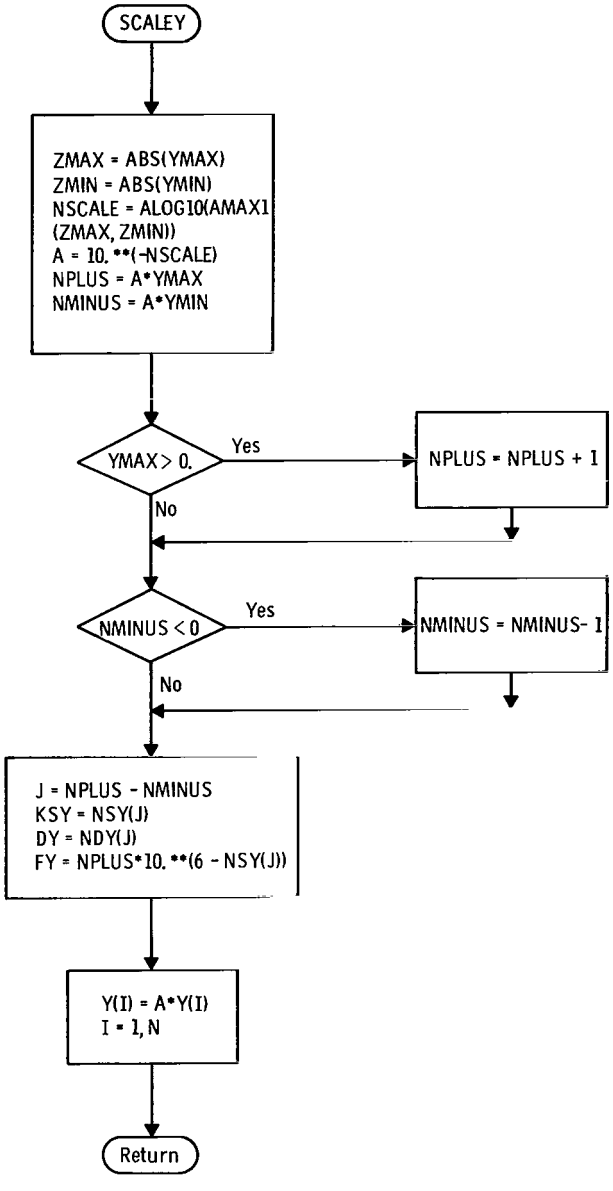


Figure 32. - Flow chart for subroutine SCALEY.

```
$IBFTC SCALEY

      SUBROUTINE SCALEY(N,Y,YMAX,YMIN,KSY,FY,DY,NSCALE)
      DIMENSION Y(N),NDY(20),NSY(20)
      DATA NDY/2,4,625,2*1,125,1429,3*2,3*25,4*3333,3*5/
      DATA NSY/2*4,2,2*5,3,2,3*5,3*4,4*2,3*5/
      REAL KSY
      ZMAX=ABS(YMAX)
      ZMIN=ABS(YMIN)
      NSCALE = ALOG10(AMAX1(ZMAX,ZMIN))
      A =10.**(-NSCALE)
      NPLUS  = YMAX * A
      NMINUS = YMIN * A
      IF(YMAX .GT.0.) NPLUS=NPLUS+1
      IF(NMINUS.LT.0 ) NMINUS=NMINUS-1
      J = NPLUS - NMINUS
      KSY = NSY(J)
      DY =-NDY(J)
      FY = NPLUS *10**(6-NSY(J))
      DO 10  I = 1,N
   10 Y(I) = A*Y(I)
      RETURN
      END
```

Auxiliary FORTRAN IV Program Descriptions

The programs given in this section are necessary to generate the binary formatted data decks discussed in the sections Tabulated Distributions and Preparation of Input Tables. For an example of input to and printed output from these programs, see appendix F. The ranges of the independent variables used in programs MFP, ARGON, and ARGINV were chosen specifically for the gas used in the sample problem. If the user wishes to change these ranges, for a different gas, he must also change the subroutines PATH and STOSS of ENEC accordingly. In other words, ENEC expects the tables to be given for specific ranges.

CVEL

Purpose:

To construct and punch on cards a table of the functional values of -ln R for 1024 equally spaced values of R over the range from 0 to 1

Method:

The table is punched on cards by subroutine BCDUMP.

Program called:

BCDUMP (appendix E)

FORTRAN listing:

```
$IBFTC CVEL

      DIMENSION VEL(1024)
      DELX = 1.0/1024.0
      DO 1 I=1,1024
      X = DELX*(FLOAT(I-1)+0.5)
    1 VEL(I) = -ALOG(X)
      CALL BCDUMP(VEL(1),VEL(1024))
      DO 2 L=1,4
      WRITE (6,201)
      DO 2 I=1,256,8
      I1 = I+(L-1)*256
      I2 = I1+7
      WRITE (6,202) I1,(VEL(J),J=I1,I2)
    2 CONTINUE
  201 FORMAT (1H1,///,10X,1HI,44X,3HVEL,//)
  202 FORMAT (1H ,5X,I5,8F11.4)
      STOP
      END
```

MFP

Purpose:

To construct and punch on cards a table of $\lambda(E) = 1.01/\sigma(E)$ for 100 values of E equally spaced over the range from 0 to 1 and 25 values equally spaced over the range from 1 to 13.5

Method:

The values of $\lambda(E)$ are obtained by curve fitting known values of $\sigma(E^{1/2})$ with a cubic spline (appendix B) and interpolating. The last value in the table is set to 0.012 and is used by ENEC for energies greater than 13.5 electron volts. The table is then punched on cards by subroutine BCDUMP.

Programs called:

BCDUMP (appendix E)

SPLINE and related subprograms (appendix B)

FORTRAN listing:

```
$IBFTC MFP

C     COMPUTES MFP TABLE FROM SPLINE FIT OF SIGMA S
      REAL MFP
      DIMENSION SQTEV(21),SIGMAS(21),MFP(126),E(125)
      READ (5,101) (SQTEV(I),SIGMAS(I),I=1,21)
      WRITE (6,201) (SQTEV(I),SIGMAS(I),I=1,21)
      CALL SPLINE(SQTEV,SIGMAS,21,0.0,0.0,2)
      DO 2 I=1,100
    2 E(I) = .005+.01*FLOAT(I-1)
      DO 3 I=101,125
    3 E(I) = 1.25+.5*FLOAT(I-101)
      DO 4 I=1,125
    4 MFP(I) = 1.01/F(SQRT(E(I)))
      MFP(126) = .012
      CALL BCDUMP(MFP(1),MFP(126))
      WRITE (6,202) (I,E(I),MFP(I),I=1,125)
      I = 126
      WRITE (6,203) I,MFP(126)
      STOP
  101 FORMAT (2E10.0)
  201 FORMAT (1H1,///,11X,8HSQRT(EV),10X,7HSIGMA-S,//,(F20.7,F15.2))
  202 FORMAT (1H1,///,20X,2HEV,13X,3HMFP,//,(I4,F20.3,F20.8))
  203 FORMAT (I4,20X,F20.8)
      END
```

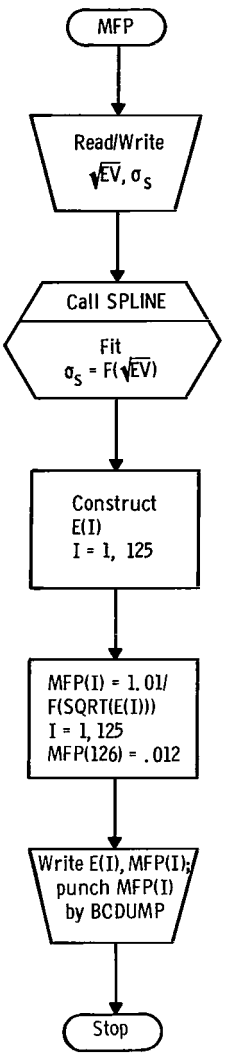


Figure 33. - Flow chart for subroutine MFP.

ARGON, G, SIMPS

Purpose:

To compute values on the surface defined by equation (11) for input to program ARGINV

Method:

These surface values are computed by surface fitting (appendix B) known values of the differential cross sections $\sigma(\theta, E)$ for the gas under consideration and performing Simpson's Rule integration.

Remarks:

ARGON uses two function subprograms G and SIMPS to perform the Simpson's Rule integration

Programs called:

SIMPS

G

SPLIN2 and related subprograms (appendix B)

FORTRAN listings:

```
$IBFTC ARGON

      DIMENSION X(20),Y(20),Z(20,20),X1(20),Z1(20,20),SIGS(20),S(20,20)
      COMMON /FUN/ YY
      EXTERNAL G
      READ (5,101) (X1(I),I=1,13)
      READ (5,102) (Y(J),(Z1(I,J),I=1,13),J=1,20)
      DO 1 I=1,13
      K = 14-I
      X(I) = COS(X1(K)/57.2957795)
      DO 1 J=1,20
    1 Z(I,J) = Z1(K,J)
      CALL SPLIN2(X,Y,Z,13,20)
      DO 2 J=1,20
      YY = Y(J)
    2 SIGS(J) = SIMPS(-1.0,1.0,G)
      DO 3 I=1,11
      X(I) = -1.0+0.2*FLOAT(I-1)
      DO 3 J=1,20
      YY = Y(J)
    3 S(I,J) = SIMPS(-1.0,X(I),G)/SIGS(J)
      WRITE (6,203) (X(I),I=1,11)
      DO 5 J=1,20
    5 WRITE (6,204) Y(J),(S(I,J),I=1,11),SIGS(J)
      STOP
  101 FORMAT (5X,13E5.0)
  102 FORMAT (14E5.0)
  203 FORMAT (1H1,///,60X,5HSIGMA,/,5X,13HEV/COS(THETA),11F8.1,2X,7HSIGM
     *A-S,//)
  204 FORMAT (3X,F5.2,10X,11F8.4,F8.3)
      END

$IBFTC G

      FUNCTION G(X)
      COMMON /FUN/ YY
      G = F(X,YY)
      RETURN
      END
```

```
$IBFTC SIMPS

      FUNCTION SIMPS(XMIN,XMAX,F)
      DELX = (XMAX-XMIN)/128.0
      SUM = 0.0
      DO 1 I=1,128,2
      X1 = XMIN+DELX*FLOAT(I-1)
      X2 = X1+DELX
      X3 = X2+DELX
    1 SUM = SUM+F(X1)+4.0*F(X2)+F(X3)
      SIMPS = DELX*SUM/3.0
      RETURN
      END
```

Flow charts:

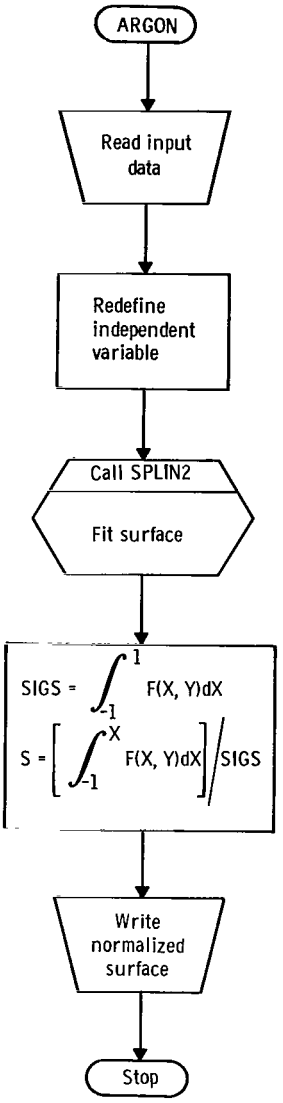


Figure 34. - Flow chart for subroutine ARGON.

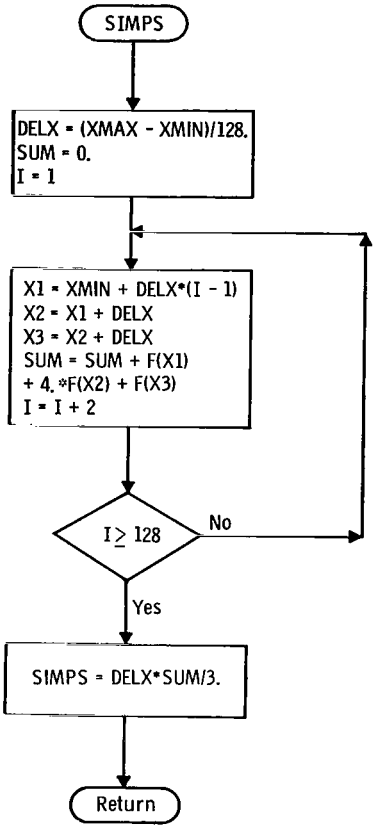


Figure 35. - Flow chart for subroutine SIMPS.

ARGINV

Purpose:

To solve equation (11) with R being taken at 64 equally spaced points over the range from 0 to 1 and 20 and 44 values of E equally spaced over the ranges 0 to 1.25 and 1.25 to 12.25, respectively, and to punch this data on cards

Method:

The equation is solved by Newton-Raphson iteration of functional values obtained from a two-dimensional surface fit (appendix B) of the output from ARGON. Convergence of the Newton-Raphson iteration is assumed when the difference between two consecutive iterations is less than 0.0001. The constructed table is punched on cards by BCDUMP.

Programs called:

BCDUMP (appendix E)

SPLIN2 and related subprograms (appendix B)

FORTRAN listing:

```
$IBFTC ARGINV

C       FINDS THE INVERSE FOR THE ARGON SURFACE
        DIMENSION X(20),Y(20),Z(20,20),R(64),E(64),ARG(64,64)
        READ (5,101) (X(I),I=1,11)
        READ (5,102) (Y(J),(Z(I,J),I=1,11),J=1,20)
        WRITE (6,201)
        CALL SPLIN2(X,Y,Z,11,20)
        DELR = 1.0/64.0
        R(1) = DELR/2.0
        DO 1 I=2,64
      1 R(I) = R(I-1)+DELR
        DELE = 1.25/20.0
        E(1) = DELE/2.0
        DO 2 J=2,20
      2 E(J) = E(J-1)+DELE
        DELE = (12.25-1.25)/44.0
        E(21) = 1.25+DELE/2.0
        DO 7 J=22,64
      7 E(J) = E(J-1)+DELE
        A = -0.8
        ERROR = 1.0E-4
        DO 4 I=1,64
        DO 4 J=1,64
        EE = E(J)
        DO 3 K=1,60
        DELA = (F(A,EE)-R(I))/FX(A,EE)
        IF(ABS(DELA).LT.ERROR)  GO TO 4
      5 IF(ABS(A-DELA).LT.1.0)  GO TO 6
        DELA = DELA/2.0
        GO TO 5
      6 A = A-DELA
      3 CONTINUE
        WRITE (6,209)
        WRITE (6,210) I,J,R(I),E(J),A,DELA
      4 ARG(I,J) = A
        CALL BCDUMP(ARG(1,1),ARG(64,64))
        WRITE (6,202)
        DO 8 I=1,64,8
        I1 = I
        I2 = I1+7
      8 WRITE (6,203) I1,(R(K),K=I1,I2)
        WRITE (6,204)
        DO 9 J=1,64,8
```

```
      J1 = J
      J2 = J1+7
    9 WRITE (6,205) J1,(E(K),K=J1,J2)
      DO 10 L=1,16
      WRITE (6,206)
      DO 10 M=1,4
      I = 4*(L-1)+M
      WRITE (6,207)
      DO 10 J=1,64,8
      J1 = J
      J2 = J1+7
      WRITE (6,208) I,J1,(ARG(I,K),K=J1,J2)
   10 CONTINUE
      STOP
  101 FORMAT (6X,11E6.0)
  102 FORMAT (12E6.0)
  201 FORMAT (1H1)
  202 FORMAT (1H1,///,5X,1HI,50X,1HR,//)
  203 FORMAT (1H ,I5,5X,8F11.7)
  204 FORMAT (1HL,10X,1HJ,45X,2HEV,//)
  205 FORMAT (1H ,5X,I5,8F11.7)
  206 FORMAT (1H1,//,5X,1HI,4X,1HJ)
  207 FORMAT (1HK)
  208 FORMAT (1H ,2I5,8F10.4)
  209 FORMAT (54HKTHE FOLLOWING DID NOT CONVERGE AFTER SIXTY ITERATIONS)
  210 FORMAT (3HKI=,I4,5X,2HJ=,I4,5X,2HR=,E17.8,5X,2HE=,E17.8,5X,2HA=,E1
     *7.8,5X,5HDELA=,E17.8)
      END
```

Flow chart:

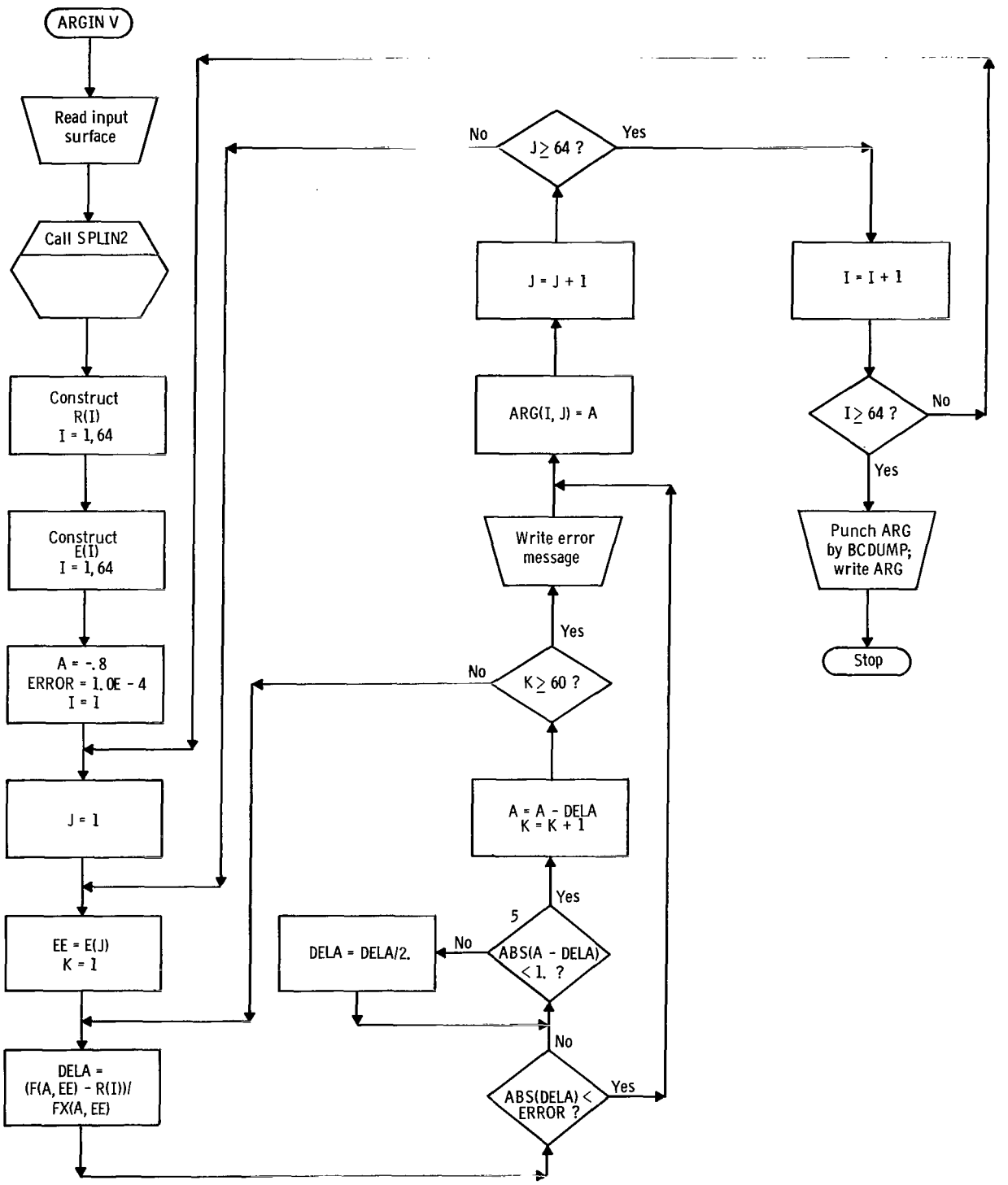


Figure 36. - Flow chart for subroutine ARGINV.

APPENDIX A

SYMBOLS

A	normalization factor
C	space charge parameter
E	electron kinetic energy
e	electron charge
$f(u,x)$	marginal probability function of $f(u,V,x)$
$f(u,V,x)$	probability function
$f(\hat{u},\hat{v},\hat{w})$	Maxwellian velocity distribution function
$G_X(t)$	cumulative distribution function, $P(X \leq t)$
$g(u,V)$	nondimensionalized flux probability function
$g(u,x)$	marginal probability function of $g(u,V,x)$
$g(u,V,x)$	probability function
J	current to collector
J_o	emission current
k	Boltzmann constant
L	electrode separation
l	path length along electron trajectory
l_c	path length along electron trajectory to collision
M	maximum value of dimensionless electron density distribution
m	electron mass
N_c	electron histories that reach collector
N_o	total electron histories for one iteration
$n(x)$	dimensionless electron density distribution, $\rho(\hat{x})/\rho(0)^+$
P_g	scattering gas pressure
P_o	reduced pressure, $P_g \frac{273}{T_g}$
$P(\theta \leq \Theta)$	probability that random variable θ is less than or equal to some Θ
R	random number uniformly distributed over range from 0 to 1

R_X	random number uniformly distributed over range from 0 to 1 associated with a random variable X
T	emitter temperature
T_g	temperature of scattering gas
u	dimensionless velocity component in x-direction, $\hat{u}(2kT/m)^{1/2}$
$\hat{u}, \hat{v}, \hat{w}$	velocity components
V	dimensionless velocity component transverse to u, $\left[(2kT/m)(\hat{v}^2 - \hat{w}^2)\right]^{1/2}$
x	dimensionless spatial variable normal to electrodes, $\hat{x}/L$
$\hat{x}$	spatial variable normal to electrodes
x_c	point of collision
x_k	Chebyshev abscissas for curve fit
x_o	location of last electron "event" (cell boundary or collision)
ϵ	permittivity of free space
θ	polar angle and polar angle in laboratory system after collision, (eq. (12))
θ_o	polar angle in laboratory system at point of (before) collision, (eq. (12))
θ'	polar angle in center-of-mass system after scattering
$\lambda(E)$	energy-dependent mean free path
$\lambda(x)$	mean free path
ν	potential distribution, (eq. (1))
$\nu(x)$	computed potential distribution
$\rho(\hat{x})$	electron density distribution
$\rho(0)^+$	electron density of emitted electrons
$\sigma(E)$	total scattering cross section
$\sigma(\theta, E)$	differential scattering cross section
φ	azimuthal angle of incidence
φ'	azimuthal angle in center-of-mass system after scattering, (eq. (12))
$\varphi(x)$	dimensionless potential distribution, $e\nu(x)/kT$

Superscript:

$(\bar{\ })$	average

APPENDIX B

SPLINE CURVE AND SURFACE FITS

It is the purpose of this appendix to describe the subroutines needed to perform one- and two-dimensional interpolatory spline curve fits. The subprograms presented herein are used by the programs that prepare the input tables for ENEC.

ONE-DIMENSIONAL SPLINE CURVE FIT

The specific functions used in the spline curve fit are piecewise polynomials of the third degree with matching first and second derivatives at the data points. These functions yield excellent approximations to the curve being fit as well as to the first derivatives of the curve.

Method

For a given set of n distinct data points x_i in increasing order ($x_i < x_{i+1}$), the corresponding functional values $f(x_i)$, and either $f'(x_1)$ and $f'(x_n)$ or $f''(x_1)$ and $f''(x_n)$, a cubic polynomial may be fit between each two data points subject to the following constraints:

$$\left.\begin{aligned} g_i'(x_{i+1}) &= g_{i+1}'(x_{i+1}) \\ g_i''(x_{i+1}) &= g_{i+1}''(x_{i+1}) \end{aligned}\right\} \qquad \text{(B1)}$$

where g_i denotes the cubic between the data points x_i and x_{i+1}, and the primes denote differentiation with respect to x. From the Hermite interpolation formula (ref. 9)

$$\left.\begin{aligned} g_i(x) &= h_i(x)f(x_i) + h_{i+1}(x)f(x_{i+1}) + \bar{h}_i(x)f'(x_i) + \bar{h}_{i+1}(x)f'(x_{i+1}) \\ x_i &\leq x \leq x_{i+1} \qquad i = 1, 2, \ldots, n \end{aligned}\right\} \qquad \text{(B2)}$$

and from the constraints given in equation (B1), the following formula is obtained:

$$(x_{i+1} - x_i)f'(x_{i-1}) + 2(x_{i+1} - x_{i-1})f'(x_i) + (x_i - x_{i-1})f'(x_{i+1}) = \frac{3}{(x_{i+1} - x_i)(x_i - x_{i-1})}$$

$$\times \left\{(x_i - x_{i-1})^2 f(x_{i+1}) + \left[(x_{i+1} - x_i)^2 - (x_i - x_{i-1})^2\right]\right.$$

$$\left.\times f(x_i) - (x_{i+1} - x_i)^2 f(x_{i-1})\right\} \qquad i = 2, 3, \ldots, n-1 \qquad \text{(B3)}$$

Given values V_1 and V_2 for $f'(x_1)$ and $f'(x_n)$, the following equations may be written:

$$\left.\begin{aligned} f'(x_1) &= V_1 \\ f'(x_n) &= V_2 \end{aligned}\right\} \qquad \text{(B4)}$$

If the values for $f''(x_1)$ and $f''(x_n)$ are given, equation (B2) may be used to obtain

$$\left.\begin{aligned} 2f'(x_1) + f'(x_2) &= \frac{3}{(x_2 - x_1)}\left[f(x_2) - f(x_1)\right] - \frac{1}{2}(x_2 - x_1)V_3 \\ f'(x_{n-1}) + 2f'(x_n) &= \frac{3}{(x_n - x_{n-1})}\left[f(x_n) - f(x_{n-1})\right] + \frac{1}{2}(x_n - x_{n-1})V_4 \end{aligned}\right\} \qquad \text{(B5)}$$

where V_3 and V_4 are the given values of $f''(x_1)$ and $f''(x_n)$, respectively.

The set of equations defined by equations (B3) and (B4), or the set defined by equations (B3) and (B5), form a system of n linear equations that may be solved for $f'(x_i)$. This system may be written in matrix notation where the matrix to be inverted in solving the system is tridiagonal

$$\begin{pmatrix} B_1 & C_1 & & & & & \\ A_2 & B_2 & C_2 & & & & \\ & A_3 & B_3 & C_3 & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & & A_{n-1} & B_{n-1} & C_{n-1} \\ & & & & & A_n & B_n \end{pmatrix} \qquad F' = D \qquad \text{(B6)}$$

where

$$F' = \left[f'(x_1)\ f'(x_2)\ \ldots\ f'(x_n)\right]$$

$$D = \left[D_1\ D_2\ \ldots\ D_n\right]$$

and

$$A_i = x_{i+1} - x_i$$

$$B_i = 2(x_{i+1} - x_{i-1})$$

$$C_i = x_i - x_{i-1}$$

$$D_i = \frac{3}{(x_{i+1} - x_i)(x_i - x_{i-1})}\left\{(x_i - x_{i-1})^2 f(x_{i+1}) + \left[(x_{i+1} - x_i)^2 - (x_i - x_{i-1})^2\right] f(x_i) - (x_{i+1} - x_i)^2 f(x_{i-1})\right\} \qquad i = 2,\ 3,\ \ldots,\ n-1$$

By specifying $f'(x_1) = V_1$ and $f'(x_n) = V_2$,

$$A_n = 0$$

$$B_1 = B_n = 1$$

$$C_1 = 0$$

$$D_1 = V_1$$

$$D_n = V_2$$

By specifying $f''(x_1) = V_3$ and $f''(x_n) = V_4$,

$$A_n = 1$$

$$B_1 = B_n = 2$$

$$C_1 = 1$$

$$D_1 = \frac{3}{(x_2 - x_1)}\left[f(x_2) - f(x_1)\right] - \frac{1}{2}(x_2 - x_1)V_3$$

$$D_n = \frac{3}{(x_n - x_{n-1})}\left[f(x_n) - f(x_{n-1})\right] + \frac{1}{2}(x_n - x_{n-1})V_4$$

This tridiagonal system may be solved by the following algorithm according to Peaceman and Rachford (ref. 10).

Let

$$W_1 = B_1$$

$$W_i = B_i - A_i b_{i-1} \qquad i = 2, 3, \ldots, n$$

$$b_i = \frac{C_i}{W_i} \qquad i = 1, 2, \ldots, n - 1$$

$$d_1 = \frac{D_1}{W_1}$$

$$d_i = \frac{D_i - A_i d_{i-1}}{W_i} \qquad i = 2, 3, \ldots, n$$

then

$$f'(x_n) = d_n$$

$$f'(x_i) = d_i - b_i f'(x_{i+1}) \qquad i = 1, 2, \ldots, n - 1$$

Now that all the $f'(x_i)$ have been determined, the desired set of cubic interpolation equations $g_i(x)$ may be obtained by returning to the Hermite interpolation formula equation (B2) and deriving equation (B7):

$$g_i(x) = (x - x_i)^3\left[\frac{2f(x_i) - 2f(x_{i+1}) + (x_{i+1} - x_i)f'(x_i) + (x_{i+1} - x_i)f'(x_{i+1})}{(x_{i+1} - x_i)^3}\right]$$

$$+ (x - x_1)^2\left[\frac{-3f(x_i) + 3f(x_{i+1}) - 2(x_{i+1} - x_i)f'(x_i) - (x_{i+1} - x_i)f'(x_{i+1})}{(x_{i+1} - x_i)^2}\right]$$

$$+ (x - x_i)f'(x_i) + f(x_i) \qquad x_i \leq x \leq x_{i+1} \tag{B7}$$

A set of second-order polynomial interpolation equations for $f'(x)$ may be obtained by simply differentiating equation (B7) with respect to x.

FORTRAN IV Subprograms for SPLINE Curve Fit

Subroutine SPLINE (XX, Y, NN, B1, BN, J):

XX Array of independent variable in increasing order

Y Array of functional values of curve to be fit; order must correspond to XX array

NN Number of values in XX array; NN $\leq$ 100

B1 Boundary condition at XX(1)

BN Boundary condition at XX(NN)

J Boundary conditions specified are first derivatives; J = 1

J Boundary conditions specified are second derivatives; J = 2

Purpose:

To construct and solve the tridiagonal system of equations (eqs. (B6)) and to compute the coefficients of the interpolation equations $g_i(x)$. These coefficients are stored in COMMON/SPLN/.

Labeled COMMON:

/SPLN/A(100), B(100), C(100), D(100), X(100), N

A, B, C, D Arrays of coefficients of zero, first, second, and third degree terms of interpolation equations (eq. (B7))

X Array of independent variable corresponding to user's XX array

N Number of values in X array corresponding to user's NN

FORTRAN listing:

```
$IBFTC SPLINE

      SUBROUTINE SPLINE(XX,Y,NN,B1,BN,J)
      DIMENSION XX(1),Y(1),DY(100)
      COMMON /SPLN/ A(100),B(100),C(100),D(100),X(100),N
      N = NN
      DO 7 I=1,N
    7 X(I) = XX(I)
      N1 = N-1
      GO TO (4,5),J
    4 B(1) = 1.0
      C(1) = 0.0
      D(1) = B1
      A(N) = 0.0
      B(N) = 1.0
      D(N) = BN
      GO TO 6
    5 B(1) = 2.0
      C(1) = 1.0
      D(1) = 3.0*(Y(2)-Y(1))/(X(2)-X(1))-0.5*(X(2)-X(1))*B1
      A(N) = 1.0
      B(N) = 2.0
      D(N) = 3.0*(Y(N)-Y(N1))/(X(N)-X(N1))+0.5*(X(N)-X(N1))*BN
    6 DO 1 I=2,N1
      A(I) = X(I+1)-X(I)
      B(I) = 2.0*(X(I+1)-X(I-1))
      C(I) = X(I)-X(I-1)
    1 D(I) = 3.0*(Y(I+1)*C(I)**2+Y(I)*(A(I)**2-C(I)**2)-Y(I-1)*A(I)**2)/
     *(C(I)*A(I))
      D(1) = D(1)/B(1)
      DO 2 I=2,N
      B(I) = B(I)-A(I)*C(I-1)/B(I-1)
    2 D(I) = (D(I)-A(I)*D(I-1))/B(I)
      DY(N) = D(N)
      DO 3 I=1,N1
      K = N-I
    3 DY(K) = D(K)-C(K)*DY(K+1)/B(K)
      A(1) = X(2)-X(1)
      DO 8 I=1,N1
      D(I) = (2.0*Y(I)-2.0*Y(I+1)+A(I)*DY(I)+A(I)*DY(I+1))/A(I)**3
      C(I) = (-3.0*Y(I)+3.0*Y(I+1)-2.0*A(I)*DY(I)-A(I)*DY(I+1))/A(I)**2
      B(I) = DY(I)
    8 A(I) = Y(I)
      RETURN
      END
```

FUNCTION F(X1):

X1 Independent variable $XX(1) \leq X1 \leq XX(NN)$

Purpose:

To apply the coefficients in COMMON/SPLN/ and to determine the interpolated value of the curve fit at X1 by using equation (B7).

Labeled COMMON:

/SPLN/

FORTRAN listing:

```
$IBFTC F

      FUNCTION F(X1)
      COMMON /SPLN/ A(100),B(100),C(100),D(100),X(100),N
      IF(X1.LT.X(1))  STOP
      DO 1 I=2,N
      J = I-1
    1 IF(X1.LE.X(I))  GO TO 2
      STOP
    2 Z = X1-X(J)
      F = A(J)+Z*(B(J)+Z*(C(J)+Z*D(J)))
      RETURN
      END
```

FUNCTION DF(X1):

X1 Independent variable XX(1) $\leq$ X1 $\leq$ XX(NN)

Purpose:

To apply the coefficients in COMMON/SPLN/ and to determine the interpolated value of the first derivative of the curve fit at X1 by using the first derivative of equation (B7).

Labeled COMMON:

/SPLN/

FORTRAN listing:

```
$IBFTC DF

      FUNCTION DF(X1)
      COMMON /SPLN/ A(100),B(100),C(100),D(100),X(100),N
      IF(X1.LT.X(1))  STOP
      DO 1 I=2,N
      J = I-1
    1 IF(X1.LE.X(I))  GO TO 2
      STOP
    2 Z = X1-X(J)
      DF = B(J)+Z*(2.0*C(J)+3.0*7*D(J))
      RETURN
      END
```

Program use:

The user must call subroutine SPLINE, with the proper arguments, only once for the curve to be fit. After this call, the user has available to him labeled COMMON /SPLN/ and may use both functions F and DF. The user must be sure that F and DF are only used with an argument in the range of the independent variable because of stops built into the functions. This curve fit should not be used for extrapolation.

TWO-DIMENSIONAL SPLINE SURFACE FIT

The two-dimensional spline is completely analogous to the one-dimensional case in that the surface to be fit, defined on a rectangular grid with boundary conditions along the

edges, is redefined in terms of one-dimensional spline curve fits along the grid lines parallel to the coordinate axes. Two-dimensional interpolation is achieved by multiple one-dimensional interpolations of the function and its derivative.

Method

The method of solving the tridiagonal system of equations and obtaining the interpolation equations is identical to the one-dimensional case and will not be repeated; however, the boundary conditions that must be supplied need clarification. The surface to be fit must be defined by a set of $n \times m$ functional values on a $n \times m$ rectangular grid. Also, boundary conditions consisting of first or second partials (of the function being fit) must be supplied for the four boundaries. For example, if $f(x,y)$ is the surface to be fit, it must be defined by functional values $f(x_i, y_j)$ on the rectangular grid $x_i \times y_j$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$, and by one of the following sets of boundary conditions:

$$\left(\frac{\partial f}{\partial x}\right)_{\substack{x=x_1 \\ y=y_j}} \quad \left(\frac{\partial f}{\partial x}\right)_{\substack{x=x_n \\ y=y_j}} \quad \left(\frac{\partial f}{\partial y}\right)_{\substack{x=x_i \\ y=y_1}} \quad \left(\frac{\partial f}{\partial y}\right)_{\substack{x=x_i \\ y=y_m}} \tag{B8}$$

$$\left(\frac{\partial f}{\partial x}\right)_{\substack{x=x_1 \\ y=y_j}} \quad \left(\frac{\partial f}{\partial x}\right)_{\substack{x=x_n \\ y=y_j}} \quad \left(\frac{\partial^2 f}{\partial y^2}\right)_{\substack{x=x_i \\ y=y_1}} \quad \left(\frac{\partial^2 f}{\partial y^2}\right)_{\substack{x=x_i \\ y=y_m}} \tag{B9}$$

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_{\substack{x=x_1 \\ y=y_j}} \quad \left(\frac{\partial^2 f}{\partial x^2}\right)_{\substack{x=x_n \\ y=y_j}} \quad \left(\frac{\partial f}{\partial y}\right)_{\substack{x=x_i \\ y=y_1}} \quad \left(\frac{\partial f}{\partial y}\right)_{\substack{x=x_i \\ y=y_m}} \tag{B10}$$

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_{\substack{x=x_1 \\ y=y_j}} \quad \left(\frac{\partial^2 f}{\partial x^2}\right)_{\substack{x=x_n \\ y=y_j}} \quad \left(\frac{\partial^2 f}{\partial y^2}\right)_{\substack{x=x_i \\ y=y_1}} \quad \left(\frac{\partial^2 f}{\partial y^2}\right)_{\substack{x=x_i \\ y=y_m}} \tag{B11}$$

$$i = 1, 2, \ldots, n$$

$$j = 1, 2, \ldots, m$$

FORTRAN IV Subprograms for SPLINE Surface Fit

Subroutine SPLIN2(X, Y, Z, N, M):

X Array of independent variable x_i in increasing order

Y Array of independent variable y_j in increasing order

Z Two-dimensional array of dependent variable $f(x_i, y_j)$ corresponding to X and Y arrays

This array must be dimensioned (20, 20) in the calling program.

N Number of values in X array; $N \leq 20$

M Number of values in Y array; $M \leq 20$

Purpose:

To set up arrays for calling subroutine SPLIN1 and to store results from SPLIN1 for later use in the interpolation function subprograms.

Program called:

SPLIN1

Labeled COMMON:

/BNDRY/BX1(20), BXN(20), JX, BY1(20), BYM(20), JY

This labeled COMMON allows the user to specify one of the four sets of boundary conditions given by equations (B8) to (B11):

$$\text{BX1(J)} = \begin{cases} \left(\dfrac{\partial f}{\partial x}\right)_{\substack{x=x_1 \\ y=y_j}} & \text{JX} = 1 \\ \left(\dfrac{\partial^2 f}{\partial x^2}\right)_{\substack{x=x_1 \\ y=y_j}} & \text{JX} = 2 \end{cases} \qquad \begin{matrix} \text{J} = 1,\ \text{M} \\ j = 1, 2, \ldots, m \end{matrix}$$

$$\text{BXN(J)} = \begin{cases} \left(\dfrac{\partial f}{\partial x}\right)_{\substack{x=x_n \\ y=y_j}} & \text{JX} = 1 \\ \left(\dfrac{\partial^2 f}{\partial x^2}\right)_{\substack{x=x_n \\ y=y_j}} & \text{JX} = 2 \end{cases} \qquad \begin{matrix} \text{J} = 1,\ \text{M} \\ j = 1, 2, \ldots, m \end{matrix}$$

$$\text{BY1(I)} = \begin{cases} \left(\dfrac{\partial f}{\partial y}\right)_{\substack{x=x_i \\ y=y_1}} & \text{JY} = 1 \\ & \text{I} = 1, \text{N} \\ & \text{i} = 1, 2, \ldots, n \\ \left(\dfrac{\partial^2 f}{\partial y^2}\right)_{\substack{x=x_i \\ y=y_1}} & \text{JY} = 2 \end{cases}$$

$$\text{BYM(I)} = \begin{cases} \left(\dfrac{\partial f}{\partial y}\right)_{\substack{x=x_i \\ y=y_m}} & \text{JY} = 1 \\ & \text{I} = 1, \text{N} \\ & \text{i} = 1, 2, \ldots, n \\ \left(\dfrac{\partial^2 f}{\partial y^2}\right)_{\substack{x=x_i \\ y=y_m}} & \text{JY} = 2 \end{cases}$$

If the user does not specify any boundary conditions, the subroutines assume that the second particles are zero; that is, the arrays in labeled COMMON/BNDRY/ are initialized to zero, and JX and JY are initialized to 2 by the block data subprogram.

/SPLIN/N1, M1, X1(20), Y1(20), Z1(20,20), ZX(20,20), ZY(20,20), ZYX(20,20)

This labeled COMMON is used to transmit data from subroutine SPLIN2 to the function subprograms.

FORTRAN listings:

```
$IBFTC SPLIN2

      SUBROUTINE SPLIN2(X,Y,Z,N,M)
      DIMENSION X(20),Y(20),Z(20,20),ZI(20),ZP(20)
      COMMON /SPLIN/ N1,M1,X1(20),Y1(20),Z1(20,20),ZX(20,20),ZY(20,20),
     * ZYX(20,20)
      COMMON /BNDRY/ BX1(20),BXN(20),JX,BY1(20),BYM(20),JY
      N1 = N
      M1 = M
      DO 10 I=1,N1
   10 X1(I) = X(I)
      DO 11 J=1,M1
   11 Y1(J) = Y(J)
      DO 12 I=1,N1
      DO 12 J=1,M1
   12 Z1(I,J) = Z(I,J)
      DO 3 I=1,N1
      DO 1 J=1,M1
    1 ZI(J) = Z1(I,J)
      CALL SPLIN1(Y1,ZI,M1,ZP,BY1(I),BYM(I),JY)
      DO 2 J=1,M1
    2 ZY(I,J) = ZP(J)
```

```
      3 CONTINUE
        DO 6 J=1,M1
        DO 4 I=1,N1
      4 ZI(I) = Z1(I,J)
        CALL SPLIN1(X1,ZI,N1,ZP,BX1(J),BXN(J),JX)
        DO 5 I=1,N1
      5 ZX(I,J) = ZP(I)
      6 CONTINUE
        DO 9 J=1,M1
        DO 7 I=1,N1
      7 ZI(I) = ZY(I,J)
        CALL SPLIN1(X1,ZI,N1,ZP,0.0,0.0,2)
        DO 8 I=1,N1
      8 ZYX(I,J) = ZP(I)
      9 CONTINUE
        RETURN
        END
```

```
$IBFTC BDATA

      BLOCK DATA
      COMMON /BNDRY/ BX1(20),BXN(20),JX,BY1(20),BYM(20),JY
      DATA BX1,BXN,BY1,BYM,JX,JY/80*0.0,2*2/
      END
```

Subroutine SPLIN1:

Purpose:

This subroutine is called by SPLIN2 and is used to fit single spline curves through the data points on the grid lines.

Calling program:

SPLIN2

Labeled COMMON:

None

FORTRAN listing:

```
$IBFTC SPLIN1

      SUBROUTINE SPLIN1(X,Y,N,DY,B1,BN,J)
      DIMENSION X(1),Y(1),DY(1),A(20),B(20),C(20),D(20)
      N1 = N-1
      GO TO (4,5),J
    4 B(1) = 1.0
      C(1) = 0.0
      D(1) = B1
      A(N) = 0.0
      B(N) = 1.0
      D(N) = BN
      GO TO 6
    5 B(1) = 2.0
      C(1) = 1.0
      D(1) = 3.0*(Y(2)-Y(1))/(X(2)-X(1))-0.5*(X(2)-X(1))*B1
      A(N) = 1.0
      B(N) = 2.0
      D(N) = 3.0*(Y(N)-Y(N1))/(X(N)-X(N1))+0.5*(X(N)-X(N1))*BN
```

```
6 DO 1 I=2,N1
  A(I) = X(I+1)-X(I)
  B(I) = 2.0*(X(I+1)-X(I-1))
  C(I) = X(I)-X(I-1)
1 D(I) = 3.0*(Y(I+1)*C(I)**2+Y(I)*(A(I)**2-C(I)**2)-Y(I-1)*A(I)**2)/
 *(C(I)*A(I))
  D(1) = D(1)/B(1)
  DO 2 I=2,N
  B(I) = B(I)-A(I)*C(I-1)/B(I-1)
2 D(I) = (D(I)-A(I)*D(I-1))/B(I)
  DY(N) = D(N)
  DO 3 I=1,N1
  K = N-I
3 DY(K) = D(K)-C(K)*DY(K+1)/B(K)
  RETURN
  END
```

FUNCTION F(X1, Y1), FUNCTION FX(X1, Y1), FUNCTION FY(X1, Y1), and

FUNCTION FXY(X1, Y1):

X1 The independent variable x. $X(1) \leq X1 \leq X(N)$

Y1 The independent variable y. $Y(1) \leq Y1 \leq Y(M)$

Purpose:

These four functions give interpolated values for the surface f(x, y) and the following partial derivatives $\partial f/\partial x$, $\partial f/\partial y$, and $\partial^2 f/\partial x \partial y$, respectively.

Labeled COMMON:

/SPLIN/

FORTRAN listings:

```
$IBFTC F

      FUNCTION F(X1,Y1)
      COMMON /SPLIN/ N,M,X(20),Y(20),Z(20,20),ZX(20,20),ZY(20,20),
     * ZYX(20,20)
      G(FI,FI1,DFI,DFI1,DEL,S) = (2.0*(FI-FI1)+DEL*(DFI+DFI1))*(S/DEL)**
     * 3 + (3.0*(FI1-FI)-DEL*(2.0*DFI+DFI1))*(S/DEL)**2 + DFI*S + FI
      IF(X1.LT.X(1))  STOP
      DO 1 I=2,N
      K = I-1
      IF(X1.LE.X(I))  GO TO 2
    1 CONTINUE
      STOP
    2 IF(Y1.LT.Y(1))  STOP
      DO 3 J=2,M
      L = J-1
      IF(Y1.LE.Y(J))  GO TO 4
    3 CONTINUE
      STOP
```

```
    4 DEL = X(K+1)-X(K)
      S = X1-X(K)
      FXJ = G(Z(K,L),Z(K+1,L),ZX(K,L),ZX(K+1,L),DEL,S)
      FXJ1 = G(Z(K,L+1),Z(K+1,L+1),ZX(K,L+1),ZX(K+1,L+1),DEL,S)
      FYXJ = G(ZY(K,L),ZY(K+1,L),ZYX(K,L),ZYX(K+1,L),DEL,S)
      FYXJ1 = G(ZY(K,L+1),ZY(K+1,L+1),ZYX(K,L+1),ZYX(K+1,L+1),DEL,S)
      DEL = Y(L+1)-Y(L)
      S = Y1-Y(L)
      F = G(FXJ,FXJ1,FYXJ,FYXJ1,DEL,S)
      RETURN
      END

$IBFTC FX

      FUNCTION FX(X1,Y1)
      COMMON /SPLIN/ N,M,X(20),Y(20),Z(20,20),ZX(20,20),ZY(20,20),
     * ZYX(20,20)
      G(FI,FI1,DFI,DFI1,DEL,S) = (2.0*(FI-FI1)+DEL*(DFI+DFI1))*(S/DEL)**
     * 3 + (3.0*(FI1-FI)-DEL*(2.0*DFI+DFI1))*(S/DEL)**2 + DFI*S + FI
      DG(FI,FI1,DFI,DFI1,DEL,S) = (6.0*(FI-FI1)+3.0*DEL*(DFI+DFI1))*
     * (S**2/DEL**3)+(6.0*(FI1-FI)-2.0*DEL*(2.0*DFI+DFI1))*(S/DEL**2)+
     * DFI
      IF(X1.LT.X(1))  STOP
      DO 1 I=2,N
      K = I-1
      IF(X1.LE.X(I))  GO TO 2
    1 CONTINUE
      STOP
    2 IF(Y1.LT.Y(1))  STOP
      DO 3 J=2,M
      L = J-1
      IF(Y1.LE.Y(J))  GO TO 4
    3 CONTINUE
      STOP
    4 DEL = X(K+1)-X(K)
      S = X1-X(K)
      FXJ =DG(Z(K,L),Z(K+1,L),ZX(K,L),ZX(K+1,L),DEL,S)
      FXJ1 =DG(Z(K,L+1),Z(K+1,L+1),ZX(K,L+1),ZX(K+1,L+1),DEL,S)
      FYXJ =DG(ZY(K,L),ZY(K+1,L),ZYX(K,L),ZYX(K+1,L),DEL,S)
      FYXJ1 =DG(ZY(K,L+1),ZY(K+1,L+1),ZYX(K,L+1),ZYX(K+1,L+1),DEL,S)
      DEL = Y(L+1)-Y(L)
      S = Y1-Y(L)
      FX =G(FXJ,FXJ1,FYXJ,FYXJ1,DEL,S)
      RETURN
      END

$IBFTC FY

      FUNCTION FY(X1,Y1)
      COMMON /SPLIN/ N,M,X(20),Y(20),Z(20,20),ZX(20,20),ZY(20,20),
     * ZYX(20,20)
      G(FI,FI1,DFI,DFI1,DEL,S) = (2.0*(FI-FI1)+DEL*(DFI+DFI1))*(S/DEL)**
     * 3 + (3.0*(FI1-FI)-DEL*(2.0*DFI+DFI1))*(S/DEL)**2 + DFI*S + FI
      DG(FI,FI1,DFI,DFI1,DEL,S) = (6.0*(FI-FI1)+3.0*DEL*(DFI+DFI1))*
     * (S**2/DEL**3)+(6.0*(FI1-FI)-2.0*DEL*(2.0*DFI+DFI1))*(S/DEL**2)+
     * DFI
      IF(X1.LT.X(1))  STOP
      DO 1 I=2,N
```

```
      K = I-1
      IF(X1.LE.X(I))  GO TO 2
    1 CONTINUE
      STOP
    2 IF(Y1.LT.Y(1))  STOP
      DO 3 J=2,M
      L = J-1
      IF(Y1.LE.Y(J))  GO TO 4
    3 CONTINUE
      STOP
    4 DEL = X(K+1)-X(K)
      S = X1-X(K)
      FXJ = G(Z(K,L),Z(K+1,L),ZX(K,L),ZX(K+1,L),DEL,S)
      FXJ1 = G(Z(K,L+1),Z(K+1,L+1),ZX(K,L+1),ZX(K+1,L+1),DEL,S)
      FYXJ = G(ZY(K,L),ZY(K+1,L),ZYX(K,L),ZYX(K+1,L),DEL,S)
      FYXJ1 = G(ZY(K,L+1),ZY(K+1,L+1),ZYX(K,L+1),ZYX(K+1,L+1),DEL,S)
      DEL = Y(L+1)-Y(L)
      S = Y1-Y(L)
      FY =DG(FXJ,FXJ1,FYXJ,FYXJ1,DEL,S)
      RETURN
      END

$IBFTC FXY

      FUNCTION FXY(X1,Y1)
      COMMON /SPLIN/ N,M,X(20),Y(20),Z(20,20),ZX(20,20),ZY(20,20),
     * ZYX(20,20)
      DG(FI,FI1,DFI,DFI1,DEL,S) = (6.0*(FI-FI1)+3.0*DEL*(DFI+DFI1))*
     * (S**2/DEL**3)+(6.0*(FI1-FI)-2.0*DEL*(2.0*DFI+DFI1))*(S/DEL**2)+
     * DFI
      IF(X1.LT.X(1))  STOP
      DO 1 I=2,N
      K = I-1
      IF(X1.LE.X(I))  GO TO 2
    1 CONTINUE
      STOP
    2 IF(Y1.LT.Y(1))  STOP
      DO 3 J=2,M
      L = J-1
      IF(Y1.LE.Y(J))  GO TO 4
    3 CONTINUE
      STOP
    4 DEL = X(K+1)-X(K)
      S = X1-X(K)
      FXJ =DG(Z(K,L),Z(K+1,L),ZX(K,L),ZX(K+1,L),DEL,S)
      FXJ1 =DG(Z(K,L+1),Z(K+1,L+1),ZX(K,L+1),ZX(K+1,L+1),DEL,S)
      FYXJ =DG(ZY(K,L),ZY(K+1,L),ZYX(K,L),ZYX(K+1,L),DEL,S)
      FYXJ1 =DG(ZY(K,L+1),ZY(K+1,L+1),ZYX(K,L+1),ZYX(K+1,L+1),DEL,S)
      DEL = Y(L+1)-Y(L)
      S = Y1-Y(L)
      FXY = DG(FXJ,FXJ1,FYXJ,FYXJ1,DEL,S)
      RETURN
      END
```

Program use:

The user must call subroutine SPLIN2 with the proper arguments only once for the surface to be fit. After this call the user may use any or all of the function subprograms. The user must make sure that the arguments used in the functions are within the ranges of the original rectangular grid. These programs should not be used for extrapolation.

APPENDIX C

CONVERGENCE EXPERIMENT

Problem Description

Given the equation

$$y''(x) = f(x,y) \tag{C1}$$

it is desired to investigate the nature of the convergence by Picard iteration to the solution $y^*(x)$ when $f(x,y)$ is, for a given x, a normally distributed random variable. In particular, it is desired to determine the effect of the standard deviation of $f(x,y)$, at a given x, on the convergence.

A general analytical analysis has not been attempted. Instead a particular expression has been chosen for $f(x,y)$. Solutions of equation (C1) have been obtained numerically by the Clenshaw-Norton method (ref. 2) for different standard deviations, and the results compared.

Numerical Experiment

Let $f(x,y) = 4y^2 + \delta(x)$, where $\delta(x)$ is, for a given x, a normally distributed random variable with zero mean and standard deviation σ_δ. Hence, $f(x,y)$ is, for a given x, a normally distributed random variable with mean $4y^2$ and standard deviation σ_δ. The equation

$$y''(x) = 4y^2 + \delta(x) \qquad 0 \leq x \leq 1 \tag{C2}$$

with the boundary conditions $y(0) = 0$ and $y(1) = 1$ was numerically solved for the following values of the standard deviation σ_δ: 0, 0.05, 0.1, 0.2, 0.5. Note that the Clenshaw-Norton method evaluates the right side of equation (C2) only at the N arguments of an $N - 1$ degree Chebyshev curve fit

$$4y^2 + \delta(x) = \sum_{k=0}^{N-1} a_k T_k(x) \tag{C3}$$

where $T_k(x)$ are the Chebyshev ploynomials of degree k. For this experiment, $N = 17$.

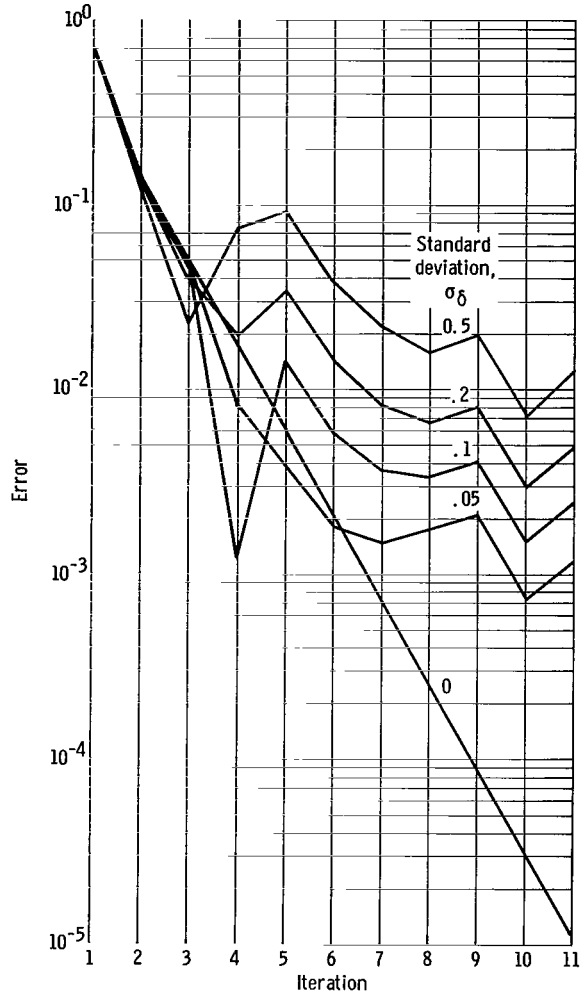


Figure 37. - Effect of σ_δ on the relative rate of convergence.

Results

The relative rate of convergence is shown in figure 37 for the various standard deviations. The curve labeled $\sigma_\delta = 0$ is, of course, the exact solution. The error in figure 37 is defined as the maximum difference in two successive iterations of the corresponding coefficients a_k (eq. (C3)). For $\sigma_\delta > 0$, a general tendency to oscillate about a mean error is noted as the number of iterations increase; the magnitude of this mean error is proportional to σ_δ, as might be expected.

The results of figure 37, while interesting, are not of major significance in relation to the problem encountered in this report. Of greater interest is the relative error of the solution. Since equation (C2) is solved as a boundary value problem, the relative error was investigated at $x = 0.5$. The results are given in table IV.

TABLE IV. - EFFECTS OF STANDARD DEVIATION

(a) On relative error during convergence

Iteration	Standard deviation, σ_δ				
	0	0.5	0.2	0.1	0.05
	Relative error at x = 0.5, (y(x) - y* (x))/y* (x)				
1	-0.685	-0.685	-0.685	-0.685	-0.685
2	.235	.189	.217	.226	.231
3	-.098	-.098	-.098	-.098	-.098
4	.033	-.042	.003	.018	.026
5	-.012	.154	.054	.021	.005
6	.004	-.094	-.034	-.015	-.005
7	-.001	.012	.004	.001	-.000
8	.000	-.034	-.013	-.006	-.003
9	-.000	-.093	-.037	-.019	-.009
10	.000	-.052	-.021	-.010	-.005
11	-.000	-.042	-.017	-.008	-.004

(b) On standard deviation σ_y

Iterations used in computing σ_y	Standard deviation, σ_δ				
	0	0.5	0.2	0.1	0.05
	Standard deviation of y(x) at x = 0.5, σ_y				
2 to 11	0.010	0.013	0.010	0.010	0.010
4 to 11	.002	.011	.004	.002	.002

Discussion of Results

The effect of increasing the standard deviation σ_δ is shown in table IV. The relative error tabulated in table IV(a) is taken with respect to the "true" value of y(0.5) (i.e., with $\sigma(x) = 0$ in eq. (C2)). Of greatest interest is the standard deviation, σ_y, of y(0.5) about the true value, as shown in table IV(b). There, σ_y is given, calculated from two choices of sample iterations; iterations 2 to 11 and iterations 4 to 11. It is observed that when the last 10 iterations are employed in the calculation of σ_y, σ_y is insensitive to σ_δ. This observation tends to corroborate the method employed in this report of averaging successive iterations.

Averaging also accomplishes another economy. The evaluations of f(x,y), with the associated random errors, at the Chebyshev arguments x_i can be likened to experimentally obtained data at this same argument. In the case of experimental data, the usual procedure would be to use a least-squares fit. This could be accomplished in the present

program by simply truncating the expansion of equation (C3) at some $M < N$. A test was made by setting $M = 5$. While the resulting fit was smoother (after each iteration) the results, as presented in table IV, were not affected.

Conclusions

This numerical experiment on the convergence of equation (C2) is not necessarily expected to apply to all second-order differential stochastic equations. In fact, there is no assurance that this experiment accurately represents the problem in ENEC where

$$f(x,y) = C \cdot n(x) \tag{C4}$$

This experiment was primarily intended to give some insight into the problem. With this in mind it must be noted that, by the Central Limit Theorem (appendix A of ref. 1), $n(x) + \delta(x)$ approaches $n(x)$ as the number of electron histories becomes very large.

APPENDIX D

IMPROVED SQUARE ROOT ROUTINE

Standard computer library subroutines for the calculation of square roots have accuracy greater than is needed for Monte Carlo work. Sacrificing some of this accuracy for a faster square root subroutine has proved to be helpful in increasing the efficiency of Monte Carlo codes.

Three modifications were made to the IBM FORTRAN IV version 13 library square root subroutine to effect the increase in speed:

(1) The number of Newton-Raphson iterations, applied to the starting approximation, was reduced from three to two or one.

(2) The indexing on the iteration loop is removed, and the iteration equation was written twice or once.

(3) The starting approximation was changed.

The last modification was necessary to maintain good accuracy with the reduced number of iterations. The starting approximation used is given as equation (11'') in reference 11. The modifications added several words to the subroutine, but it was felt that the increase in speed more than compensated for this.

Table V lists results from tests run on square root subroutines with various combinations of the three modifications. Each test consisted of 100 000 arguments exponentially distributed over the range from 0 to 10^{38} and was conducted on an IBM 7094 computer.

Test 1 is the library square root. Test 2 is the starting approximation used in the library square root with two iterations and no indexing. Test 3 is the same as test 2 but

TABLE V. - SQUARE ROOT SUBROUTINE TEST RESULTS

[F and G represent true and test square roots, respectively.]

Test	Ratio of library to test square root time	$\left(\frac{1}{N}\sum_{i=1}^{N}\left\|\frac{F-G}{F}\right\|^2\right)^{1/2}$	$\text{MAX}\left\|\frac{F-G}{F}\right\|$
1	1.0	3.1×10^{-9}	7.25×10^{-9}
2	1.65	3.65×10^{-7}	1.5×10^{-6}
3	2.41	5.8×10^{-4}	1.7×10^{-3}
4	1.77	9.7×10^{-8}	2.0×10^{-7}
5	2.55	3.7×10^{-4}	6.4×10^{-4}

with one iteration. Tests 4 and 5 use the more accurate starting approximation of the reference with two and one iterations, respectively, without indexing.

As a final test, the square root subroutine that gave a time ratio of 1.77 was tried in ENEC which used the square root subroutine 44 percent of the running time. The saving in computer time for this code was considerable, while the results compared favorably with those using the slower library subroutine. The listing of FSQR, the subroutine used in this final test, is given in appendix E.

APPENDIX E

MACHINE LANGUAGE SUBROUTINES

RANDOM

Purpose:

A double-entry subroutine for generating pseudorandom numbers within the range from 0 to 1.

Calling sequence:

CALL SAND(RO)

CALL RAND(R)

Use:

The first call to this subroutine must be CALL SAND(RO). This call is necessary to set up addresses in RAND. The statement CALL RAND(R) will cause the next pseudorandom number to be generated. The normalized floating point number is stored in R, while the fixed point number is stored in RO.

Method:

The pseudorandom number sequence is generated by the low order 36 bits of the product $r_{i-1}K$, where $K = 5^{15}$, r_{i-1} is the previous pseudorandom number, and $r_0 = 1$ (see ref. 4). This fixed point number is then floated and normalized so that the result is in the range from 0 to 1.

Remarks:

This method is dependent on the computer word length and was specifically designed for the IBM 7094.

Map listing:

```
$IBMAP RANDOM
       ENTRY   RAND
       ENTRY   SAND
SAND   CLA     3,4
       STA     B        MULTIPLIER IN RANDOM NO. GENERATOR
       CLA     ONE      SET DAM = TO 1 FOR FIRST RANDOM NO.
DAM    STO*    3,4
       TRA     1,4
RAND   SAVE    (4)
       LDQ*    B
       MPY     CONS     BY DAM
     B STQ             STORE THE LOW ORDER PART AT DAM
       CLA     FLC      FLOAT NORMALIZE , AND
       LLS     27       ROUND THE
       FAD     C        RANDOM NO.
R      STO*    3,4
       TRA     1,4
   FLC OCT     000000000200      EXPONENT OF RANDOM NO.
```

```
CONS  DEC     30517578125        5 EXP 15
 ONE  DEC     1                  ONE
   C  OCT     170000000200       NORMALIZING CONSTANT
      END
```

BCREAD

Purpose:

Subroutine BCREAD allows the programer to read absolute binary cards with a maximum of 22 words per card. BCREAD is to be used in conjunction with BCDUMP.

Calling sequence:

CALL BCREAD(A,B), where A is the first data word to be read, and B is the last data word to be read. If the address of A equals the address of B, one word is read.

Remarks:

The address of A must be less than or equal to the address of B. BCREAD makes use of the file definition subroutine .READ5.

Map listings:

```
$IBMAP BCREAD
       TTL     BCREAD SUBROUTINE FOR IBSYS
       LBL     BCREAD
       ENTRY   BCREAD
BCREAD SAVE    1,2,4
       CLA     3,4             PICK UP THE FIRST ARGUMENT
       LDQ     4,4             PICK UP THE SECOND ARGUMENT
       TLQ     *+2             MAKE SURE THE LARGEST
       XCA                     ARGUMENT IS IN THE AC
       STQ     TEMP
       SUB     TEMP
       PAX     0,1             PUT WORD COUNT + 1
       TXI     *+1,1,1         INTO INDEX 1
       LXA     TEMP,2          PICK UP THE FIRST LOAD ADDRESS
       SXA     IX1,1
       SXA     IX2,2
       CLA*    IN5
       STA     *+2
       TSX     .CLOSE,4
       MON     **
       CLA*    READ5
       STA     MON
       STA     READ2
       STA     SHUT
       TSX     .OPEN,4
MON    MON     **
IX1    AXT     **,1            HOLDS THE WORD COUNT
IX2    AXT     **,2            HOLDS THE LOADING ADDRESS
SXA    SXA     IO,2
       TXL     LASTC,1,22
READ   TSX     .READ,4
READ2  PZE     **,,EOB
```

```
        PZE     EOF,,ERR
        IOCPN   **,,2           SKIP FIRST WORD AND CHECKSUM
IO      IOCD    **,,22
        TXI     *+1,1,-22
        TXI     *+1,2,22
        TRA     SXA
LASTC   CLA     DONE
        STO     *-2
        SXD     IO,1            REDUCE THE WORD COUNT
        TRA     READ
DONE    TRA     *+1
        AXT     SXA,4
        SXA     LASTC-1,4
        AXT     22,4
        SXD     IO,4
        TSX     .CLOSE,4
SHUT    MON     **
        RETURN  BCREAD
EOB     CALL    .FXEM.(EOB2)
        TRA     EOF
ERR     CALL    .FXEM.(ERR2)
EOF     CALL    EXIT
ERR2    PZE     35
EOB2    PZE     36
IN5     PZE     .UN05.
READ5   PZE     .READ5
TEMP    PZE
        END

$IBMAP  .READ5
        ENTRY   .READ5
.READ5  PZE     READ5
READ5   FILE    ,IN1,READY,INPUT,BLK=28,MULTIREEL,MXBIN,NOLIST
        END
```

BCDUMP

Purpose:

Subroutine BCDUMP allows the programer to punch out data in an absolute binary format with a maximum of 22 words per card. BCDUMP is meant to be used in conjunction with BCREAD.

Calling sequence:

CALL BCDUMP(A,B,K), where A is the first data word to be punched and B is the last. If the address of A equals the address of B, one word is punched. K controls card numbering. If K equals zero or is missing, each call to BCDUMP will start numbering cards with 000. If K is not equal to zero, the numbering continues in sequential order starting with 000.

Remarks:

The address of A must be less than or equal to the address of B. BCDUMP makes use of the file definition subroutine .PCH..

Map listings:

```
$IBMAP BCDUMP
       TTL     BCDUMP ROUTINE FOR IBSYS
       ENTRY   BCDUMP
BCDUMP SAVE    1,2,4
       CLA     1,4             IS THERE A
       PDX     0,2               THIRD
       TXL     *+2,2,2             ARGUMENT
       NZT*    5,4             YES, IS IT = 0
       SXA     CNUM,0                YES
       CLA     3,4             PICK UP FIRST ARGUMENT
       LDQ     4,4             PICK UP SECOND ARGUMENT
       TLQ     *+2
       XCA
       STQ     WD1             WD1 HAS THE FIRST ADDRESS
       LXA     WD1,1           FIRST LOCATION IN INDEX 1
       SUB     WD1
       PAX     0,2             THE NO. OF WORDS OUTPUTED IN INDEX 2
       TXI     *+1,2,1         TRUE WORD COUNT
       SXA     IX1,1
       SXA     IX2,2
       CLA*    OUT
       STA     RITE+1
       STA     MON
       STA     CLSE
       PAX     0,1
       TXI     *+1,1,1
       SXA     *+1,1
       LDI     **
       LNT     040000
       TRA     *+2
       TRA     *+3
       TSX     .OPEN,4
MON    MON     **
IX1    AXT     **,1
IX2    AXT     **,2
TEST   TXL     LASTC,2,22      ONLY 22 WORDS OR LESS LEFT
       TIX     *+1,2,22
       SXA     IX2,2
       AXT     22,2
TEST4  TXI     *+1,2,320
       SXD     WD1,2
       TIX     *+1,2,320
       SXA     CLA,1
       SXA     WD1,1
LOOP   TXI     *+1,1,22
       SXA     IX1,1
       AXT     23,4
CLEAR  STZ     CKSUM+23,4      CLEAR THE BUFFER
       TIX     *-1,4,1
       AXT     0,4
CLA    CLA     **,4            FILL THE BUFFER WITH
       STO     CKSUM+1,4       NEW DATA
       TXI     *+1,4,-1
       TIX     *-3,2,1
CNUM   AXT     **,1            CONSECUTIVELY
       CLA     HUNBIT          NUMBER
       ARS     1               THE
```

```
         TXL     *+2,1,99        BCDUMP
         TXI     *-2,1,-100      CARDS
         STA     GP+1            FROM
         CLA     BITT            ZERO
         ARS     1               TO
         TXL     *+2,1,9         999
         TXI     *-2,1,-10
         STO     WORD3
         CLA     BITU
         ARS     1
         TXL     *+2,1,0
         TXI     *-2,1,-1
         ORS     WORD3
         LXA     CNUM,1
         TXI     *+1,1,1
         TXL     *+2,1,999
         AXT     0,1
         SXA     CNUM,1
         AXT     22,1            COMPUTE
         CAL     WD1             THE
         ACL     GP,1            CHECK SUM
         TIX     *-1,1,1
         SLW     CKSUM
RITE     TSX     .WRITE,4        WRITE THE BINARY CARD ON THE OUTPUT TAPE
         PZE     **,,EOF
         IOCT    WD1,,28         CHANGE TO  TRANSMIT FOR DIRECT COUPLE  LMLR
         TRA     IX1
RETURN   TRA     *+1
         AXT     IX1,1
         SXA     RETURN-1,1
         TSX     .CLOSE,4
CLSE     MON     **
         RETURN  BCDUMP
LASTC    CLA     RETURN
         STO     RETURN-1
         TRA     TEST4
EOF      TSX     .FXEM.,4        ERROR ON END OF FILE                    LMLR
         TXI     *+3,,1                                                  LMLR
         PZE     *,,OUT+1                                                LMLR
         PTH     EOF2,,2                                                 LMLR
         CALL    EXIT
EOF2     BCI     2,EOF BCDUMP
WD1      PZE
CKSUM    BSS     23
GP       OCT     420041004040
         OCT     104020400000
WORD3    PZE
         PZE
HUNBIT   OCT     2000
BITU     OCT     20000000
BITT     OCT     200000000000
OUT      PZE     .PCH.
         END

$IBMAP   .PCH.
         ENTRY   .PCH.
.PCH.    PZE     PCH
PCH      FILE    ,PP,READY,OUTPUT,BLK=28,MULTIREEL,BIN,NOLIST
         END
```

FSQR

Purpose:

A double-entry subroutine to compute the square root by the method described in appendix D.

Calling sequence:

SQRT(X), ASQRT(X)

Remarks:

The entry point SQRT computes the square root of the argument and returns with the result in the accumulator. The entry point ASQRT forces the sign of the argument positive before computing the square root.

Map listing:

```
$IBMAP FSQR
       ENTRY   ASQRT
       ENTRY   SQRT
ASQRT  CLA*    3,4        SQUARE ROOT SUBROUTINE
       TZE     1,4
       SSP                EVE-S FIRST APPROXIMATION (11--)
       TRA     BEGIN      TWO NEWTON-RAPHSON ITERATIONS
SQRT   CLA*    3,4        03/08/66
       TZE     1,4
       TMI     ERROR
BEGIN  STO     BUFF
       ANA     K1
       TZE     *+2
       SUB     K3
       ADD     BUFF
       ARS     1
       ADD     K2
       STO     BUFF+1
       CLA     BUFF
       FDP     BUFF+1
       XCA
       FAD     BUFF+1
       SUB     K1
       STO     BUFF+1
       CLA     BUFF
       FDP     BUFF+1
       XCA
       FAD     BUFF+1
       SUB     K1
       TRA     1,4
ERROR  SXA     SYSLOC,4
       SXA     LINK,4
       CALL    .FXEM.(ESQRT)
       ORG     *-1
ESQRT  MTW     .SQRTN,,3
       LXA     LINK,4
       SSP
       TRA     BEGIN
```

```
   K1    OCT     001000000000
   K2    OCT     100356300000
   K3    OCT     000356300000
BUFF     BSS     2
LINK     LDIR
         END
```

APPENDIX F

SAMPLE PROBLEM

The example given in this appendix is ENEC's solution to the equation

$$\varphi''(x) = 50n(x)$$

for $\varphi'(0) = -13$ and $\varphi(0) = 0$, where $\varphi(x)$ is the potential distribution and $n(x)$ is the dimensionless electron density distribution. The scattering gas is to be argon at a temperature of 1000^{o} K. The reduced pressure in torr times the interelectrode spacing in centimeters P_0L is taken to be 2. The number of particles for each iteration is 500, and averaging is to take place over 10 iterations after convergence to 0.8 is achieved. The Chebyshev fit is to use 17 data points and the initial fit of $\varphi(x)$ was obtained from a previous ENEC run.

Tables VI to XII give the input and sample printed output of the programs that construct the tables needed by ENEC (see section Preparation of Input Tables). The input data were obtained for argon. To get a three dimensional perspective for the argon electron differential scattering cross-section surface $\sigma(\theta, E)$, the surface was drawn on a digital mechanical plotter and is depicted in figure 38. Output from ENEC (described in the section ENEC Output) for this example is given in figure 39 and table XIII. This sample problem was taken from reference 12. This reference contains many results obtained from the ENEC code.

TABLE VI. - FIRST PAGE OF PRINTED OUTPUT FROM PROGRAM CVEL -ln R

I	VEL							
1	7.6246	6.5260	6.0152	5.6787	5.4274	5.2267	5.0597	4.9166
9	4.7914	4.6802	4.5801	4.4891	4.4057	4.3288	4.2573	4.1906
17	4.1281	4.0693	4.0137	3.9611	3.9110	3.8634	3.8180	3.7745
25	3.7328	3.6928	3.6543	3.6173	3.5816	3.5471	3.5137	3.4815
33	3.4502	3.4199	3.3905	3.3619	3.3342	3.3071	3.2808	3.2552
41	3.2302	3.2058	3.1820	3.1587	3.1360	3.1138	3.0920	3.0707
49	3.0499	3.0295	3.0095	2.9899	2.9707	2.9518	2.9333	2.9151
57	2.8972	2.8797	2.8624	2.8455	2.8288	2.8124	2.7963	2.7804
65	2.7648	2.7494	2.7343	2.7193	2.7046	2.6901	2.6759	2.6618
73	2.6479	2.6342	2.6207	2.6073	2.5942	2.5812	2.5684	2.5557
81	2.5432	2.5309	2.5187	2.5066	2.4947	2.4830	2.4713	2.4598
89	2.4485	2.4372	2.4261	2.4151	2.4043	2.3935	2.3829	2.3723
97	2.3619	2.3516	2.3414	2.3313	2.3213	2.3114	2.3016	2.2919
105	2.2823	2.2728	2.2633	2.2540	2.2447	2.2355	2.2265	2.2174
113	2.2085	2.1997	2.1909	2.1822	2.1736	2.1650	2.1566	2.1482
121	2.1398	2.1316	2.1234	2.1152	2.1072	2.0992	2.0912	2.0834
129	2.0755	2.0678	2.0601	2.0525	2.0449	2.0374	2.0299	2.0225
137	2.0151	2.0078	2.0006	1.9934	1.9863	1.9792	1.9721	1.9651
145	1.9582	1.9513	1.9444	1.9376	1.9309	1.9242	1.9175	1.9109
153	1.9043	1.8978	1.8913	1.8848	1.8784	1.8720	1.8657	1.8594
161	1.8532	1.8470	1.8408	1.8347	1.8286	1.8225	1.8165	1.8105
169	1.8045	1.7986	1.7927	1.7869	1.7811	1.7753	1.7695	1.7638
177	1.7582	1.7525	1.7469	1.7413	1.7357	1.7302	1.7247	1.7193
185	1.7138	1.7084	1.7030	1.6977	1.6924	1.6871	1.6818	1.6766
193	1.6714	1.6662	1.6610	1.6559	1.6508	1.6457	1.6407	1.6357
201	1.6307	1.6257	1.6207	1.6158	1.6109	1.6060	1.6012	1.5963
209	1.5915	1.5867	1.5820	1.5772	1.5725	1.5678	1.5632	1.5585
217	1.5539	1.5493	1.5447	1.5401	1.5356	1.5310	1.5265	1.5221
225	1.5176	1.5132	1.5087	1.5043	1.4999	1.4956	1.4912	1.4869
233	1.4826	1.4783	1.4740	1.4698	1.4655	1.4613	1.4571	1.4529
241	1.4488	1.4446	1.4405	1.4364	1.4323	1.4282	1.4241	1.4201
249	1.4160	1.4120	1.4080	1.4040	1.4001	1.3961	1.3922	1.3882

TABLE VII. - OUTPUT OF INPUT

TO PROGRAM MFP $\sigma(E^{1/2})$

SQRT(EV)	SIGMA-S
0.	36.40
0.1000000	25.00
0.2000000	13.25
0.3000000	6.70
0.4000000	3.30
0.5000000	1.70
0.6000000	1.40
0.7000000	1.60
0.8000000	2.35
0.9000000	3.40
1.0000000	4.80
1.4142136	12.00
1.7888543	20.20
2.0000000	25.40
2.2360680	32.20
2.5884358	45.00
2.8284271	54.50
3.0000000	63.00
3.2093612	72.20
3.5355338	80.00
3.8729833	84.16

TABLE VIII. - PRINTED OUTPUT OF PROGRAM MFP $\lambda(E)$

	EV	MFP
1	0.005	0.03545675
2	0.015	0.04547802
3	0.025	0.05670404
4	0.035	0.06943773
5	0.045	0.08314588
6	0.055	0.09728152
7	0.065	0.11181470
8	0.075	0.12682515
9	0.085	0.14252448
10	0.095	0.15929276
11	0.105	0.17744290
12	0.115	0.19708641
13	0.125	0.21831543
14	0.135	0.24120887
15	0.145	0.26582742
16	0.155	0.29220729
17	0.165	0.32035189
18	0.175	0.35019431
19	0.185	0.38156812
20	0.195	0.41421016
21	0.205	0.44774819
22	0.215	0.48169303
23	0.225	0.51543956
24	0.235	0.54827990
25	0.245	0.57943085
26	0.255	0.60808366
27	0.265	0.63362976
28	0.275	0.65575616
29	0.285	0.67432405
30	0.295	0.68935928
31	0.305	0.70103466
32	0.315	0.70964184
33	0.325	0.71555825
34	0.335	0.71921379
35	0.345	0.72106143
36	0.355	0.72155368
37	0.365	0.72111481
38	0.375	0.71989438
39	0.385	0.71779265
40	0.395	0.71471675
41	0.405	0.71059322
42	0.415	0.70536911
43	0.425	0.69901259
44	0.435	0.69151314
45	0.445	0.68288124
46	0.455	0.67314735
47	0.465	0.66236047
48	0.475	0.65058606
49	0.485	0.63790367
50	0.495	0.62440746
51	0.505	0.61027010
52	0.515	0.59571803
53	0.525	0.58094945
54	0.535	0.56613175
55	0.545	0.55140359

TABLE VIII. - Continued. PRINTED OUTPUT OF PROGRAM MFP $\lambda(E)$

56	0.555	0.53687751
57	0.565	0.52264289
58	0.575	0.50876888
59	0.585	0.49530750
60	0.595	0.48229624
61	0.605	0.46976049
62	0.615	0.45771582
63	0.625	0.44616985
64	0.635	0.43512379
65	0.645	0.42457313
66	0.655	0.41449207
67	0.665	0.40483951
68	0.675	0.39557862
69	0.685	0.38667697
70	0.695	0.37810599
71	0.705	0.36984034
72	0.715	0.36185761
73	0.725	0.35413782
74	0.735	0.34666324
75	0.745	0.33941796
76	0.755	0.33238783
77	0.765	0.32556009
78	0.775	0.31892335
79	0.785	0.31246728
80	0.795	0.30618261
81	0.805	0.30006094
82	0.815	0.29409487
83	0.825	0.28828214
84	0.835	0.28262451
85	0.845	0.27712286
86	0.855	0.27177718
87	0.865	0.26659668
88	0.875	0.26154993
89	0.885	0.25666498
90	0.895	0.25192939
91	0.905	0.24734043
92	0.915	0.24289504
93	0.925	0.23858999
94	0.935	0.23442183
95	0.945	0.23038704
96	0.955	0.22648200
97	0.965	0.22270304
98	0.975	0.21904650
99	0.985	0.21550869
100	0.995	0.21208598
101	1.250	0.15176453
102	1.750	0.09863517
103	2.250	0.07346389
104	2.750	0.05875729
105	3.250	0.04919928
106	3.750	0.04247487
107	4.250	0.03735838
108	4.750	0.03320196
109	5.250	0.02966338
110	5.750	0.02667541
111	6.250	0.02423191
112	6.750	0.02226726
113	7.250	0.02066121
114	7.750	0.01922455
115	8.250	0.01785111

TABLE VIII. - Concluded. PRINTED OUTPUT OF

PROGRAM MFP $\lambda(E)$

116	8.750	0.01658869
117	9.250	0.01553171
118	9.750	0.01469191
119	10.250	0.01404402
120	10.750	0.01356194
121	11.250	0.01320282
122	11.750	0.01292968
123	12.250	0.01271599
124	12.750	0.01254188
125	13.250	0.01239486
126		0.01199999

01 EXIT IN MFP

TABLE IX - INPUT TO PROGRAM ARGON FOR $\sigma(\theta, E)$

$DATA

	0.	15.	28.	43.	59.	74.5	90.	105.5	121.	137.	152.5	167.5	180.
.01	2.73	2.53	2.38	2.21	2.05	1.91	1.79	1.70	1.62	1.56	1.52	1.49	1.49
.05	2.36	1.96	1.67	1.38	1.13	.933	.780	.662	.574	.509	.468	.447	.441
.1	2.06	1.55	1.20	.880	.624	.444	.316	.228	.169	.130	.107	.096	.093
.2	1.68	1.07	.687	.386	.188	.081	.027	.005	.000	.002	.006	.008	.009
.3	1.45	.788	.417	.165	.039	.001	.007	.028	.052	.072	.085	.091	.093
.4	1.31	.613	.262	.062	.001	.017	.060	.102	.133	.153	.162	.166	.166
.5	1.23	.502	.170	.018	.009	.062	.123	.168	.192	.199	.198	.195	.194
.6	1.20	.431	.116	.003	.033	.108	.173	.207	.214	.204	.190	.180	.177
1.1	.00	.030	.050	.100	.163	.190	.199	.172	.131	.090	.063	.032	.000
2.0	.06	.100	.170	.300	.390	.420	.359	.280	.180	.115	.120	.100	.07
2.4	.15	.200	.240	.380	.490	.500	.420	.320	.200	.140	.160	.150	.12
2.8	.28	.300	.300	.450	.570	.590	.470	.350	.216	.160	.200	.210	.18
3.2	.42	.425	.370	.540	.660	.670	.530	.390	.240	.188	.240	.282	.26
4.0	.84	.700	.540	.680	.820	.820	.630	.450	.280	.270	.336	.460	.44
5.0	1.68	1.10	.770	.850	1.00	1.00	.750	.530	.340	.410	.660	.800	.80
6.7	4.85	2.42	1.36	1.13	1.22	1.18	.915	.620	.420	.660	1.37	1.88	2.14
8.0	7.00	4.45	2.06	1.18	1.19	1.16	.885	.640	.458	.824	1.80	2.68	3.17
9.0	8.25	5.75	2.84	1.60	1.33	1.19	.947	.690	.563	1.06	2.10	3.03	3.50
10.3	9.60	7.15	4.42	2.27	1.27	1.11	1.00	.775	.610	1.16	2.38	3.30	3.70
12.5	13.0	9.30	4.93	2.38	1.31	.960	.830	.690	.690	1.23	2.58	3.49	3.90

TABLE X. - OUTPUT FROM PROGRAM ARGON $\int_{-1}^{\cos\theta} \frac{\sigma(\cos^{-1}t, E)dt}{\sigma(E)}$

EV/COS(THETA)	-1.0	-0.8	-0.6	-0.4	-0.2	SIGMA -0.0	0.2	0.4	0.6	0.8	1.0	SIGMA-S
0.01	-0.	0.0816	0.1661	0.2536	0.3446	0.4392	0.5378	0.6417	0.7517	0.8693	1.0000	3.713
0.05	-0.	0.0520	0.1098	0.1745	0.2471	0.3290	0.4219	0.5289	0.6538	0.8033	1.0000	1.791
0.10	-0.	0.0235	0.0535	0.0918	0.1406	0.2029	0.2824	0.3862	0.5223	0.7081	1.0000	0.901
0.20	-0.	0.0041	0.0052	0.0053	0.0079	0.0192	0.0466	0.1094	0.2289	0.4601	1.0000	0.303
0.30	-0.	0.0870	0.1572	0.2084	0.2400	0.2549	0.2553	0.2621	0.2990	0.4407	1.0000	0.197
0.40	-0.	0.1440	0.2773	0.3935	0.4878	0.5561	0.5925	0.6071	0.6085	0.6493	1.0000	0.226
0.50	-0.	0.1429	0.2865	0.4244	0.5487	0.6518	0.7234	0.7651	0.7735	0.7806	1.0000	0.276
0.60	-0.	0.1221	0.2555	0.3940	0.5292	0.6519	0.7485	0.8144	0.8383	0.8386	1.0000	0.308
1.10	-0.	0.0416	0.1128	0.2117	0.3352	0.4773	0.6240	0.7628	0.8845	0.9663	1.0000	0.270
2.00	-0.	0.0421	0.0879	0.1573	0.2566	0.3796	0.5235	0.6790	0.8249	0.9421	1.0000	0.538
2.40	-0.	0.0472	0.0920	0.1554	0.2481	0.3654	0.5040	0.6568	0.8065	0.9283	1.0000	0.656
2.80	-0.	0.0521	0.0956	0.1547	0.2417	0.3532	0.4900	0.6449	0.7950	0.9187	1.0000	0.762
3.20	-0.	0.0557	0.0994	0.1561	0.2395	0.3475	0.4809	0.6327	0.7824	0.9097	1.0000	0.884
4.00	-0.	0.0653	0.1135	0.1660	0.2418	0.3426	0.4696	0.6174	0.7646	0.8911	1.0000	1.116
5.00	-0.	0.0936	0.1478	0.1968	0.2660	0.3578	0.4756	0.6149	0.7533	0.8741	1.0000	1.447
6.70	-0.	0.1416	0.2007	0.2431	0.3003	0.3787	0.4780	0.5952	0.7136	0.8237	1.0000	2.053
8.00	-0.	0.1638	0.2261	0.2656	0.3164	0.3823	0.4651	0.5631	0.6623	0.7603	1.0000	2.399
9.00	-0.	0.1596	0.2273	0.2677	0.3142	0.3738	0.4471	0.5320	0.6249	0.7343	1.0000	2.849
10.30	-0.	0.1523	0.2155	0.2531	0.2977	0.3533	0.4163	0.4836	0.5597	0.6852	1.0000	3.330
12.50	-0.	0.1541	0.2178	0.2570	0.2958	0.3397	0.3892	0.4454	0.5186	0.6433	1.0000	3.531

01 EXIT IN ARGON

TABLE XI. - INPUT TO PROGRAM ARGINV $\int_{-1}^{\cos\theta} \frac{\sigma(\cos^{-1}t, E)dt}{\sigma(E)}$

```
$DATA
```

	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
0.01	0.0	.0816	.1661	.2536	.3446	.4392	.5378	.6417	.7517	.8693	1.0
0.05	0.0	.0520	.1098	.1745	.2471	.3290	.4219	.5289	.6538	.8033	1.0
0.10	0.0	.0235	.0535	.0918	.1406	.2029	.2824	.3862	.5223	.7081	1.0
0.20	0.0	.0041	.0052	.0053	.0079	.0192	.0466	.1094	.2289	.4601	1.0
0.30	0.0	.0870	.1572	.2084	.2400	.2549	.2553	.2621	.2990	.4407	1.0
0.40	0.0	.1440	.2773	.3935	.4878	.5561	.5925	.6071	.6085	.6493	1.0
0.50	0.0	.1429	.2865	.4244	.5487	.6518	.7234	.7651	.7735	.7806	1.0
0.60	0.0	.1221	.2555	.3940	.5292	.6519	.7485	.8144	.8383	.8386	1.0
1.10	0.0	.0416	.1128	.2117	.3352	.4773	.6240	.7628	.8845	.9663	1.0
2.00	0.0	.0421	.0879	.1573	.2566	.3796	.5235	.6790	.8249	.9421	1.0
2.40	0.0	.0472	.0920	.1554	.2481	.3654	.5040	.6568	.8065	.9283	1.0
2.80	0.0	.0521	.0956	.1547	.2417	.3532	.4900	.6449	.7950	.9187	1.0
3.20	0.0	.0557	.0994	.1561	.2395	.3475	.4809	.6327	.7824	.9097	1.0
4.00	0.0	.0653	.1135	.1660	.2418	.3426	.4696	.6174	.7646	.8911	1.0
5.00	0.0	.0936	.1478	.1968	.2660	.3578	.4756	.6149	.7533	.8741	1.0
6.70	0.0	.1416	.2007	.2431	.3003	.3787	.4780	.5952	.7136	.8237	1.0
8.00	0.0	.1638	.2261	.2656	.3164	.3823	.4651	.5631	.6623	.7603	1.0
9.00	0.0	.1596	.2273	.2677	.3142	.3738	.4471	.5320	.6249	.7343	1.0
10.3	0.0	.1523	.2155	.2531	.2977	.3533	.4163	.4836	.5597	.6852	1.0
12.5	0.0	.1541	.2178	.2570	.2958	.3397	.3892	.4454	.5186	.6433	1.0

TABLE XII. - FIRST THREE PAGES OF PRINTED OUTPUT FROM ARGINV (R, EV, COS θ)

I	R							
1	0.0078125	0.0234375	0.0390625	0.0546875	0.0703125	0.0859375	0.1015625	0.1171875
9	0.1328125	0.1484375	0.1640625	0.1796875	0.1953125	0.2109375	0.2265625	0.2421875
17	0.2578125	0.2734375	0.2890625	0.3046875	0.3203125	0.3359375	0.3515625	0.3671875
25	0.3828125	0.3984375	0.4140625	0.4296875	0.4453125	0.4609375	0.4765625	0.4921875
33	0.5078125	0.5234375	0.5390625	0.5546875	0.5703125	0.5859375	0.6015625	0.6171875
41	0.6328125	0.6484375	0.6640625	0.6796875	0.6953125	0.7109375	0.7265625	0.7421875
49	0.7578125	0.7734375	0.7890625	0.8046875	0.8203125	0.8359375	0.8515625	0.8671875
57	0.8828125	0.8984375	0.9140625	0.9296875	0.9453125	0.9609375	0.9765625	0.9921875

J	EV							
1	0.0312500	0.0937500	0.1562500	0.2187500	0.2812500	0.3437500	0.4062500	0.4687500
9	0.5312500	0.5937500	0.6562500	0.7187500	0.7812500	0.8437500	0.9062500	0.9687500
17	1.0312500	1.0937500	1.1562500	1.2187500	1.3750000	1.6250000	1.8750000	2.1250000
25	2.3750000	2.6250000	2.8750000	3.1250000	3.3750000	3.6250000	3.8750000	4.1250000
33	4.3750000	4.6250000	4.8750000	5.1250000	5.3750000	5.6250000	5.8750000	6.1250000
41	6.3750000	6.6250000	6.8750000	7.1250000	7.3750000	7.6250000	7.8750000	8.1250000
49	8.3750000	8.6250000	8.8750000	9.1250000	9.3750000	9.6250000	9.8750000	10.1250000
57	10.3750000	10.6250000	10.8750000	11.1250000	11.3750000	11.6250000	11.8750000	12.1250000

I	J								
1	1	-0.9757	-0.9387	-0.5294	-0.8949	-0.9785	-0.9873	-0.9894	-0.9894
1	9	-0.9885	-0.9871	-0.9853	-0.9831	-0.9805	-0.9772	-0.9733	-0.9685
1	17	-0.9630	-0.9567	-0.9503	-0.9443	-0.9358	-0.9450	-0.9589	-0.9654
1	25	-0.9677	-0.9701	-0.9721	-0.9734	-0.9744	-0.9754	-0.9768	-0.9786
1	33	-0.9806	-0.9825	-0.9842	-0.9856	-0.9867	-0.9877	-0.9885	-0.9891
1	41	-0.9897	-0.9902	-0.9907	-0.9910	-0.9913	-0.9916	-0.9917	-0.9917
1	49	-0.9917	-0.9917	-0.9915	-0.9914	-0.9913	-0.9913	-0.9911	-0.9911
1	57	-0.9911	-0.9910	-0.9910	-0.9910	-0.9910	-0.9910	-0.9910	-0.9910
2	1	-0.9273	-0.8225	-0.2829	-0.5185	-0.9351	-0.9620	-0.9681	-0.9681
2	9	-0.9655	-0.9613	-0.9559	-0.9495	-0.9417	-0.9323	-0.9210	-0.9079
2	17	-0.8930	-0.8773	-0.8620	-0.8486	-0.8291	-0.8432	-0.8774	-0.8958
2	25	-0.9025	-0.9095	-0.9156	-0.9194	-0.9224	-0.9256	-0.9298	-0.9353
2	33	-0.9413	-0.9471	-0.9523	-0.9565	-0.9600	-0.9628	-0.9653	-0.9673
2	41	-0.9691	-0.9706	-0.9720	-0.9731	-0.9740	-0.9747	-0.9751	-0.9752
2	49	-0.9751	-0.9748	-0.9744	-0.9740	-0.9737	-0.9735	-0.9733	-0.9731
2	57	-0.9730	-0.9730	-0.9730	-0.9730	-0.9730	-0.9730	-0.9731	-0.9731
3	1	-0.8793	-0.7200	-0.1459	0.0906	-0.8909	-0.9364	-0.9469	-0.9470
3	9	-0.9426	-0.9356	-0.9268	-0.9163	-0.9037	-0.8890	-0.8718	-0.8526
3	17	-0.8320	-0.8115	-0.7926	-0.7764	-0.7509	-0.7574	-0.7976	-0.8247
3	25	-0.8353	-0.8465	-0.8566	-0.8631	-0.8683	-0.8740	-0.8813	-0.8907
3	33	-0.9011	-0.9110	-0.9197	-0.9269	-0.9328	-0.9376	-0.9417	-0.9452
3	41	-0.9482	-0.9508	-0.9530	-0.9550	-0.9565	-0.9576	-0.9583	-0.9585
3	49	-0.9583	-0.9578	-0.9572	-0.9566	-0.9560	-0.9556	-0.9553	-0.9550
3	57	-0.9548	-0.9547	-0.9547	-0.9547	-0.9547	-0.9549	-0.9550	-0.9552
4	1	-0.8319	-0.6290	-0.0470	0.2212	-0.8452	-0.9108	-0.9256	-0.9258
4	9	-0.9197	-0.9100	-0.8979	-0.8837	-0.8671	-0.8480	-0.8266	-0.8036
4	17	-0.7801	-0.7574	-0.7366	-0.7187	-0.6886	-0.6861	-0.7216	-0.7517
4	25	-0.7644	-0.7786	-0.7923	-0.8018	-0.8099	-0.8187	-0.8297	-0.8435
4	33	-0.8586	-0.8731	-0.8857	-0.8963	-0.9047	-0.9117	-0.9176	-0.9226
4	41	-0.9269	-0.9306	-0.9338	-0.9365	-0.9387	-0.9403	-0.9413	-0.9416
4	49	-0.9413	-0.9406	-0.9397	-0.9388	-0.9381	-0.9375	-0.9370	-0.9366
4	57	-0.9364	-0.9362	-0.9362	-0.9362	-0.9362	-0.9365	-0.9367	-0.9369

TABLE XII. - Concluded. FIRST THREE PAGES OF PRINTED OUTPUT FROM ARGINV (R, EV, COS θ)

I	J								
5	1	-0.7852	-0.5464	0.0324	0.2985	-0.7969	-0.8849	-0.9042	-0.9046
5	9	-0.8968	-0.8845	-0.8694	-0.8520	-0.8320	-0.8096	-0.7854	-0.7602
5	17	-0.7349	-0.7105	-0.6883	-0.6690	-0.6353	-0.6257	-0.6534	-0.6804
5	25	-0.6919	-0.7054	-0.7204	-0.7322	-0.7436	-0.7566	-0.7725	-0.7918
5	33	-0.8127	-0.8325	-0.8497	-0.8638	-0.8753	-0.8847	-0.8926	-0.8992
5	41	-0.9049	-0.9098	-0.9140	-0.9176	-0.9205	-0.9226	-0.9239	-0.9243
5	49	-0.9239	-0.9231	-0.9219	-0.9208	-0.9198	-0.9189	-0.9183	-0.9178
5	57	-0.9174	-0.9172	-0.9172	-0.9172	-0.9174	-0.9176	-0.9178	-0.9181
6	1	-0.7394	-0.4705	0.0990	0.3542	-0.7451	-0.8587	-0.8828	-0.8834
6	9	-0.8740	-0.8593	-0.8416	-0.8213	-0.7986	-0.7739	-0.7477	-0.7208
6	17	-0.6940	-0.6683	-0.6448	-0.6243	-0.5879	-0.5732	-0.5942	-0.6158
6	25	-0.6238	-0.6337	-0.6462	-0.6578	-0.6710	-0.6876	-0.7082	-0.7332
6	33	-0.7607	-0.7872	-0.8101	-0.8288	-0.8438	-0.8560	-0.8661	-0.8747
6	41	-0.8820	-0.8882	-0.8936	-0.8981	-0.9017	-0.9044	-0.9060	-0.9065
6	49	-0.9061	-0.9050	-0.9036	-0.9021	-0.9009	-0.8998	-0.8990	-0.8984
6	57	-0.8979	-0.8976	-0.8976	-0.8976	-0.8978	-0.8980	-0.8984	-0.8988
7	1	-0.6945	-0.4005	0.1560	0.3984	-0.6886	-0.8321	-0.8611	-0.8622
7	9	-0.8513	-0.8344	-0.8143	-0.7917	-0.7669	-0.7404	-0.7126	-0.6843
7	17	-0.6561	-0.6292	-0.6047	-0.5832	-0.5448	-0.5266	-0.5426	-0.5592
7	25	-0.5633	-0.5692	-0.5780	-0.5873	-0.5996	-0.6165	-0.6391	-0.6678
7	33	-0.7008	-0.7342	-0.7643	-0.7891	-0.8088	-0.8247	-0.8376	-0.8484
7	41	-0.8575	-0.8654	-0.8721	-0.8777	-0.8821	-0.8854	-0.8874	-0.8881
7	49	-0.8876	-0.8863	-0.8845	-0.8828	-0.8813	-0.8800	-0.8789	-0.8781
7	57	-0.8775	-0.8772	-0.8770	-0.8770	-0.8773	-0.8776	-0.8781	-0.8786
8	1	-0.6502	-0.3356	0.2059	0.4356	-0.6264	-0.8050	-0.8395	-0.8409
8	9	-0.8286	-0.8098	-0.7876	-0.7631	-0.7366	-0.7086	-0.6794	-0.6498
8	17	-0.6205	-0.5926	-0.5671	-0.5449	-0.5050	-0.4843	-0.4965	-0.5091
8	25	-0.5099	-0.5125	-0.5183	-0.5252	-0.5350	-0.5498	-0.5707	-0.5992
8	33	-0.6343	-0.6728	-0.7098	-0.7419	-0.7679	-0.7889	-0.8057	-0.8195
8	41	-0.8310	-0.8407	-0.8490	-0.8560	-0.8614	-0.8654	-0.8679	-0.8687
8	49	-0.8681	-0.8666	-0.8645	-0.8625	-0.8606	-0.8590	-0.8577	-0.8567
8	57	-0.8559	-0.8555	-0.8553	-0.8553	-0.8556	-0.8560	-0.8566	-0.8572

TABLE XIII. - PRINTED OUTPUT FROM ENEC

```
$IN1
NO    =          500, N1    =           17, KI    =           10, ALPHA =  2.0000000E 00, CONST =  5.0000000F 01,
NFLAG =            2, BC    = -1.3000000E 01, KODE  =            2, MODE  =           1, ERROR =  7.9999999E-01,
NS    =            8, KFLAG =            2, TEMPK = 1.0000000E 03, LFLAG =           2,
$ ENC
    THERMIONIC EMISSION 152

AI=
    -0.21969125E 01    -0.23765674F 00     0.62491354E 00    -0.15334123E 00     0.50910291F-01    -0.19946524F-01
     0.68098435E-02    -0.20800841E-02     0.13114373E-02    -0.82991723E-03     0.57191579F-03    -0.43561954E-03
     0.13841715E-04     0.95622907F-04    -0.62333257F-04     0.19405210E-03    -0.12663246F-04     0.
     0.                 0.
```

	XD	MEAN N	STD. N
1	-0.	967.900	4.537
2	0.00960736	856.800	5.957
3	0.03806023	613.200	10.594
4	0.08426519	390.200	8.821
5	0.14644661	253.600	8.384
6	0.22221488	163.000	7.467
7	0.30865828	111.000	4.869
8	0.40245484	76.600	4.387
9	0.49939999	61.800	4.320
10	0.59754515	51.400	4.248
11	0.69134171	45.000	4.768
12	0.77778511	41.400	5.012
13	0.85355338	38.400	5.431
14	0.91573481	35.200	4.791
15	0.96193976	35.000	4.879
16	0.99039263	33.000	5.013
17	1.00000000	32.000	4.556

	MEAN AI	STD. AI	MEAN DA	STD. DA	MEAN B	STD. B
1	-3.66167679	0.17469033	-6.51143974	0.39980942	41.15168715	0.60209867
2	-1.04503354	0.11007060	5.43993092	0.16972015	-30.33697963	0.23735175
3	0.54961915	0.02118890	-2.33130553	0.06242871	19.39196372	0.27361134
4	-0.15537757	0.00624523	1.04297768	0.04069808	-11.68653517	0.39588887
5	0.05116830	0.00363047	-0.45677465	0.02416211	6.87623149	0.48478547
6	-0.01727193	0.00195082	0.22428477	0.01892346	-4.21814072	0.36597438
7	0.00658920	0.00065380	-0.12133606	0.01879106	2.39053601	0.19135751
8	-0.00266536	0.00080612	0.06614400	0.00935168	-1.30607525	0.24036396
9	0.00110793	0.00050411	-0.04670603	0.00578302	0.53850392	0.21118020
10	-0.00122989	0.00023203	0.03069014	0.00844622	0.18851788	0.25728086
11	0.00094552	0.00036242	-0.00242998	0.00321652	-0.56634136	0.32823963
12	-0.00014676	0.00017453	-0.00713085	0.00795397	0.28571755	0.19339832
13	0.00003092	0.00016670	0.00402765	0.00709825	-0.25258386	0.20521291
14	-0.00016668	0.00016971	-0.00861512	0.00366031	0.09239006	0.24925443
15	-0.00005572	0.00012649	0.01269518	0.00407287	0.19540282	0.29722011
16	0.00037266	0.00013171	-0.00549471	0.00404726	-0.61854020	0.30506335
17	-0.00008585	0.00006323	-0.00966468	0.00476661	0.52508580	0.13166470

TABLE XIII. - Concluded. PRINTED OUTPUT FROM ENEC

	MEAN Y	STD. Y	MEAN DY	STD. DY	MEAN DDY	STD. DDY
1	0.00000010	0.00000001	-13.00000048	0.00000004	97.27428341	1.95648837
2	-0.12043346	0.00011887	-12.14068496	0.01643634	85.69345951	1.59331208
3	-0.43435966	0.00082861	-10.01470077	0.03406554	63.62157488	1.25241616
4	-0.84042504	0.00725005	-7.77618700	0.05459954	36.78868532	0.94069674
5	-1.26187769	0.00642534	-5.87558651	0.09958778	26.71593428	1.09402619
6	-1.63894701	0.01630090	-4.19454759	0.18267000	17.47496819	1.12554373
7	-1.94463196	0.03347100	-2.95429876	0.21243631	12.34195209	0.76109815
8	-2.17403677	0.05201884	-2.00364363	0.20529954	7.75743997	0.51680201
9	-2.33571520	0.07211596	-1.33246121	0.24044415	6.85151982	0.65472865
10	-2.43513528	0.09561784	-0.73212643	0.27012233	4.69202816	0.63100371
11	-2.48597673	0.12031564	-0.37907971	0.28503805	3.66198051	0.34921714
12	-2.50515202	0.14432648	-0.05743694	0.29545915	3.31606522	0.31601627
13	-2.50015339	0.16649128	0.17689936	0.31401187	3.31563717	0.67820865
14	-2.49312446	0.18585514	0.37558117	0.32977037	2.45997602	0.31569276
15	-2.46316043	0.20063088	0.47336564	0.33487796	2.62418780	0.39094478
16	-2.44876626	0.20988189	0.56060471	0.34028953	2.22343302	0.26049839
17	-2.44303799	0.21302953	0.56557281	0.34145060	2.07499668	0.22437462

	MEAN	STD.
CURRENT	0.06400000	0.00911165
VOLT	-2.44303799	0.21302953
DPHIO	-13.00000048	0.00000004
KNTR	702.100	15.755

PHIMIN= -2.50568336

XMIN= 0.79492188

EQUALLY SPACED DATA

X	PHI(X)	DPHI(X)	N(X)
0.	0.00000013	-13.00000036	1.94548561
0.05000000	-0.54952634	-9.31162477	1.10364881
0.09999999	-0.95850401	-7.22976363	0.64567125
0.15000000	-1.28757124	-5.78223091	0.52781617
0.20000000	-1.54111873	-4.61686987	0.40628888
0.25000000	-1.74916059	-3.74006915	0.29716718
0.30000000	-1.91863452	-3.06029117	0.25245698
0.34999999	-2.05670783	-2.48759082	0.21138898
0.40000000	-2.16908213	-2.02345943	0.15734602
0.45000000	-2.26104784	-1.66022669	0.13258342
0.50000000	-2.33571520	-1.33246116	0.13703039
0.55000000	-2.39395094	-1.00442715	0.12625750
0.59999999	-2.43692946	-0.72007295	0.09216407
0.65000000	-2.46764058	-0.51697391	0.07136576
0.70000000	-2.48909181	-0.34924910	0.07421897
0.75000000	-2.50197527	-0.16180167	0.07216772
0.80000000	-2.50563982	0.01771381	0.06333854
0.84999999	-2.50078127	0.16612880	0.06619857
0.90000000	-2.48856401	0.32855473	0.05539818
0.95000000	-2.46876404	0.44869374	0.04913158
1.00000000	-2.44303799	0.56557281	0.04149993

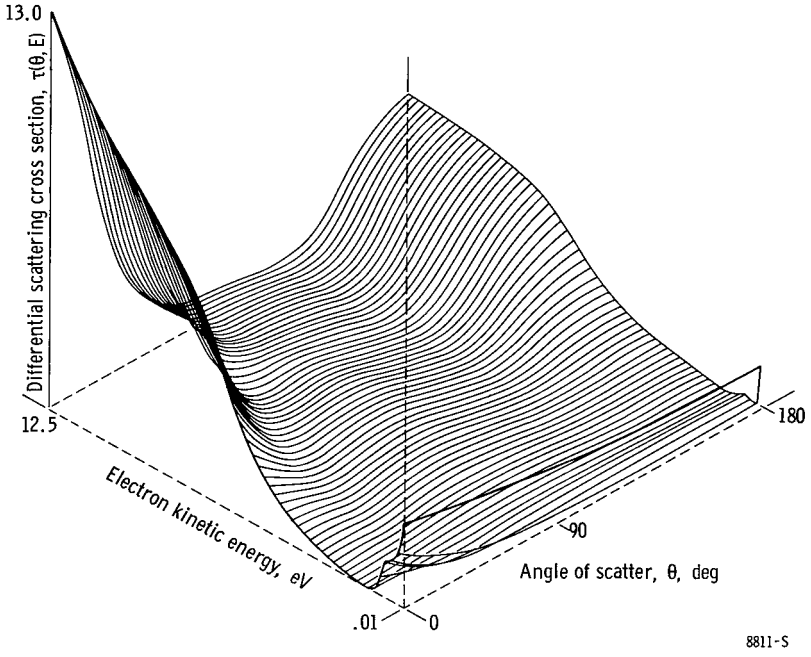
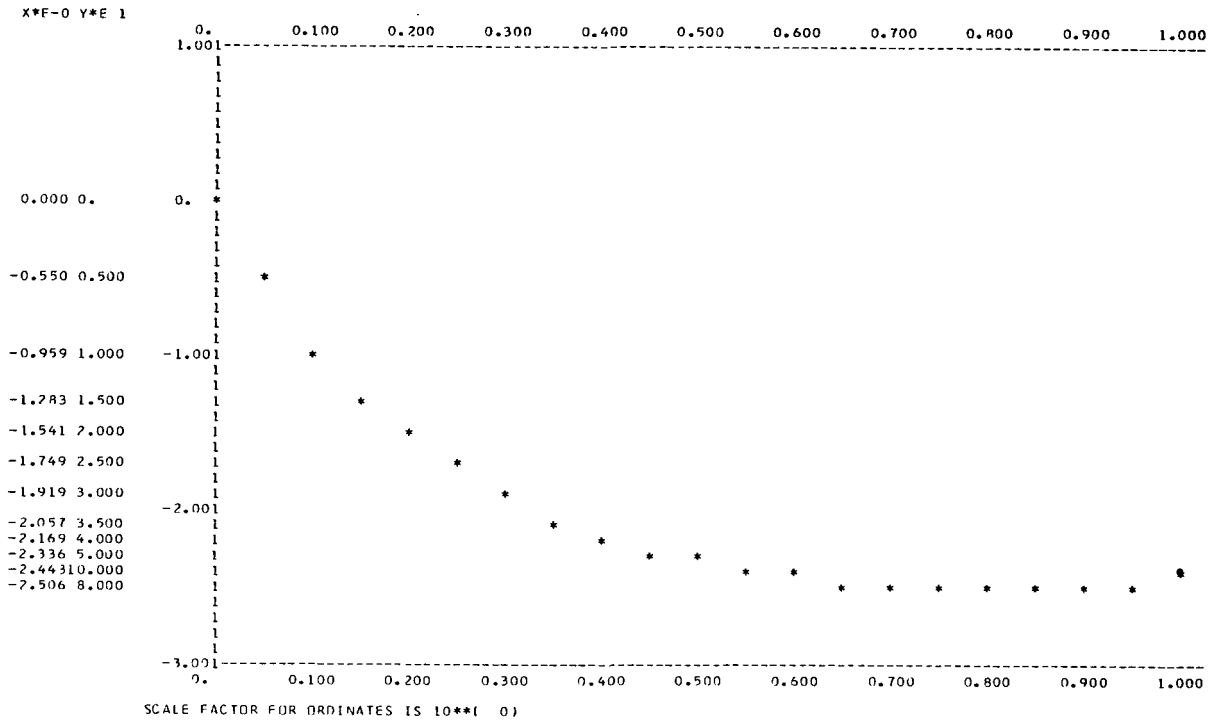
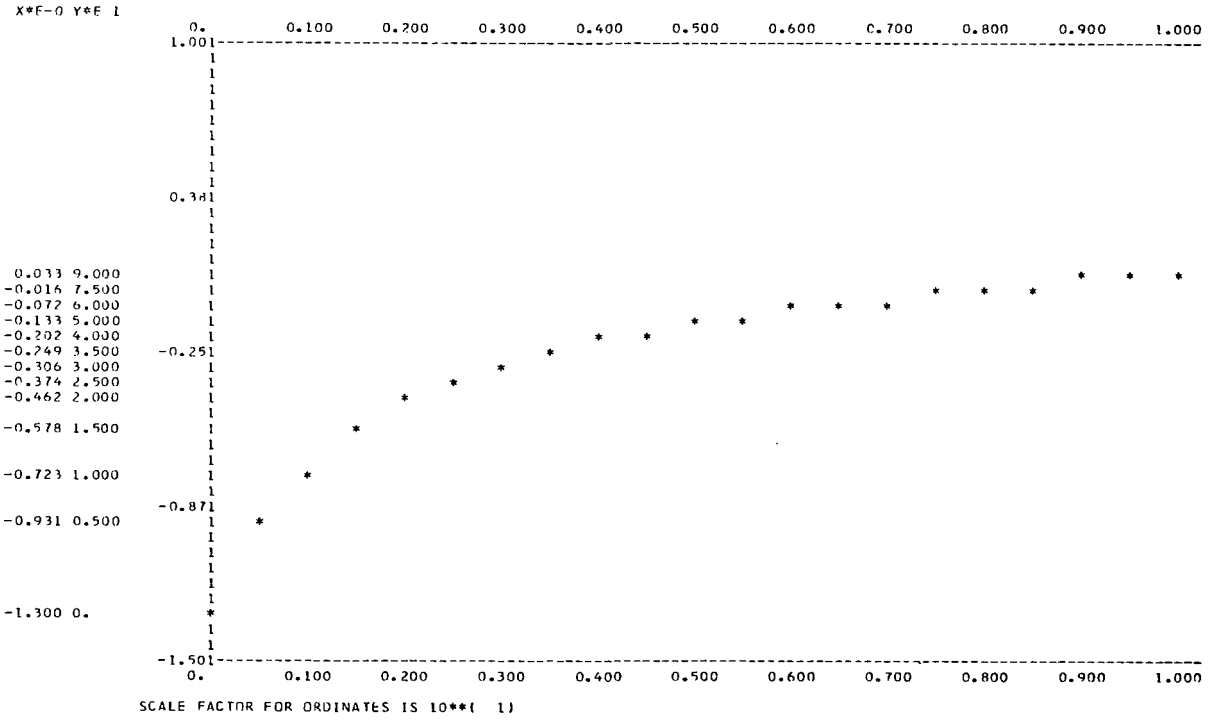


Figure 38. - Argon electron differential scattering cross section.

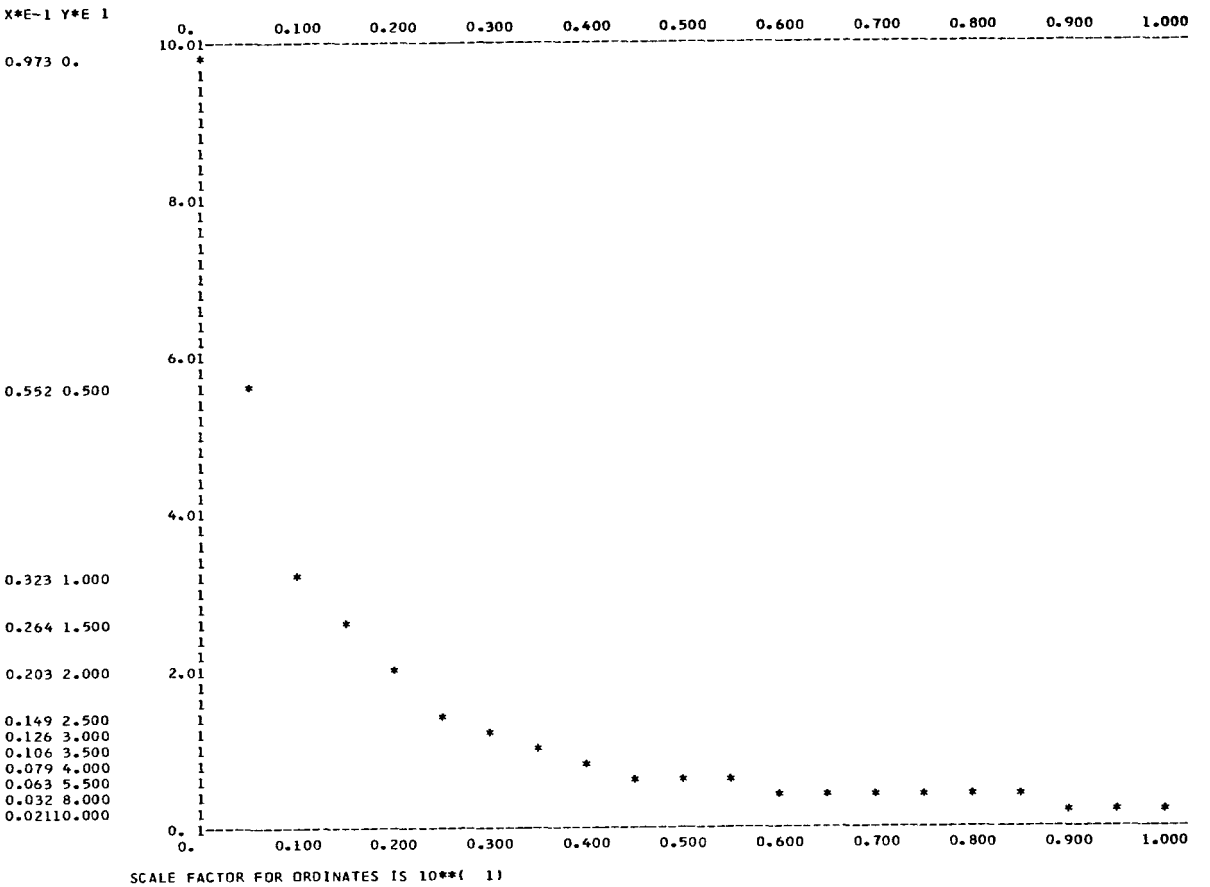


(a) Printer plot of $\varphi(X)$.



(b) Printer plot of $\varphi'(X)$.

Figure 39. - Output from ENEC.



(c) Printer plot of $\varphi''(X)$.

Figure 39. - Concluded.

REFERENCES

1. Goldstein, Charles M.: Monte Carlo Method for the Calculation of Transport Properties in a Low-Density Ionized Gas. NASA TN D-2959, 1965.

2. Clenshaw, C. W.; Norton, H. J.: The Solution of Nonlinear Ordinary Differential Equations in Chebyshev Series. Computer J., vol. 6, 1963, pp. 88-92.

3. Hammersley, J. M.; and Handscomb, D. C.: Monte Carlo Methods. John Wiley & Sons, Inc., 1964, p. 25ff.

4. Taussky, Olga; and Todd, John: Generation and Testing of Pseudo-Random Numbers. Symposium on Monte Carlo Methods. Herbert A. Meyer, ed., John Wiley and Sons, Inc., 1956, pp. 15-28.

5. Scarborough, James B.: Numerical Mathematical Analysis. Fifth ed., Johns Hopkins Press, 1962.

6. Swigert, Paul: Computer Generation of Quadrature Coefficients Utilizing the Symbolic Manipulation Language FORMAC. NASA TN D-3472, 1966.

7. Clenshaw, C. W.: Chebyshev Series for Mathematical Functions. Vol. 5 of the National Physical Laboratory Mathematical Tables. Her Majesty's Stationary Office, London, 1962.

8. Dellner, Lois T.; and Moore, Betty Jo: An Optimized Printer Plotting System Consisting of Complementary 7090 (FORTRAN) and 1401 (SPS) Subroutines. Part I - Instructions for Users. NASA TN D-2174, 1964.

9. Kopal, Zdenek: Numerical Analysis. Second ed., John Wiley & Sons, Inc., 1961, p. 33.

10. Peaceman, D. W.; and Rachford, H. H., Jr.: The Numerical Solution of Parabolic and Elliptic Differential Equations. J. Soc. Ind. Appl. Math., vol. 3, no. 1, Mar. 1955, pp. 28-41.

11. Eve, J.: Starting Approximations for the Iterative Calculation of Square Roots. Computer J., vol. 6, 1963, pp. 274-276.

12. Goldstein, Charles M.: Electron Flow in Low-Density Argon Gas Including Space-Charge and Elastic Collisions. NASA TN D-4087, 1967.

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Notes, and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION DIVISION

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546