UCRL - 4789

PARACANTOR

A two group, two region reactor code

by

Stuart P. Stone
UCRL - Livermore

Programmers:
Paracantor I -
R. Brousseau
Paracantor II -
R. Brousseau
M. Ferris
Report Compiled by:
S. Stone
K. Purdum

DISCLAIMER

PARACANTOR I

Introduction

Paracantor I is a two energy group, two region, time independent reactor code, which obtains a closed solution for a critical reactor assembly. The code deals with cylindrical reactors of finite length and with a radial reflector of finite thickness. It is programmed for the I.B.M. Magnetic Drum Data-Processing Machine, Type 650. The limited memory space available does not permit a flux solution to be included in the basic Paracantor code. A supplementary code, Paracantor II, has been programmed which computes fluxes, including adjoint fluxes, from the output of Paracantor I.

General Description of Paracantor I

A. Physics

Paracantor I, in general, follows the two energy group theory found in The Elements of Nuclear Reactor Theory[1] by Glasstone and Edlund. This report will give a brief resume of the theory which is pertinent to the Paracantor code. The notation used in the report and the code conforms to that of the book. The form of the equations has been modified where either ease of programming or saving of machine time results.

A two energy group approximation to reactor neutron behavior divides the neutrons into two groups: slow or thermal and fast or fission neutrons. The general equation for the conservation or balance of neutrons may be written

$$\text{Production} - \text{Leakage} - \text{Absorption} = \frac{\partial n}{\partial t}$$

where $\partial n/\partial t$ is the time rate of change of the neutron density. If a system is in an equilibrium state, referred to here as critical, $\frac{\partial n}{\partial t}$ is zero. If the equilibrium state does not exist, an assumption, such as

$$\phi(r,t) \equiv \phi_o(r)e^{w_i t}$$

permits a solution of the "steady state" condition for a given w_i.

1. Samuel Glasstone and Milton C. Edlund, The Elements of Nuclear Reactor Theory, D. Van Nostrand Company, Inc., New York, 1952, Fourth Print.

Unfortunately, a true solution is severely complicated by the process of delay neutron emission. However, the assumption that all neutrons emitted in the fission process are prompt, ($w_i \rightarrow w_{prompt}$), does permit a ready solution of the above type.

Using the equation for the conservation of neutrons given above, four differential diffusion equations may be obtained for a two energy group, two region problem.[2] These may be solved upon the application of the usual neutronic boundary conditions, i.e., the neutron fluxes must be finite and non-negative, there must be continuity of fluxes and current densities at the core-reflector interface, and the fluxes must become zero at the extrpolated boundary of the finite reflector. As a result, four flux equations are obtained, which represent the solutions of the differential equations and involve four unknown constants. For a consistent set of solutions, the determinant of the coefficients must be zero.

Paracantor I determines each element of the fourth order determinant from various stored constants and input data, which includes one predesignated variable parameter, such as the core radius. This parameter is varied by a "receipe" until the desired degree of convergence is obtained. The main parameters, from which the unknown variable may be chosen, are:

1. The core radius - R
2. The moderator to fuel atom ration - (M/u)
3. η - contained in $k = \eta \epsilon p f$
4. The cylindrical height - H
5. The reflector thickness - T

The computing time thus depends upon the nearness of the input value of the variable parameter to the value which satisfies the critical equation. In general, 6 to 10 iterations are required to obtain a convergence of the parameter to an accuracy of $\approx$ ¼%. The average machine running time, for normal production, is 5 - 8 minutes per problem. The output, which includes volumes, masses and all of the functions necessary to compute fluxes, is punched out on cards. The computer is then left in readiness for the succeeding problem, which is automatically read in and solved.

---

2. Ibid - p. 240

B. Equations

No attempt will be made to justify or prove the numerous assumptions and conditions necessary to solve the two group problem outlined below. Discussions of Fermi age theory, conditions for separability of time and space variables, treatments which consider delayed neutrons, and other concepts involved in neutron diffusion theory may be found in such references as Glasstone and Edlund.

The two group, two region diffusion equations used in Paracantor I may be written as

1) $$\nabla^2\phi_{1c} - \left(\frac{\Sigma_{1c}}{D_{1c}} + \frac{\Sigma_{1a}}{D_{1c}} + \frac{W_p}{v_1 D_{1c}}\right)\phi_{1c} + \frac{k}{L^2}\cdot\frac{D_{2c}}{D_{1c}}\phi_{2c} = 0$$

2) $$\nabla^2\phi_{2c} - \left(\frac{\Sigma_{2c}}{D_{2c}} + \frac{W_p}{v_2 D_{2c}}\right)\phi_{2c} + \frac{\Sigma_{1c}}{D_{2c}}\phi_{1c} = 0$$

3) $$\nabla^2\phi_{1r} - \left(\frac{\Sigma_{1r}}{D_{1r}} + \frac{W_p}{v_1 D_{1r}}\right)\phi_{1r} = 0$$

4) $$\nabla^2\phi_{2r} - \left(\frac{\Sigma_{2r}}{D_{2r}} + \frac{W_p}{v_2 D_{2r}}\right)\phi_{2r} + \frac{\Sigma_{1r}}{D_{2r}}\phi_{1r} = 0$$

where equations 1 and 2 refer to the core (c) and equations 3 and 4 refer to the reflector (r). The subscripts 1 and 2 refer to fast and slow neutrons respectively.

In the above equations:

ϕ_i is the neutron flux $= nv_i$ where n is the neutron density
v_i is the velocity assigned to the ith group

D represents the diffusion coefficients

$\Sigma_{1/D_1} \equiv 1/\tau$ where τ is the Fermi age length

Σ_{1a} is the macroscopic asorption cross section in the fast core equation. It is to be noted that this term does not appear in many two group presentations.

L^2 is defined on page 8.

Σ_{2c} is the <u>total</u> asorption cross section in the slow group of core. Σ_{2c} includes both moderator absorption and fuel absorption.

Σ_{2r} is the <u>total</u> absorption cross section in the slow group of the reflector.

The terms involving w_p come from the assumption $\phi(r,t) = \phi_o(r)e^{w_p t}$, where w_p may be considered the inverse reactor period with all neutrons assumed to be prompt. Thus,

$$\frac{\partial n}{\partial t} = \frac{1}{v}\frac{\partial \phi(r,t)}{\partial t} = \frac{w_p}{v}\ \phi(r,t)$$

$k = \eta \epsilon p f$ This is sometimes referred to as the four factor formula. In Paracantor I both the fast fission factor, ϵ, and the resonance escape probability, p, are assumed to be unity. Therefore, k is computed as the product,

$$\eta \times f$$

where f is the thermal utilization

$$f = \frac{\text{Thermal neutrons absorbed in fuel}}{\text{Total thermal neutrons absorbed}}$$

and $\eta = \dfrac{\sigma_f}{\sigma_f + \sigma_c}\ \nu$

ν = number of neutrons per fission (2.5)

σ_f = thermal fission cross section for fuel

σ_c = thermal radiative capture cross section for fuel

The factor η is considered a constant in any one Paracantor problem.

The condition necessary for solutions of the core equations (1 and 2) in the form

$$\nabla^2 \phi_c = -B^2 \phi_c$$

may be shown to be,

Eqn. 5
$$\frac{k}{(1 + \tau_c B^2 + \gamma)(1 + L^2 B^2)} = 1$$

where $\gamma = \dfrac{\Sigma_{1a}}{\Sigma_{1c}} + \dfrac{w_p c}{v_1 D_{1c}}$

Thus, the flux solution in the core involves a linear combination of the two roots of the above equation. For a cylinder the permissible solutions are

$$\phi_{1c} = AX + CY$$

$$\phi_{2c} = S_1 AX + S_2 CY$$

where

$$X = J_o(\mu r)$$
$$Y = I_o(\nu r) \quad \text{and}$$

J_o and I_o are normal and modified Bessel functions, respectively, of the first kind and zero order. (See Appendix A)

μ and ν, are the two roots of Equation 5 modified for a cylinder of finite length

$$\mu = \left\{\frac{1}{2}\left[-\left(\frac{1}{\tau_c}+\frac{1}{L^2}+\frac{\gamma}{\tau_c}\right)+\sqrt{\left(\frac{1}{\tau_c}+\frac{1}{L^2}+\frac{\gamma}{\tau_c}\right)^2+\frac{4(k-1-\gamma)}{\tau_c L^2}}\right]-\left(\frac{\pi}{H'}\right)^2\right\}^{\frac{1}{2}}$$

$$\nu = \left\{\frac{1}{2}\left[\left(\frac{1}{\tau_c}+\frac{1}{L^2}+\frac{\gamma}{\tau_c}\right)+\sqrt{\left(\frac{1}{\tau_c}+\frac{1}{L^2}+\frac{\gamma}{\tau_c}\right)^2+\frac{4(k-1-\gamma)}{\tau_c L^2}}\right]+\left(\frac{\pi}{H'}\right)^2\right\}^{1/2}$$

where H' = the actual height plus the extrapolated height

S_1 and S_2 are the coupling coefficients.

In a similar manner, the solutions to the reflector equations (3 and 4), may be written as

$$\phi_{1r} = FZ_1$$
$$\phi_{2r} = GZ_2 + S_3\phi$$

where

$$Z_1 = I_o(K_{1r}r) - \frac{I_o}{K_o}\left[K_{1r}(R+T')\right] K_o(K_{1r}r)$$

K_o is a modified Bessel function of the second kind, zero order. (See Appendix A)

T' = the actual thickness plus the extrapolated distance.

$$K_{1r} = \left[\frac{1}{\tau_r}+\frac{\omega\rho}{D_{1r}V_1}+\left(\frac{\pi}{H'}\right)^2\right]^{1/2}$$

Z_2 is a function in K_{2r} corresponding to Z_1.

$$K_{2r} = \left[\frac{\Sigma_{2r}}{D_{2r}}+\frac{\omega\rho}{D_{2r}V_2}+\left(\frac{\pi}{H'}\right)^2\right]^{1/2}$$

S_3 is the reflector coupling coefficient

The application of the boundary conditions that the fluxes and current densities be continuous at R, the core-reflector interface, leads to four equations in A, C, F, and G, the unknown constants. For a consistent set of solutions, the determinant, Δ, of the coefficients must

vanish; that is,

$$\begin{vmatrix} X & Y & -Z_1 & 0 \\ S_1X & S_2Y & -S_3Z_1 & -Z_2 \\ D_{1c}X' & D_{1c}Y' & -D_{1r}Z_1' & 0 \\ S_1D_{2c}X' & S_2D_{2c}Y' & -S_3D_{2r}Z_1' & -D_{2r}Z_2' \end{vmatrix} = \Delta \to 0$$

where X', Y', Z_1', and Z_2' are the first derivatives of X, Y, Z_1, and Z_2, respectively, all evaluated at the core-reflector interface, (w.r.t.) r. This is the critical equation for a two group, two region reactor. An expanded form of the determinant is used in Paracantor I.

Equations Used in Paracantor I Code

A. Input Data

The normal form of the input data used in Paracantor I may not be convenient for all uses. Various constants, which are normally stored in main program, can be changed by insertion of what may be described as optional input data. A sample input is included with the Test Problem in Appendix C.

In general, the input data may be broken into four groups of information: 1) moderator data, 2) reflector data, 3) fuel data and 4) geometry constants.

The moderator and reflector materials are specified by six constants:

$$\frac{1}{\tau} \quad \frac{\Sigma_2}{D_2} \quad D_1, D_2, \rho_n, \rho/\rho_n$$

The first four are always given for normal material density, (ρ_n). Paracantor I adjusts these material constants for any density variation desired by using the ratio of the desired density to normal density, (ρ/ρ_n). The above constants adjusted for density are:

$$\frac{1}{\tau} = \left(\frac{1}{\tau}\right)_n \left(\rho/\rho_n\right)^2$$

$$\frac{\Sigma_2}{D_2} = \left(\frac{\Sigma_2}{D_2}\right)_n \left(\rho/\rho_n\right)^2$$

$$D_1 = (D_1)_n \left(\frac{\rho_n}{\rho}\right)$$

$$D_2 = (D_2)_n \left(\frac{\rho_n}{\rho}\right)$$

$$\rho = \rho_n \left(\frac{\rho}{\rho_n}\right)$$

where n denotes the value of the constant at normal material density.

The fuel is pecified by the moderator/fuel atom ration, (M/u). Optional input permits changes in η, Σ absorption for fuel, and c, the fuel concentration. These are normally constants or may be computed by relations involving constants germane to a specific moderator.

$$\eta = \nu \frac{\sigma_f}{\sigma_f + \sigma_c}$$

$$\Sigma_a^f = \frac{A(\%\rho_n)}{M/u}$$

$$c = \frac{B(\%\rho_n)}{M/u}$$

The value of η and the constants A and B must be adjusted to give the desired η, Σ_a^f and c values for a given problem.

The geometry constants are given by R, T, and H, with all values given in the actual dimensions. The extrapolated end points, which are assumed to be identical for the fact and slow groups, are computed by the following relationships

$$T' = T + \Delta T \qquad \text{where } \Delta T = 2.13\, D_{2r}$$

$$H' = H + 2\Delta H \qquad \text{where } \Delta H = 2.13\, D_{2c}$$

The 2.13 constant may be changed by an optional input constant.

The core fast group absorption term expressed as $\frac{\Sigma_{1a}}{\Sigma_{1c}}$, is normally set equal to zero. However, any desired value may be specified in the input.

It is to be noted that if new optional input is desired to run a specific problem, the memory locations of such data remain occupied with the information from the preceding problem until it is restored to normal value or the program is re-read into the memory. Input constants which are set equal to zero should be so loaded with the problem input.

The basic input is, therefore:

$$M/u,\ H,\ T,\ R,\ 1/t,\ \frac{\Sigma_2}{D_2},\ D_1,\ D_2,\ \rho_n,\ \%\rho_n$$

Typical optional input would be:

$\omega_p, \frac{\Sigma_{1a}}{\Sigma_{1c}}, \eta, \nu_1, \nu_2$ A, B, and the extrapolation constant

Intermediately Computed Constants

The following constants are computed from the input data:

$$L^2 = \left(\frac{1}{\frac{\Sigma_{2c}}{D_{2c}} + \frac{\omega_p}{D_{2c}V_2}}\right) \div \left[1 + \frac{\Sigma_a^f}{D_{2c}}\left(\frac{1}{\frac{\Sigma_{2c}}{D_{2c}} + \frac{\omega_p}{D_{2c}V_2}}\right)\right]$$

$$K_{1r} = \left[\frac{1}{\tau_r} + \frac{\omega_p}{D_{1r}V_1} + \left(\frac{\pi}{H'}\right)^2\right]^{1/2}$$

$$K_{2r} = \left[\frac{\Sigma_{2r}}{D_{2r}} + \frac{\omega_p}{D_{2r}V_2} + \left(\frac{\pi}{H'}\right)^2\right]^{1/2}$$

$$k = \eta\left(\frac{\Sigma_a^f}{\Sigma_a^f + D_{2c}\left[\frac{\Sigma_{2c}}{D_{2c}} + \frac{\omega_p}{D_{2c}V_2}\right]}\right)$$

Solution of the Critical Equation

The critical equation for a two group, two region reactor is given above, (page 6). An expanded form of the determinant is used in Paracantor I:[3] that is,

$$\Delta = \left(D_{1c}\frac{X'}{X} - D_{1r}\frac{Z_1'}{Z_1}\right)\left(S_2D_{2c}\frac{Y'}{Y} - S_2D_{2r}\frac{Z_1'}{Z_1} - (S_2 - S_3)D_{2r}\frac{Z_2'}{Z_2}\right)$$

$$- \left(D_{1c}\frac{Y'}{Y} - D_{1r}\frac{Z_1'}{Z_1}\right)\left(S_1D_{2c}\frac{X'}{X} - S_3D_{2r}\frac{Z_1'}{Z_1} - (S_1 - S_3)D_{2r}\frac{Z_2'}{Z_2}\right)$$

For a solution of the above equation it is necessary to compute the following quantities:

$$Z_1 = I_o(a_1) - \frac{I_o(t_1)}{K_o(t_1)}K_o(a_1)$$

$$Z_1' = K_{1r}\left[I_1(a_1) + \frac{I_o(t_1)}{K_o(t_1)}K_1(a_1)\right]$$

---

3. Ibid p. 246

$$Z_2 = I_o(a_2) - \frac{I_o(t_2)}{K_o(t_2)} K_o(a_2)$$

$$Z_2' = K_{2r}\left[I_1(a_2) + \frac{I_o(t_2)}{K_o(t_2)} K_1(a_2)\right]$$

where

$$a_1 = K_{1r} \times R \qquad t_1 = K_{1r}(R + T')$$
$$a_2 = K_{2r} \times R \qquad t_2 = K_{2r}(R + T')$$

If

$$a = 1$$
$$b = \left(\frac{1}{\tau c} + \frac{1}{L^2} + \frac{\gamma}{\tau c}\right)$$
$$-c = \frac{k-1-\gamma}{\tau_c L^2}$$

then the condition

$$\frac{k}{(1 + \tau_c B^2 + \gamma)(1 + L^2 B^2)} = 1$$

leads to

$$\mu_{H=\infty} = \left[1/2(-b + \sqrt{b^2 - 4ac}\right]1/2$$
$$\nu_{H=\infty} = \left[1/2\ (b + \sqrt{b^2 - 4ac})\right]^{1/2}$$

which, when modified for a finite cylinder, become

$$\mu = \left[\mu^2_{H=\infty} - (\frac{\pi}{H'})^2\right]1/2$$

$$\nu = \left[\nu^2_{H=\infty} + (\frac{\pi}{H'})^2\right]1/2$$

if

$$\mu^2 > 0$$
$$X = J_o(\mu R) \qquad Y = J_o(\nu R)$$
$$X' = -\mu J_1(\mu R) \qquad Y' = \nu I_1(\nu R)$$

If

$$\mu^2 < 0$$
$$X = I_o(\sqrt{|\mu^2|} \times R)$$
$$X' = \sqrt{|\mu^2|}\, I_1\ (\sqrt{|\mu^2|} \times R)$$

The coupling coefficients, S_1, S_2, and S_3, are computed by the following equations:

$$S_1 = \frac{1}{\tau_c} \frac{D_{1c}}{D_{2c}} \left(\frac{1}{\frac{1}{L^2} + \mu_{H=0}^2} \right)$$

$$S_2 = \frac{1}{\tau_c} \frac{D_{1c}}{D_{2c}} \left(\frac{1}{\frac{1}{L^2} - \nu_{H=0}^2} \right)$$

$$S_3 = \frac{1}{\tau_r} \frac{D_{1r}}{D_{2r}} \left(\frac{1}{K_{2r}^2 - K_{1r}^2} \right)$$

The procedure used to evaluate the determinant, or critical equation, is given below for the variable parameter R. The same procedure would be followed if one of the other variable parameters was used.

In the R iteration version, the determinant is evaluated for an initial guess of the critical R, an input number. Depending upon the sign of Δ, the R used is multiplied by an appropriate constant, such as, 1.1 or .75, and this new R is used in the next evaluation of Δ. This procedure is continued until an opposite sign for Δ is obtained. The two R values which bracket the critical R, that is the R for which $\Delta \to 0$, are used in an averaging scheme until the difference between two choices of R, resulting in an opposite sign of Δ, are less than a stored constant. At the conclusion of each iteration a single card is punched with values of μ, (the variable) and Δ.

The general relationship between the value of Δ and the variables R and M/u is given in Fig. I.

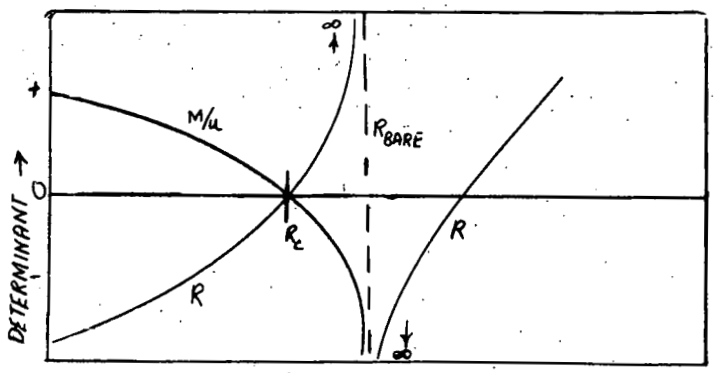


Figure I

As can be seen, other roots exist which are not normally of physical interest. Some indication of the value of the variable parameter must, therefore, be obtained from some other source. In the case of R, $\Delta \rightarrow \pm\infty$ in the region of the radius of the equivalent bare core and other roots exist for $R \geq R_{Bare}$. Therefore, the initial guess for R must be less than R_{Bare} and the constants used to modify the initial R must be sufficiently small to avoid "jumping" to the wrong branch. A solution obtained for one of the other roots may usually be easily detected since $\mu R \gg 2.405$ and thus X may have a negative sign. The $\mu^2 < 0$ loop allows a somewhat greater latitude for the M/u iterations. However, other roots may be obtained, usually for $\mu^2 < 0$. It is to be noted that $-\mu$ on the Paracantor I printout shows as $\mu^2 < 0$.

Output or Edit

When the necessary convergence of the variable parameter is reached, Paracantor I then computes and punches the rest of the output. If output is desired before convergence is reached, manual transfer to memory cell (1688) will force edit, and all resets necessary for solution of a new problem are performed.

Included in the output is the critical parameter together with values of the various functions, all evaluated at the critical parameter. In addition, the volume of the core and reflector and the masses of the fuel, moderator, and reflector are computed and punched. Also included in the output are two modified albedo :

$$\beta_1 = \frac{1 + 2D_{1r}\frac{Z_1'}{Z_1}}{1 - 2D_{1r}\frac{Z_1'}{Z_1}}$$

$$\beta_2 = \frac{1 + 2D_{2r}\frac{Z_2'}{Z_2}}{1 - 2D_{2r}\frac{Z_2'}{Z_2}}$$

and the adjoint coupling coefficients:

$$S_1^* = \frac{1}{S_2}\left(-\frac{D_{1c}}{D_{2c}}\right)$$

$$S_2^* = \frac{1}{S_1}\left(-\frac{D_{1c}}{D_{2c}}\right)$$

$$S_3^* = \left(-\frac{D_{2r}}{D_{1r}}\right) S_3$$

$K_1(x)$ and $I_1(x)$ are also evaluated for $x = K_{1r}(R + T')$ and $x = K_{2r}(R + T')$. These would be of interest if it was desired to evalute integrals such as $\int_{VOL} \phi \, dVOL$.

A sample output is included with the Test Problem in Appendix C.

After the output has been punched, all counters and instructions are reset to the original form and the computer is automatically transferred to the load routine to read the input for the following problem.

Operation of the Paracantor Code

The basic deck, for both Paracantor I and Paracantor II, contains all of the load, punch and interpretative routines, tables, miscellaneous constants, and the main program necessary for the running of the codes. Also included are all of the necessary resets and constants which are needed to automatically restore both programs to the initial computing conditions. A single card, which precedes the deck, will clear the memory, if it is desired. However, this is not necessary for operation of the codes.

The basic Paracantor I deck iterates on the moderator to fuel atom ratio, (M/u). Short subdecks are available which convert this deck to iterate on one of the other variables, R, η, T or H. These subdecks should immediately follow the main deck and precede the input cards. (Figure 2) The variable to be iterated upon may be changed at any time, after the main deck has been read into the memory, by inserting the desired subdeck before the particular problem input. There is also a subdeck available to convert back to the basic M/u version. The coding for each subdeck version is given in Appendix B.

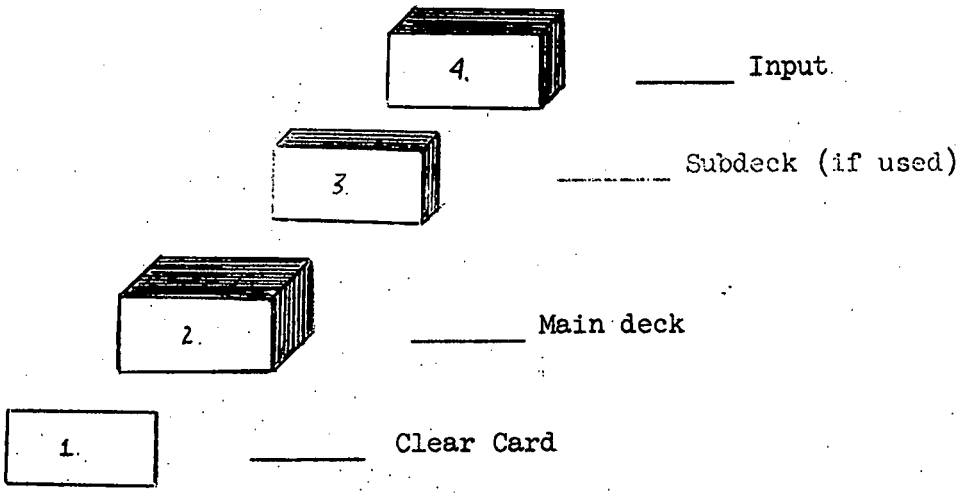


Fig. 2. Arrangement of Cards in Paracantor I

The control console switch settings for both Paracantor I and Paracantor II are:

1. Programmed switch is in the STOP position. Programming will be stopped whenever a stop code (01) is encountered.
2. Half-cycle switch is in the RUN position. The program will proceed automatically once it is started.
3. Control switch is in the RUN position. The program will proceed automatically once it is started.
4. Display switch is in other than READ-OUT STORAGE or READ-IN STORAGE, since the machine will not operate with the control switch in the RUN position if the display switch is in either of these two positions.
5. Overflow switch is set to SENSE. The overflow will be indicated but the machine will not stop except on quotient overflow.
6. Error switch is set to STOP. The machine will stop under any of the following conditions:
   - A. Validity error
     - a. Program register
     - b. Accumulator
     - c. Distributor
   - B. Clocking (timing) error.

Two methods may be used to start the Paracantor programs. (See Appendix F)

I. If the memory is not cleard:

a. Set 70 1851 1851 on the storage entry switches.

b. Depress the program or computer-reset key.

c. Depress the program-start key.

d. The read light on both the console and read punch unit will be on. Press the read key of the read punch unit to read entire code into the memory.

II. If the memory is to be cleared:

a. Set 70 0001 0003 in the storage entry switches and proceed as outlined above.

The memory locations of the constants and other pertinent data for Paracantor I are given in Appendix B. The convergence criteria constant, (1684), may be changed to give the desired convergence for any problem. The constants, (1621 and 1622), which determine the rate of change of the variable being iterated upon may also be changed. However, the relation of these constants to unity must be preserved, i.e., the increase constant must remain greater than unity and the decrease constant must remain less than unity. Manual transfer to 1688 will force edit, including all necessary resets, if output is desired before convergence is reached.

It is to be noted that Paracantor does not contain a punch routine. Therefore, to obtain a memory dump, a punch routine, (SLPR), must be read into the memory, thus destroying a small portion of the code, (1950-2000).

A standard 80-80 read and punch board is used. The UCRL board has been modified to permit suppression of any memory cells in the punch band by the insertion of 8's in the corresponding positions in the last memory cell of the punch band. For example, to suppress the third memory cell, an 8 would be inserted in the third position of the last memory cell in the punch band.

In all coding and in the sample printouts which are included the minus sign always follows the number.

The following are some of the difficulties which may be encountered in the operation of the Paracantor I code:

1. e^x for $x > 115$ cannot be correctly evaluated by the 650 Floating Point routine, since an exponent greater than 99 is encountered. The quantity $K_{ir}(R + T')$ is most likely to be affected, thus limiting the thickness of the reflector usable in Paracantor I.
2. The two common causes of quotient overflow are:
   a. v_1 and v_2 are not loaded. These must NOT be zero.
   b. The square root of a negative number is attempted.
3. A program stop will be encountered if $K_{ir}r < .22$.

PARACANTOR II

The limited memory space available in the I.B.M. 650 does not permit flux solutions to be included in the Paracantor I code. A second code, Paracantor II, has, therefore, been programmed which computes the fluxes, including the adjoint fluxes, from the output of Paracantor I. The solution of the "critical" reactor by Paracantor I provides the critical parameters, which may be shown to apply to the solution of the adjoint "critical" conditions.

The normal two group, two region flux equations are: (See p. 4 & 5)

$$\phi_{1c} = AJ_0(\mu r) + CI_0(\nu r)$$
$$\phi_{2c} = S_1 AJ_0(\mu r) + S_2 C I_0(\nu r)$$
$$\phi_{1r} = F\left\{I_0(K_{1r}r) - \frac{I_0}{K_0}\left[K_{1r}(R_c+T')\right]K_0(K_{1r}r)\right\}$$
$$\phi_{2r} = S_3\phi_{1r} + G\left\{I_0(K_{2r}r) - \frac{I_0}{K_0}\left[K_{2r}(R_c+T')\right]K_0(K_{2r}r)\right\}$$

The adjoint flux, represented by ϕ^*, is obtained by interchanging the rows and columns of the matrix operator and solving for a new flux. The calculation of the adjoint fluxes is of particular value in the application of perturbation theory.* The adjoint equations may be written:

$$\nabla^2\phi_{1c}^* - \left(\frac{\Sigma_{1c}}{D_{1c}} + \frac{\Sigma_{1a}}{D_{1c}} + \frac{\omega\rho}{V_1 D_{1c}}\right)\phi_{1c}^* + \frac{\Sigma_{1c}}{D_{1c}}\phi_{2c}^* = 0$$
$$\nabla^2\phi_{2c}^* - \left(\frac{\Sigma_{2c}}{D_{2c}} + \frac{\omega\rho}{V_2 D_{2c}}\right)\phi_{2c}^* + \frac{k}{L^2}\phi_{1c}^* = 0$$
$$\nabla^2\phi_{1r}^* - \left(\frac{\Sigma_{1r}}{D_{1r}} + \frac{\omega\rho}{V_1 D_{1r}}\right)\phi_{1r}^* + \frac{\Sigma_{1r}}{D_{1r}}\phi_{2r}^* = 0$$
$$\nabla^2\phi_{2r}^* - \left(\frac{\Sigma_{2r}}{D_{2r}} + \frac{\omega\rho}{V_2 D_{2r}}\right)\phi_{2r}^* = 0$$

where solutions may be expressed as follows:

$$\phi_{1c}^* = a^* J_0(\mu r) + b^* I_0(\nu r)$$
$$\phi_{2c}^* = S_1^* a^* J_0(\mu r) + S_2^* b^* I_0(\nu r)$$
$$\phi_{1r}^* = g^*\left\{I_0(K_{1r}r) - \frac{I_0}{K_0}\left[K_{1r}(R+T')\right]K_0(K_{1r}r)\right\} + S_3^*\phi_{2r}^*$$
$$\phi_{2r}^* = f^*\left\{I_0(K_{2r}r) - \frac{I_0}{K_0}\left[K_{2r}(R+T')\right]K_0(K_{2r}r)\right\}$$

A relative plot of the normal and adjoint flux equations may be obtained simultaneously and in a parallel manner. In Paracantor II an arbitrary choice

* Glasstone & Edlund, p.372

of one constant is made, and the other constants are determined from the boundary conditions. The normalizing factors, N and N*, are obtained such that

$$\phi_{1c} = \phi_{1c}^{*} = 1.0 \quad \text{AT} \quad R = 0$$

The equations which are used in the code are, therefore:

$$A = \bar{A}N \qquad a^{*} = \bar{a}^{*}N^{*}$$

$$C = \bar{C}N \qquad b^{*} = \bar{b}^{*}N^{*}$$

$$F = \bar{F}N \qquad f^{*} = \bar{f}^{*}N^{*}$$

$$G = \bar{G}N \qquad g^{*} = \bar{g}^{*}N^{*}$$

assuming $\bar{F} = \frac{1}{Z_1}$ AND $\bar{f}^{*} = \frac{1}{Z_2}$

then $\bar{A} = \frac{1}{X}(1 - \bar{C}Y)$ $\qquad \bar{a}^{*} = \frac{1 - S_2^{*}\bar{C}Y}{S_1^{*}X}$

$$\bar{C} = \frac{\frac{D_{1r}}{D_{1c}}\frac{Z_1'}{Z_1} - \frac{X'}{X}}{Y' - \frac{X'}{X}Y} \qquad \bar{b}^{*} = \frac{\frac{D_{2r}}{D_{2c}}\frac{Z_2'}{Z_2} - \frac{X'}{X}}{S_2^{*}\left(Y' - \frac{X'}{X}Y\right)}$$

$$G = \frac{1}{Z_2}\left[S_1\bar{A}X + S_2\bar{C}Y - S_3\right] \qquad g^{*} = \frac{1}{Z_1}\left[\bar{a}^{*}X + \bar{b}^{*}Y - S_3^{*}\right]$$

and the normalizing constants are

$$N = \frac{1}{\bar{A} + \bar{C}} = \frac{X}{1 + C(X - Y)} \qquad N^{*} = \frac{S_1^{*}X}{1 + b^{*}(S_1^{*}X - S_2^{*}Y)}$$

The four fluxes ϕ_1, ϕ_2, ϕ_1^{*}, and ϕ_2^{*} and the cross products $\phi_1\phi_1^{*}$, $\phi_2\phi_2^{*}$, $\phi_1\phi_2^{*}$ and $\phi_2\phi_1^{*}$ are computed for values of the raduis, which may be specified as input in the following manner.

1. The flux for r = 0 is determined without the use of series, since X(0) = 1.
2. A Δr_c, an input number, is used to evaluate the flux at r_n.

   $$r_n = n(\Delta r_c) \qquad n = 1, 2, 3, \ldots.$$

   The varying r_n is tested against R_c, until the next r_n would be greater than R_c.
3. The flux is evaluated at R_c using the core equations.
4. A second input number specifies the first $r_n > R_c$ at which the fluxes are to be evaluated.
5. A Δr_R, an input number, determines the spacing in the reflector in the same manner as in the core. The variable, r, is tested

against R + T' and the last flux is determined at R + T'. This value should be zero or approach zero and, thus, provides a check on the calculation.

The fluxes for a given r are punched as soon as they are computed. The Bessel Functions are determined in the same manner as in Paracantor I. (See Appendix A).

Operation of the Paracantor II Code

Paracantor II is a completely self-contained and automatic code. Once the main program has been read into the memory it is not necessary to reload it in order to run as many problems as may be desired. The running time of the code is directly proportional to the number of flux points required, which is determined by the first card of the input. A typical problem, determining ten fluxes in both the core and the reflector, takes about five minutes.

The console switch settings for Paracantor II are the same as for Paracantor I. (p.13). A memory clear card again is optional and is not necessary for operation of the code. The code contains no programmed stop, since K_{ir} is guaranteed to be greater than 0.22 by a Paracantor I solution. If output is desired before the completion of a problem, manual transfer to memory location 1365 will compute the fluxes at R + T' and reset the program for the next problem.

The test problem which is given for Paracantor II, (Appendix E), is a continuation of the same problem used in Paracantor I. No flow diagram is included for Paracantor II, since the calculational procedures follow Paracantor I in detail. The memory locations of the constants and other pertinent data is given in Appendix D.

The input for Paracantor II is in floating point form and consists of seven unaddressed cards. The first card of the input contains Δr_c, R_{1R} (the first r in the reflector for which fluxes are desired), and Δr_R. These quantities occupy locations 1-30 on the I.B.M. card. The next six cards are the last six cards from the Paracantor I output. These can be distinguished by the numbers 1-6 in the 71-80 positions of the cards. These must be in numerical order, since there is no check on the order being correct once the cards are read in. The last card in the main program is a transfer card (to 0646) to those commands which read the input into the memory. Upon the completion of a problem the code automatically transfers to 0646, therefore, no transfer card (s) are needed with any input. However, the transfer card in the main program must be the last card of the program.

The form of the input is:

Δr_c	R_{1R}	Δr_R					
M/u	T	H	w_p	S_1^*	S_2^*	S_3^*	1.0
R_c	S_1	S_2	S_3	μ	ν	V_c	2.0
K_{1r}	K_{2r}	Z_1'/Z_1	Z_2'/Z_2	M^f	M^m	V_R	3.0
M^R	β_1	β_2	η	D_{1r}	D_{2r}	D_{1c}	4.0
Z_1	Z_2	X	X'/X	Y	Y'/Y	$\frac{Io}{Ko}(t_1)$	5.0
$\frac{Io}{Ko}(t_2)$	$I_1(t_1)$	$K_1(t_1)$	$I_1(t_2)$	$K_1(t_2)$	$\frac{\Sigma_{1a}}{\Sigma_{1c}}$	D_{2c}	6.0

The output for Paracantor II is also in floating point form. The arrangement of the output is:

R	ϕ_1	ϕ_2	ϕ_1^*	ϕ_2^*
-	$\phi_1\phi_1^*$	$\phi_2\phi_2^*$	$\phi_1\phi_2^*$	$\phi_2\phi_1^*$

The following are given as aids in the operation of the Paracantor II code:

1. If Δr_c or Δr_R is set equal to zero, the program will go into a loop and the problem will not be completed.
2. If $\Delta r_c > R_c$, the flux calculation in the core may be skipped. The program will compute the flux at r = 0 and r = R_c, from core

equations and then proceed in the normal fashion. In a similar manner, if $\Delta r_R > T'$, the program will bypass the reflector points, except for $r = R_{1R}$ and $r = R_c + T'$.

3. If $\Delta r_c = R_c/n$, where n is an integer, $\phi(R_c)$ is evaluated twice using the core equations. Similarly, if $\Delta r_R = T'/n$, a double evaluation of $\phi(R+T')$ is made.

4. If $R_{1R} = R_c$, the fluxes for R_{1R} are calculated by use of the the reflector equations, thus providing a check on the flux boundary conditions.

5. If $R_{1R} < R_c$ or $R_{1R} \geq R_c + T'$, the program automatically replaces the R_{1R} by R_c and proceeds.

6. The fluxes at $R_c + T'$ are frequently identically zero, but occasionally will have absolute values of 10^{-8} to 10^{-12}. This is generally because the $\vec{z}$'s are not zero due to round off in the $\frac{Io}{K_a}\left[K_{ir}\ (R_c + T')\right]$ terms in Paracantor I.

Appendix A

Bessel Functions

The Bessel Functions used in Paracantor I are all evaluated by series with the exception of $K_o(x)$ and $K_1(x)$ for $x < 2$.

In the case of $J_o(x)$ and $J_1(x)$ a single series is used, since the desired value of the argument for solution of the critical equation usually lies below the first zero, $(x < 2.405)$. The series used are:

$$J_o(x) = 1 - \frac{x^2}{(2)^2(1!)^2} + \frac{x^4}{2^4(2!)^2} - \frac{x^6}{2^6(3!)^2} + \frac{x^8}{2^8(4!)^2} - \frac{x^{10}}{2^{10}(5!)^2} + \frac{x^{12}}{2^{12}(6!)^2} - \frac{x^{14}}{2^{14}(7!)^2} \qquad \text{(p.1)}$$

$$J_1(x) = \frac{x}{2} - \frac{x^2}{2^3 1!2!} + \frac{x^5}{2^5 2!3!} - \frac{x^7}{2^7 3!4!} + \frac{x^9}{2^9 4!5!} - \frac{x^{11}}{2^{11} 5!6!} + \frac{x^{13}}{2^{13} 6!7!} - \frac{x^{15}}{2^{15} 7!8!} \qquad \text{(p.1)}$$

Two series are used to evaluate $I_o(x)$ and $I_1(x)$, since the arguments of these Bessel functions may vary widely depending upon the reactor materials used. The series used are, therefore:

A. $x \leq 5$

$$I_o(x) = 1 + \frac{x^2}{2^2(1!)^2} + \frac{x^4}{2^4(2!)^2} + \frac{x^6}{2^6(3!)^2} + \frac{x^8}{2^8(4!)^2} + \frac{x^{10}}{2^{10}(5!)^2} + \frac{x^{12}}{2^{12}(6!)^2} + \frac{x^{14}}{2^{14}(7!)^2} \qquad \text{(p. 213)}$$

$$I_1(x) = \frac{x}{2} + \frac{x^3}{2^3 1!2!} + \frac{x^5}{2^5 2!3!} + \frac{x^7}{2^7 3!4!} + \frac{x^9}{2^9 4!5!} + \frac{x^{11}}{2^{11} 5!6!} + \frac{x^{13}}{2^{13} 6!7!} + \frac{x^{15}}{2^{15} 7!8!} \qquad \text{(p. 213)}$$

British Association for the Advancement of Science, Mathematical Tables Vol. VI, Bessel Functions, Part I, University Press, Cambridge, 1950

B. $x > 5$

The asymptotic series are used, where the last coefficient has been adjusted to give the best fit as $x \to 5$. (1 part in 10^4)

$$I_o(x) = \frac{e^x}{(2\pi x)^{1/2}} \left\{ 1 + \frac{1^2}{1!8x} + \frac{1^2 3^2}{2!(8x)^2} + \frac{1^2 3^2 5^2}{3!(8x)^3} + .15094433 \frac{1}{x^4} \right\}$$

(p.271)

$$I_1(x) = \frac{e^x}{(2\pi x)^{1/2}} \left\{ 1 - \frac{1.3}{1!8x} - \frac{1^2 3.5}{2!(8x)^2} - \frac{1^2 . 3^2 . 5.7}{3!(8x)^3} - .19129591 \frac{1}{x^4} \right\}$$

(p. 271)

$K_o(x)$ and $K_1(x)$ are evaluated by an asymptotic series for arguments greater than 2. The last coefficient has again been adjusted to give a smooth fit as $x \to 2$. (1 part in 10^4)

$$K_o(x) = \left(\frac{\pi}{x}\right)^{1/2} e^{-x} \left\{ 1 - \frac{1^2}{1!8x} + \frac{1^2 . 3^2}{2!(8x)^2} - \frac{1^2 . 3^2 . 5^2}{3!(8x)^3} + .058964527 \frac{1}{x^4} \right.$$

(p. 271)

$$K_1(x) = \left(\frac{\pi}{x}\right)^{1/2} e^{-x} \left\{ 1 + \frac{1.3}{1!8x} - \frac{1^2 . 3.5}{2!(8x)^2} + \frac{1^2 . 3^2 . 5.7}{3!(8x)^3} - .077874253 \frac{1}{x^4} \right\}$$

$K_o(x)$ and $K_1(x)$ for $x < 2$ are evaluated by a table look up plus a seven point LaGrangian interpolation scheme. The smallest argument which can be evaluated is $x = 0.22$. The form is as follows:

$$f_n = f_o + nA + B^{II}C + B^{III}D + B^{IV}E + B^{V}F + B^{VI}G$$

where

f_o is the first tabular value smaller than the f being searched for.

$A = (f_1 - f_o)$

$C = (f_2 - f_1 - f_o + f_{-1})$

$D = (f_2 - 3f_1 + 3f_o - f_{-1})$

$E = (f_3 - 3f_2 + 2f_1 + 2f_o - 3f_{-1} + f_{-2})$

$F = (f_3 - 5f_2 + 10f_1 - 10f_o + 5f_{-1} - f_{-2})$

$G = (f_4 - 5f_3 + 9f_2 - 5f_1 - 5f_o + 9f_{-1} - 5f_{-2} + f_{-3})$

$$B^{II} = \frac{n(n-1)}{4}$$

$$B^{III} = \frac{n(n^2 - \frac{3}{2}n + \frac{1}{2})}{6}$$

$$B^{IV} = \frac{n(n^3 - 2n^2 - n + 2)}{48}$$

$$B^{V} = \frac{n(n^4 - \frac{5}{2}n^3 + \frac{5}{2}n + 1)}{120}$$

$$B^{VI} = \frac{n(n^5 - 3n^4 - 5n^3 + 15n^2 + 4n - 12}{1440}$$

Problem: Paracantor I - Subdecks

Loc. Instr.	Operation		Data Addr.	Instr. Addr.	Remarks
M/u ITERATION					
1658	LD	69	1627	1659	M/u → 1679 for modification
1573		-00	1627	0000	.95 M/u → 1627
1574	BR	-12	0931	0000	Recycle
1666	LD	69	1627	1667	M/u → 1678 for modification
1576		-00	1627	0000	1.05 M/u → 1627
1577	BR	-12	0931	0000	Recycle
1582		-00	1627	0000	$\frac{M/u_i + M/u_j}{2}$ → 1627
1583	BR	-12	0931	0000	Recycle
1621		49	9500	0000	Constant - decreases M/u
1622		50	1050	0000	Constant - increases M/u
1684		50	1000	0000	Convergence criteria
1124	LD	69	1627	1125	Transfers M/u to punch each iteration
R ITERATION (All D and I addresses of 1627 are changed to 0570)					
1621		50	1100	0000	Constant - Increases R
1622		49	7500	0000	Constant - Decreases R
1684		49	2500	0000	Convergence Criteria
η ITERATION (All D and I addresses of 1627 are changed to 0584) The recycle F.D. BR commands are changed to:					
		-12	1165	0000	
1621		50	1050	0000	Constant - Increases η
1622		49	9500	0000	Constant - Decreases η
1684		47	1000	0000	Convergence criteria
H ITERATION (All D and I addresses of 1627 are changed to 0569) The recycle F.D. BR commands are changed to:					
		-12	1163	0000	
The constants for R may be used.					
1163	A-B→C	-06	0569	1153	H' - 2Δ H = H
1164		-00	1629	0000	
1124	LD	69	1629	1125	Transfers H to punch each iteration
T ITERATION (All D and I addresses of 1627 are changed to 0568) The recycle F.D. BR commands are changed to:					
		-12	1163	0000	
The constants for R may be used.					
1163	A-B→C	-06	0568	1152	T' - Δ T = T
1164		-00	1628	0000	
1124	LD	69	1628	1125	Transfers T to punch each iteration

Appendix B

Problem: Paracantor I - Memory Locations of Constants and Pertinent Data

Loc. Instr.	Operation	Data Addr.	Instr. Addr.	Remarks
567				Empty
568				T'
569				H'
570				R (Input)
573				$1/\tau_c$
574				Σ_{2c}/D_{2c}
575				D_{1c}
576				D_{2c} — Moderator Data (Input)
577				ρ_c
578				$1/\tau_r$
579				Σ_{2r}/D_{2r}
580				D_{1r}
581				D_{2r} — Reflector Data (Input)
582				ρ_r
583				Σ_a^{fuel}
584				η (Optional Input)
585				Extrapolation Constant (Optional Input)
586				2.0
587				Zero (all)
588				π
589				K_{1r}
590				K_{2r}
591				1.0
592				L^2
593				5.0
594				0.22 (Lower Argument Limit)
637	24	0029	0026	Σ_{1c}/Σ_{1c}
697				
698	00	0001	0000	
699				Temporary Storage
702				Empty
703				Empty
704				Empty
705				Empty
719				$(\rho/\rho_n)^2$ (Input)

Problem: Paracantor I - Memory Locations of Constants and Pertinent Data

Loc. Instr.	Operation	Data Addr.	Instr. Addr.	Remarks
720				$(\rho/\rho_n)_c$ (Input)
721				v_2 } (Optional Input) - must
722				v_1 } be filled - non-zero
723				Empty
725				$\nu_H = \infty$
726				$\mu_H = \infty$
740				Z_2
741				Y'/Y
742				Y
743				X'/X
744				X
745				$I_o/K_o \left[K_{2r} (R + T')\right]$
746				Z_1
747				$I_o/K_o \left[K_{1r} (R + T')\right]$
748	00	0000	0797	} Constants in K_o, K_1 locator
749	00	0000	0997	} scheme
760	01	0000	0000	Stop
762	00	0001	0000	
763				n - Argument Fraction
764				3.0
765				} Temporary Storage
766				
767				
768	51	1000	0000	} Constants for K_o and K_1 Interpolation Scheme
769	50	9000	0000	
770	52	1200	0000	
771	50	4000	0000	
772	49	5000	0000	
773	50	6000	0000	
774	51	4800	0000	
775	53	1440	0000	
800				} K_o Table
↓				
895				
959	00	0000	0001	
1000				} K_1 Table
↓				
1095				
1120				Empty
1121				Empty
1127				ρ Punch Band used
1128				Variable } in Iteration
1129				Determinant } cycle
1131				Empty
1132				Empty
1133				Empty
899				Empty

Problem: Paracantor I - Memory Locations of Constants and Pertinent Data

Loc. Instr.	Operation	Data Addr.	Instr. Addr.	Remarks
1134				Empty
1135				Empty
1136	00	0888	8888	Punch Band (1127) Field Control
1149				Empty
1150				Empty
1151				Empty
1152				Empty
1153				Empty
1157	00	0000	0007	
1160				c - fuel concentration (gm/cm^3)
1161				Empty
1162				Empty
1216	49	2500	0000	
1217	48	1562	5000	
1218	46	4340	2778	
1219	44	6781	6840	$I_o(x)$ constants $x < 5.0$
1220	42	6781	6840	
1221	40	4709	5028	
1222	38	2402	8075	
1223				Argument Storage
1236	49	5000	0000	
1643	48	6250	0000	
1644	47	2604	1667	
1645	45	5425	3472	$I_1(x)$ constants $x < 5.0$
1646	43	6781	6840	
1647	41	5651	4033	
1648	39	3363	9306	
1649	37	1501	7547	
1255	49	1250	0000	
1256	48	7031	2500	$I_o(x)$ constants $x > 5.0$
1257	48	7324	2187	
1258	49	1509	4433	
1277	49	3750	0000	
1278	49	1171	8750	$I_1(x)$ constants $x > 5.0$
1279	49	1025	3906	
1280	49	1912	9591	
1281	08	0000	0000	Table
1282	00	0000	0050	Constants
1379	49	1250	0000	
1380	48	7031	2500	$K_o(x)$ constants $x > 2.0$
1381	48	7324	2187	
1382	48	5896	4527	

Problem: Paracantor I - Memory Locations of Constants and Pertinent Data

Loc. Instr.	Operation	Data Addr.	Instr. Addr.	Remarks
1406	49	3750	0000	$K_1(x)$ constants $x > 2.0$
1407	49	1171	8750	
1408	49	1025	3906	
1409	48	7787	4253	
1417				A - constants $\Sigma_a^F = \frac{A(\rho/\rho_n)^{m/\mu}}{m/\mu}$
1429				B - constant $c = \frac{B(\rho/\rho_n)^{m/\mu}}{m/\mu}$
1482	49	2500	0000	$J_0(x)$ constants
1483	48	1562	5000	
1484	46	4340	2778	
1485	44	6781	6840	
1486	42	6781	6840	
1487	40	4709	5028	
1488	38	2402	8075	
1500	49	5000	0000	$J_1(x)$ constants
1501	48	6250	0000	
1502	47	2604	1667	
1503	45	5425	3472	
1504	43	6781	6840	
1505	41	5651	4033	
1506	39	3363	9306	
1507	37	1501	7547	
1584				Empty
1621				Multiplication constants in
1622				new variable selection
1627				Edit Punch Band
↓ 1634				
1639				Working Storage
↓ 1642				
1675				a counter
1676				b counter
1677	00	0000	0002	
1678				(Variable)a
1679				(Variable)b
1684				Convergence Criteria Constant
1951				Temporary Working Storage
↓ 1960				
1979				Empty
1980				Short Load Routine
↓ 1999				

PARACANTOR I - TEST PROBLEM

The test problem which follows uses water as the moderator, U^{235} as the fuel, and thick graphite as the reflector. R, the core radius, is the variable parameter. A basic deck, a subdeck, and the input deck are, therefore, assembled in order.

Input

All numbers in the input are in floating point form. The order is immaterial except for the transfer card, (to 0026), which must always be the last card of the input. The input constants $\frac{1}{\tau}$, $\frac{\Sigma_2}{D_2}$, D_1, and D_2 are given for normal density. All of the constants used are from Glasstone and Edlund.

A. Basic Input

First Card: M/u, T, H, w_p

Second Card: R initial

Third Card: $1/\tau_a$, Σ_{2c}/D_{2c}, D_{1c}, D_{2c}, $(\rho_n)c$

Fourth Card: $1/\tau$, Σ_{2r}/D_{2r}, D_{1r}, D_{2r}, $(\rho_n)r$

Fifth Card: $(\rho/\rho_n)_r$, $(\rho/\rho_n)_c$

B. Optional Input

Sixth Card: η

Seventh Card: v_1, v_2

Eighth Card: Σ_{1a}/Σ_{1s}

Ninth Card: B c = $\frac{B(\rho/\rho_n)}{M/u}$

Tenth Card: A Σ_a^f = $\frac{A(\rho/\rho_n)}{M/u}$

Last Two Cards

Eleventh Card: Loads the i counter (0029) of the floating point routine.

Twelfth Card: Transfers to the floating point routine.

Output

All numbers in the output are in floating point form. The numbers in the eighth position on each card serve as markers for Paracantor II input. The arrangement of the output is as follows:

M/u	T	H	w_p	S_1^*	S_2^*	S_3^*	1.0
R	S_1	S_2	S_3	μ	ν	v_c	2.0
K_{1r}	K_{2r}	Z_1'/Z_1	Z_2'/Z_2	M^f	M^m	v_r	3.0
M^r	β_1	β_2	η	D_{1r}	D_{2r}	D_{1c}	4.0
Z_1	Z_2	X	X'/X	Y	Y'/Y	$I_o^{(t_1)}/K_o(t_1)$	5.0
$I_o^{(t_2)}/K_o(t_2)$	$I_1(t_1)$	$K_1(t_1)$	$I_1(t_2)$	$K_1(t_2)$	$\frac{\Sigma_{1a}}{\Sigma_{1c}}$	D_{2c}	6.0

where

$$t_1 = K_{1r}(R + T')$$

$$t_2 = K_{2r}(R + T')$$

APPENDIX D

PARACANTOR II - Memory Locations of Constants and Pertinent Data

Location of Instruction	Code	Data	Instruction	Remarks
568				R = 0
569	49	2500	0000	
570	48	1562	5000	
571	46	4340	2778	
572	44	6781	6840	$J_o(x)$ constants
573	42	6781	6840	
574	40	4709	5028	
575	38	2402	8075	
577	49	2500	0000	
578	48	1562	5000	
579	46	4340	2778	
580	44	6781	6840	$I_o(x)$ Constants
581	42	6781	6840	$x < 5.0$
582	40	4709	5028	
583	38	2402	8075	
584	49	1250	0000	
585	48	7031	2500	$I_o(x)$ constants
586	48	7324	2187	$x > 5.0$
587	49	1509	4433	
590				2.0
591				π
592				Extrapolation Constant
594				3.0
596				5.0
597				10.0

Location of Instruction	Code	Data	Instruction	Remarks
598				9.0
599				4.0
609				Empty
610				Empty
611				C
612				A
613				G
614				F
615				$\bar{b}$*
616				$\bar{a}$*
617				$\bar{f}$*
618				$\bar{g}$*
619				b*
620				a*
621				f*
622				g*
627				Initial R=0
628 ↓ 636				Set Punch Band to Zero
650				Working Storage
659				Empty
660				Empty
667				1.0
677				Working Storage
678				Working Storage
709				Empty

Location of Instruction	Code	Data	Instruction	Remarks
710				Empty
727 ↓ 736				Set Punch Band to zero
759				Empty
760				Empty
766				0.5
767				48.0
768				120.0
799	00	0000	897	Constant in K_o, locator scheme
809				Empty
810				Empty
816				6.0
817				1440.0
859				Empty
860				Empty
864				Working Storage
878	49	1250	0000	$K_o(x)$ constants $x > 2.0$
879	48	7031	2500	
880	48	7324	2187	
881	48	5896	4527	
1951 ↓ 1960				Working Storage

UTILITY PROGRAMS USED IN PARACANTOR

I. Setting of Storage Entry Switches.

If the clear card shown below is used, the storage entry switches should be set to

70 0001 0003

If this card is not used, the Storage Entry keys should be set to

70 1851 1851

II. Storage Clear Card.

Description

This card places +0 in locations 0002 through 1999 inclusive, and places the instruction 70 1851 1851 in location 0000. It concludes by executing the instruction 70 1851 1851.

Card

```
0001:  00  0001  0000
0002:  16  0001  8002
0003:  69  0005  0004
0004:  24  0000  0006
0005:  70  1851  1851
0006:  65  0007  8002
0007:  21  1999  0002
0008:  00  0000  0000
```

III. The first card (or the second card, if a clear card is used) is the Initial loader. This is read into locations 1851-1856.

```
1851:  70  1901  1852
1852:  69  1902  1901
1853:  69  1904  1903
1854:  69  1906  1905
1855:  69  1908  1907
1856:  70  1851  1996
```

This card willload cards of the following format:

```
1901  24  L(word 1)  1853
1902  Word 1
```

1903 24 L(word 2) 1854

1904 Word 2

1905 24 L(word 3) 1855

1906 Word 3

1907 24 L(word 4) 1851

1908 Word 4.

The initial loader is used to load the standard load routine, which is loaded, four words per card, in the format shown above. (These cards must not be load cards.).

In the last card of the load routines of Paracantor I, the Instruction Address of word 7 is 1856. This is followed by a card having all 80 columns punched with zeros. This will clear the 1851-1860 read buffer. Control is automatically transferred to the standard load routine for reading the rest of the cards.

In the last card of the "Long Load and Punch Routine" used with Paracantor II, word 7 is 24 1851 1851 and word 8 is 70 1851 1996. This card is followed by a card with eighty zeros, which clears out the read band 1851-1860. Control is then automatically transferred to the "Long Load and Punch Routine" to read in the rest of the program.

IV. Standard Load Routine for Paracantor I.

This program will load n words per card, $1 \leq n \leq 7$, into consecutive locations, the first word being placed in the location punched in column 7 - 10. n is punched in column 6. If n = 0, the card serves to transfer control, the next instruction being taken from the address specified in columns 7 -10. The program is included below.

V. Long Load and Punch Routine LLPR

The Long Load and Punch Routine, used in Paracantor II, is

is loaded four words per card by the initial loader described above. A description of this routine, and the coding for both the load routine and punch routine is included below. The constant in 1981 in the load routine should read 24 0000 1999.

VI. Interpretive Routine

This problem uses the Floating Decimal Interpretive Routine described in IBM Technical Newsletter No. 8, pp. 17-43. This has been modified in that the multiplication routine, page 25 in the newsletter, has been changed so that it will yield a normal zero when one of the factors is zero. A constant, 98 9898 9898 is placed in location zero, so that a transfer of control to location zero due to the omission of cards will stop the machine with a recognizable number in the program register. The coding for the revised multiply routine is included below. An asterisk indicates a change from Technical Newsletter No.8.

VII. Exponential Routine

The interpretive system described above handles floating point arithmetic. This problem also uses the exponential function. In the interpretive system, a command of the form -22 AB will compute e^x, where x is the floating point number in location A, and store e^x in location B. The calculation uses a Chebyshev polynomial. The coding is included below.

IBM TYPE 650 PROGRAM SHEET

PROBLEM: Standard Load Routine for Paracantor I WRITTEN BY:

INSTR NO.	LOCATION OF INSTRUCTION	OPERATION		ADDRESS		REMARKS
		ABBRV.	CODE	DATA	INSTRUCTION	
	1996	RD	70	1951	1994	
199	1994	RAL	65	1951	1981	
	1981	SL	16	1998	1982	
	1982	BRMIN	46	1951	1983	is n = o ? (transfer card)
	1983	SLT	35	0004	1989	No
	1989	AL	15	1951	1984	n + A
	1984	LD	69	1988	1991	
	1991	STDA	22	1988	1992	Last word control
	1992	SLT	35	0004	1999	
	1999	STDA	22	1959	1986	First store command
	1986	RAL	65	1990	1987	First load command
	1987	AU	10	1959	8002	
*	8002	LD	69	(1952+i)	8003	
*	8003	STD	24	(A+i)	1995	
	1995	AU	10	1998	1980	Modify store command
	1980	AL	15	8001	1985	Modify load command
	1985	SU	11	1988	1993	
199	1993	BRNZU	44	1997	1996	End of card:?
	1997	AU	10	8001	8002	
	Constants:					
	1988		24	(0000)	1995	
	1990		69	1952	8003	
	1998		00	0001	0000	
	Note: Instructions marked with (*) do not appear on Drum.					

RL. 1315

PAGE _____ OF _____

IBM TYPE 650 PROGRAM SHEET

PROBLEM: Revised Multiply Routing WRITTEN BY:

INSTR NO.	LOCATION OF INSTRUCTION	OPERATION ABBRV.	OPERATION CODE	ADDRESS DATA	ADDRESS INSTRUCTION	REMARKS
	0134	RAL	65	0037	0141	
	0141	SLT	35	0002	0097	
	0097	STL	20	0151	0106	
	0106	RAU	60	8003	0063	
	0063	SRT	30	0002	0119	
	0119	RAABL	67	8002	0128	
	0128	SL	16	0131	0085	
	0085	AU	10	0089	0093	
	0093	BRMIN	46	0096	0147	
	0096	SU	11	8001	0103	
	0103	SL	16	8001	0161	
	0161	SLT	35	0002	0117	
	0117	AU	10	8002	0075	
	0075	RSU	61	8003	0138	
	0138	SRT	30	0002	0095	
	0147	SU	11	8001	0155	
	0155	AL	15	8001	0113	
	0113	SLT	35	0002	0121	
	0121	AU	10	8002	0115	
	0115	RAU	60	8003	0138	
*	0095	MPY	19	0151	0109	
*	0109	LD	69	8003	0173	
*	0173	SLT	35	0002	0181	
*	0181	SRT	30	0001	0042	
*	0042	SCT	36	0001	0100	
*	0100	SRT	30	0009	0170	
*	0170	BROV2	47	0124	0125	OV if product of form X.XXX XXX
*	0124	SRD	31	0002	0082	Product of form X.XXXXXX
*	0082	BRNZ	45	0086	0083	
*	0086	STDA	22	0089	0047	
*	0047	STIA	23	0052	0163	
*	0163	RAL	65	8001	0083	Answer into lower
	0083	STL	20	0037	0026	Finis
*	0125	SRD	31	0003	0135	Product of form XX.XXXXXX
*	0135	STDA	22	0089	0094	
*	0094	STIA	23	0052	0107	
*	0107	RAL	65	8001	0114	
*	0114	BRMIN	46	0171	0123	
*	0171	SL	16	0126	0083	Adjust exponent
*	0123	AL	15	0126	0083	
Constants:						
	0000		98	9898	9898	
	0126		01	0000	0000	
	Note: An asterisk indicates a change from the program in Technical Newsletter No. 8.					

IBM TYPE 610 PROGRAM SHEET

PROBLEM: LOAD ROUTINE (Modified) for LLPR WRITTEN BY: LASNIK and KEIRSTEAD

INSTR NO	LOCATION OF INSTRUCTION	OPERATION ABBRV.	OPERATION CODE	ADDRESS DATA	ADDRESS INSTRUCTION	REMARKS
	1996	RD	70	1951	1997	Load Entry
	1997	RAU	60	1951	1962	
	1962	SRT	30	0004	1973	
	1973	BRNZU	44	1978	8001	Exit If n = 0
	1978	BRMIN	46	1982	1983	Plus Cards or Minus Cards?
	1982	AU	10	1985	1989	Minus Cards
	1989	BRMIN	46	1993	1994	n = 9 on Minus Card?
	1994	RAABL	67	1951	1986	No |n| → 8002
	1983	SRT	30	0002	1990	Plus Cards
	1990	LD	69	1981	1965	
	1965	STDA	22	1981	1970	Set Address for Plus Card
	1970	SRT	30	0004	1986	Plus or Minus Cards
	1986	AL	15	1992	1972	
	1972	STDA	22	1987	1966	Set Store Loop Count
	1993	SRT	30	0002	1967	n =9 on Minus Cards
	1967	LD	69	1981	1988	
	1988	STDA	22	1981	1996	Set Address for Minus Card
	1966	RAU	60	1992	1968	
	1968	AL	15	1981	8003	Initiate Loop
	*8003	LD	69	[1952]	8002	
	*8002	STD	24	[xxx]	1961	Store Card
	1999	AU	10	1964	1969	
	1969	AL	15	8001	1977	
	1977	STL	20	1981	1984	
	1984	SU	11	1987	1991	Up Loop
	1991	BRNZU	44	1995	1996	Done?
	1995	AU	10	8001	8003	No, Continue Loop
	1985		00	0000	0008	
	1992		69	1952	8002	
	*1987		69	1952	8002	Constants
	1964		00	0001	0000	
	1981		24	0000	1961	
		NOTE:	INSTRUCTIONS MARKED WITH A (*) DO NOT			
			OCCUPY PERMANENT STORAGE POSITIONS			

RL-1315

PAGE _____ OF _____

IBM TYPE 650 PROGRAM SHEET

PROBLEM: $e^A \longrightarrow B$ WRITTEN BY:

Storage locations 0466-0566

INSTR NO.	LOCATION OF INSTRUCTION	OPERATION ABBRV.	OPERATION CODE	ADDRESS DATA	ADDRESS INSTRUCTION	REMARKS
	0022	SLT	35	0002	0503	
	0503	LD	69	0559	0466	
	0466	STDA	22	0478	0550	Set exit to Store Ans in B
	0550	RAABL	67	0037	0558	
	0558	SL	16	0561	0515	
	0515	BRMIN	46	0468	0518	is $\|A\| \geq$ 50 in 10?
	0518	RAU	60	0037	0510	Yes
	0510	BRMIN	46	0478	6005	is A negative? If NOT STOP
	0468	AL	15	0471	0475	$\|A\| \leq$ 50 in 10
	0475	BRMIN	46	0466	0531	is $\|A\| > 10^{-10}$
	0531	RAL	65	0037	00557	Yes
	0557	SLT	35	0002	0513	
	0513	STL	20	0467	0470	A $\longrightarrow$ 0467
	0470	RAABL	67	8003	0477	
	0477	SL	16	0480	0485	
	0485	SLT	35	0004	0545	
	0545	BRMIN	46	0500	0501	is $\|A\| > 1$
	0500	SL	16	0553	0509	No: Modify Shift Right Inst.
	0501	AL	15	0554	0509	Yes: Modify Shift Left Inst.
	0509	STL	20	0521	0514	
	0514	RAL	65	0467	0521	
	0521	SRT	30	000X	0527	
		SLT	35	000X		
	0527	DIV	14	0481	0469	A/10 = I+r
	0469	STL	20	0473	0476	I $\longrightarrow$ 0473
	0476	SRT	30	0009	0547	r=j+s; j $\rightarrow$ 8003, s $\rightarrow$ 8002
	0547	STL	20	0552	0555	S $\rightarrow$ 0552
	0555	SL	16	8001	0511	
	0511	SLT	35	0004	0519	Modify Last Mult. Inst
	0519	AU	10	0522	0532	
	0532	STU	21	0499	0512	
	0512	RAU	60	0484	0543	
	0543	MULT	19	0552	0517	$A_{10} \cdot S = K_{10}$
	0517	RAU	60	8003	0533	
	0533	AU	10	0486	0494	
	0494	MULT	19	0552	0474	$(A_9 + K_{10})S = K_9$
	0474	RAU	60	8003	0534	
	0534	AU	10	0487	0544	
	0544	MULT	19	0552	0566	$(A_8 + K_9)S = K_8$
	0566	RAU	60	8003	0524	
	0524	AU	10	0483	0488	
	0488	MULT	19	0552	0525	$(A_7 + K_8)S = K_7$
	0525	RAU	60	8003	0536	
	0536	AU	10	0489	0496	
	0496	MULT	19	0552	0526	$(A_6 + K_7)S = K_6$
	0526	RAU	60	8003	0537	
	0537	AU	10	0490	0546	
	0546	MULT	19	0552	0479	$(A_5 + K_6)S = K_5$
	0479	RAU	60	8003	0538	

IBM TYPE 650 PROGRAM SHEET

PROBLEM: $e^A \longrightarrow B$ WRITTEN BY:

INSTR NO	LOCATION OF INSTRUCTION	OPERATION ABBRV.	OPERATION CODE	ADDRESS DATA	ADDRESS INSTRUCTION	REMARKS
	0538	AU	10	0491	0497	
	0497	MULT	19	0552	0529	$(A_4 + K_5)S = K_4$
	0529	RAU	60	8003	0539	
	0539	AU	10	0492	0549	
	0549	MULT	19	0552	0482	$(A_3 + K_4)S = K_3$
	0482	RAU	60	8003	0540	
	0540	AU	10	0493	0548	
	0548	MULT	19	0552	0530	$(A_2 + K_3)S = K_2$
	0530	RAU	60	8003	0541	
	0541	AU	10	0495	0498	
	0498	MULT	19	0552	0562	$(A_1 + K_2)_s = K_1$
	0562	RAU	60	8003	0542	
	0542	AU	10	0495	0499	
	0499	MULT	19	0506+j	0528	$(A_0 + K_1)y \overset{*}{=} e^A$ (see note below)
	0528	SCT	36	0000	0502	
	0502	STU	21	0560	0563	$e^A \longrightarrow 0560$
	0563	SLT	35	0008	0523	
	0523	RSL	66	8002	0564	
	0564	SIT	35	0002	0520	-count to 8003
	0520	AU	10	0473	0535	I-Count
	0535	AU	10	0551	0556	51 + I - count = exp
	0556	AL	15	0560	0516	
	0516	SRD	31	0002	0478	e^A
	0565	RAL	65	0472	0478	If A $\leq 10^{-10}$ set Ans to
	0478	STL	20	B	0026	

Note

*If j = -2 mult. by $1/e^2$
j = -1 mult. by $1/e$
j = 0 mult. by 1
j = 1 mult. by e
j = 2 mult. by e^2

IBM TYPE 650 PROGRAM SHEET

PROBLEM: $e^A \longrightarrow B$ WRITTEN BY:

Constants

INSTR NO.	LOCATION OF INSTRUCTION	OPERATION ABBRV.	OPERATION CODE	ADDRESS DATA	ADDRESS INSTRUCTION	REMARKS
	0559	STL	20	0000	0026	
	0561		52	1151	2925	50 in 10
	0471		12	1151	2925	
	0480		00	0000	0050	
	0553		30	0000	0527	
	0554		35	0000	0527	
	0481		23	0258	5093	In 10
	0522		19	0506	0528	
	0504		01	3533	5283	$1/e^2$
	0505		03	6787	9442	$1/e$
	0506		10	0000	0000	e^0
	0507		27	1828	1828	e
	0508		73	8905	6097	e^2
	0484		00	0000	0282	a_{10}
	0486		00	0000	2825	a_9
	0487		00	0002	4794	a_8
	0483		00	0019	8343	a_7
	0489		00	0138	8893	a_6
	0490		00	0833	3364	a_5
	0491		00	4166	6666	a_4
	0492		01	6666	6661	a_3
	0493		05	0000	0000	a_2
	0495		10	0000	0000	a_1, a_0
	0472		50	1000	0000	

THE LOAD AND PUNCH ROUTINE

1. THE LOAD ROUTINE

Definitions

Addressed Card: see Plus Card.

Control Field: Columns 1-10 on a card.

Field k: Columns 10k+1 through 10(k+1) on the card. k= 1,2,3, ... or 7.

Minus Card: One having an "X" Punch in column 10 along with whatever other digit may be necessary.

Plus Card: One having no "X" punch in column 10.

Un-addressed Card: see Minus Card.

Units column of Field k: Column 10(k+1).

Word: Any 10 digit number with sign. It may represent either an instruction or a piece of data. It is entered in field k with the most significant digit in column 10k+1 and the least significant digit in column 10(k+1). The sign, if the word is negative, is double punched an "X" in the units column of the field. No extra punch is necessary if the word is positive. Every column of the field must be punched and only the units column should be double punched.

The Control Field

Columns 1-5: punched 0,0,0,0,0 respectively.

Column 6: Punched with a single digit, n, permissible values of which will be enumerated in the following sections, Plus Cards and Minus Cards.

Columns 7-9: Punched with the first 3 digits of a 4 digit number having significance to the load routine in the circumstances to be described in the two sections, Plus Cards and Minus Cards, below.

Column 10: Punched with the fourth digit of the above-mentioned four digit number in the case of Plus Cards. In addition to this fourth digit, column 10 has an "X" punch if the card is a Minus Card.

Plus Cards

If n (column 6 in the Control Field) is zero, fields 1-7 on the card are ignored, and the load routine will transfer control to that instruction whose location is determined by the four digit number in columns 7-10 of the Control Field of the card.

If n is one of the digits 1,2,3,... or 7, fields 1 through n, respectively, of the card are transferred to the consecutive locations on the drum starting with that one whose address is given by the four digit number in columns 7-10 of the Control Field of the Card.

Values of n other than those listed above are not permissible for Plus Cards.

Minus Cards

If n (column 6 of the Control Field of the card) is zero, fields 1-7 on the card are ignored and the load routine will transfer control to that location whose adddress is determined by the four digit number in columns 7-10 of the Control Field of the card.

If n equals 9, an address is entered into the load routine which is that of the first of a sequence of locations to which information from succeeding cards (on which $n \neq 0$ or 9) will be sent up by the load routine. Fields 1-7 of the card are ignored. This address is taken from columns 7-10 of the Control Field of the card.

If n is one of the digits 1,2,3,... or 7, the n words, fields 1 through n respectively, of the card are stored on the drum in consecutive locations starting with the location whose address is one greater than that of the location to which the last word on the preceeding card (on which $n \neq 0$ or 9) was sent. The four digit number in columns 7-10 of the Control Field of the card is ignored by the load routine. Hence these digits may be used for card identification purposes.

Values of n other than those listed above are not permissible on Minus Cards.

Entering from Program

The load routine may be entered from any program by transferring control to drum location 1996. Control is returned to the program by the use of a card having n=0 (column 6 of the control field) behind the deck of cards to be entered.

Example

Consider the following load deck, containing many of the possible combinations that can be handled by the load routine.

Card 1: A plus card for which n=3 and columns 7-10 of the control field contain the address 0300.
The load routine will send the word in field 1 to location 0300; the word in field 2 to location 0301; the word in field 3 to location 0302 and read in.

Card 2: A minus card on which n = 2
The load routine will send the word in field 1 to location 0303; the word in field 2 to location 0304 and read in.

Card 3: A minus card on which n=9 and columns 7-10 of the control field contain the address 0320.
The load routine will set its address counter to 0320 and read in.

Card 4: A minus card on which n=1.
The load routine stores field 1 of the card into location 0320 and read in.

Card 5: A minus card on which n=3.
The load routine stores the word in field 1 of the card into location 0321; the word in field 2 into 0322; the word in field 3, into 0323 and read in.

Card 6: A plus card for which n=0 and columns 7-10 of the control field contain the address 0303.
The load routine transfers control to the instruction in drum location 0303 and the program loaded on the preceeding five cards begins at that point.

2. THE PUNCH ROUTINE

A. How to Get a Memory Dump:

By loading the following word into the lower accumulator:

on A B ± (A), where n = 1,2,3,... or 7

and then transferring control to drum location 1971, the punch routine will then punch out, n words to the card, all the words in drum locations A through B, inclusive. Minus cards will be punched if the word is negative and plus cards will be punched if it is positive.

The routine may be entered from a program and when the punch routine is finished, control will be transferred to that instruction whose drum location is entered by the programmer in the instruction address of location 1902.

B. The Punch Routine may also be used to emit an X-punch in column 75 or 76 on a card. See the How to Use the 650 sheet for details.

3. STORAGE REQUIREMENTS

The Punch and Load routines are scrambled together and occupy the entire last two bands of drum, locations 1901 -1999 inclusive. If location 1959 is used for temporary storage by the programmer, it must contain zeros before the Punch Routine is used.

K. B. Williams/lm
June 12, 1956

SHORT LOAD AND PUNCH ROUTINE (SLPR)

The essential differences between this routine and the regular Load and Punch Routine (LPR) are as follows:

1. SLPR occupies permanently the locations 1951-1999 in memory and uses the read buffer 1901-10 and the punch buffer 1927-36. While punching, the spaces 1901-1903 are used in addition to the punching buffer zone.

2. SLPR treats all cards to be Loaded as Plus (addressed) cards so that columns 7-10 must be filled in with the address to which the information on the card is to be sent. This is true whether or not the first field is positive or negative. SLPR will punch either Plus or Minus cards suitable for direct reloading with SLPR (or LPR with the use of the 9 card in case of Minus cards). CAUTION: If minus cards are to be punched the number of words to be punched must be an integral multiple of the number of words per card desired. Otherwise, the control field of the last card will not be set properly.

3. The exit from the punch routine is in 1953, otherwise all entrances are the same as in LPR; i.e. Load is entered at 1996 and Punch at 1971. Other than is noted above, the instructions and definitions for LPR hold for SLPR.

4. The Punch and Load portions SLPR are coded in separate blocks. The Punch portion occupies locations 1951-1979 and the Load routine occupies locations 1980-1999.

K. B. Williams/lm
June 12, 1956

SHORT LOAD ROUTINE (SLR)

This is a load only routine, identical to SLPR in that it will load only Plus (addressed) cards.
It occupies spaces 1980-1999 and uses the buffer 1951-1960.
It will not punch.
It is entered, as with the other load routines at 1996.

K. B. Williams/lm
June 12, 1956

FORMAT FOR INITIAL LOADER CARDS

June 13, 1956

I. Long Load and Punch Routine (LLPR)

8000 70 1851 1851

Card 1

(1851) 70 1951 1852

(1852) 69 1952 1951

(1853) 69 1954 1953

(1854) 69 1956 1955

(1855) 69 1958 1957

Card 2 (and succeeding cards)

(1951) 24 L(word 1) 1853

(1952) Word 1

(1953) 24 L(word 2) 1854

(1954) Word 2

(1955) 24 L(word 3) 1855

(1956 Word 3

(1957) 24 L(word 4) 1851

(1958) Word 4

The last card contains all zeros. The last 2 words of the card preceding it are:

(1957) 24 1851 1851

(1958) 70 1851 1996

This clears read buffer 1851-1858 to zeros before transferring control to the Load Routine.

II. Short Load and Punch Routine (SLPR)

8000 70 1851 1851

Card 1

(1851) 70 1901 1852

(1852) 69 1902 1901

(1853) 69 1904 1903

(1854) 69 1906 1905

(1855) 69 1908 1907

Card 2 (and succeeding cards)

Same as Long Load and Punch Routine

III. Short Load and Non-Zero Punch Routine

8000 70 1851 1851
Card 1

(1851)	70	1951	1852
(1852)	69	1952	1951
(1853)	69	1954	1953
(1854)	69	1956	1955
(1855)	69	1958	1957
(1856)	70	1851	1996

Card 2

Same as Long Load and Punch Routine.

The last card is all zeros. The C(1957) from the preceding card is 24 XXXX 1856, which clears the read buffer 1851-1858 before transferring control to the load routine.

IV. Short Load Routine

Same as Short Load and Non-Zero Punch Routine

V. Multiple Storage Check (MSC)

8000 70 1851 or 70 0001 0003*
Card 1 Same as card 1 SLPR
Card 2 (and succeeding cards) same as card 2 SLPR.
Last card reads:

(1901)	24	0000	1903
(1902)	00	0000	0000
(1903)	24	0001	1904
(1904)	24	1851	1905
(1905)	24	1852	1906
(1906)	24	1853	1907
(1907)	24	1954	1908
(1908)	24	1855	1996

This card clears the storages used by the storage clear card as well as the read buffer 1851-1858 before transferring control to MSC.

* A storage clear card should precede the MSC.

K. B. Williams
lm

AN INTERPRETATIVE FLOATING DECIMAL SYSTEM FOR THE IBM TYPE 650

Introduction

This floating decimal system will perform 23 basic operations using a floating decimal number system by means of interpretative programming. It was designed with coding convenience as a prime objective.

Floating decimal instructions, that is, instructions which are to be interpreted, have a negative sign. A floating decimal instruction consists of a two digit operation code, a 4 digit address specifying the location of the first factor, a 4 digit address specifying the location of the second factor (if required) and another 4 digit address specifying the location of the third factor (if required). It is a variable address system in that only as many addresses as are needed for the particular operation are required. Thus, some operations use only one address, some require two addresses and others require 3 addresses.

Floating decimal instructions will be taken from consecutive memory locations. If an instruction requires one or two addresses, the instruction is stored in one memory location. If three addresses are required, the third address is stored in the memory location immediately following the one containing the instruction.

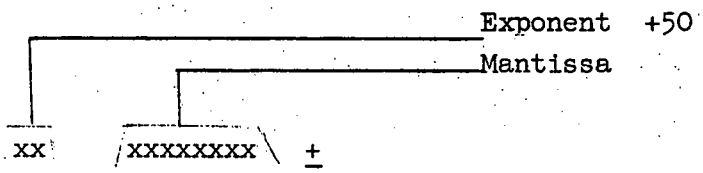
Should a positive instruction appear, it is interpreted as a normal 650 instruction and subsequent instructions are not interpreted. Thus, upon occurrence of a positive instruction the 650 operates in its usual D-I mode. This continues until an instruction address of 0026 is given which causes control to return to the interpretative routine. The program will return to the floating decimal mode of operation at the point of departure.

For example, consider the following sequence:

Location	Contents
n	floating point instruction (-)
n+1	floating point instruction (-)
n+2	floating point instruction (-)
n+3	normal 650 instruction(+)

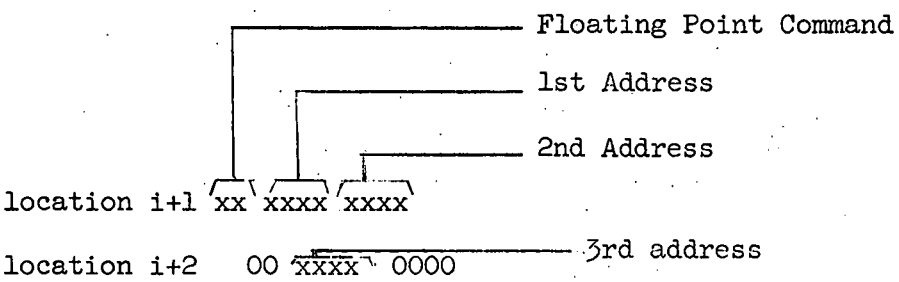
(The contents of n+3 are interpreted as a normal 650 instruction including the instruction address for sequencing. Normal execution of instructions continues until 0026 is used as an instruction address, at which time the program returns to the floating decimal mode of operation beginning with the instruction location n+4).

NUMBER FORM



where the Mantissa is in the form $1 \leq x.xxxxxxx < 10$

INSTRUCTION FORM



i+2 has this form only if the 3rd address is needed, otherwise the form of i+2 is the same as form of i+1.

START OF PROGRAM

To enter the interpretive routine for the first time to begin a sequence of interpreted instructions beginning at memory location i+1, the programmer must arrange to insert the number i+1 in the data address of location 0029. Then a programmed transfer to 0029 will begin the proper sequence of interpretive instructions. (See the "Hints and Aids" following this paper).

FLOATING DECIMAL ACCUMULATOR

A floating point accumulator referred to as k(location 0057) is used for accumulation and accumulative multiplication. It need not be addressed in operations which make use of it but can be addressed for other operations.

OPERATIONS

Below is a list of the operations with their codes, addresses required and estimated average time of execution.

Code and Addresses	Operation	Estimated Average Time
01, A, B	$A+B \longrightarrow K$	62.8
02, A	$A+K \longrightarrow K$	54.7
03, A, B	$A-B \longrightarrow K$	63.1
04, A	$K-A \longrightarrow K$	59.5
05, A, B, C	$A+B \longrightarrow C$	81.8
06, A, B, C	$A-B \longrightarrow C$	96.2
07, A, B, C	$AxB \longrightarrow C$	84.8
08, A, B	$AxB \longrightarrow K$	78.4
09, A, B, C	$A/B \longrightarrow C$	92.2
10, A, B	$A/B \longrightarrow K$	72.2
11, A, B	BR MIN A	28.4
12, A	BR	18.7
13, A, B	BRNZ A	30.8
14, A, B	$(AxB)+K \rightarrow K$	102.5
15, A, B	$K-(AxB) \rightarrow K$	102.5
16, A, B	$\sqrt{A} \longrightarrow B$	157.9
17, A, B, C	$\sqrt{A^2+B^2+C^2} \longrightarrow K$	416.3
18, A_1, B_1, n	$\sum_{i=1}^{n} A_i B_i \longrightarrow K$	(32.4 92.3n)

Code and Address	Operation
19 A, B	Sin A → B
20 A, B	Cos A → B
21 A, B	ln A → B
22 A, B	$e^A \rightarrow B$
23 A, B	$\tan^{-1} A \rightarrow B$

EXPLANATION OF PROGRAMS

1. General Interpretation

The general interpretation routine takes instructions from consecutive memory locations and analyzes them to see if tney are normal 650 instructions or if they must be interpreted. If the instruction is to be interpreted, the routine obtains the factor at address A and stores it in location 0037. Control is then transferred to the proper sub-routine. Each subroutine (operation codes 01 - 23) begins at the drum location identical to the operation code. Thus, if operation code 08 is used, control is transferred to location 0008.

2. Operation Code 16 Square Root

This sub-interpretative routine computes the required square root and stores it in location B. The initial approximation is $X_o = (1+4A)/(4+A)$, obtained from the RAND Corporation "Approximations in Numerical Analysis" form 15S, Notes.

3. Operation Code 18 Vector Multiplication

Operation 18 is a vector by vector multiplication. Address A_1 and address B_1 are the addresses of the first elements of each of the vectors. Succeeding elements of the vectors are then taken from consecutive memory locations starting with the initial locations A_1 and B_1. The third address, n, indicates the number of elements in each vector.

Use is made of the sub-interpretative routine "Interpretation of 14" to perform accumulative multiplication. As each pair of factors are multiplied together and added to the previous sum, n is reduced by 1 and zero tested. When n has been reduced to zero, the last two factors have been multiplied and the operation is complete.

4. Operation Codes 19 and 20 - Computes sin θ or cos θ

$/\theta/ < \pi/2 \times 10^{10}$ if θ exceeds this range an incorrect answer will result. θ is positioned so that division by $\pi/2$ results in $\theta = I + f$ where $f \leq \pi/2$ and I is an integral. NOTE: θ MUST BE IN RADIANS.

If cos θ is to be computed then $/I/ + 1 \rightarrow I$ and $/f/ \rightarrow f$.

If sin θ is to be computed and θ is negative then $/I/ + 2 \rightarrow I$.

The adjusted I is divided by 4 so that $I/4 = Q+R$.

if R = 0 then $/f/ \rightarrow f$

if R = 1 then $\pi/2 - /f/ \rightarrow f$

if R = 2 then $- /f/ \rightarrow f$

if R = 3 then $/f/ - \pi/2 \rightarrow f$

$(f.2/\pi) = x$ where $-1 < x < 1$

and $\sin(x) = 1.570796318(x) - .645963711(x^3)$
$+ .079689679(x^5) - .004673765(x^7)$
$+ .00015148419(x^9)$

Locations 0466 → 0569 are used for this routine. The relative error is not more than 5 in the 9th decimal.

5. Operation Code 21, Computes ln A

Let $x = y \cdot 10^K$

where $.1 \leq y \; 1$

if $y \leq \frac{\sqrt{10}}{10}$

we use

(1) $\ln x = (K - 3/4) \ln 10 + 2\Sigma^*$

where $2\Sigma^* = t\left(\frac{2}{3}t^2 + \frac{2}{5}t^4 + \frac{2}{7}t^6 + \frac{2\lambda^*}{9}t^8\right) + 2t$

and $t = \frac{y - 10^{-3/4}}{y + 10^{-3/4}}$

$\frac{2\lambda^*}{9} = \frac{(22/81)}{(11/9) - t^2}$

and if $y > \frac{\sqrt{10}}{10}$

we set $y = \frac{\sqrt{10}}{10} \cdot y$

and $K = K + \frac{1}{2}$

and go back (1)

This routine uses locations 0466-0566

6. Operation Code 22 - Computes $e^{\pm x}$
$-50\text{Ln}10 < x < 50 \text{ Ln } 10$

Procedure:

$\frac{x}{\text{Ln}10} = \frac{(i \text{ Ln}10 + r)}{\text{Ln}10}$ where: i is the integral part of quot. and r, the remainder

$e^x = e^{i \text{ Ln } 10 + r} = e^{i \text{ Ln } 10} \quad e^r = 10^i e^r$

Following division by Ln10, r is in the range

$-\text{Ln}10 < r < \text{Ln}10$

or approximately $-2.3 < r < 2.3$

we now split r into an integral part j and a fractional part s.

$e^r = e^{j+s} = e^j e^s$ j may be -2 -1 0 1 2

$$\left[1/e^2 \quad \frac{1}{e} \quad 1 \quad e \quad e^2 \right]$$

$X = i \text{ Ln } 10 + (j+s)$

$e^x = 10^i \cdot e^j \cdot e^s$

where e^j is one of the above values. We then compute only e^s using a Chebyshev Polynomial. The error is not more than 5 in the 9th decimal. This routine uses locations 0466-0566.

7. Operation Code 23 - Computes $\tan^{-1} x$

If $|x| < 1$, $\tan^{-1}x = \sum_{i=1}^{8} a_i \, x^{2i-1}$

If $|x| > 1$, $\tan^{-1}x = \pi/4 + \sum_{i=1}^{8} a_i \left(\frac{x-1}{x+1}\right)^{2i-1}$

The coefficients used are taken from RAND Corporation Approximations in Numerical Analysis, form 15S.

Locations 0466-0566 are used. The error is not more than 4 in the eighth decimal.

STORAGE

Operations 01-18 require 466 storage locations. It uses locations 0000 through 0465. Every location within this block is used, thus making it easy to incorporate this program with other programs.

LOADING

The interpretive routine and the commands for operations 01 through 18 can be loaded with the regular load routine. These commands have been punched sever per card and are entered into locations 0000-0465.

Operations 19-23 have been coded so that only the operations to be used need be entered on the drum. Therefore each subroutine was coded starting with location 0466. The sin and cos routines are coded as one unit and therefore must always be entered as one block. When more than one of these subroutines is used all but one subroutine must be translated, which can be done by using the translating routine. This translation must always be by an even amount to preserve the even-odd conditions.

A program using the interpretive routine can be loaded on with addressed or unaddressed cards. It is suggested that they can be loaded as unaddressed cards with only one instruction per card until the program is checked out. Since all instructions using the interpretive routine must be in sequence it will be easier to add and delete instructions in a program if the instructions are entered one per card. After the program is checked out, the instructions can be punched out 7 per cards for ease in future loading.

LOADING PROCEDURE SUMMARY

1. Load Routine
2. Interpretative Routine (Operations 01-18)
3. Translating Routine [if more than one subroutine (operations 19-23) is used] .
4. Transfer to translating routine (Card punched 000000 1729 in cols. 1-10)
5. Subroutines to be translated with a translator card preceding each subroutine.
6. Transfer to Load Routine (Card punched 9900001996 in cols 1-10)
7. General program using Interpretative Routine

NOTE: If translating routine is not used, do only steps 1,2,7.

K. B. Williams, 4/23/56
lm