Theoretical Notes
Note 46

MEMORANDUM
RM-4942
MARCH 1966

NUMERICAL PROGRAMS FOR SOLVING HYPERBOLIC SYSTEMS BY THE METHOD OF CHARACTERISTICS: RADIO EMISSION FROM A NUCLEAR EXPLOSION: PART I

G. H. Peebles

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA • 90406

PREFACE

This Memorandum is an input for a larger RAND project on the electromagnetic signal from a nuclear explosion. The numerical calculations of the signal are in theory simple to perform, but in practice are difficult because of the abrupt behavior of the functions which appear. The role of this report is to present some ways of overcoming the difficulties so that results are obtained efficiently and accurately.

SUMMARY

Two programs are presented which calculate the solution for a hyperbolic system of partial differential equations with three constant characteristic directions. The computations are on a net of characteristics of variable mesh, the size of the mesh being determined by a criterion of accuracy. The first program ranges over four mesh sizes; the second, over nine.

ACKNOWLEDGMENT

The writer wishes to thank Donald C. MacNeilage for assistance in the development of these numerical programs and in particular for his scheme of handling inputs, which is very convenient, flexible, and to some extent foolproof.

CONTENTS

TABLE OF SYMBOLS AND THEIR DEFINITIONS*

Symbol in Listing	Symbol in Flow Diagram	Definition
A	a	Horizontal characteristic bounding area of interest.
ADIVΘ		$\alpha\Delta t/\theta$, where $\theta = 1,2,4,6,12$.
ALBET	α	Floating point symbol for α, see IAB below.
AMU(1,μ)	t_μ	t at index μ in the μ-file.
AMU(2,μ)	x_μ	x at index μ in the μ-file.
AMU(3,μ)	v'_μ	dv/dt at index μ in the μ-file.
AMU(4,μ)	u'_μ	du/dt at index μ in the μ-file.
AMU(5,μ)	w'_μ	dw/dt at index μ in the μ-file.
AMU(6,μ)	v_μ	v at index μ in the μ-file.
AMU(7,μ)	u_μ	u at index μ in the μ-file.
AMU(8,μ)	w_μ	w at index μ in the μ-file.
B	B	$(w-u)/2x$.
CAPU	U	Approximate error in the value of u computed at $(i+2\alpha, j+2\alpha)$.
CAPV	V	Approximate error in the value of v computed at $(i+2\alpha, j+2\alpha)$.
CAPW	W	Approximate error in the value of w computed at $(i+2\alpha, j+2\alpha)$.
DELTAU	Δt	$t_{i+\alpha,j+\alpha} - t_{ij}$.
DELX	Δx	The increment of x between successive points of the μ-file on the first forward-running characteristic.
E	e	Maximum value accepted for U,V,or W.
EPSLØN	ϵ	e/Q.
ER	E_R	v/x.
ET	E_T	$(w+u)/2x$.
I	i	First subscript of a point in the 289-point array.

*The absence of a symbol from the table implies that the statement in which the symbol is found, or one close by, provides a definition.

I1		$i+\alpha$.
I2		$i+2\alpha$.
I3		$i+\alpha/2$.
I4		$i+3\alpha/2$.
I5		$i-\alpha$.
IAB	α	The ratio of a side of one of the squares from the 289-point array to the side of the smallest square.
IABY2		$\alpha/2$.
IAL(I,J)	α_{ij}	α at the point (i,j) in the array.
IALFAO	α_o	The value of α at every point of the μ-file on the first forward-running characteristic.
IAMU(2,IN)		Alternate symbol for AMU(I,J). See comment in listing for OPEN.
IB	β	An index associated with α.
IG	g	A subscript. See XG(IG).
IH	h	A subscript. See TH(IH).
IHALT		See comment in listing of subroutine HALT.
IR	r	$i+\alpha$ or $i+2\alpha$.
ISB		A switch set in accordance with the success or failure of a calculation to pass a test of accuracy.
ISIG		See comments prefacing the listing of VER1.
ITAUB(IB)	T_β	A tally of the successes of a module of index β.
ITB(IB)	t_β	An index locating a square in its module.
ITEMP		A fixed point variable of general use.
J	j	Second subscript of a point in the 289-point array.
J1		$j+\alpha$.
J2		$j+2\alpha$.
J3		$j+\alpha/2$.
J4		$j+3\alpha/2$.
J5		$j-\alpha$.
KOOOFX		A divide-check index.

LAMθ	λ_θ	A fixed point variable of general use. $\theta = 1,2,3,4$.
M1	μ_1	Index of the first point in the ℓ-file.
MAL(μ)	$\alpha(\mu)$	The value of α in the μ-file for index μ.
MAXSIG		The number of floating-point quantities at a point in both the μ-file and the 289-point array.
MCAP	M	The number of points in the μ-file.
MUO	μ_o	Index of the first point in the k-file.
MUK	μ_k	Running index for the k-file.
MUL	μ_ℓ	Running index for the ℓ-file.
MUR	μ_R	The number of points in the μ-file open for use.
NE	ω	The running sum of the values of α along the side of a module.
P		P(t,x).
Q	Q	The maximum value attained by the absolute value of u, v, or w at the point (17,17) in the 289-point array.
S		S(t,x).
TABLE(1,I,J)	t_{ij}	t at the point (i,j) in the array.
TABLE(2,I,J)	x_{ij}	x at the point (i,j) in the array.
TABLE(3,I,J)	v'_{ij}	dv/dt at the point (i,j) in the array.
TABLE(4,I,J)	u'_{ij}	du/dt at the point (i,j) in the array.
TABLE(5,I,J)	w'_{ij}	dw/dt at the point (i,j) in the array.
TABLE(6,I,J)	v_{ij}	v at the point (i,j) in the array.
TABLE(7,I,J)	u_{ij}	u at the point (i,j) in the array.
TABLE(8,I,J)	w_{ij}	w at the point (i,j) in the array.
TEMP1		A floating point variable of general use.
TH(IH)	t_h	A value of t for which outputs are wanted.
TP	t_p	t-coordinate for an output.
UP	u_p	u-value for an output.
V(IN)		A parametric input array.
VP	v_p	v-value for an output.
WP	w_p	w-value for an output.

XG(IG)	x_g	A value of x for which outputs are wanted.
XI	ξ	t-x.
XIH	ξ_H	The largest value of ξ assumed to be of interest.
XP	x_p	x-coordinate for an output.

I. INTRODUCTION

The propagation of radio signals from a nuclear burst in the atmosphere is represented, under suitable assumptions, by the system*

$$\frac{\partial E_R}{\partial t} = \frac{2}{x} B - S(t,x)\ E_R - P(t,x),$$

$$\frac{\partial E_T}{\partial t} + \frac{\partial B}{\partial x} = - \frac{B}{x} - S(t,x)\ E_T, \qquad (1)$$

$$\frac{\partial B}{\partial t} + \frac{\partial E_T}{\partial x} = - \frac{E_T}{x} - \frac{E_R}{x}.$$

The initial data are: $E_R = E_T = B = 0$, when $x = t$; and $E_T = 0$, when $x = \text{const.} > 0$. The functions $S(t,x)$ and $P(t,x)$, when they resemble those from an actual explosion, are abrupt and therefore difficult to handle numerically.

A subtraction and an addition of the last two equations lead to the equivalent system:

$$\frac{\partial E_R}{\partial t} = \frac{2}{x} B - S(t,x)\ E_R - P(t,x),$$

$$\frac{\partial (E_T - B)}{\partial t} - \frac{\partial (E_T - B)}{\partial x} = \frac{E_R}{x} + \frac{E_T}{x} - \frac{B}{x} - S(t,x) E_T,$$

$$\frac{\partial (E_T + B)}{\partial t} + \frac{\partial (E_T + B)}{\partial x} = - \frac{E_R}{x} - \frac{E_T}{x} - \frac{B}{x} - S(t,x) E_T,$$

a system, it will be seen, in which each equation represents the derivative of a particular quantity in a particular direction.** To

*See V. Gilinsky, The Kompaneet's Model for Radio Emission from a Nuclear Explosion, The RAND Corporation, RM-4134, August 1964, p. 10, Eqs. (3.21), (3.22), (3.23).

**For a general treatment of the notions implicit in these maneuvers, see R. Courant and D. Hilbert, Methods of Mathematical Physics, John Wiley & Sons, Inc., New York, 1962, pp. 424 ff.

capitalize on this fact, change the dependent variables by means of the relations

$$u = E_T - B,$$

$$v = E_R,$$

$$w = E_T + B.$$

The new system is

$$\frac{du}{dt} = \frac{1}{x}(u+v) - \tfrac{1}{2} S(t,x)(u+w), \qquad \text{if } \frac{dx}{dt} = -1,$$

$$\frac{dv}{dt} = \frac{1}{x}(w-u) - S(t,x)v - P(t,x), \qquad \text{if } \frac{dx}{dt} = 0, \qquad (2)$$

$$\frac{dw}{dt} = -\frac{1}{x}(u+v) - \tfrac{1}{2} S(t,x)(u+w), \qquad \text{if } \frac{dx}{dt} = 1.$$

The initial data become: $u = v = w = 0$, when $x = t$; and $w = -u$, when $x = \text{const.} > 0$.

The last system can be regarded as three ordinary differential equations: the first valid along the family of straight lines $x + t = \text{const.}$, the so-called backward-running characteristics; the second, along the lines $x = \text{const.}$, called here the horizontal characteristics; and the third, along the lines $x - t = \text{const.}$, the forward-running characteristics. In the light of the above remarks, consider Fig. 1. At both A and B, u, v, and w are known so that the derivative of u,

$$\frac{du}{dt} = u',$$

along BC and the derivative of v,

$$\frac{dv}{dt} = v',$$

along AC are also known. Approximate values at C are

$$u_{C1} = u_B + u'_B \Delta t/2,$$

$$v_{C1} = v_A + v'_A \Delta t,$$

$$w_{C1} = - u_{C1},$$

$$u'_{C1} = f(u_{C1}, v_{C1}, w_{C1}),$$

$$v'_{C1} = g(u_{C1}, v_{C1}, w_{C1}),$$

$$w'_{C1} = h(u_{C1}, v_{C1}, w_{C1}),$$

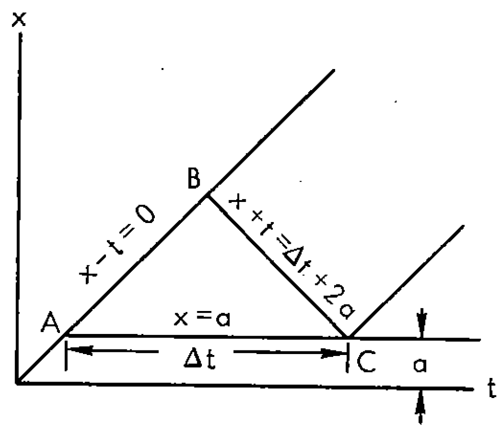


Fig.1

where f(u,v,w), g(u,v,w), and h(u,v,w) are the right-hand members respectively of the formulas for $\frac{du}{dt}$, $\frac{dv}{dt}$, and $\frac{dw}{dt}$ in system (2). Improved values at C, accurate to and including the term in $\overline{\Delta t}^2$, are

$$u_{C2} = u_B + (u'_B + u'_{C1})\Delta t/4,$$

$$v_{C2} = v_A + (v'_A + v'_{C1})\Delta t/2,$$

$$w_{C2} = - u_{C2},$$

$$u'_{C2} = f(u_{C2}, v_{C2}, w_{C2}),$$

etc.

The method of numerical integration will be recognized as that of Heun. Its accuracy cannot be improved if only the two points at the ends of the interval are used.

If Δt is chosen small enough to make the last approximation at C as accurate as needed, the calculation of values at E in Fig. 2 can begin. Dropping the second subscript at C to indicate that the values there are acceptably accurate, one has

$$u_{E1} = u_D + u_D' \Delta t/2,$$

$$v_{E1} = v_B + v_B' \Delta t,$$

$$w_{E1} = w_C + w_C' \Delta t/2,$$

$$u_{E1}' = f(u_{E1}, v_{E1}, w_{E1}),$$

etc.

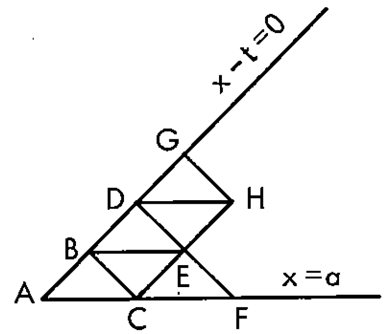


Fig. 2

These values in turn give the improvements

$$u_{E2} = u_D + (u_D' + u_{E1}')\Delta t/4,$$

$$v_{E2} = v_B + (v_B' + v_{E1}')\Delta t/2,$$

$$w_{E1} = w_C + (w_C' + w_{E1}')\Delta t/4,$$

$$u_{E1}' = f(u_{E1}, v_{E1}, w_{E1}),$$

etc.

It is clear that the values either at H or at F are now ready for computation and that to reach the point Z in Fig. 3 some such net as shown is necessary.

The general outlines of a simple procedure appear. There are some decisions to be made on the storing of the values of u, v, etc., at the nodes, but the problem is straightforward for the simple net of Fig. 3.

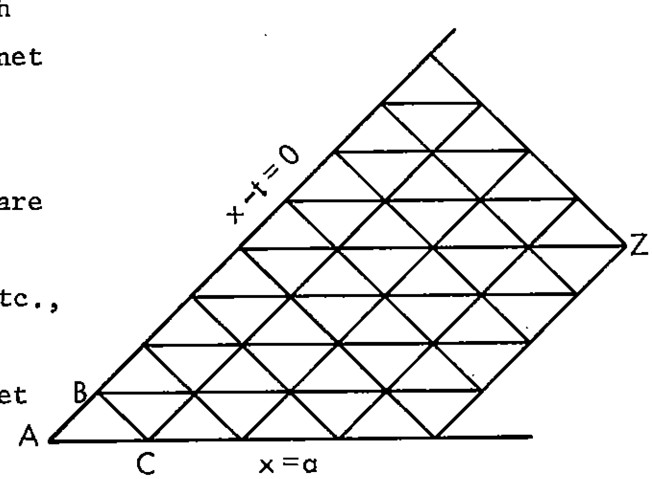


Fig. 3

The numerical solution, however, is kept from being routine by the nature of the input functions S(t,x) and P(t,x). The latter, when they are defined so as to resemble those from a nuclear explosion, have an extremely violent behavior which swamps the simple program described. Solutions of meaningful problems calculated on a mesh of constant size are either totally inaccurate or prohibitively long. But a net of variable mesh size turns out to be a very effective way of dealing with the rapid variations in S(t,x) and P(t,x). That the variable mesh is necessary is attested by the fact that it is not unusual for the solution of a problem to use 50 percent of its time in as little as 1/100 of one percent of its area. See Fig. 4.

It is the intention here to develop numerical programs which vary the size of the mesh throughout a calculation in such a way that solutions of system (2) [or (3)] are reached with something like maximum efficiency.

The schemes for doing this are specifically designed for the problem of the signal from a nuclear burst. However, the only thing unusual about this particular instance of one-dimensional wave propagation is the lengths to which it is necessary to go to get a numerical solution. It is believed, therefore, that the devices developed below are of some general interest. In any event, it is no restriction on much of what follows to replace system (2) by the more general system:

$$\frac{du}{dt} = f(t,x,u,v,w), \qquad \text{if } \frac{dx}{dt} = -1,$$

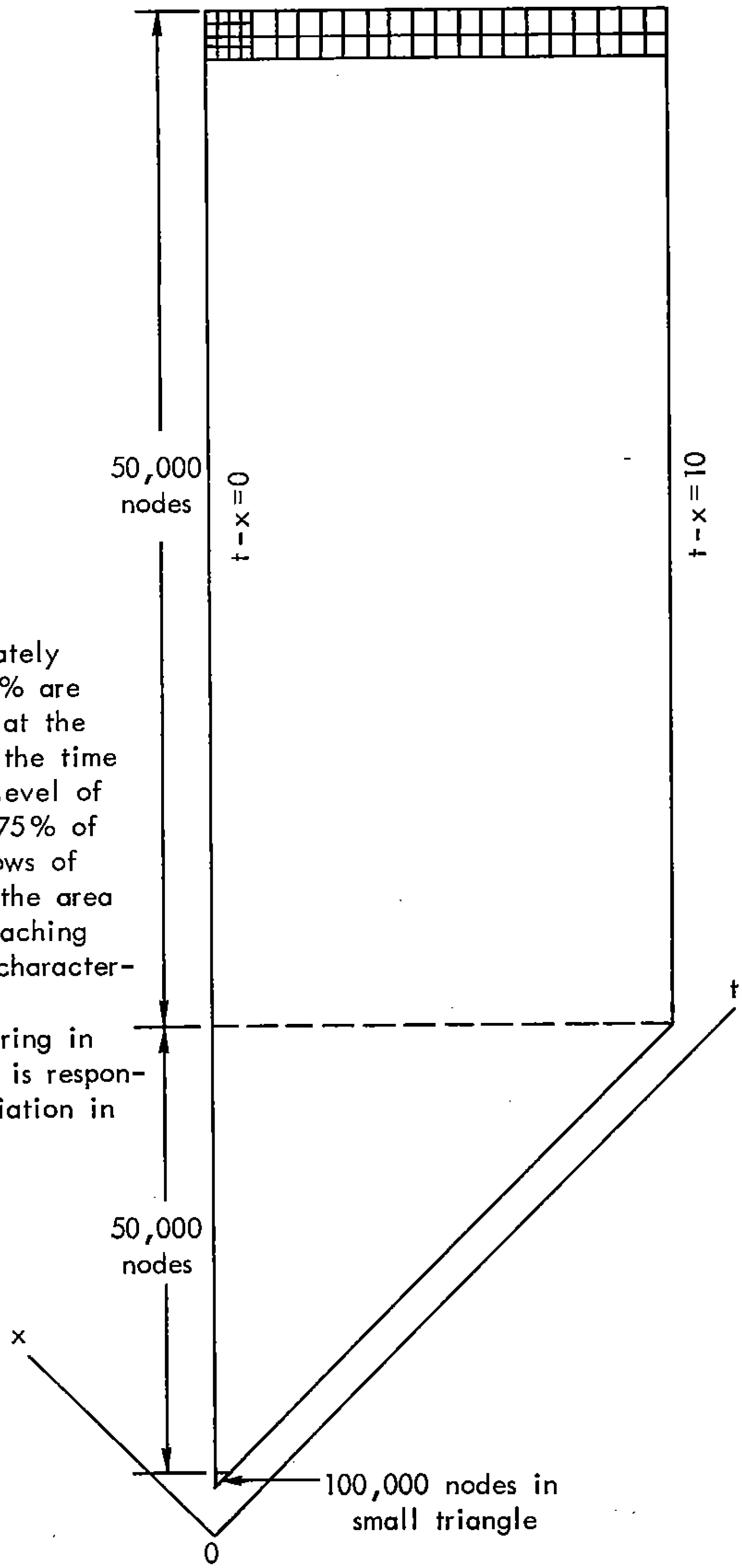
$$\frac{dv}{dt} = g(t,x,u,v,w), \qquad \text{if } \frac{dx}{dt} = 0, \tag{3}$$

$$\frac{dw}{dt} = h(t,x,u,v,w), \qquad \text{if } \frac{dx}{dt} = 1,$$

where the quantities u, v, and w are supposed known on some forward-running characteristic, $x + t = \text{const.} \geq 0$ and where a formula, say,

$$w = \varphi(t,u,v),$$

supplies w on a horizontal characteristic $x = a > 0$.



Of a total of approximately 200,000 nodes, about 50 % are within the small triangle at the tip of the trapezoid. By the time the solution reaches the level of the obtuse angle, about 75% of the work is done. The rows of squares along the top of the area show the mesh size on reaching the last forward-running characteristic.

A factor e^{-x}/x^2 appearing in both $S(t,x)$ and $P(t,x)$ is responsible for the extreme variation in mesh size.

Fig.4—A not unusual variation in mesh size

When the argument allows generality, system (3) is the model. When it is necessary to be specific, the model is system (2).

II. THE OVERALL PLAN

The most convenient way to change mesh size is to halve or to double. Doubling is only a matter of ignoring the proper points, but halving requires interpolation. If the interpolation is two-point, the result is only linearly accurate. Since the values of u, v, w, etc., are quadratically accurate, three points are needed to retain accuracy. This suggests that the calculation should be based on a module of four squares of the same size as shown in Fig. 5. Known values of the variables are at A, B, C, D, and G. Values are calculated first at E, then, say, at F, followed by those at H and I. But if the simple module of one square is abandoned, perhaps a method of order higher than Heun's could be profitably used. The difficulty with this expedient is that only two points of the module are available with which to calculate v at F and H. Of course it is not strictly necessary to use only the nine points of the module, but it is much simpler and more compatible with subsequent maneuvers if the calculation is contained entirely within the module. Four squares and the two-point method of Heun are the base for what follows.

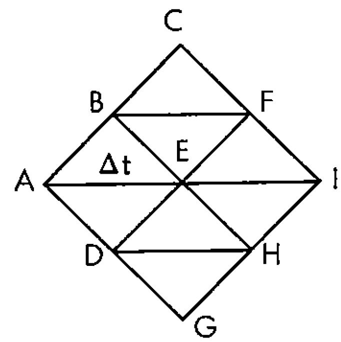


Fig. 5

Suppose now that the values at E, F, H, and I (Fig. 5) are calculated. Implicit in what has gone before is a criterion of the acceptability of their accuracy. The module of four readily provides such a criterion. Consider the relations

$$u_{I3} = u_C + (u'_C + 4u'_{F2} + u'_{I2})\Delta t/12,$$

$$v_{I3} = v_A + (v'_A + 4v'_{E2} + v'_{I2})\Delta t/6,$$

and

$$w_{I3} = w_G + (w'_G + 4w'_{H2} + w'_{I2})\Delta t/12.$$

As the notation indicates, the values at I given by these formulas are accurate to and including the term in $\overline{\Delta t}^3$. If these third-order values differ by an acceptably small amount from the second-order values, there is reason to expect u_{I2}, v_{I2}, and w_{I2} to be sufficiently accurate.* Looked at another way, the differences between the two approximations are

$$u_{I2} - u_{I3} = (u'_C - 2u'_{F2} + u'_{I2})\Delta t/96,$$

$$v_{I2} - v_{I3} = (v'_A - 2v'_{E2} + v'_{I2})\Delta t/48,$$

and

$$w_{I2} - w_{I3} = (w'_G - 2w'_{H2} + w'_{I2})\Delta t/96,$$

and they show, not surprisingly, that the criterion amounts to insisting that the second-order differences on u', v', and w' be small along the directions $\frac{dx}{dt} = -1$, 0, and 1 respectively.

Suppose it turns out that, according to the criterion, the second-order values at I are insufficiently accurate. Then the procedure used in this program is to subdivide each square of the module of four into four modules of four. The values at E are calculated and tested on the smaller module. If the criterion is satisified, the values at F are attempted, and so on until the point I is reached. Suppose, however, that the module of four, which is to yield the values of F, also turns out to be too large. Then its four squares are each subdivided into modules of four squares. In theory, subdividing could continue indefinitely, but in practice, three subdivisions seem to be about the optimum, all things considered. Thus the values at I might, as a consequence of critical subdivision, be

---

*Similar criteria are examined and recommended by Mark Lotkin in "On the Improvement of Accuracy in Integration," Quarterly of Applied Mathematics, Vol. XIII, No. 1, April 1955, pp. 47-54.

reached on a network such as shown in Fig. 6.

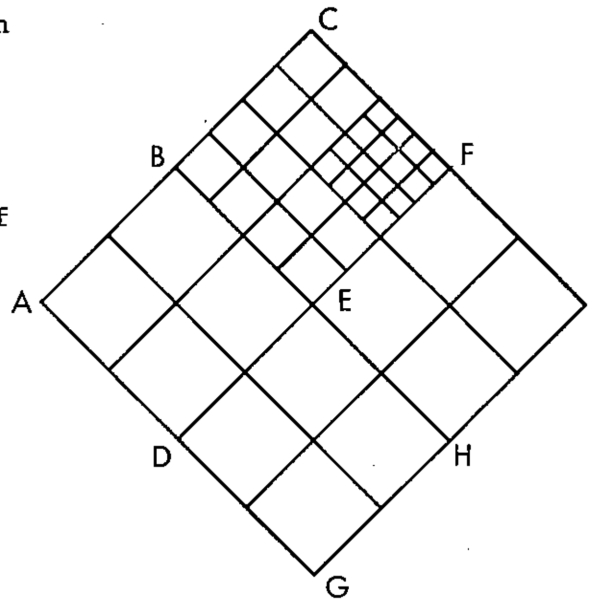


Fig. 6

It will be seen now what is contemplated. The area of Fig. 3 is to be worked over in modules of 4 squares (modules of one square and two right-angle equilateral triangles at the first horizontal characteristic), any one of which can be broken down into 256 sub-squares (120 sub-squares and 16 sub-triangles along x = a). To implement this idea a double-subscripted array of 289 (= 17^2) points* is set up. Known values of the variables are set in along the upper and lower left edges of the array. The operations described obtain values along the upper and lower right edges of the array. The quantities on the lower right edge are stored, those on the upper right are transferred to the lower left edge, and new values are read into the upper left edge. This process is repeated over and over until the last backward-running characteristic is reached. The area covered is a strip whose edges are forward-running characteristics. One edge of the strip provides the known values for the array; the other edge, as it were, provides storage for quantities from the lower right edge of the array. In this way an area like that of Fig. 3 is covered strip by strip until the last forward-running characteristic is reached.

The procedure sketched above, when suitably implemented with subroutines, becomes a program which operates on a variable net of four mesh sizes. This program can do many problems, but there are also many that are too difficult for it. Implicit in its formulation is the constraint that all strips be of the same width. This

*Since several quantities reside at each point of the array, the array is really triply subscripted.

constraint is not necessary. It is set merely to simplify. Removing it leads to a program more complex and, since it uses more mesh sizes, more powerful.

The first version described below assumes strips of the same width; the second version does not. Other versions will remove other implicit constraints.

III. FIRST VERSION

The main loop begins, as described above, with a file of discrete values for u, v, w, etc., along a forward-running characteristic. Sets of these values are transferred to the upper left edge of the 289-point array, where they are transformed into another set, which goes to make up a second file along a second forward-running characteristic. Another circuit of the loop is then made in which the second file plays the role of the first file, and so on. Hereafter the first file is called the k-file; the second, the ℓ-file.

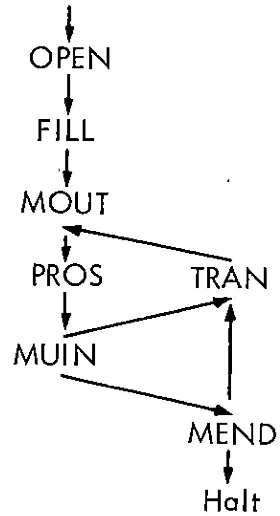


Fig. 7

The whole system of loops consists of seven major operations or subroutines (see Fig. 7), called here: OPEN, FILL, MOUT, PROS, MUIN, TRAN, and MEND. The first routine, OPEN, is the starting routine. FILL is the operation of making up the first k-file from the initial values of u, v, etc., given along the first forward-running characteristic. MOUT is the transfer of the values from the k-file to the array. PROS is the process of generating from the transferred values the values for the upper and lower righthand edges of the array. MUIN is the transfer of the set on the lower right edge to the ℓ-file. TRAN is the transfer of the set on the upper right to the lower left edge. MEND is the operation of ending the ℓ-file and preparing it for its role as the k-file.

THE MAIN ROUTINE

A knowledge of the workings of PROS is preliminary to understanding any one of the other subroutines, because they serve PROS by bringing up, preparing, or taking away data. Figure 8 displays and explains the operations of PROS.

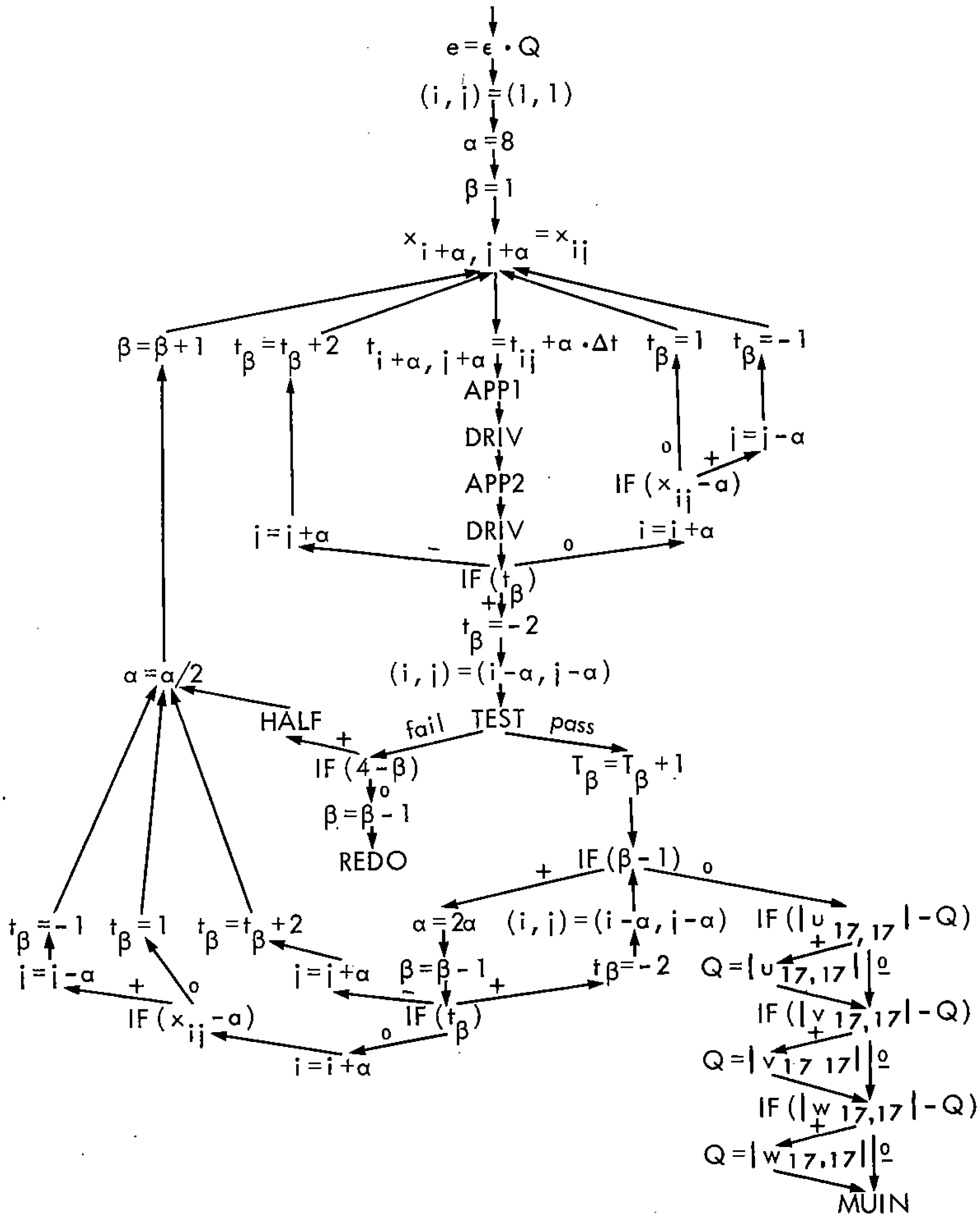


Fig. 8—Subroutine PROS

LEGEND FOR FIG. 8

e — is the approximate error set for u, v, and w.

ϵ — is the approximate error relative to the maximum reached anywhere by $|u|, |v|$, or $|w|$. PROS usually obtains u, v, and w with the accuracy implied by ϵ, but sometimes the error becomes as large as, say 10ϵ. It is, therefore, desirable to set ϵ at about 1/10 of the acceptable relative error.

Q — is the maximum of $|u|, |v|$, or $|w|$. It is estimated initially and is increased when PROS finds the estimate too small.

i,j — are the indices affixed to the points of the array of 289. The leftmost point has the pair (1,1); the rightmost, the pair (17,17); the topmost, the pair (1,17), etc.

α — is the number of points less one along a side of the square under consideration. It has four values: 8, 4, 2, 1.

β — is the index for α. $\alpha = 2^{\beta-1}$.

t_{ij}, x_{ij} — are the values of the independent variables at the point (i,j).

Δt — is the length of the diagonal of a square whose α is 1.

APP1 — and APP2 are the subroutines which make respectively the first-order and the second-order approximations to u,v, and w. See Fig. 10.

DRIV — is the subroutine which calculates u',v', and w'. Since it is straightforward, it is not discussed.

t_β — is the index showing the square under consideration in a module of index β. See Fig. 9.

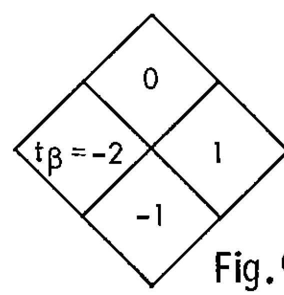


Fig.9

a — is the value of x on the first horizontal characteristic. Along this characteristic $w = \varphi(t,u,v)$.

TEST — is the subroutine testing the accuracy of the results of APP2. See Fig. 11.

HALF — is the subroutine interpolating for the four half-points in the upper and lower left edges of a module. See Fig. 12.

REDO — is a subroutine taking appropriate action when a module of index $\beta = 4$ fails to yield acceptably accurate values. In the first version it can consist simply of the command to halt the calculation.

T_β — is a tally recording the number of times a module of index β has been accepted.

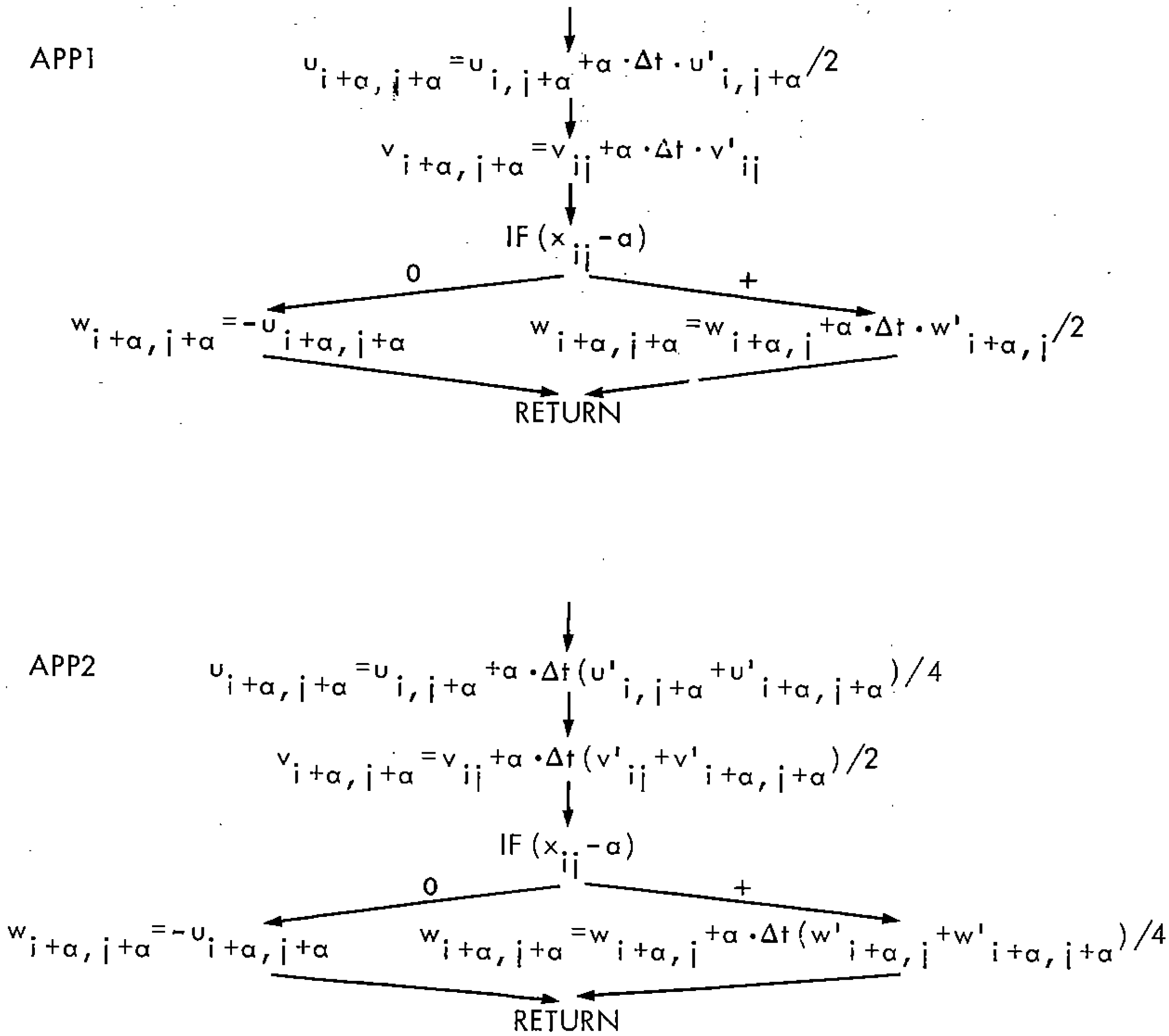


Fig. 10—Subroutines APP1 and APP2

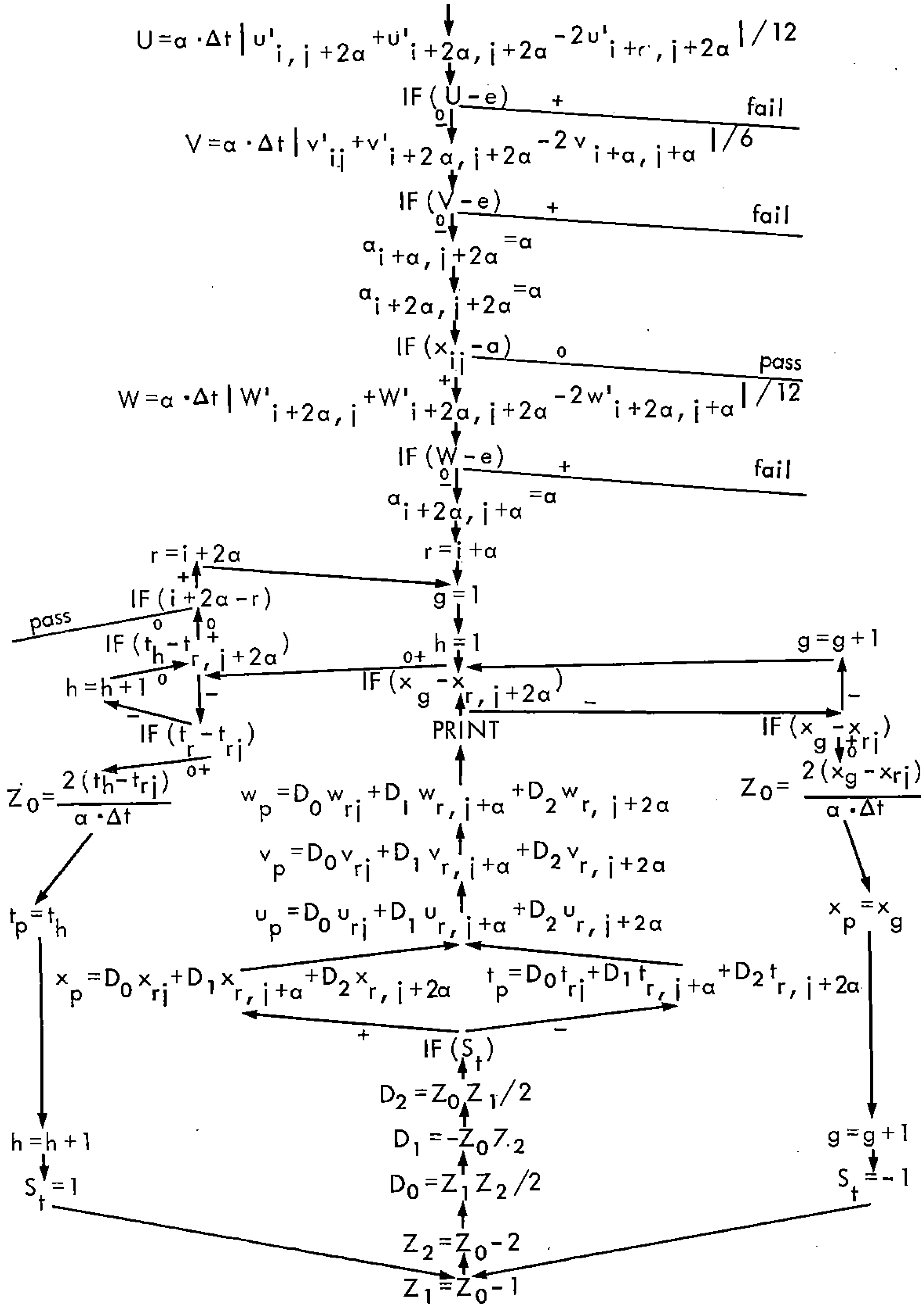


Fig. 11—Subroutine TEST

LEGEND FOR FIG. 11

g and h are subscripts. The first ten lines of TEST accomplish its previously stated purposes, which are testing the values obtained by PROS for acceptability and recording acceptance, when found, by storing the value α at each node on the perimeter of the module. After its first ten statements, TEST turns to the ultimate purpose of the program, namely, the production of visible results. The program assumes that results are wanted on lines of constant x and constant t and gives a way to obtain them in the case of the dependent variables u, v, and w. The constants associated with x and those associated with t are ordered according to their magnitudes and set in the subscripted quantities x_g and t_h. Two dummy values, larger than the program can possibly reach in its run, terminate the set of constant x and the set of constant t. The reason for these large terminating values is most easily discovered by studying TEST's flow diagram. Quadratic interpolations, once on the points $(i+\alpha,j)$, $(i+\alpha,j+\alpha)$, $(i+\alpha,j+2\alpha)$, and once on the points $(i+2\alpha,j)$, $(i+2\alpha,j+\alpha)$, $(i+2\alpha,j+2\alpha)$, yield u_p, v_p, and w_p, the values of the dependent variables at the points, where the lines of constant x or constant t intersect two of the forward-running characteristics of the module.

S_t is an integer used as a switch.

D_o, D_1, and D_2 are Lagrangian interpolation coefficients.

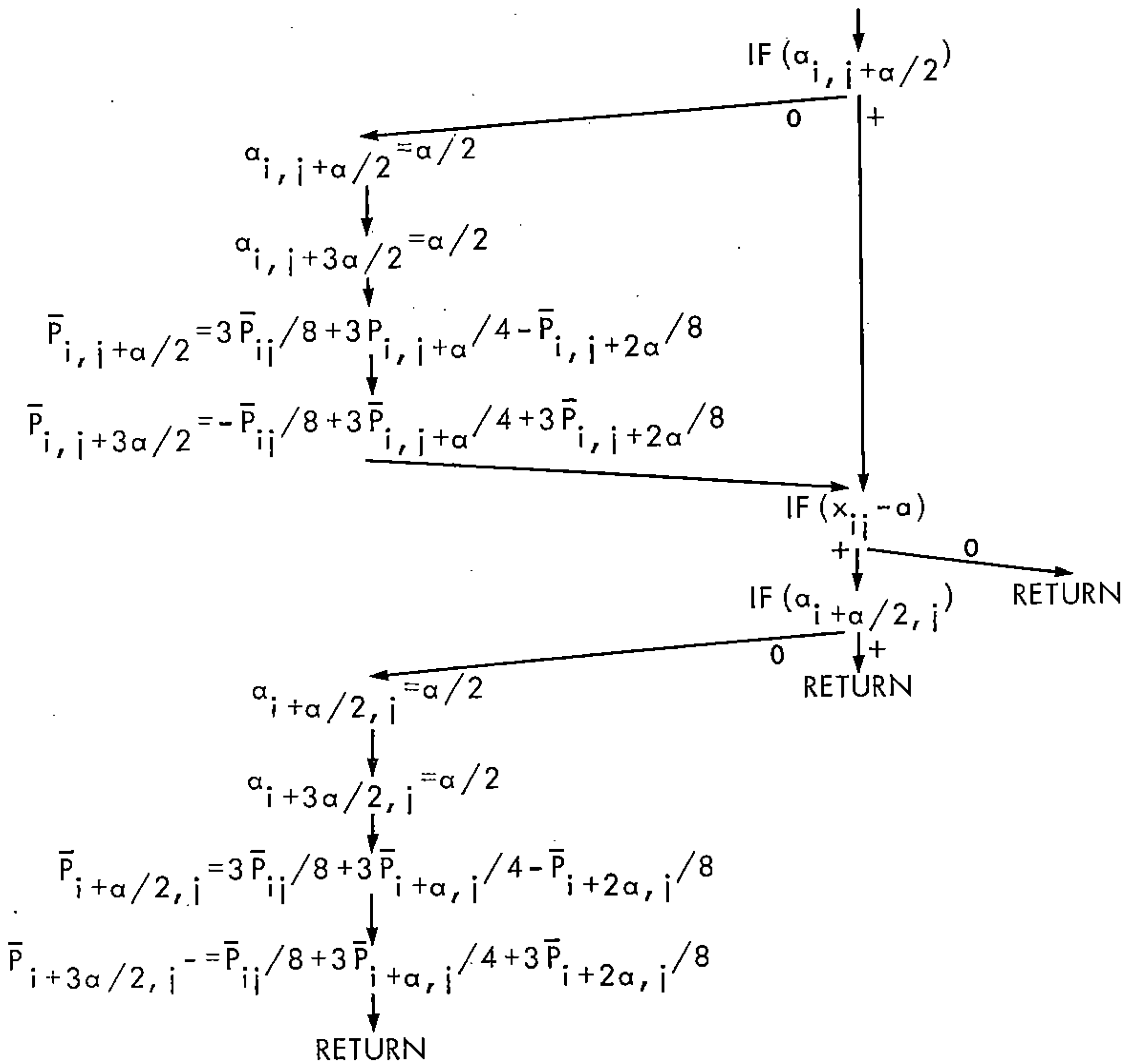


Fig. 12—The Subroutine HALF

Stored at each point of the array are the local values of t, x, u, v, w, u', v', w', and α. The last is defined as the value of α used in obtaining the stored values of u, v, etc. The symbol P_{ij} represents the entire set of nine quantities residing at (i, j). The symbol $\bar{P}_{ij}$ is P_{ij} without α. When the set $\bar{P}_{ij}$ is acceptably accurate, α_{ij} is set to α. A zero value for α_{ij} signifies that $\bar{P}_{ij}$ is not yet accurately known.

THE SUBROUTINES FILL, MOUT, AND MUIN

The two files, the k- and ℓ-files, which contain the values for and from the edges of the array, are themselves sub-files of one large file. Let the latter be called the μ-file. Then a point of the μ-file P_μ, or alternatively $P(\mu)$, holds nine quantities: t,x,u, etc., the same as P_{ij} does. The smallest value of μ is 1 and the largest is, say, M.

The subroutine FILL fills the first k-file, starting with index $\mu = 1$, with enough points to reach as far along the first forward-running characteristic as the backward-running characteristic which bounds the area of interest. PROS, receiving inputs from the k-file via MOUT, generates values from which MUIN sets up the ℓ-file with the first point following immediately after the last point in the k-file. When the ℓ-file is complete, it becomes the k-file. If this circuit is repeated enough times, the ℓ-file runs out of space in the μ-file. That is, MUIN may call for a point beyond $\mu = M$. When this happens, MUIN sets μ to 1 and continues. MOUT must be able to make the same maneuver.

Figure 13 displays the flow diagrams for the three routines.

THE SUBROUTINES TRAN AND MEND

After MUIN has completed its operations, the array is prepared for its next turn with PROS. This preparation consists of MOUT's operations supplemented by those of TRAN. TRAN performs the transfer of values from the upper right to the lower left edge and sets to zero all α_{ij} except those on the line i = 1. The latter operation is necessary because the test of whether or not a point in the array contains correct information is whether or not its α is non-zero.

Figure 15 shows the flow diagram for TRAN.

When the end of the k-file is reached, a contingency revealed by the equality of μ_1 and μ_k, MEND begins the operations associated with changing the ℓ-file to a k-file. First, however, MEND makes sure that these operations are needed by determining whether point (17,17) of the array lies on or to the right of the last forward-running characteristic bounding the area of interest. If the solution

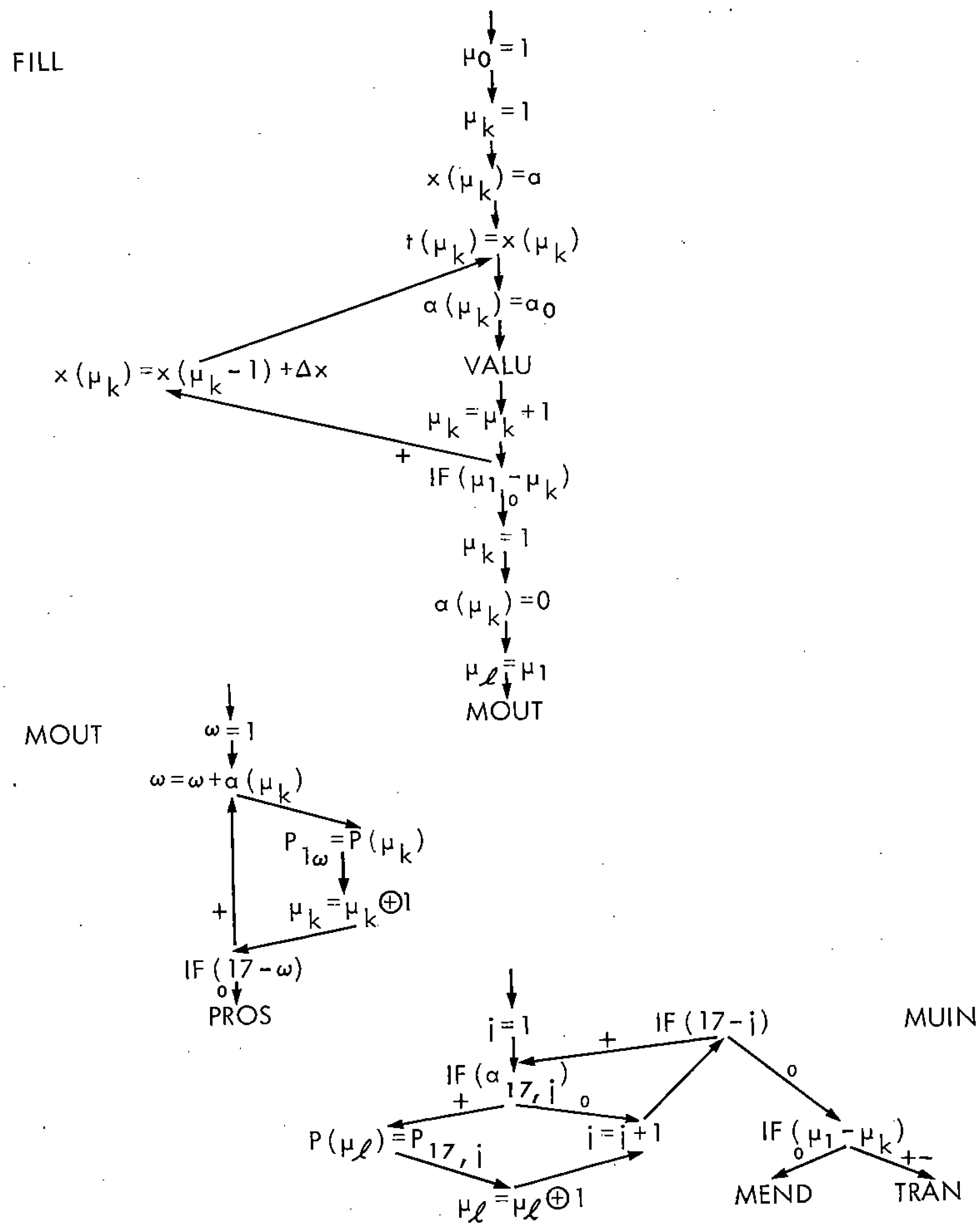


Fig. 13—Subroutines FILL, MOUT, and MUIN

μ_k is an index from the k-file

μ_ℓ is an index from the ℓ-file

LEGEND FOR FIG. 13

μ_o is the first index of the k-file.

μ_k is the running index for the k-file.

a is the value of x on the first horizontal characteristic.

$x(\mu_k)$, $t(\mu_k)$, and $\alpha(\mu_k)$ are respectively the values x, t, and α for index μ_k.

VALU is the subroutine which supplies u, v, w, u', v', and w' at μ_k.

α_o is the value for α which FILL assigns to every point in the first file. It is an input and is given a small or large value according as the first forward-running characteristic is in a region of bad or good behavior.

μ_1 is the first index in the ℓ-file and follows immediately after the last index of the k-file.

Δx is the increment in x between successive values of μ_k. μ_1 and Δx are inputs which must be so chosen that the solution can develop over an adequate area. It is also necessary that μ_1-2 be a multiple of 2^β, where β corresponds to α_o. This insures that the last point in the k-file falls on the top corner of the array.

$\oplus$ is the symbol used here for addition with modulus M. It is shorthand for the operations of Fig. 14.

μ_ℓ is the running index for the ℓ-file.

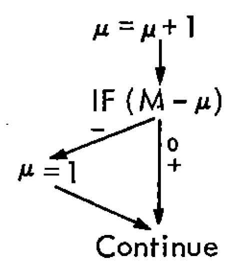


Fig. 14

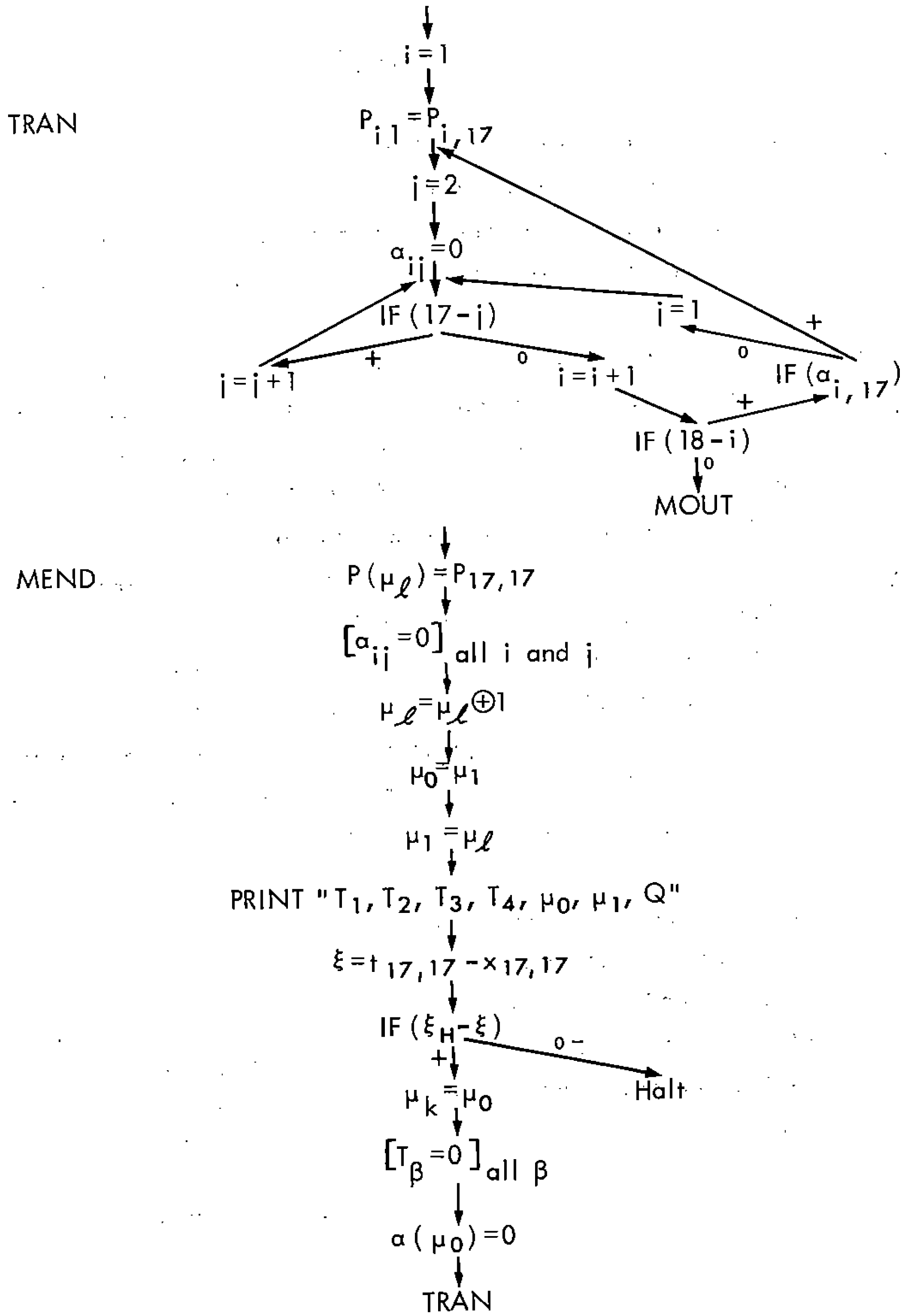


Fig. 15—Subroutines TRAN and MEND

ξ_H corresponds to the last forward-running characteristic bounding the area of interest.

is not yet completed, MEND sets all the α's to zero, including, now, those for i = 1. MUIN does not transfer $P_{17,17}$ to the ℓ-file, since $P_{17,17}$ goes in from the next array as $P_{1,17}$. MEND makes the transfer. Next MEND sets μ_o to μ_1 and gives μ_1 its position immediately after the last index of the new k-file. MEND also prints out information relative to operations for the strip just finished. This, of course, is optional, but experience is that T_β, β = 1,2,3,4, and μ_o, μ_1, and Q are likely to be interesting quantities to have available.

The flow diagram for MEND, always the final routine, the one out of which comes the command to halt the calculations, is in Fig. 15.

THE SUBROUTINE OPEN

Twelve quantities receive initial values at the beginning of a run. Of these, nine, α_o, Δt, μ_1, ϵ, Q, a, x_g, t_h, and ξ_H are chosen by the operator according to the requirements of the run. Three are always given the same value:

$$t_\beta = -2, \ \beta = 1, 2, 3, 4,$$

$$\alpha_{ij} = 0, \ i,j = 1, 2, \ldots, 17,$$

$$T_\beta = 0, \ \beta = 1, 2, 3, 4.$$

IV. SECOND VERSION

The effectiveness of the first version is limited at one end of the range of mesh size by failure of one of the smallest modules to pass TEST and at the other end by its inability to take advantage of larger modules. The processing of the k-file depends on the α's in the k-file and on the Δt chosen for PROS, but the formulas of PROS show that as long as the product $\alpha \cdot \Delta t$ is constant the results are the same. That is to say, if all the α's were doubled and the Δt halved, the results would be the same except that the strip covered would be half as wide, and two strips would be needed instead of one. Again, if all the α's were halved and Δt doubled, the chief consequence would be a strip twice as wide.

This suggests a means of expanding the range of the first version. Suppose that TEST is failed when $\alpha = 1$, and suppose further that instead of halting the calculations, Δt is halved, all the α's are doubled, and the processing of the k-file is started again. On this passage through the k-file, $\alpha = 1$ corresponds to a smaller diagonal, and TEST may pass the module. The only difficulty is the possible existence of α's of 8 in the original k-file. But this difficulty can be got around by interpolating so that each pair of points having α's of 8 is replaced by a set of four points with α's of 4. All α's can then be doubled.

Doubling the width of the strip is always possible when no α in the k-file is 1. To be sure, after doubling, the error criteria may sh that halving, not doubling, is the correct change in Δt, but it is so inefficient to use modules smaller than necessary that strip width is doubled, whenever possible. To cover an area with modules whose diagonals are twice, four times, or eight times too large is, respectively, a waste of about 25, 31, or 33 percent. On the other hand, to cover an area with modules half as large as necessary is 300 percent wasted effort. In view of these percentages it is best to fall on the large side of the optimum size module.

THE NEW AND MODIFIED SUBROUTINES

The main changes in the program in the second version (see the overall plan of Fig. 16) are in the new subroutines REDO and HAFM, shown in Figs. 17 and 18 respectively. These two subroutines make the arrangements for starting over with a narrower strip, whenever PROS finds that its smallest module cannot pass TEST.

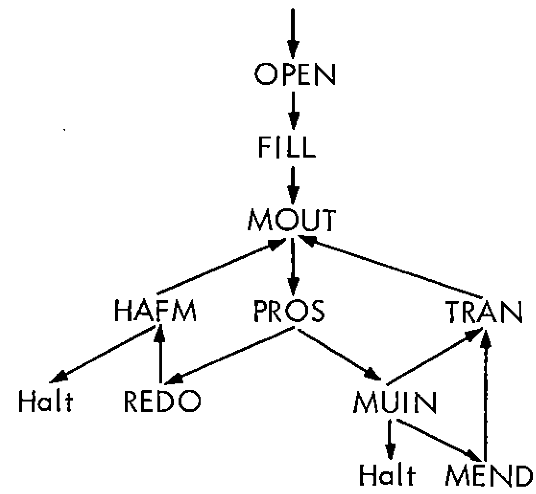


Fig. 16

The program is also modified to guard against a rather disastrous consequence of halving the strip width. Halving, repeated enough times, overflows the μ-file. Both HAFM and MUIN, the latter slightly modified, (see Fig. 17) keep watch on the unfilled space in the μ-file.

HAFM's function is modifying the k-file for the narrower strip. It begins with an attempt to double all the α's in the k-file, which is the correct operation unless the k-file contains an α of 8. If HAFM encounters an α of 8, it sets all the doubled α's back to their original values and then constructs an ℓ-file. This it does when $\alpha < 8$, by a simple transfer of data from the k-file to the ℓ-file with α doubled. When $\alpha = 8$, HAFM transfers and interpolates, producing in the ℓ-file double the number of α's of 8. The completed ℓ-file is converted to a k-file in the usual way.

The operations doubling the strip width are an addition to MEND (see Fig. 19). Two criteria must be met before Δt can be doubled. T_4 must be zero, showing that no α's of 1 are in the k-file, and the sum of all the α's in the k-file must be evenly divisible by 32. Unless the last criterion is satisfied, the k-file does not hold enough points to fill the edge of the array on doubled spacing.

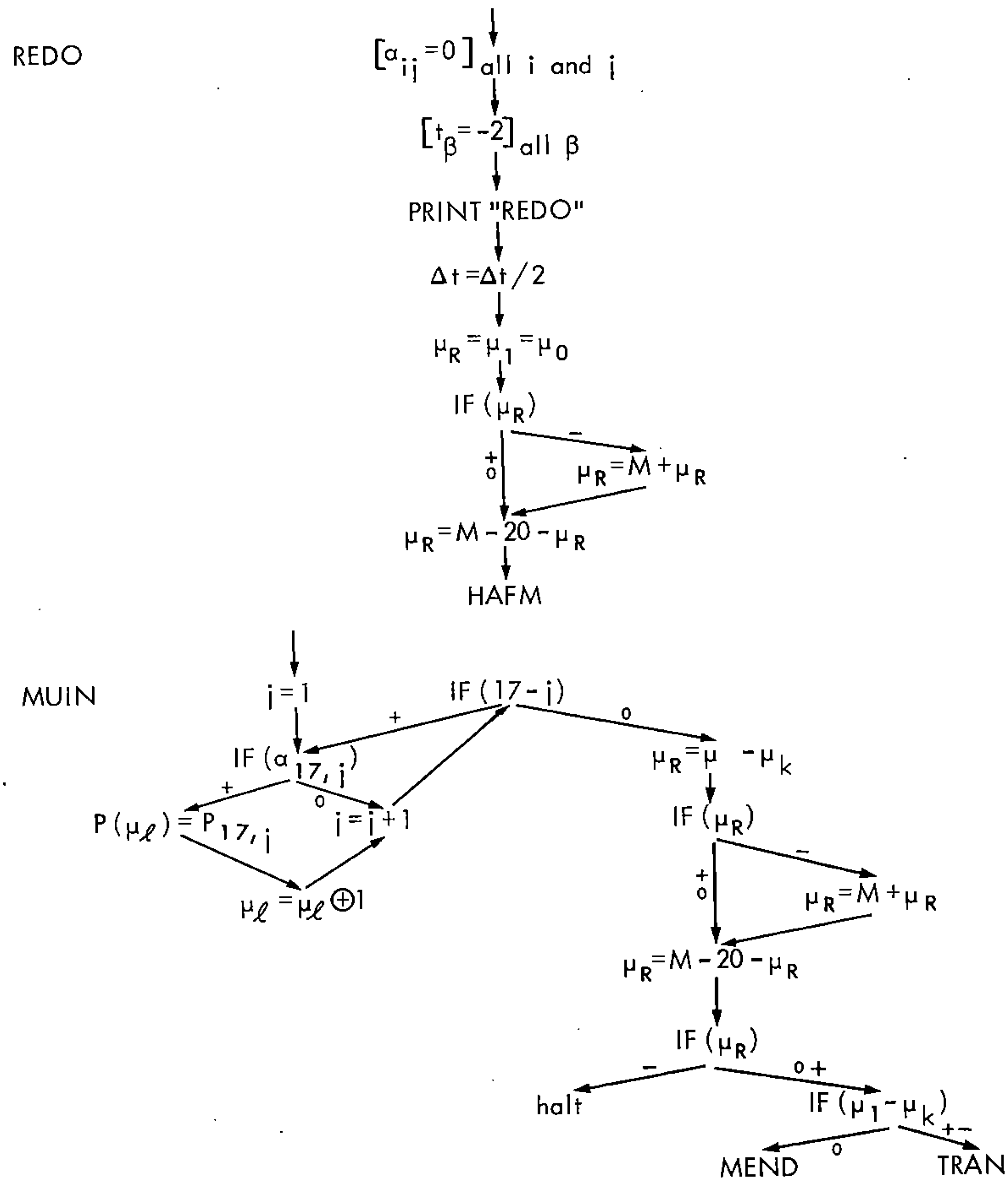


Fig. 17—Subroutines REDO and MUIN. Second version

μ_R is an underestimate of the number of unfilled points remaining in the μ-file.

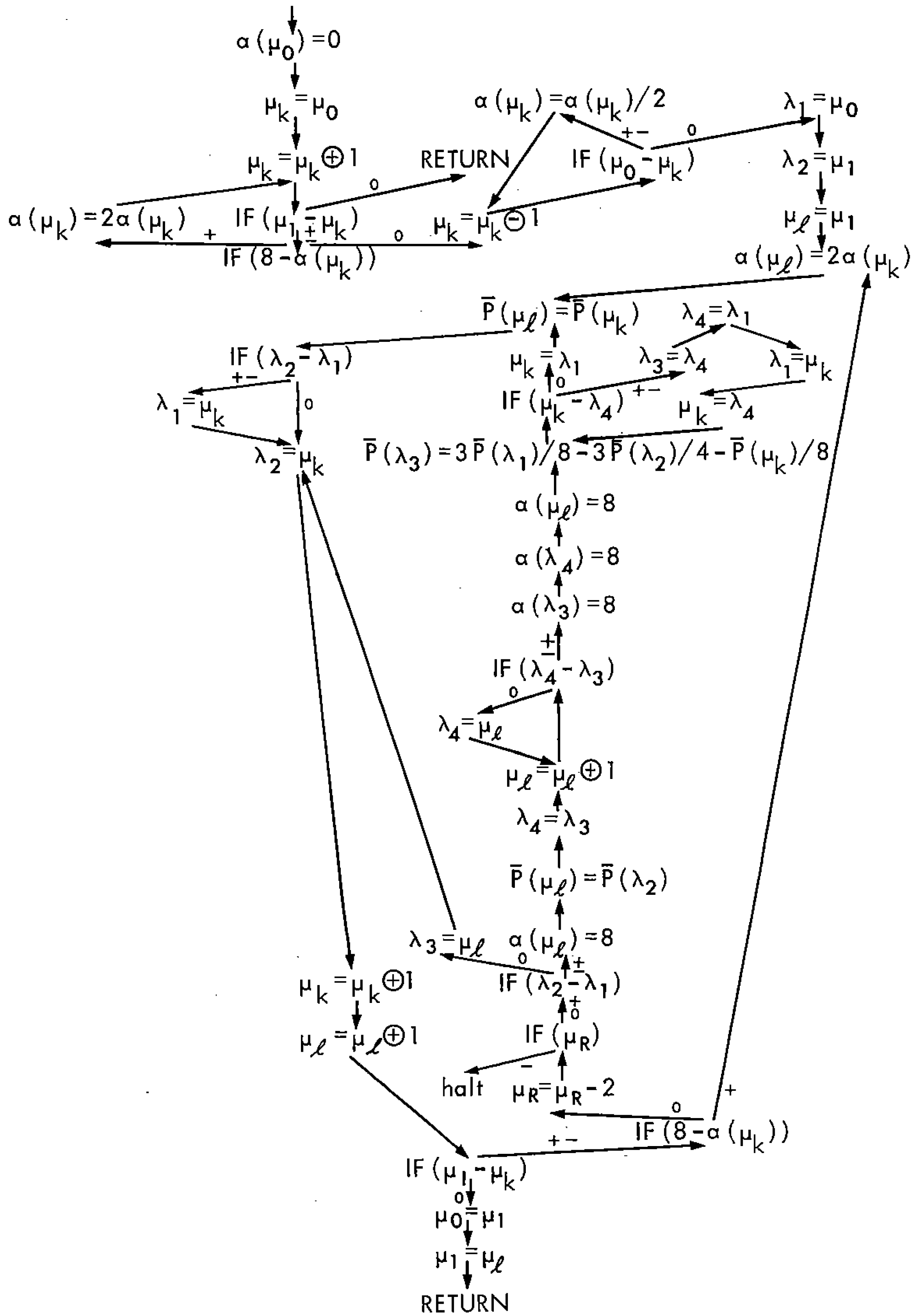


Fig. 18—Subroutine HAFM

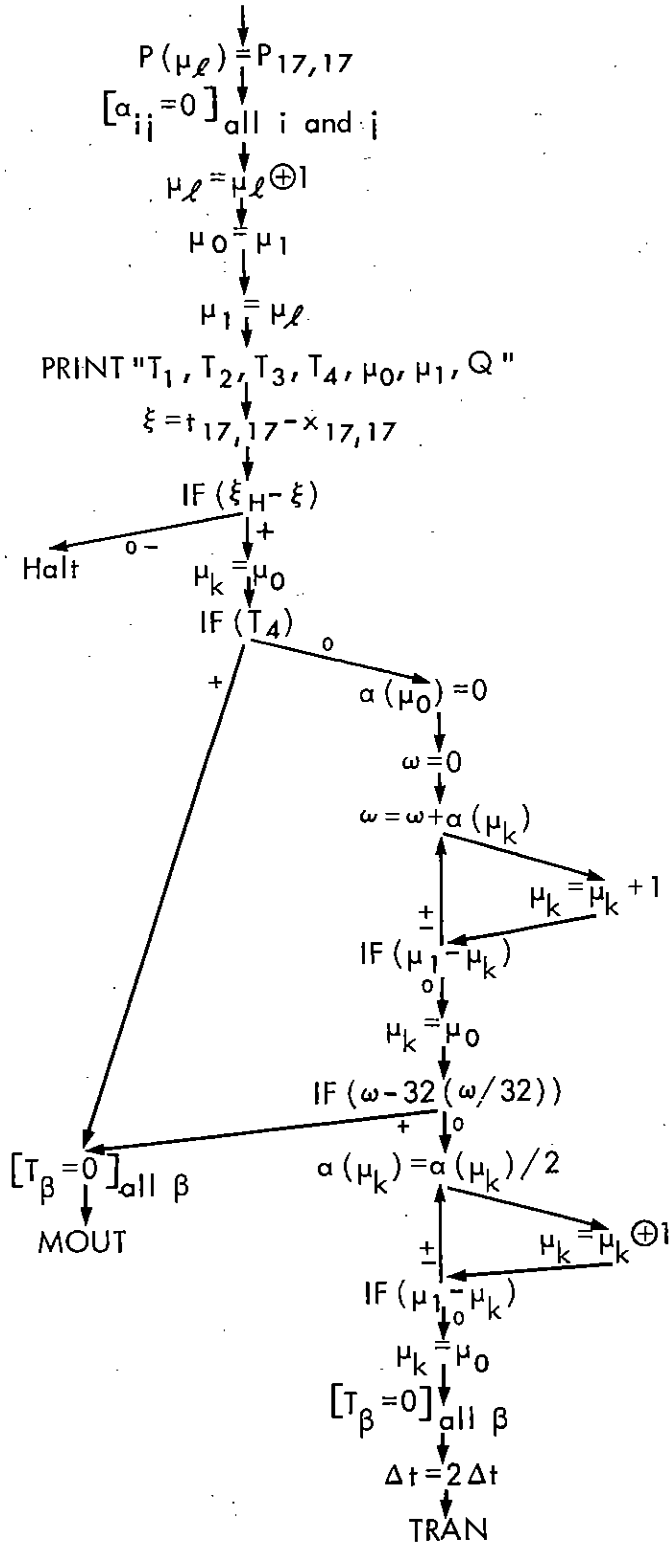


Fig. 19—Subroutine MEND. Second version

Appendix

LISTING OF THE ROUTINES*

```
C     THESE STATEMENTS APPLICABLE TO ALL ROUTINES, BOTH VERSIONS.
      COMMON /AMUX/AMU(8,1000)        /TABLX/TABLE(8,17,17)
      COMMON /MALX/MAL(  1000)        / IALX/  IAL(  17,17)
      COMMON /ITAUBX/ITAUB(4)/ITBX/ITB(4)/THX/TH(18)/VX/V(30)/XGX/XG(18)
      COMMON A,ADIV1,ADIV2,ADIV4,ADIV6,ADIV12,ALBET
      COMMON CAPU,CAPV,CAPW
      COMMON D0,D1,D2,DELTAU,DELX
      COMMON E,EPSLON
      COMMON I,I1,I2,I3,I4,I5,IAB,IABY2,IALFAO,IB,IG,IH,IR,ISB,ISEND
      COMMON ISIG,IST,ITEMP
      COMMON J,J1,J2,J3,J4,J5
      COMMON K000FX
      COMMON M1,MAXSIG,MCAP,MU0,MUK,MUL
      COMMON NE
      COMMON P
      COMMON Q
      COMMON S
      COMMON TP
      COMMON UP
      COMMON VP
      COMMON WP
      COMMON XI,XIH,XP
      COMMON Y0,Y1,Y2
      COMMON Z0,Z1,Z2
C
C     P-BAR QUANTITY NOS., ISIG, AS ORDERED AT THE POINTS MU, NU, AND
C     (I,J) IN THE ARRAYS AMU(ISIG,MU), ANU(ISIG,NU), AND TABLE(ISIG,I,J).
C        QUANTITY= T,X,V',U',W',V,U,W
C            ISIG= 1,2,3 ,4 ,5 ,6,7,8
```

---

*A change of the dependent variables of system (1) from E_R, E_T, and B to xE_R, xE_T, and xB makes the calculations easier. In the following routines

$$u = xE_T - xB,$$

$$v = xE_R,$$

$$w = xE_T + xB.$$

C . . . V E R 1 . . .

```
      SUBROUTINE APP1
      TABLE(7,I1,J1)= TABLE(7,I,J1)+ADIV2 *TABLE(4,I,J1)
      TABLE(6,I1,J1)= TABLE(6,I,J)+ADIV1*TABLE(3,I,J)
      IF(TABLE(2,I,J)-A)1,2,3
    1 CALL HALT (68)
    2 TABLE(8,I1,J1)= -TABLE(7,I1,J1)
      GO TO 4
    3 TABLE(8,I1,J1)= TABLE(8,I1,J)+ADIV2 *TABLE(5,I1,J)
    4 RETURN
      END


      SUBROUTINE APP2
      TABLE(7,I1,J1)=TABLE(7,I,J1)+ADIV4*(TABLE(4,I,J1)+TABLE(4,I1,J1))
      TABLE(6,I1,J1)=TABLE(6,I,J)+ADIV2*(TABLE(3,I,J)+TABLE(3,I1,J1))
      IF(TABLE(2,I,J)-A)4,1,2
    4 CALL HALT (73)
    1 TABLF(8,I1,J1)= -TABLE(7,I1,J1)
      RETURN
    2 TABLE(8,I1,J1)=TABLE(8,I1,J )+ADIV4*(TABLE(5,I1,J)+TABLE(5,I1,J1))
      RETURN
      END


      SUBROUTINE DRIV
      CALL SP(TABLE(1,I1,J1)-TABLE(2,I1,J1),TABLE(2,I1,J1))
      TABLE(4,I1,J1)= TABLE(6,I1,J1)/TABLE(2,I1,J1)-0.5*S*
     1                (TABLE(7,I1,J1)+TABLE(8,I1,J1))
      TABLE(3,I1,J1)=(TABLE(8,I1,J1)-TABLE(7,I1,J1))/TABLE(2,I1,J1)-
     1                S*TABLE(6,I1,J1)-TABLE(2,I1,J1)*P
      TABLE(5,I1,J1)= -2.0*TABLE(6,I1,J1)/TABLE(2,I1,J1)+TABLE(4,I1,J1)
      RETURN
      END


      SUBROUTINE FILL
      MUO    = 1
      MUK    = 1
      AMU(2,MUK)= A
    1 AMU(1,MUK)= AMU(2,MUK)
      MAL(MUK)= IALFAO
      CALL VALU
      MUK    = MUK+1
      IF(M1-MUK)4,3,2
    4 CALL HALT (22)
    2 AMU(2,MUK)= AMU(2,MUK-1)+DELX
      GO TO 1
    3 MUK    = 1
      MAL(MUK) = 0
      MUL    = M1
      CALL MOUT
      END


      SUBROUTINE HALF
      IF(IAL(I,J3))143,36,62
  143 CALL HALT (43)
```

```
36 IAL(I,J3)    = IABY2
   IAL(I,J4)    = IABY2
   DO 52 ISIG=1,MAXSIG
   TABLE(ISIG,I,J3)  = .375*TABLE(ISIG,I,J)+.75*TABLE(ISIG,I,J1)
  1                   -.125*TABLE(ISIG,I,J2)
   TABLE(ISIG,I,J4)  =-.125*TABLE(ISIG,I,J)+.75*TABLE(ISIG,I,J1)
  1                   +.375*TABLE(ISIG,I,J2)
52 CONTINUE
62 IF(TABLE(2,I,J)-A)142,18,64
142 CALL HALT (42)
64 IF(IAL(I3,J))63,39,18
63 CALL HALT (45)
39 IAL(I3,J)    = IABY2
   IAL(I4,J)    = IABY2
   DO 53 ISIG=1,MAXSIG
   TABLE(ISIG,I3,J)  = .375*TABLE(ISIG,I,J)+.75*TABLE(ISIG,I1,J)
  1                   -.125*TABLE(ISIG,I2,J)
   TABLE(ISIG,I4,J)  =-.125*TABLE(ISIG,I,J)+.75*TABLE(ISIG,I1,J)
  1                   +.375*TABLE(ISIG,I2,J)
53 CONTINUE
18 RETURN
   END


      SUBROUTINE HALT (IHALT)
C  THIS ROUTINE PRINTS A 'HALT NUMBER' AND CALLS EXIT.  'HALT NUMBERS'
C  IDENTIFY THE ROUTINE (VIA THE FOLLOWING TABLE) AND THE TRAP (VIA THE
C  CALL STATEMENT) WITHIN THE ROUTINE.  TRAPS ARE LACED THROUGHOUT THE
C  CODE TO DETECT, ABORT, AND IDENTIFY ILLOGICAL BEHAVIOR.  SELECTIVE
C  PRINT-OUTS, OR LIMITED CORE DUMPS, EXECUTE NATURALLY HERE WHEN SUCH
C  BEHAVIOR IS ANTICIPATED.
C
C       TABLE OF HALT NUMBERS..
C                0      1      2      3      4      5      6      7      8      9
C         0             MUIN   HAFM   MUIN   REDO   STOR   STOR   OPEN   MEND   MEND
C        10      VALU   OPEN   JUMP   PROS   PROS   DISC   MEND   REDO
C        20                    FILL                                PROS
C        30
C        40                    HALF   HALF   TEST   HALF   FILL   PROS   PROS   PROS
C        50             MOUT   MUIN          TRAN   NUIN   NUIN   NUIN   NOUT   NOUT
C        60      NOUT          NOUT   NOUT          FIXN   FIXN   HAFM   APP1
C        70      NEND   NEND   NEND   APP2          TEST   TEST   HAFM   HAFM
C        80      HAFN   HAFN   TERP   TERP
C        90      NEND                                                           SP
C
      PRINT 1, IHALT
    1 FORMAT (/33H ***PROGRAMMED HALT***  HALT NO.=I3 )
      CALL EXIT
      END


      SUBROUTINE MEND
      MAL(MUL)= IAL(17,17)
      DO 4 ISIG= 1,MAXSIG
    4 AMU(ISIG,MUL)= TABLE(ISIG,17,17)
      DO 33 I=1,17
      DO 3 J=1,17
    3 IAL(I,J)= 0
   33 CONTINUE
      MUL    = MUL+1
      IF(MCAP-MUL)5,6,6
    5 MUL    = 1
    6 MUO    = M1
      M1     = MUL
```

```
      PRINT 7
    7 FORMAT (3X4HT(1)3X4HT(2)3X4HT(3)3X4HT(4)2X5HMU(0)2X5HMU(1)12X1HQ)
    8 FORMAT (6I7,E13.4)
      PRINT 8,(ITAUB(IB),IB=1,4),MU0,M1,Q
      XI     = TABLE(1,17,17)-TABLE(2,17,17)
      IF(XIH-XI)1,1,2
    1 PRINT 10
   10 FORMAT (//65X5HFINIS//)
      CALL EXIT
    2 MUK    = MU0
      DO 9 IB=1,4
    9 ITAUB(IB)= 0
      MAL(MU0)= 0
      CALL TRAN
      END


      SUBROUTINE MOUT
      NE     = 1
   25 NE     = NE+MAL(MUK)
      IAL(1,NE)    = MAL(MUK)
      DO 28 ISIG=1,MAXSIG
   28 TABLE(ISIG,1,NE)   = AMU(ISIG,MUK)
      MUK    = MUK+1
      IF(MCAP-MUK)26,51,51
   26 MUK    = 1
   51 IF(17-NE)999,29,25
   29 CALL PROS
  999 CALL HALT (51)
      END


      SUBROUTINE MUIN
      J      = 1
   35 IF(IAL(17,J))999,37,36
  999 CALL HALT (52)
   36 MAL(MUL)      = IAL(17,J)
      DO 43 ISIG=1,MAXSIG
   43 AMU(ISIG,MUL)       = TABLE(ISIG,17,J)
      MUL    = MUL+1
      IF(MCAP-MUL)41,37,37
   41 MUL    = 1
   37 J      = J+1
       IF(17-J)20,10,35
   20 CALL HALT (3)
   10 IF(M1-MUK)200,100,200
  100 CALL MEND
  200 CALL TRAN
      END


C            ...OPEN...  THE INITIATING ROUTINE
      DIMENSION IAMU(8,1000)
      EQUIVALENCE (SYMBOL,ISYMBL),(AMU,IAMU)
      MAXSIG= 8
      CALL DVCHK (K000FX)
      GO TO (1,1),K000FX
C  INPUT READ-IN AND PRINT-OUT, BEGIN.
C  BEGIN, STD+MOD INPUT
C
C  NOTE..  THIS INPUT ROUTINE UTILIZES SPACE, AMU(I,J), ASSIGNED FOR
C          LATER USE.  IT REQUIRES A STANDARD DATA INPUT DECK, DEFIN-
C          ING DATA INPUT NAMES AND STANDARD VALUES, AND A MODIFYING
```

```
C           DATA INPUT DECK.  THE MODIFYING DECK CARD FORMAT IS, FOR ALL
C           INPUT, (A6,1X,E12.8)...MODE DISTINCTIONS ARE MADE IN THE
C           CODE.  ADVANTAGES ARE FLEXIBILITY, EASE OF INPUT, AND IN-
C           DIFFERENCE TO SEQUENCE.
C
    1 READ 4,(AMU(1,IN),IN=1,73)
    4 FORMAT (/(6E12.8))
      READ 1100,(AMU(2,IN),IN=1,73)
 1100 FORMAT (12A6)
      READ 2
      READ 2
 1004 FORMAT (A6,1X,E12.8)
      DO 1005 MODIN=1,300
      READ 1004,SYMBOL,VALUE
      IF(ISYMBL)1101,1007,1101
 1101 DO 1103 IN=1,73
      IF(ISYMBL-IAMU(2,IN))1103,1102,1103
 1103 CONTINUE
      CALL HALT (7)
 1102 AMU(1,IN)=VALUE
 1005 CONTINUE
 1009 CALL HALT (11)
 1007 A      = AMU(1,  1)
      DELTAU= AMU(1,  2)
      EPSLON= AMU(1,  3)
      XIH    = AMU(1,  4)
      M1     = AMU(1,  5)
      MCAP   = AMU(1,  6)
      Q      = AMU(1,  7)
      PRINT 2
      DO 1001 IN=8,25
 1001 TH(IN- 7)    = AMU(1,IN)
      DO 1002 IN=26,55
 1002 V(IN-25)     = AMU(1,IN)
      DO 1003 IN=56,73
 1003 XG(IN-55)    = AMU(1,IN)
      IF(TH(1)*XG(1)-1.E7)2000,1009,1009
 2000 DO 1008 IN=1,7
      IF(AMU(1,IN))1009,1009,1008
 1008 CONTINUE
C     END, STD+MOD INPUT
    2 FORMAT (72H
     1
    5 FORMAT (////////////)
    6 FORMAT (/////)
    7 FORMAT (6E20.8)
    8 FORMAT (12I10)
    9 FORMAT(/8X2HM16X4HMCAP)
      PRINT 9
      PRINT 8,M1,MCAP
   10 FORMAT (/19X1HA14X6HDELTAU14X6HEPSLON19X1HQ17X3HXIH)
      PRINT 10
      PRINT 7,A,DELTAU,EPSLON,Q,XIH
   13 FORMAT (/14H TH(I), I=1,18     )
      PRINT 13
      PRINT 7,(TH(I),I=1,18)
   14 FORMAT (/13H V(I), I=1,30      )
      PRINT 14
      PRINT 7,(V(I),I=1,30)
   15 FORMAT (/14H XG(I), I=1,18     )
      PRINT 15
      PRINT 7,(XG(I),I=1,18)
      PRINT 6
C   INPUT READ-IN AND PRINT-OUT, END.
C        * * * INITIALIZATION, BEGIN * * *
      DO 260 IB=1,4
```

```
260 ITB(IB)= -2
    IALFA0= 8
    DELX  = 4.*DELTAU
C     * * * INITIALIZATION, END * * *
    CALL FILL
    END


    SUBROUTINE PRNT
    ER    = VP/XP
    TEMP1 = 2.*XP
    ET    = (WP+UP)/TEMP1
    B     = (WP-UP)/TEMP1
201 FORMAT(/)
 91 FORMAT (9X8E15.7)
 92 FORMAT (1H 22X1HT14X1HX11X4HE(R)11X4HE(T)14X1HB)
    IF(MSAVE-MU0)203,204,203
203 MSAVE = MU0
    GO TO 211
204 IF(ILABEL)211,211,212
211 PRINT 201
    PRINT 92
    ILABEL= 25
212 PRINT 91,TP,XP,ER,ET,B
    ILABEL= ILABEL-1
    RETURN
    END


    SUBROUTINE PROS
    E     = EPSLON*Q
    I     = 1
    J     = 1
    IAB   = 8
    ALBET = 8.0
    IB    = 1
  1 ISEND = 2
1003 I1    = I+IAB
    J1    = J+IAB
    ITEMP = 2*IAB
    I2    = I+ITEMP
    J2    = J+ITEMP
    IABY2 = IAB/2
    I3    = I+IABY2
    J3    = J+IABY2
    ITEMP = 3*IABY2
    I4    = I+ITEMP
    J4    = J+ITEMP
    I5    = I-IAB
    J5    = J-IAB
    GO TO (1001,1002),ISEND
1002 ADIV1 = ALBET*DELTAU
    ADIV2 = ADIV1/2.0
    ADIV4 = ADIV1/4.
    ADIV6 = ADIV1/6.
    ADIV12= ADIV1/12.
    TABLE(2,I1,J1)    = TABLE(2,I,J)
    TABLE(1,I1,J1)    = TABLE(1,I,J)+ADIV1
    CALL APP1
    CALL DRIV
    CALL APP2
    CALL DRIV
    IF(ITB(IB))6,7,8
  6 J     = J1
```

```
      ITB(IB)      = ITB(IB)+2
      GO TO 1
    7 I       = I1
      IF(TABLE(2,I,J)-A)131,9,10
  131 CALL HALT (47)
    9 ITB(IB)      = 1
      GO TO 1
   10 J       = J5
      ITB(IB)      =-1
      GO TO 1
    8 ITB(IB)      =-2
      I       = I5
      J       = J5
      ISEND = 1
      GO TO 1003
 1001 CALL TEST
      IF(ISB)11,19,143
  143 CALL HALT (49)
   11 CONTINUE
C  FAIL TEST.
      IF(4-IB)132,13,12
  132 CALL HALT (48)
   13 IB      = IB-1
      CALL REDO
   12 CALL HALF
      GO TO 18
   19 CONTINUE
C  PASS TEST.
      ITAUB(IB)    = ITAUB(IB)+1
   20 IF(IB-1)134,40,23
   23 IAB     = 2*IAB
      ALBET = IAB
      IB      = IB-1
      IF(ITB(IB))24,25,28
  134 CALL HALT (27)
   28 ITB(IB)      =-2
      I       = I-IAB
      J       = J-IAB
      GO TO 20
   24 J       = J+IAB
      ITB(IB)      = ITB(IB)+2
      GO TO 18
   25 I       = I+IAB
      IF(TABLE(2,I,J)-A)135,26,27
  135 CALL HALT (14)
   26 ITB(IB)      = 1
      GO TO 18
   27 J       = J-IAB
      ITB(IB)      =-1
   18 IAB     = IAB/2
      ALBET = IAB
      IB      = IB+1
      GO TO 1
   40 TEMP1 = ABS(TABLE(7,17,17))
      IF(TEMP1-Q)42,42,41
   41 Q       = TEMP1
   42 TEMP1 = ABS(TABLE(6,17,17))
      IF(TEMP1-Q)44,44,43
   43 Q       = TEMP1
   44 TEMP1 = ABS(TABLE(8,17,17))
      IF(TEMP1-Q)46,46,45
   45 Q       = TEMP1
   46 CALL DVCHK (K000FX)
      GO TO(136,47),K000FX
  136 CALL HALT (13)
   47 CALL MUIN
      END
```

```
      SUBROUTINE REDO
      CALL HALT (4)
      END


      SUBROUTINE SP (XISP,XSP)
C      . . .
C      AS DEFINED.
C      . . .
C  EXPERIENCE DICTATES THE PRESENCE OF THE FOLLOWING DVCHK---
C  A CONSEQUENCE OF THE HIGH CALL FREQUENCY UPON THIS ROUTINE AS WELL
C  AS THE FREQUENT REDEFINITION WITH EACH NEW PROBLEM.
      CALL DVCHK (K000FX)
      GO TO (11,12),K000FX
   11 CALL HALT (99)
   12 RETURN
      END


      SUBROUTINE TEST
      REAL LAG
      LAG(Y0,Y1,Y2)= D0*Y0+D1*Y1+D2*Y2
C  TEST, ENTER.
      CAPU = ADIV12*ABS(TABLE(4,I,J2)+TABLE(4,I2,J2)-2.0* TABLE(4,I1,J2)
     1)
      IF(CAPU-E)50,50,11
   50 CAPV = ADIV6 *ABS(TABLE(3,I ,J)+TABLE(3,I2,J2)-2.0* TABLE(3,I1,J1)
     1)
      IF(CAPV-E)34,34,11
   34 IAL(I1,J2)= IAB
      IAL(I2,J2)= IAB
      IF(TABLE(2,I,J)-A)60,103,61
   60 CALL HALT (44)
   61 CAPW = ADIV12*ABS(TABLE(5,I2,J)+TABLE(5,I2,J2)-2.0* TABLE(5,I2,J1)
     1)
      IF(CAPW-E)2000,2000,11
 2000 IAL(I2,J1)= IAB
C  TEST, EXIT.
C  INTERPOLATION (AT XG, AND TH), ENTER.
      IR    = I1
  200 IG    = 1
      IH    = 1
   19 IF(XG(IG)-TABLE(2,IR,J2))1,3,3
    1 TEMP1 = XG(IG)-TABLE(2,IR,J)
      IF(TEMP1)2,13,13
    2 IG    = IG+1
      GO TO 19
   13 Z0    = TEMP1/ADIV2
      XP    = XG(IG)
      IG    = IG+1
      IST   =-1
      GO TO 12
    3 IF(TH(IH)-TABLE(1,IR,J2))8,4,4
    4 IF(I2-IR)66, 103,5
   66 CALL HALT(76)
    6 CALL HALT (75)
    5 IR    = I2
      GO TO 200
  103 ISB   = 0
      RETURN
```

```
   11 ISB    =-1
      RETURN
    8 TEMP1 = TH(IH)-TABLE(1,IR,J)
      IF(TEMP1)9,10,10
    9 IH     = IH+1
      GO TO 3
   10 Z0     = TEMP1/ADIV2
      TP     = TH(IH)
      IH     = IH+1
      IST    = 1
   12 Z1     = Z0-1.
      Z2     = Z0-2.
      D0     = Z1*Z2/2.
      D1     =-Z0*Z2
      D2     = Z0*Z1/2.
      IF(IST)15,6,14
   14 XP     = LAG(TABLE(2,IR,J),TABLE(2,IR,J1),TABLE(2,IR,J2))
      GO TO 16
   15 TP     = LAG(TABLE(1,IR,J),TABLE(1,IR,J1),TABLE(1,IR,J2))
   16 UP     = LAG(TABLE(7,IR,J),TABLE(7,IR,J1),TABLE(7,IR,J2))
      VP     = LAG(TABLE(6,IR,J),TABLE(6,IR,J1),TABLE(6,IR,J2))
      WP     = LAG(TABLE(8,IR,J),TABLE(8,IR,J1),TABLE(8,IR,J2))
      CALL PRNT
C  INTERPOLATION, END.
      GO TO 19
      END


      SUBROUTINE TRAN
      I      = 1
  105 IAL(I,1)      = IAL(I,17)
      DO 112  ISIG=1,MAXSIG
  112 TABLE(ISIG,I,1)    = TABLE(ISIG,I,17)
      J      = 2
  106 IAL(I,J)      = 0
      IF(17-J)999,108,107
  999 CALL HALT (54)
  107 J      = J+1
      GO TO 106
  108 I      = I+1
      IF(18-I)999,109,110
  110 IF(IAL(I,17))999,111,105
  111 J      = 1
      GO TO 106
  109 CALL MOUT
      END


      SUBROUTINE VALU
      XIFILL= 0.
      CALL SP (XIFILL,AMU(2,MUK))
      AMU(3,MUK)=-AMU(2,MUK)*P
      DO 1 ISIG=4,MAXSIG
    1 AMU(ISIG,MUK)= 0.0
      RETURN
      END


VER1 STANDARD DATA INPUT DECK, BEGIN.
+0        +00+0     +00+0     +00+0     +00+0     +00+1000  +04
+0        +00+1     +06+1     +06+1     +06+1     +06+1     +06
+1        +06+1     +06+1     +06+1     +06+1     +06+1     +06
+1        +06+1     +06+1     +06+1     +06+1     +06+1     +06
```

```
+1        +06+0       +00+0       +00+0       +00+0       +00+0       +00
+0        +00+0       +00+0       +00+0       +00+0       +00+0       +00
+0        +00+0       +00+0       +00+0       +00+0       +00+0       +00
+0        +00+0       +00+0       +00+0       +00+0       +00+0       +00
+0        +00+0       +00+0       +00+0       +00+0       +00+0       +00
+0        +00+1       +06+1       +06+1       +06+1       +06+1       +06
+1        +06+1       +06+1       +06+1       +06+1       +06+1       +06
+1        +06+1       +06+1       +06+1       +06+1       +06+1       +06
+1        +06
A     DELTAUEPSLONXIH     M1    MCAP  Q     TH1   TH2   TH3   TH4   TH5
TH6   TH7   TH8   TH9   TH10  TH11  TH12  TH13  TH14  TH15  TH16  TH17
TH18  V1    V2    V3    V4    V5    V6    V7    V8    V9    V10   V11
V12   V13   V14   V15   V16   V17   V18   V19   V20   V21   V22   V23
V24   V25   V26   V27   V28   V29   V30   XG1   XG2   XG3   XG4   XG5
XG6   XG7   XG8   XG9   XG10  XG11  XG12  XG13  XG14  XG15  XG16  XG17
XG18
VER1 STANDARD DATA INPUT DECK, END.
```

```
C              ...EXAMPLE OF MODIFYING DATA INPUT DECK...
1RUN 112/VER1/7-28-65/TEST CASE
A      =+5         +00
DELTAU=+15625      -01
EPSLON=+5          -02
XIH    =+24999     +01
M1     =+18        +02
Q      =+1         +01
TH1    =+1         +01
TH2    =+15        +01
TH3    =+2         +01
XG1    =+1         +01
000000=                    ENDMOD INPUT
```

```
C                               . . . V E R 2 . . .

C     VER2 IS CONSTRUCTED FROM VER1 AND THE FOLLOWING ROUTINES BY
C     SUBSTITUTION WHEN ROUTINES OF THE SAME NAME APPEAR IN BOTH GROUPS,
C     BY ADDITION WHEN THE ROUTINE APPEARS ONLY IN THE LATTER.

      SUBROUTINE HAFM
      COMMON /VER2X/LAM1,LAM2,LAM3,LAM4,MUR
      MAL(MUO)     = 0
      MUK    = MUO
    1 MUK    = MUK+1
      IF(MCAP-MUK)8,9,9
    8 MUK    = 1
    9 IF(M1-MUK)3,2,3
    3 IF(8-MAL(MUK))151,5,4
  151 CALL HALT (67)
    4 MAL(MUK)=2*MAL(MUK)
      GO TO 1
    5 MUK    = MUK-1
      IF(MUK)152,10,11
  152 CALL HALT (77)
   10 MUK    = MCAP
   11 IF(MUO-MUK)6,7,6
    6 MAL(MUK)= MAL(MUK)/2
      GO TO 5
    7 LAM1   = MUO
      LAM2    = M1
      MUL    = M1
  100 MAL(MUL)       = 2*MAL(MUK)
  110 DO 111 ISIG=1,MAXSIG
  111 AMU(ISIG,MUL)=AMU(ISIG,MUK)
      IF(LAM2-LAM1)120,130,120
  120 LAM1   = MUK
  130 LAM2   = MUK
      MUK    = MUK+1
      IF(MCAP-MUK)131,132,132
  131 MUK    = 1
  132 MUL    = MUL+1
      IF(MCAP-MUL)133,134,134
  133 MUL    = 1
  134 IF(M1-MUK)150,140,150
  140 MUO    = M1
      M1     = MUL
    2 RETURN
  150 IF(8-MAL(MUK))153,20,100
  153 CALL HALT (78)
   20 MUR    = MUR-2
      IF(MUR)21,160,160
   21 CALL HALT (2)
  160 IF(LAM2-LAM1)180,170,180
  170 LAM3   = MUL
      GO TO 130
  180 MAL(MUL)=8
      DO 181 ISIG=1,MAXSIG
  181 AMU(ISIG,MUL)= AMU(ISIG,LAM2)
      LAM4   = LAM3
  190 MUL    = MUL+1
      IF(MCAP-MUL)191,192,192
  191 MUL    = 1
  192 IF(LAM4-LAM3)210,200,210
  200 LAM4   = MUL
      GO TO 190
  210 MAL(LAM3)= 8
```

```
      MAL(LAM4)= 8
      MAL(MUL) = 8
  220 DO 221 ISIG=1,MAXSIG
  221 AMU(ISIG,LAM3)= .375*AMU(ISIG,LAM1)+.75*AMU(ISIG,LAM2)
     1                -.125*AMU(ISIG,MUK)
      IF(MUK-LAM4)240,230,240
  230 MUK   = LAM1
      GO TO 110
  240 LAM3  = LAM4
      LAM4  = LAM1
      LAM1  = MUK
      MUK   = LAM4
      GO TO 220
      END



      SUBROUTINE MEND
      COMMON /VER2X/LAM1,LAM2,LAM3,LAM4,MUR
      MAL(MUL)= IAL(17,17)
      DO 4 ISIG= 1,MAXSIG
    4 AMU(ISIG,MUL)= TABLE(ISIG,17,17)
      DO 3 I=1,17
      DO 3 J=1,17
    3 IAL(I,J)= 0
      MUL   = MUL+1
      IF(MCAP-MUL)5,6,6
    5 MUL   = 1
    6 MUO   = M1
      M1    = MUL
      PRINT 7
    7 FORMAT (3X4HT(1)3X4HT(2)3X4HT(3)3X4HT(4)2X5HMU(0)2X5HMU(1)12X1HQ)
    8 FORMAT (6I7,F13.4)
      PRINT 8,(ITAUB(IB),IB=1,4),MUO,M1,Q
      XI    = TABLE(1,17,17)-TABLE(2,17,17)
      IF(XIH-XI)1,1,2
    1 PRINT 10
   10 FORMAT (//65X5HFINIS//)
      CALL EXIT
    2 MUK   = MUO
      IF(ITAUB(4))11,20,13
   20 MAL(MUO)= 0
      NE    = 0
   21 NE    = NE+MAL(MUK)
      MUK   = MUK+1
      IF(MCAP-MUK)24,25,25
   24 MUK   = 1
   25 IF(M1-MUK)21,22,21
   22 MUK   = MUO
      IF(NE-32*(NE/32))23,12,13
   23 CALL HALT (9)
   11 CALL HALT(8)
   12 MAL(MUK)= MAL(MUK)/2
      MUK   = MUK+1
      IF(MCAP-MUK)14,15,15
   14 MUK   = 1
   15 IF(M1-MUK)12,16,12
   16 MUK   = MUO
      DELTAU= 2.*DELTAU
   13 DO 39 IB=1,4
   39 ITAUB(IB)= 0
      CALL TRAN
      END
```

```
      SUBROUTINE MUIN
      COMMON /VER2X/LAM1,LAM2,LAM3,LAM4,MUR
      J      = 1
   35 IF(IAL(17,J))999,37,36
  999 CALL HALT (52)
   36 MAL(MUL)     = IAL(17,J)
      DO 43 ISIG=1,MAXSIG
   43 AMU(ISIG,MUL)       = TABLE(ISIG,17,J)
      MUL    = MUL+1
      IF(MCAP-MUL)41,37,37
   41 MUL    = 1
   37 J      = J+1
      IF(17-J)998,38,35
  998 CALL HALT (1)
   38 MUR    = MUL-MUK
      IF(MUR)3,4,4
    3 MUR    = MCAP+ MUR
    4 MUR    = MCAP-20-MUR
      IF(MUR)20,10,10
   20 CALL HALT (3)
   10 IF(M1-MUK)200,100,200
  100 CALL MEND
  200 CALL TRAN
      END


      SUBROUTINE REDO
      COMMON /VER2X/LAM1,LAM2,LAM3,LAM4,MUR
      DO 2 I=1,17
      DO 2 J=1,17
    2 IAL(I,J)=0
      DO 403 IB=1,4
  403 ITB(IB)= -2
      PRINT 1
    1 FORMAT (//63X4HREDO//)
      DELTAU= DELTAU/2.0
      MUR    = M1-MUO
      IF(MUR)3,4,4
    3 MUR    = MCAP+MUR
    4 MUR    = MCAP-20-MUR
      CALL HAFM
      END


C   VER2 STANDARD DATA INPUT AND MODIFYING DATA INPUT DECKS IDENTICAL
C   TO VER1.
```