Atomic Energy of Canada Limited

FORSIM: A FORTRAN-ORIENTED SIMULATION PACKAGE FOR THE AUTOMATED SOLUTION OF PARTIAL AND ORDINARY DIFFERENTIAL EQUATION SYSTEMS

by

M.B. CARVER

Chalk River Nuclear Laboratories

Chalk River, Ontario

November 1973

AECL-4608

FORSIM: A FORTRAN-ORIENTED SIMULATION PACKAGE FOR THE AUTOMATED SOLUTION OF PARTIAL AND ORDINARY DIFFERENTIAL EQUATION SYSTEMS

by

M.B. Carver

ABSTRACT

The FORSIM program is a versatile general package for the solution of partial and/or ordinary differential equation systems. The program has been in use as an ordinary differential equation package for some time, both at Atomic Energy of Canada Limited and at several other institutions and an adequate user's manual (AECL-4311) already exists for this application.

The program has now been extended to provide automatic solution of a variety of linear and non-linear partial differential equations. The partial differential equations are converted into a set of coupled ordinary differential equations, which are then integrated in time using an error controlled variable step integration algorithm.

The equations, initial conditions and any special instructions are specified by the user in a single FORTRAN subroutine. This is then loaded with the control routines which perform the solution and any necessary input or output.

This report contains a description of the application of the FORSIM system to partial differential equations, instructions for its use, and several illustrative examples. A more detailed description of the philosophy and content of the FORSIM system has been given in Atomic Energy of Canada Limited report AECL-4311.

Chalk River Nuclear Laboratories
Chalk River, Ontario
November 1973

AECL-4608

FORSIM - Programme de simulation à vocation FORTRAN pour la solution automatique de systèmes d'équations différentielles, partielles et ordinaires

par

M.B. Carver

Résumé

FORSIM est un programme général polyvalent permettant de solutionner des systèmes d'équations différentielles, partielles et/ou ordinaires. Ce programme est employé pour résoudre les équations différentielles ordinaires depuis quelque temps à l'EACL et dans plusieurs autres établissements. Il existe déjà un manuel destiné aux utilisateurs (AECL-4311).

Le programme FORSIM a été agrandi pour permettre de solutionner automatiquement un assortiment d'équations différentielles partielles, linéaires et non-linéaires. Les équations différentielles partielles sont converties en une série d'équations différentielles ordinaires connectées, lesquelles sont ensuite intégrées dans le temps au moyen d'un algorithme d'intégration à pas variable dont les erreurs sont contrôlées.

Les équations, les conditions initiales et les instructions spéciales sont spécifiées par l'utilisateur au moyen d'un sous-programme FORTRAN simple. Celui-ci est alors chargé avec les sous-programmes de contrôle qui établissent la solution et avec toute entrée ou sortie nécessaire.

On trouvera dans ce rapport une description de l'application du programme FORSIM aux équations différentielles partielles, des instructions pour son emploi et plusieurs exemples. Une description plus détaillée de la philosophie et de contenu du programme FORSIM se trouve dans le manuel AECL-4311.

L'Energie Atomique du Canada, Limitée
Laboratoires Nucléaires de Chalk River
Chalk River, Ontario

Novembre 1973

AECL-4608

CONTENTS

FORSIM: A FORTRAN-ORIENTED SIMULATION PACKAGE FOR THE AUTOMATED SOLUTION OF PARTIAL AND ORDINARY DIFFERENTIAL EQUATION SYSTEMS

by

M.B. Carver

1. INTRODUCTION

The study of partial differential equations (PDE's) is a classical branch of mathematical analysis with many important applications in the physical sciences and engineering. Few PDE's have analytical solutions, but with the advent of digital computers, an ever growing number of numerical techniques have been developed for their solution.(2)

Partial differential equations are frequently classified according to their order, linearity and type. The order is the highest order partial derivative which appears in the equation. A linear PDE is of first degree in the dependent variables throughout, a quasilinear PDE is linear in the highest order derivative, and in a nonlinear PDE the coefficients of the highest order derivatives are functions of the dependent variable. Finally the type of equation is determined by analogy with a general equation

$$A\frac{\partial^2 u}{\partial x^2} + 2B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0 \qquad (1)$$

and is called elliptic, hyperbolic or parabolic according to whether the determinant

$$\begin{vmatrix} A & B \\ B & C \end{vmatrix}$$

is positive, negative or zero.

The choice of an appropriate technique to solve a given PDE is governed by its order, linearity, type and the number of independent variables, so the novice is faced with a bewildering number of algorithms, each of which has its associated limits of application, stability problems, and other quirks. For the numerical analyst, the field is intriguing and the literature immense, but the engineer or scientist frequently prefers a method which will supply answers without requiring involved study in the field.

For this reason, there has recently been interest in providing programs which will to some extent automate the approach to the solution of PDE's and thus relieve the user of a great deal of responsibility. Prominent examples of this type of program are PDEL(3), DSS(4), and LEANS(5), all of which use finite difference methods.

PDEL is a precompiler, in which one writes the PDE's in a simple special format language which is then converted into PL/1. DSS is FORTRAN oriented; the PDE's are specified in a normal FORTRAN subroutine. Each of these programs selects an algorithm appropriate to the type of equation to be solved, and so is efficient for certain types of PDE's, but neither can be used for systems of equations.

LEANS, which is also FORTRAN oriented, uses the method of lines for all types of equations. In this method, each PDE is converted into a set of ordinary differential equations, coupled in space, which are then integrated in time. The method easily extends to equation systems, and this feature gives it wide application. The FORSIM program has been used to solve PDE's in a similar way since its inception(1) and the automation of the method requires no changes in program structure.

The current version of FORSIM incorporates some of the features of LEANS together with some improvements in implementation and accuracy. Three major improvements over LEANS are the provision of a five-point finite difference scheme as alternative to the normal three-point scheme, the fact that FORSIM requires only one subroutine to describe the equations whereas LEANS requires a minimum of five, and the provision in FORSIM of an integration routine for stiff equation systems. The first feature gives greater accuracy at the expense of little extra computing time; the second is not only more convenient, but also reduces execution time; and the third can reduce the execution time by several orders of magnitude in extreme cases. These features will be discussed in more detail below.

1.1 THE METHOD OF LINES

The method of discrete ordinates, or method of lines, in which a partial differential equation is transformed into a set of coupled ordinary differential equations by discretizing the space variable, has been studied extensively, particularly in the USSR(2). In most early applications the resulting ordinary differential equations were integrated continuously in time on an analog computer. Sophisticated numerical techniques involving automated error control are now available for the solution of ordinary differential equations and may be used to integrate 'continuously' on the digital computer.

A number of finite difference schemes have been proposed for the digital solution of partial differential equations, and implicit formulae, which remain stable for large time steps, are most useful. For the one-dimensioned heat equation

$$\frac{\partial u}{\partial t} = c^2 \frac{\partial^2 u}{\partial x^2} \qquad (2)$$

it can be shown that one of the more popular implicit formulae, the Crank Nicholson scheme is equivalent to a method of lines solution using three-point spatial difference formulae, and the trapezoidal rule for integrating in time[6]. This is stable and accurate within a truncation error proportional to the square of the spatial increment. The combination of a five-point spatial difference formula with an error controlled time integration algorithm reduces this error to better than the cube of the spatial increment[7].

1.2 THE FINITE DIFFERENCE FORMULAE FOR THE SPATIAL DERIVATIVES

Finite difference formulae for the derivatives of a polynominal f(x) may be developed by considering the Taylor series expansion

$$f(x_k+h) = f(x_k) + \frac{h}{1!} f'(x_k) + \frac{h^2}{2!} f''(x_k) + \ldots + \frac{h^r}{r!} f^{(4)}(x_k) \qquad (3)$$

and the forward difference operator

$$\Delta = f(x_k+h) - f(x_h) \qquad (4)$$

Using the symbol D for the derivative operator (1) becomes

$$f(x_k+h) = (1 + \frac{hD}{1!} + \ldots + \frac{h^r}{r!} D^r) f(x_k) \qquad (5)$$

$$= e^{hD} f(x_k)$$

Combining this with (4) gives

$$\Delta = e^{hD} - 1$$

or

$$hD = \ell n(1+\Delta) \qquad (6)$$

which is expanded by Taylor series to give

$$hD = \Delta - \frac{1}{2}\Delta^2 + \frac{1}{3}\Delta^3 \qquad (7)$$

Thus the derivative D may be obtained by taking the first n terms of (7) with the n+1th term providing an estimation of truncation error.

Similar formulae may be developed for higher order derivatives, and also all derivatives may be expressed in terms of forward or central difference operators. These are discussed in standard texts such as Hildebrand(8) and are summarized concisely in Abramowitz and Stegun(9) which lists the coefficients A_i and truncation errors of the general expansion

$$D^k f(x_k) = \frac{k!}{m!h^k} \sum_{i=0}^{m} A_i f(x_i) \qquad (8)$$

where k is the order of derivative and m+1 is the number of points involved in the expansion.

In FORSIM, spatial derivative formulae are programmed for the three-point (quadratic, m=2) and five-point (quartic, m=4) approximations.

1.3 THE ALGORITHMS FOR INTEGRATION IN THE TIME DOMAIN

Once the PDE's have been transformed by the finite difference formulae into a set of coupled ODE's in time, they may be integrated by any standard algorithm. Integration algorithms have been studied extensively, and the accuracy and stability of a variety of proven routines are described in suitable texts(10,11). In FORSIM the algorithms have been chosen to cover a wide spectrum of requirements and range from the simplest possible fixed step method to intricate variable order, error controlled methods. They are, in order of increasing complexity:

(a) Eulers Method

Integration is performed by the application of the first two terms of the Taylor expansion

$$y(t+h) = y(t) + hy'(t) \qquad (9)$$

with a fixed step size h specified by the user. This first order routine requires only one evaluation of the derivatives y' specified in the UPDATE routine and has a truncation error of order h^2. It is prone to rapid accumulation of error as the integration proceeds, and becomes unstable at large step sizes h.

(b) Trapezoidal or Modified Euler Integration

The trapezoidal rule is normally stated

$$y(t+h) = y(t) + \frac{h}{2}(y'(t)+y'(t+h)) \qquad (10)$$

and this can be approximated by a simple predictor corrector method where the predictor is the Euler formula above

$$y_0(t+h) = y(t) + hy'(t) \qquad (11)$$

and the predictor is used to obtain y'(t+h) as follows

$$y(t+h) = y(t) + \frac{h}{2}(y'(y(t)) + y'(y_0(t+h))) \qquad (12)$$

This is a second order method, requiring two derivative evaluations and has a truncation error of h^3.

The above two routines were added to FORSIM to provide fast simple routines suitable for use while debugging the UPDATE routine and for approximation purposes. The following routines which have been included in FORSIM since its inception, provide greater accuracy at the expense of requiring more derivative evaluations. References 1 and 12 contain a discussion of the relative merits of the routines, their stability and accuracy limits, and advice on the choice of algorithm.

(c) Fourth order Runge Kutta

This method requires four derivative evaluations and is accurate to order h^5. It may be used in fixed or variable step mode. In the latter case the step size is maniuplated to maintain the local error in each variable below a specifiable maximum. This requires a minimum number of eight derivative evaluations per time step.

(d) Adams' Variable Order Variable Step Size Predictor Corrector

This routine is a formulation of the Nordsieck method of step control in the Adams' predictor corrector reported by Gear(13), and has some speed advantages over Runge-Kutta when equations are highly nonlinear. Again, the step size is error controlled.

(e) Gear's Variable Order, Variable Step Size, Predictor Corrector for Stiff Equation Systems

Gear's algorithm(13) is one of the most widely accepted methods for the integration of stiff equation systems containing widely varying time constants. In such systems, standard algorithms are restricted to small step sizes by stability problems, but Gear's method permits large step sizes to be taken, once the initial transients have decayed below the level of significance. Unfortunately this algorithm requires a storage proportional to the square of the number of equations as it requires a matrix inversion. Gear's method is restricted to 70 equations in FORSIM because of storage limitations and the time required to invert the matrix.

(f) The Fowler Warten, Second Order, Variable Step Routine(14)

This routine was also intended to handle stiff equations, and while not so efficient as Gear's, it is a considerable improvement on Runge Kutta for such systems. It is included in FORSIM as a viable alternative to Gear's method, for systems with a large number of equations.

1.4 ACCURACY, COMPUTATION SPEED, AND THE SPATIAL VARIABLE

In partial differential equations the accuracy of the solution obtained will be a function of three factors: the number of spatial divisions chosen, the order of the finite difference approximation of the spatial derivatives, and the algorithm used for temporal integration. These three factors are by no means independent of each other.

The accuracy will, of course, increase with the number of spatial increments, but so does the number of equations, and hence the time required. However, computation time is not a simple function of the number of increments, as the maximum time step attained by the integration algorithm is frequently a function of the size of the space increment. Thus any decrease in the space interval gives a disproportionate increase in computer time.

A much more rewarding method of increasing accuracy is to increase the order of approximation of the spatial derivatives. Early investigations using FORSIM showed that to obtain the accuracy given by the five-point formulae, the three-point formulae require four times the number of spatial division and over ten times as much computing time(15). This is illustrated for a particular application in Figure 1.

When the finite difference formulae were incorporated into FORSIM, it was believed that the five-point formulae represented the optimum trade off between accuracy and computing time. This has been confirmed by Loeb(16), who has investigated the possibilities of linking all the spatial points.

For most applications the combination of eleven spatial points, the three-point difference formulae, and Euler integration should be used for debugging purposes. The accuracy of the solution may then be checked by proceeding to five-point formulae, and then through trapezoidal to variable step Runge-Kutta integration. The number of points should probably be increased only as a last resort.

2. WRITING THE USER'S ROUTINE, UPDATE, FOR PDE'S

2.1 INTRODUCTION

The specification of PDE's in the UPDATE subroutine differs from the ODE case in that the equation itself is not written directly. Instead, one defines the coefficients of a master partial differential equation, which encompasses most types of one-dimensional PDE's:

$$A_1\frac{\partial^2 u}{\partial t^2} + A_2\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}(A_3 u) + \frac{1}{x^c}\frac{\partial}{\partial x}\left[x^c A_4\frac{\partial u}{\partial x}\right] + A_5 u + A_6 \qquad (13)$$

The FORSIM system is designed to solve directly any equation which fits the format of this master equation, but can also be easily adapted to handle equations containing higher order derivatives or further spatial dimensions.

The master equation may have associated boundary conditions at either limit of x, expressed in the form

$$\frac{\partial}{\partial x}(B_1 u) + B_2 u = B_3 \qquad (14)$$

Coefficients A_i and B_i may be constants or functions of u, x, and/or t and the exponent c defines the coordinate system.

The PDE is defined by specifying the coefficients A and B and some control parameters and then executing a call to the FORSIM routine PARTIAL which converts the PDE into a set of coupled ODE's. Linkage to the control routines is accomplished through labelled common blocks.

2.2 OBLIGATORY COMMON BLOCKS AND THEIR PURPOSE

The UPDATE routine must contain the following blocks:

2.2.1 Integrals and Their Derivatives

```
COMMON/INTEGLT/U(10),Y
COMMON/DERIVT/DU(10),DY
```

In the common block INTEGLT, the user reserves space for the current value of integrated variables, then in the same order he reserves space for their time derivatives in block DERIVT. The example indicates an array of variables U and a single variable Y are to have derivatives

$$DU(I) = \frac{d}{dt}(U(I)) \text{ and } DY = \frac{d}{dt}Y.$$

Any symbols may be used and in the current version of the program 300 entries in each block are permitted.

The example shown could be used for an equation system consisting of one partial differential equation (first order in time) to be solved using ten spatial increments, and one ordinary differential equation.

For equations which are second order in time, space must be reserved for the second derivative in the following pattern.

```
COMMON/INTEGLT/U(20),Y,DY
COMMON/DERIVT/DU(20),DY1,DDY
```

This reserves sequential storage for the two derivative orders of the ten spatial approximations of U and also a space for the second derivative of Y. As two common blocks may not contain the same variable, the first derivative is written DY1 in the block DERIVT, and the UPDATE routine must somewhere contain the statement

```
DY1 = DY
```

as well as the definition of DDY. For the PDE, such assignations are done automatically by the routine PARTIAL.

2.2.2 The Block of Reserved Variables

```
COMMON/RESERVD/T,DT,DTMAX,DTMIN,DTOUT,EMAX,...
```

T — current value of the variable t.

DT — current time step in the time integration algorithm.

DTMAX, DTMIN — upper and lower limits set on DT.

DTOUT — printout interval.

EMAX — maximum relative local convergence error for the time integration algorithm.

This block also contains other control variables, and all of these variables except T and DT have default values which may be changed by the user if necessary. For further description, see reference 1.

2.2.3 Printout and Option Control Block

COMMON/CONTROL/IOUT,METHOD,...

IOUT is zero during integration and is set to 1 at appropriate printout intervals as specified by DTOUT. It may have other values (1).

METHOD time integration algorithm selector.

If METHOD=1 Runge-Kutta-Romberg is used (default).
=2 Adam's predictor corrector used.
=3 Gear's predictor corrector used.
=4 Fowler Warten method used.
=5 Euler's method used.
=6 Trapezoidal method used.

2.2.4 PDE Control Block

COMMON/PARTS/NDIF,NCOF,XL,XU,NL,NU,C,DX,X(300)

NDIF spatial accuracy control.

=0 three-point difference formulae are used.
=1 five-point difference formulae are used.

NCOF non-linearity control.

=0 the coefficients A are all constant or functions of time only.
=1 some or all of the coefficients A are functions of the spatial variable x or the dependent variable u.

XL,XU lower and upper limits of the spatial variable x.

NL,NU boundary condition control at upper and lower boundaries.

NL NL=-1 no imposed boundary condition, the derivative at the lower end point will be computed.

NL= 0 normal boundary condition, the value of u at the lower end point will be computed as specified in the boundary coefficients BL.

NL= 1 Non-linear boundary condition in which one or more of the coefficients BL are functions of U(1). In this case the user may choose to iterate. Partial returns a negative value of NCOF if successive approximations to U(1) do not converge within a relative error of EMAX.

C coordinate system control.

=0 cartesian coordinates in x.
=1 cylindrical coordinates in x.
=2 spherical coordinates in x.

DX spatial increment.

X(300) spatial variable. Storage for x is reserved in PARTIAL and made available to the user should he require it in his routine.

2.3 ADDITIONAL OBLIGATORY STORAGE BLOCKS

The user must also reserve storage for the coefficients of equations 1 and 2, which will be passed to the routine PARTIAL.

For linear equations the following statement is adequate:

```
DIMENSION A(6),BL(3),BU(3)
```

The coefficients are arranged such that

$$A(1) = A_1,\ BL(1) = B_1 \text{ at } X=XL, \text{ etc.}$$

For equations in which the coefficients A depend on space, a two-dimensional array must be reserved.

```
REAL A1(6,10),BL(3),BU(3)
```

The second dimension of the array must be the number of spatial increments required. The coefficients may be set using any appropriate logic:

```
A1(3,I) = U(I)**2
```

2.4 THE CALL TO PARTIAL

After all coefficients and controls have been set, the discretization is done by calling the routine PARTIAL as follows:

```
CALL PARTIAL(U,DU,A,N,BL,BU)
```

where N is the number of spatial increments and the arrays U, DU, A, BL, BU have been defined consistently with N as discussed above.

2.5 STRUCTURE OF THE UPDATE ROUTINE

The UPDATE routine will normally be organized in four distinct sections.

Storage Section

The first section will be the storage block containing the common blocks and the coefficient blocks. This may also contain data statements to set any constants.

Initial Condition Section

The first executable statement will normally check the independant variable time and transfer to the initial condition section only if T is zero.

In this section the user should set the initial values of all the variables in the /INTEGLT/ block, plus any other variable controls, or coefficients will remain constant throughout the solution.

Dynamic Section

This section will define any coefficients and controls which are time dependent and will contain calls to PARTIAL. The derivatives of any ODE's will also be evaluated here.

Printout Section

This section should be entered only when IOUT=1, and should contain appropriate output statements. It should also contain a call to the FORSIM routine FINISH to check termination conditions.

Examples of typical UPDATE routines are given in Appendix A.

3. INPUT AND OUTPUT

3.1 INPUT

Data input is quite flexible, in that the user may either utilize the routines supplied or organize his own input. Full details are given in (1), but for most PDE's the following input deck will be sufficient for a single run.

```
          Col.1
          ↓
Card 1:   title alphanumeric
Card 2:   *NOPARS*
          7
          8
          9
```

For more than one run per job, run parameters may be read in using the PARS option.

```
          Col.1      Col.11     Col.21     Col.31
          ↓
Card 1:              Title
          *PARS*
            ALF      1.0        BET        0.
          *PARS*
            ALF      2.0        BET        0.
          *FINIT*
```

The user must then include an additional common block for parameters in the UPDATE routine.

```
COMMON/PARAMS/ALF,BET
```

and the control routines will set ALF and BET at the start of each run. A new title and run number is given as the first output of each run. Pages 14-17 of reference 1 describe the various options available to read input data.

3.2 OUTPUT

3.2.1 Numeric Output

There is an automated output routine which saves the user the trouble of arranging formats, but for PDE's the user will probably prefer to structure his own output. Printing should be done only when IOUT=1 except for debugging runs when a printout of each iteration step may be desired.

3.2.2 Printer Plot

Up to ten printer plots may be obtained, up to five variables per plot (pages 15, 18 reference 1).

4. UTILITY ROUTINES

Fourteen utility routines are loaded with the FORSIM system, and may be called at any time by the UPDATE routine. They include routines for numeric output, printer plot, limits, derivatives, switches, table interpolators, delay functions and transfer functions (pages 17-23, reference 1). Any FORTRAN library routine or CRNL library routine may also be used in UPDATE.

5. ACKNOWLEDGEMENTS

The project has benefited considerably from discussions with Dr. S.Y. Ahmad, Dr. J.M. Blair and Dr. W.N. Selander of Atomic Energy of Canada Limited, Dr. W.E. Schiesser of Lehigh University, Dr. M.A. Loeb of Naval Research Laboratory, Washington and Dr. A.F. Cardenas, University of California, Los Angeles.

6. REFERENCES

(1) Carver, M.B., FORSIM: A FORTRAN Oriented Simulation Package for the Transient Solution of Simultaneous Ordinary Differential Equations, Atomic Energy of Canada Limited Report AECL-4311, February 1973.

(2) Giese, J.H., A Bibliography for the Numerical Solution of Partial Differential Equations, AD730662 US Ar Army Aberdeen R & D Centre, Maryland, 1971.

(3) Cardenas, A.F. and Karplus, W.J., PDEL: A Language for Partial Differential Equations, Comm. ACM, 13, 3 (March 1970), p.184-191.

(4) Zellner, M.G., DSS - Distributed System Simulator, Ph.D. Dissertation, Lehigh University, 1970.

(5) Schiesser, W.E., A Digital Simulation System for Mixed Ordinary/Partial Differential Equation Models, Proc. IFAC Symposium on Digital Simulation of Continuous Systems, 2, p.S2-1 to S2-9, September 1971, Gyor, Hungary.

(6) Zafurullah, A., Application of the Method of Lines to Partial Differential Equations with Error Estimates, J.ACM 17, 2, p.294-302, April 1970.

(7) Hicks, J.S. and Wei, J., Numerical Solution of Parabolic Partial Differential Equations with Two-Point Boundary Conditions by Use of the Method of Lines, J.ACM 14, 3, p.549-562, July 1967.

(8) Hildebrand, F.B., Finite-Difference Equations and Simulations, Prentice Hall, New Jersey, 1968.

(9) Abramowitz, M. and Stegun, I.A., Editors, Handbook of Mathematical Functions, National Bureau of Standards, Applied Mathematics Series, Number 55, Washington, 1965.

(10) Henrici, P., Discrete Variable Methods in Ordinary Differential Equations, Wiley, New York, 1968.

(11) Lapidus, L. and Seinfeld, J.H., Numerical Solution of Ordinary Differential Equations, Academic Press, New York, 1971.

(12) Carver, M.B., FORSIM: A FORTRAN Oriented Simulation Program with Particular Application to the Solution of Stiff Equation Systems, Proceedings, 72 Summer Simulation Conference, SCI, La Jolla, p.73-80.

(13) Gear, C.W., The Automatic Integration of Ordinary Differential Equations, Comm. ACM, March 1971, Vol. 14, No. 3, p.176-180, 185-190.

(14) Fowler, M.E. and Warten, R.M., A Numerical Integration Technique for Ordinary Differential Equations with Widely Separated Eigenvalues, IBM Journal, November 1967.

(15) Carver, M.B., A FORTRAN Simulation System for the General Solution of Partial Differential Equations, Proceedings 1973 Summer Simulation Conference, SCI, La Jolla, p.46-51.

(16) Loeb, A.M. and Schiesser, W.E., Stiffness and Accuracy in the Method of Lines Solution of Partial Differential Equations, Proceedings 1973 Summer Simulation Conference, SCI, La Jolla, p.25-39.

(17) James, M.L. et al, Applied Numerical Methods for Digital Computation with FORTRAN, International Textbook Co., Scranton, Pennsylvania, 1968.

APPENDIX A

EXAMPLES OF TYPICAL UPDATE ROUTINES

The following pages contain descriptions and listings of input decks including control cards, the UPDATE subroutine, and input data for a number of selected examples. They include:

1. The unidimensional heat equation
2. The unidimensional wave equation
3. The fourth-order beam vibration equation
4. The two-dimensional Laplace equation
5. A nonlinear equation.

In all cases, the output from the program was throughly tested, against analytical solutions when available.

Example 1: Unidimensional Heat Equation

$$\frac{\partial u}{\partial t} = c^2 \frac{\partial^2 u}{\partial x^2} , \quad c^2 = K/\rho C_p \qquad (A\text{-}1)$$

This equation describes the temperature u(x,t) in a conductor in which heat is permitted to flow only in the x direction. K is thermal conductivity, ρ density and C_p specific heat.

If this equation is given the boundary conditions

$$u(0,t) = u(1,t) = 0 ,$$

$$u(x,0) = x, \ 0<x<\tfrac{1}{2}; \ u(x,0) = 1-x, \ \tfrac{1}{2}<x<1$$

and

$$C = 1.0$$

The solution may be determined by Fourier series to be

$$u(x,t) = \left(\frac{2}{\pi}\right)^2 \left[\sum_{i=1}^{\infty} \frac{(-1)^m}{n^2} \sin(n\pi x) \ e^{-(\pi n)^2 t} \right]_{(n=2i-1, \ m=i+1)} \qquad (A\text{-}2)$$

The UPDATE routine required to solve equation (1) is shown in Figure 2. The initial conditions and the analytical solution are computed from equation (2) in subroutine EXACT which is not listed.

The storage area reserves space for 11 points and the parameter NDIV is set to 11. The arrays E and ER are not required for the basic solution, but were used to store the analytical value and the error between this and the computed value.

The initial values U(I) are set when time T=0 and, as all the coefficients of the master equation and boundary equations are constant, they are also set here. Coefficients A_2 and A_4 are 1.0 and all others zero. The upper and lower boundary coefficients are identical here so only one array B is required. For U_L=0 the boundary coefficient B_2 is set to 1.0, the remainder are zero.

The call to PARTIAL must be executed each time UPDATE is entered, but the printout section is required only when IOUT≠0.

The parameters in /PARTS/ all have their default values.

Example 2: Unidimensional Wave Equation

Lateral vibrations in an elastic string are governed by the equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \qquad \text{(A-3)}$$

An analytical solution also exists for this equation. The programming of the UPDATE routine is identical to example 1 with the following exceptions.

(1) $A_1 = 1.0$, $A_2 = 0$.

(2) One should reserve space for the second derivative in the common blocks.

```
COMMON/INTEGLT/U (at least 2N)
COMMON/DERIVT/DU (at least 2N)
```

As the system has 300 spaces reserved in each block, this is really only essential when systems of equations are to be integrated and the variable U is followed by another variable.

As the UPDATE routine is so similar, it is not shown.

Example 3: Beam Vibration a Fourth Order Equation

The lateral vibration of a continuous beam follows the equation

$$\frac{\partial^2 y}{\partial t^2} = -\lambda^4 \frac{\partial^4 y}{\partial y^4} \qquad (A-4)$$

This equation does not fit the format of the master equation as it contains a fourth order spatial derivative. However, as the routine PARTIAL may be used to fill any array with the spatial derivatives of any other array, the fourth derivative is easily handled by making two calls to PARTIAL, as shown in Figure 2.

Note that a different array of coefficients has been used for each call to partial. The first call merely sets up the array

$$M = -\frac{\partial^2 y}{\partial x^2} \qquad (A-5)$$

where M is passed in the normal position for $\partial y/\partial t$, so no second derivative in time is required. The second call, however, sets up the second derivative in time.

$$\frac{\partial^2 y}{\partial t^2} = -\frac{\partial^4 y}{\partial x^4} = +\frac{\partial^2 M}{\partial x^2} \qquad (A-6)$$

As the two calls to PARTIAL deal with different arrays, and the equation is second order in time, the user must also ensure that M(I+N) is set equal to Y(I+N) - the current values of $\partial Y(I)/\partial T$ - before executing the second call, as PARTIAL expects the first time derivative in this position. Again a minimum of 2*NDIV storage should be reserved for Y, DY and M.

Figure 3 contains the entire deck required to complete three runs of this problem, and compare against the analytical solution.

For boundary conditions

$$y(0,t) = y(1,t) = y_{xx}(0,4) = y_{xx}(1,t) = y(x,0) = 0$$

the primary mode solution is

$$y(x,t) = \sin(\pi x)\ \cos(\pi^2 \lambda t) \qquad (A-7)$$

This solution is computed at printout time and stored in array E, and the absolute error between analytical and computed values is stored in array ER.

Under the initial conditions, y(x,0) is set to sin πx and $\partial/\partial t(y(x,0))$, stored in y(I+N) is set to zero. The coefficients of the master equation have been set by data statements instead of assignment statements.

The local routine SECOND returns expired CP time. The FORSIM function FINISH prints out a message and starts the next run if time T exceeds 1.0.

The parameter cards on input are set up to run three cases, varying the number of points DIV and the three/five point difference formula selector DIF. To complete the link, the block

```
/PARAMS/ DIV,DIF
```

is used, and as names may not be duplicated in common, the statements NDIV=DIV and NDIF=DIF are required.

Example 4: The Laplace Equation in Two Dimensions

The equation describing the steady-state temperature distribution in a conducting plane is

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \text{(A-8)}$$

At first glance it appears that this equation cannot be solved by FORSIM, but a little thought reveals that it can be done quite readily. As in all such problems, an initial estimate of the temperature distribution is required, so one may treat the problem as a transient, integrating the equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad \text{(A-9)}$$

until the derivative approaches zero.

As this is a two-dimensional problem, one arranges an array U(I,J) = U(X,Y) and completes the equations

$$\frac{\partial}{\partial t}(U(I,J)) = \frac{\partial^2}{\partial x^2}(U(I,J=\text{const})) + \frac{\partial^2}{\partial y^2}(U(I=\text{const},J)) \qquad \text{(A-10)}$$

This is illustrated below.

The sample problem used is to determine the temperature distribution in the square plane shown in Figure 4. These were compared against results obtained using the Gauss-Seidel method on a 10 x 10 square mesh(16).

The axes chosen are shown in Figure 4 and the listing of the UPDATE routine in Figure 5. As the problem is two dimensional, the temperature array U(I,J) requires boundary conditions in both x and y. The boundary conditions in y are simply

$$U(I,0) = 0 \quad , \quad U(I,10) = 100$$

These are specified in arrays BLY and BUY in the initial condition section. The boundary conditions in x depend on the position along the y axis, so must be set within the y loop in the dynamic section.

The entire array U(I,J) is set to an initial value of 50 as a first estimation, and the coefficients A_2 and A_4 are set to 1 to compute only the first derivative in time and the second in space.

In the dynamic section the boundary coefficients for the x direction are specified for each y in arrays BLX and BUY. In the insulated region

$$\frac{\partial u}{\partial x} = 0$$

and outside this region

$$u = 100$$

The first call to PARTIAL calculates $\partial^2/\partial x^2$(U(I,J=const)), for all J, and stores it in array D1. Because of the storage sequence of two dimensional arrays, this can be done only by passing an array U(I=1,10,J) for each J. Therefore, to compute $\partial^2/\partial y^2$(U(I=const,J)) for each I, one must first transpose the matrix U(I,J) into the matrix F(J,I) and then compute $\partial^2/\partial y^2$(F(J,I=const)), storing it in D2. The time derivatives DU(I,J) are then computed by equation A-10.

The complete array is printed out every DTOUT seconds, and one may check the derivatives for sufficient convergence.

Example 5: A Nonlinear Equation

Burger's equation

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \mu \frac{\partial^2 u}{\partial x^2} \qquad \text{(A-11)}$$

is of interest in a variety of fluid flow programs and can be solved analytically. In this equation the coefficient u on the spatial derivative renders the equation nonlinear.

This type of equation is handled readily in the FORSIM nonlinear option. This is invoked by setting the parameter NCOF nonzero. All coefficients in the master equations must now be arrays in the spatial direction, so the coefficient array A is now two dimensional and is declared DIMENSION A(6,n) where n is the number of spatial divisions.

Comparing equation A-11 with the master equation, it is apparent that for all I, A(2,I) = 1 and A(3,I) = μ.

Finally if A(4,I) = -U(I)/2 for each I, the master equation reduces to A-11.

The UPDATE routine is shown in Figure 6.

APPENDIX B

RECENT MODIFICATIONS TO THE FORSIM SYSTEM FOR ODE'S

The Derivative Function DER

The call has been changed to F = DER(A,B,C,N) where the additional parameter N is a unique index.

Field Length Input Block

As the FORSIM program is now entirely in overlay form, the code contains calls to SFL to set the appropriate field length for each overlay. The field lengths provided will be adequate for most applications, but if a large number of routines are added with UPDATE, these field lengths may need to be increased. This is done by supplying two data cards which must precede all other input for FORSIM. They are:

```
          Col.1
          ↓
Card 1    *SFL*
Card 2    Octal number right justified in columns 1 to 10
```

The number on card 2 is an estimate of the additional field length required and is added to the appropriate field length requests before overlays are loaded.

Loading the User Routines

When only UPDATE is loaded, the COPYL method described in reference 1 is adequate.

When other user routines are to be loaded, the COPYN method is again used; however, the COPYN control cards are a little different. On page 24, reference 1, the COPYN control cards should now be

```
1,PLOT,X
UPDATE,SUB2,LGO
2,*,X
```

as the UPDATE routine is now positioned in the middle of the FORSIM file.

Editing the FORSIM Program

The FORSIM system contains a number of utility routines which may not be required for a particular project. The core required to load may be considerably reduced by editing out the larger utility routines or some of the integration algorithms. This may be done readily by using COPYN to select the routines required straight from the object file. It is not necessary to access the source code.

FIGURE 1

ACCURACY AND THE SPATIAL DERIVATIVES IN THE UNDIMENSIONAL HEAT EQUATION

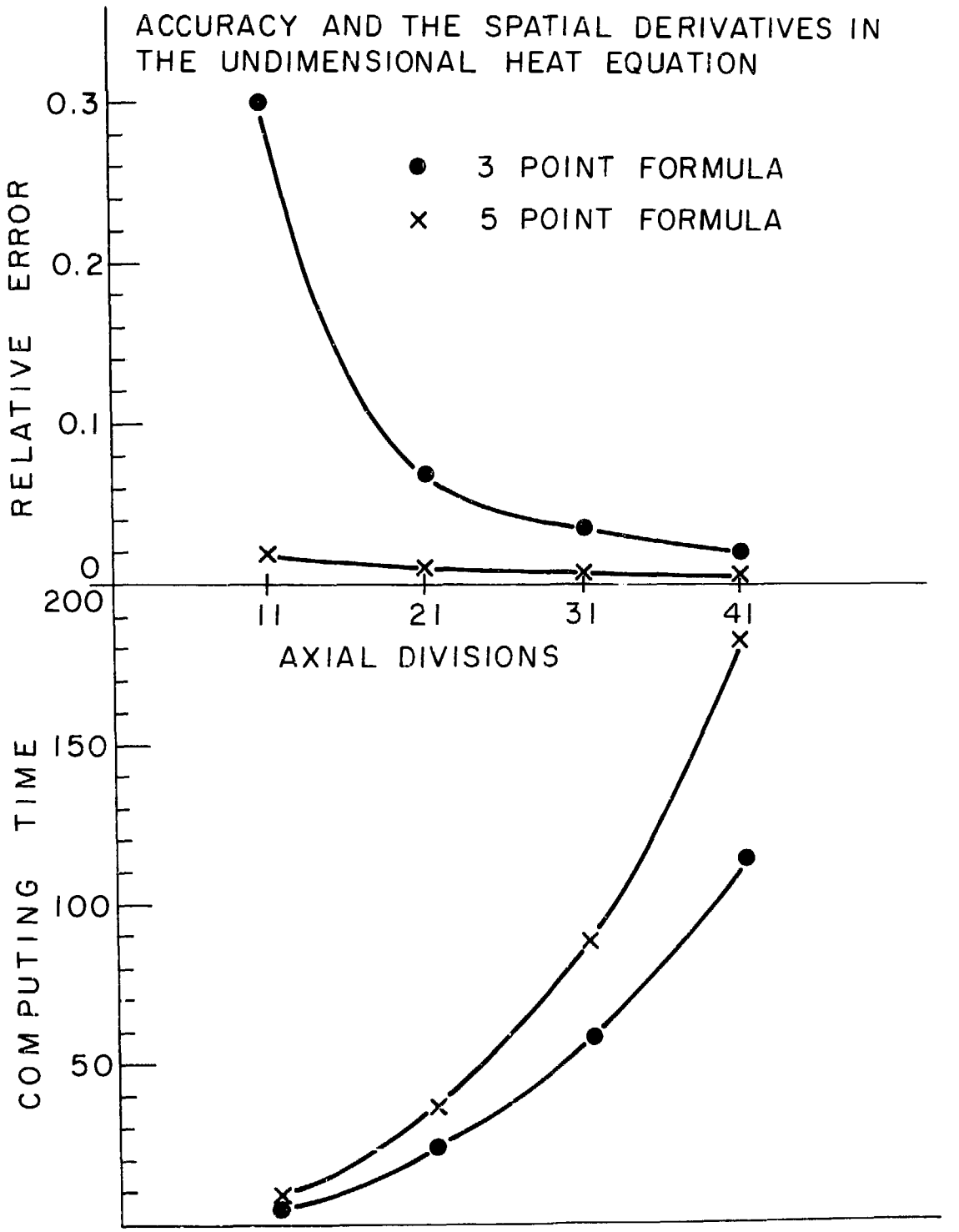

FIGURE 2

UPDATE SUBROUTINE FOR HEAT EQUATION

```
      SUBROUTINE UPDATE
C
C     FORSIM PDE TEST CASE 1
C     SOLUTION OF BASIC HEAT EQUATION  - 1 PARABOLIC EQUATION
C     EQUATION IS  DU/DT=C*D2U/DX2      TAKE C=1.0 AND 0<X<1
C     TAKE IC AS U(X)=X   0<X<0.5     U(X)=1-X   .5<X<1
C
C
C     RESERVED COMMON BLOCK STORAGE
C
      COMMON/INTEGLT/U(11)
      COMMON/DERIVT/DU(11)
      COMMON/RESERVD/T,DT
      COMMON/CONTROL/IOUT
      COMMON/PARTS/NDIF,NCOF,XL,XU,NL,NU,C,DX
      DIMENSION E(11),ER(11),A(6),B(3)
C
      IF(T)10,10,40
C
C     INITIAL CONDITIONS
C
   10 NDIV=11
      CALL EXACT(U,E,ER,T,DX,NDIV,EMX,EAV)
      DO 20 I=1,NDIV
   20 U(I)=E(I)
C
C     SET UP PARAMETERS FOR PARTIAL  (THESE ARE CONSTANT  HERE)
C
      A(1)=A(3)=A(5)=A(6)=B(1)=B(3)=0      B      A(2)=A(4)=B(2)=1.
C
C     CALL PARTIAL TO DISCRETIZE PDE EQUATIONS IN SPACE
C
   40 CALL   PARTIAL(U,DU,A,NDIV,B,B)
C
      IF(IOUT.EQ.0)RETURN
C
C     IF PRINT OUT TIME,CALCULATE EXACT VALUES
C
      CALL EXACT(U,E,ER,T,DX,NDIV,EMX,EAV)
C
C     OUTPUT AND END CHECK
C
      PRINT 100,T,DT,EMX,EAV,(U(I),E(I),ER(I),I=1,NDIV)
  100 FORMAT(1H0,4G12.5,    /(X,3G12.5))
C
      IF(T.GE.3.00)STOP
C
      END
```

```
*
*                              FIGURE 3
*
*       COMPLETE DECK TO RUN 2 CASES OF BEAM VIBRATION EQUATION
*

BEAM,B235-MC,CM1000000,T25.                    JOB CONTROL CARD
FTN.                                           COMPILE THE UPDATE ROUTINE
ATTACH(Y,FORSIM)                               ACCESS THE FORSIM PROGRAM
COPYL(Y,LGO,GO)                                INSERT THE UPDATE ROUTINE
GO.                                            EXECUTE
 00EOR
      SUBROUTINE UPDATE
*
*     FORSIM PDE TEST CASE 3
*     SOLUTION OF THE BEAM VIBRATION EQUATION
*     D2Y/DT2=D4Y/DY4   0<X<1
*     SIMPLY SUPPORTED BEAM WITH INITIAL DEFLECTION Y=SIN(PI*X)
*
      COMMON/DERIVT/DY(44) /INTEGLT/Y(44) /CONTROL/ICUT
      COMMON/RESERVD/T,DT/PARAMS/DIV,DIF
      COMMON/PARTS/NDIF,NCOF,XL,XU,NL,NU,
           REAL AY(6),AM(6),B(3),E(44),ER(44),M(44),E1(44),E2(44)
*
*     DATA FOR PARAMETERS IN PARTIAL
*
      DATA AM/1.0,0.0,0.0,-1.,0.0,0.0/,AY/0.0,1.0,0.0,1.0,0.0,0.0/
     ,,    E,ER,E1,E2,M/44*0,44*0,44*0,44*0,44*0/
     ,,    B/0.0,1.0,0.0/,PI/3.141592/
*
      IF(T)10,10,40
*
*     INITIAL CONDITIONS
*
  10  DX=(XU-XL)/(DIV-1.0)
      NDIF=DIF   $  N=DIV  $  NN=N-1   $  PISQ=PI*PI
      CALL SECOND(T1)
      DO 20 I=1,N
      Y(I)=SIN(PI*DX*(I-1) )
  20  Y(I+N)=0.0
*
*     CALL PARTIAL TWICE  1 = M=D2Y/DX2    ,    2 = D2Y/DT2=D2M/DX2
*
  40  CALL PARTIAL(Y,M,AY,N,B,B)
      DO 50 I=1,N
  50  M(I+N)=Y(I+N)
      CALL PARTIAL(M,DY,AM,N,B,B)
*
      IF(IOUT.EQ.0)RETURN
*
*     IF PRINT OUT TIME COMPUTE EXACT VALUES FOR COMPARISON
*
      EMX=EAV=0
  60  DO 70 I=2,NN
      E(I)=SIN(PI*DX*(I-1))*COS(PISQ*T)
      ER(I)=E(I)-Y(I)      $    E1(I)=-Y(I)*PISQ
      EAV=EAV+ER(I)        $    EMX=AMAX1(EMX,ABS(ER(I)))
  70  E2(I)=-E1(I)*PISQ
      EAV=EAV/(DIV-3)
*
      T2=T1    $    CALL SECOND(T1)    $    SEC=T1-T2
      PRINT 100,T,DT,SEC,EMX,EAV,
     ,(Y(I),E(I),ER(I),M(I),E1(I),DY(I+N),E2(I),DY(I),I=1,N)
 100  FORMAT(*0*5G12.5,/(X,8G12.5))
*
      FIN=FINISH(T,1.0,4HTIME)
      END
 00EOR

 EXAMPLE 4  BEAM VIBRATION EQUATION  D2Y/DT2=D4Y/DX4
*PARS*
 DINV        21.0      DIF         1.
*PARS*
DIV          21.       DIF         0.
*PARS*
DIV          11.       DIF         1.
*FINIT*
A17EOF
```

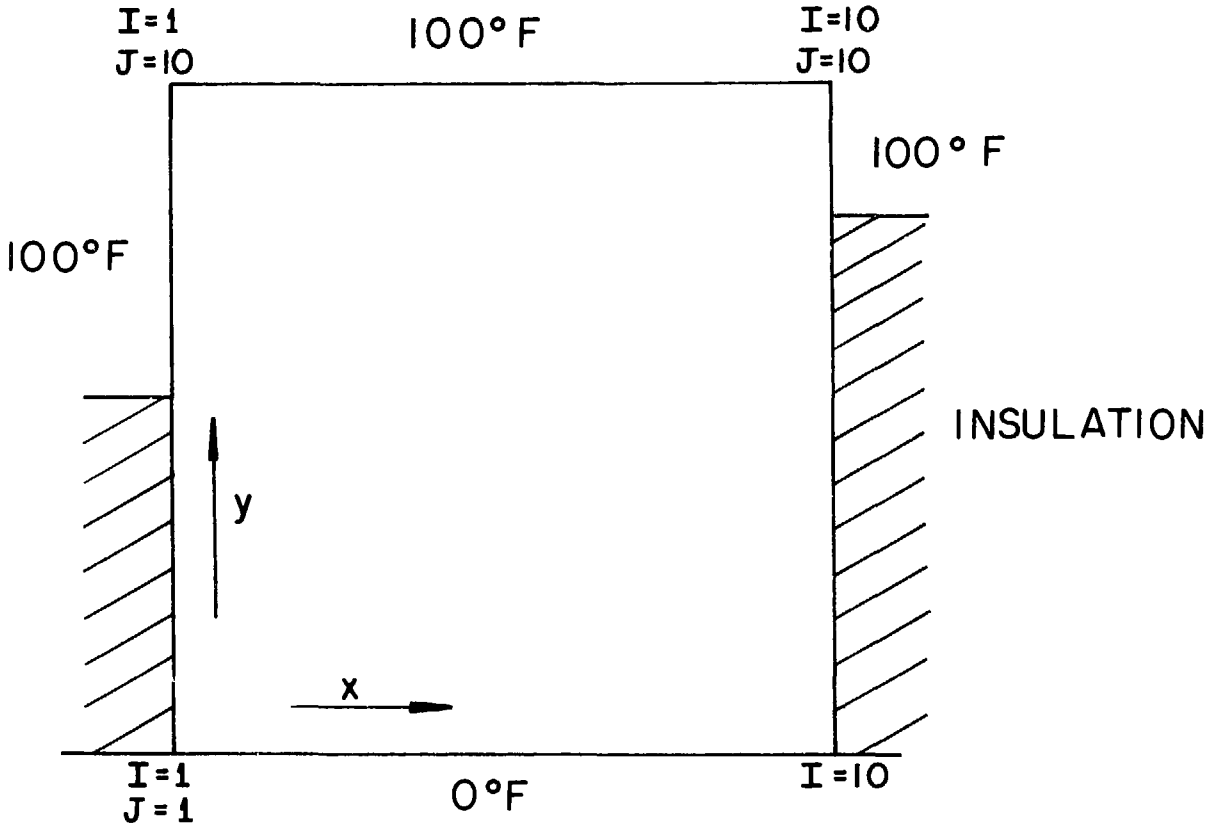

FIG. 4. STEADY STATE TEMPERATURE CALCULATION IN A CONDUCTING MEDIUM

PROBLEM CONFIGURATION FOR EXAMPLE A.

FIGURE 5

UPDATE SUBROUTINE FOR LAPLACE EQUATION

```
      SUBROUTINE UPDATE
C
CC    FCRSIM PDE TEST CASE 4
C     TEMPERATURE DISTRIBUTION IN A SQUARE PLATE
C     LAPLACE EQUATION DU/DT=0=D2U/DX2 + D2U/DY2
C
      COMMON/INTEGLT/U(10,10)
      COMMON/DERIVT/DU(10,10)
      COMMON/CONTROL/IOUT
      COMMON/PARTS/NDIF,NCOF,XL,XU,NL,NU,C,DX
      COMMON/RESERVD/TIME,STEP,DTU,DTL,DTOUT,EMAX,ICHEK
      REAL BLX(3),BUX(3),BLY(3),BUY(3),A(5)
      DIMENSION F(10,10),D1(10,10),D2(10,10)
C
CC    INTIIAL CONDITIONS
C
      DATA BLX,BUX,BLY,BUY  /0.,1.,100.,0.,1.,100.,0.,1.,0.,0.,1.,100./
     ,A                     /0.,1.,0.,1.,0.,0./
      IF(TIME.NE.0.) GO TO 120
      XL=1.0E-99
      DO 100 I=1,100
  100 U(I)=50.
C
CC         DYNAMIC SECTION
C
  120 BLX(1)=BUX(1)=1.   $    BLX(2)=BUX(2)=0.    $    BLX(3)=BUX(3)=0.
      DO 200 J=1,10
      IF(J.LT.5)GOTO130
      BLX(1)=0.          $    BLX(2)=1.           $    BLX(3)=100.
      IF(J.LT.8)GOTO130
      BUX(1)=0.          $    BUX(2)=1.           $    BUX(3)=100.
  130 CALL PARTIAL(U(1,J),D1(1,J),A,10,BLX,BUX)
      DO 200 I=1,10
  200 F(J,I)=U(I,J)

      DO 210 I=1,10
  210 CALL PARTIAL(F(1,I),D2(1,I),A,10,BLY,BUY)

      DO 211 I=1,10
      DO 211 J=1,10
      U(I,J)=F(J,I)
  211 DU(I,J)=D1(I,J)+D2(J,I)
C
CC       OUTPUT SECTION
C
      IF(IOUT.EQ.0) RETURN
      PRINT 1000,TIME,STEP
      PRINT 1000,U,DU
 1000 FORMAT(2X,10G12.5)
      END
```

FIGURE 6

```
            UPDATE SUBROUTINE FOR NON LINEAR EQUATION
      SUBROUTINE UPDATE
C
C     FORSIM PDE TEST CASE 5
C   BURGERS EQUATION  DU/DT+U(DU/DX)=V(D2U/DX2)
C
      COMMON/INTEGLT/U(41)/DERIVT/DU(41)/CONTROL/ICLT/RESERVE/TIME,DT
      COMMON/PARAMS/NDIF,NDIV,METHOD,STEP
      COMMON/PARTS/MDIF,MCOF,XL,XU,FL,NU,C,DX
      DIMENSION A(6,41),X(41),UO(41),ER(41),B(3)
C
C-----INITIAL VARIABLE ASSIGNMENT
C
      DATA V,B0,B1,B2,PI,B/1.,4.,1.,2.,3.1415926374,0.,1.,0./
     ,,                    A/246*0./,ER/41*1./
C
      IF(TIME.NE.0) GO TO 200
      C1=2*V*PI                    ;     C2=-V*PI**2
      N=NDIV                       ;     MDIF=C1
      DX=(XU-XL)/(NDIV-1)          ;     MCOF=1
      M=N-1
C
C   SET CONSTANT COEFFICIENTS
C
      DO 100 I=1,N
      X(I)=DX*(I-1)
      A(2,I)=1.
  100 A(4,I)=V
C
C       INTTIAL AND ANALYTICAL SOLUTION
C
  200 IF(IOUT.EQ.0) GO TO 300
      C3=B1*EXP(C2*TIME)           ;     C4=B2*EXP(4*C2*TIME)
      DO 150 I=1,N
  150 UO(I)=(C1*(C3*SIN(PI*X(I))+2.*C4*SIN(2.*PI*X(I)))/
     1       (B0+C3*COS(PI*X(I))+C4*COS(2.*PI*X(I)))
      IF(TIME.NE.0)GOTO300
      DO 180 I=1,N
  180 U(I)=UO(I)
C
C       SET NON LINEAR COEFFICIENT      AND CALL PARTIAL
C
  300 DO 350 I=1,N
  350 A(3,I)=-U(I)*0.5
      CALL PARTIAL(U,DU,A,N,B,3)
      IF(IOUT.EQ.0) RETURN
C
C       OUTPUT CALCULATED AND ANALYTICAL VALUES AND ERROR
C
      DO 400 I=2,M
  400 ER(I)=ABS(U(I)-UO(I))/UO(I)
      PRINT 1001,TIME,DT
      PRINT 1000,(U(I),I=1,N)
      PRINT 1000,(UO(I),I=1,N)
      PRINT 1000,(ER(I),I=1,N)
 1000 FORMAT(X11G12.5)
      CALL FINISH(TIME,2.0,4HTIME)

      END
```

Additional copies of this document
may be obtained from
Scientific Document Distribution Office
Atomic Energy of Canada Limited
Chalk River, Ontario, Canada
K0J 1J0

Price - $1.00 per copy

2427-73