ORNL/TM-10038

ornl

OAK RIDGE NATIONAL LABORATORY

MARTIN MARIETTA

# On the Use of CORLIB Subroutines for the Solution of a Model Fluid Flow Problem

S. Thompson



OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A04; Microfiche A01

ORNL/TM-10038

Engineering Physics and Mathematics Division

Mathematical Sciences Section

# ON THE USE OF CORLIB SUBROUTINES FOR THE SOLUTION OF A MODEL FLUID FLOW PROBLEM

S. Thompson

Date Published - December 1986

Research sponsored by the
Computing and Telecommunications Division
of Martin Marietta Energy Systems, Inc.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract DE-AC05-84OR21400

CONTENTS

# NOMENCLATURE

| Symbol | Property | S.I. Metric Unit |
|---|---|---|
| $\rho$ | Density | $kg / m^3$ |
| G | Mass Flux | $kg / m^2$–$s$ |
| T | Temperature | C |
| K | Frictional pressure drop coefficient = 10.0D0 | - |
| $g_a$ | Gravitational acceleration = 9.80665D0 | $m / s^2$ |
| $\theta$ | 90° | - |
| $\Phi$ | Heat flux = 1.1D5 | $w / m^2$ |
| $P_H$ | Heated perimeter = 7.97318D+2 | m |
| $A_f$ | Flow area = 3.82760D0 | $m^2$ |
| L | Length of spatial region | m |
| $\bar{T}$ | Absolute temperature = T + 273.15D0 | K |
| $\rho$ | Pressure | MPa ($10^6$ Pa) |
| v | Specific volume | $m^3 / kg$ |
| $h$ | Specific enthalpy | kJ/kg |
| $s$ | Specific entropy | kJ/Kg-K |
| $C_p^{-1}$ | Reciprocal of constant pressure specific heat | kg-K/kJ |
| $\kappa^{-1}$ | Reciprocal of isothermal compressibility | MPa ($10^{-6}$ Pa) |
| $\beta^{-1}$ | Reciprocal of coefficient of volume expansion | K |
| $a$ | Sound speed | m/s |
| $\rho_{sat}$ | Saturation density | $kg / m^3$ |

$$\beta^{-1} = v \left.\frac{\partial T}{\partial v}\right|_{\rho}$$

$$C_p^{-1} = \left.\frac{\partial T}{\partial h}\right|_{p}$$

$$a = \left[\left.\frac{\partial \rho}{\partial \rho}\right|_{s}\right]^{1/2}$$

$$\kappa^{-1} = -v \left.\frac{\partial \rho}{\partial v}\right|_{T}$$

# ON THE USE OF CORLIB SUBROUTINES FOR THE SOLUTION OF A MODEL FLUID FLOW PROBLEM

S. Thompson

Mathematical Sciences Section
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
P.O. Box Y, Bldg. 9207A
Oak Ridge, Tennessee 37831

## ABSTRACT

This reports describes the use of several subroutines from the CORLIB core mathematical subroutine library for the solution of a model fluid flow problem. The model consists of the Euler partial differential equations. The equations are spatially discretized using the method of pseudo-characteristics. The resulting system of ordinary differential equations is then integrated using the method of lines. The stiff ordinary differential equation solver LSODE [2] from CORLIB is used to perform the time integration. The non-stiff solver ODE [4] is used to perform a related integration. The linear equation solver subroutines DECOMP and SOLVE are used to solve linear systems whose solutions are required in the calculation of the time derivatives. The monotone cubic spline interpolation subroutines PCHIM and PCHFE are used to approximate water properties. The report describes the use of each of these subroutines in detail. It illustrates the manner in which modules from a standard mathematical software library such as CORLIB can be used as building blocks in the solution of complex problems of practical interest.

# 1. INTRODUCTION

The core mathematical subroutine library CORLIB has been one of the most popular and widely used mathematical software libraries within MMES for several years. CORLIB consists of a relatively small collection of high-quality software obtained from a variety of sources. The intent of CORLIB is to provide users with a basic set of efficient, well documented, and easy to use mathematical software tools. It contains subroutines to solve the most common problems in the major numerical analysis areas. In particular, it contains software for the solution of systems of linear and nonlinear equations, the integration of systems of ordinary differential equations, calculation of eigenvalues and eigenvectors, nonlinear optimization, numerical quadrature, random number generation, sorting, linear and nonlinear least squares, and spline interpolation. CORLIB is currently used on several computers within the MMES computer network. These computers include the PDP-10, the VAX 8600s, the IBM 3033s, and the CRAY XMP/1. Readers who are not familiar with CORLIB may wish to consult the HELP files available on these computers.

One of the biggest advantages of a standard mathematical software library such as CORLIB is that it provides high-quality modules that can be used as building blocks in the construction of computer programs to solve complex problems. The availability of such modules frees the program developer to concentrate on the solution of his problem without the need to develop and verify software for standard problems. The manner in which this is done is illustrated in this report for a model fluid flow problem. Along the way, the report describes several useful techniques with which some readers may not be familiar.

The model that is used is rather complex. It was chosen since it contains many of the features present in typical fluid flow models. The model consists of an initial value problem in partial differential equations (pdes). The defining pdes are the Euler fluid flow equations. A spatial mesh is first defined. The spatial derivative terms are next replaced by finite difference approximations. The pseudo-characteristic method used in this spatial discretization process is described in Section 2. As a result of the spatial discretization of the pdes, there results a system of ordinary differential equations (odes). This system of odes is solved using the CORLIB module LSODE. Section 3 describes the use of LSODE for the solution. Calculation of the derivatives for the system of odes requires the solution of a system of linear equations at each spatial node. DECOMP and SOLVE from CORLIB are used for this purpose. Their use is also described in Section 3. Very accurate water properties were previously calculated at several spatial points and the resulting values were included as tables in the computer program given in Appendix B. When properties are needed at other points, the monotone cubic spline subroutines PCHIM and PCHFE are used to provide interpolated values. It is possible to calculate the exact solution for the system of pdes. This is done using the ode solver ODE as described in Section 3. Section 4 contains a summary of selected numerical results. Section 5 discusses the solution of other related problems.

# 2. DESCRIPTION OF THE MODEL PROBLEM

The model problem is defined by applying a pseudo-characteristic spatial discretization to the one-dimensional Euler partial differential equations. This results in a system of ordinary differential equations that is solved using the method of lines. Readers not familiar with the method of lines may wish to consult Appendix A which contains a brief

description of this technique. Qualitative aspects of both the original system of pdes and the discretized system of odes are discussed in more detail in [6,7]. This problem is a mock-up of problems similar to those discussed in [8].

The underlying partial differential equations are defined as follows:

$$\frac{\partial U}{\partial t} + A \cdot \frac{\partial U}{\partial z} = C \quad (0 \leqslant t, 0 \leqslant z \leqslant L) \tag{2.1}$$

where

$$U = (\rho, G, T)^T$$

$$C = \left[0, -KG \mid G/\rho \mid -\rho g_a \sin\theta, \frac{a^2 \Phi P_H \kappa}{C_p A_f}\right]^T$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{\rho\kappa} - \frac{G^2}{\rho^2} & 2\frac{G}{\rho} & \frac{\beta}{\kappa} \\ \frac{-a^2\beta\bar{T}G}{\rho^2 C_p} & \frac{a^2\beta\bar{T}}{\rho C_p} & \frac{G}{\rho} \end{bmatrix}$$

and $a, \kappa, \beta, C_p = f(T, \rho)$ (Equation of State).

The boundary conditions are

$$\rho(0, t) = \rho_0 = 795.521$$

$$T(0, t) = T_0 = 255.000$$

$$G(L, t) = G_0 = 270.900 \ .$$

The eigenvalues of $A$ are

$$G/\rho \ , \ G/\rho + a \ , \text{ and } G/\rho - a \ .$$

Equation (2.1) may be expressed in characteristic form by multiplying by a matrix $B$ for which

$$B \ A \ B^{-1} = D$$

where $D$ is the diagonal matrix whose diagonal elements are the above eigenvalues. One such matrix is given by:

$$B = \begin{pmatrix} \beta a^2 \bar{T} & 0 & -\rho C_p \\ -G\kappa a + 1 & \rho\kappa a & \rho\beta \\ G\kappa a + 1 & -\rho\kappa a & \rho\beta \end{pmatrix} .$$

The resulting characteristic form of the defining equations is then:

$$B \cdot \frac{\partial U}{\partial t} + D \cdot B \cdot \frac{\partial U}{\partial z} = B \cdot C \quad .$$

At each node of the spatial mesh $\{z_1, \cdots, z_{M+1}\}$, one-sided differences are calculated for the spatial derivatives:

$$\rho_{z,0}, G_{z,0}, T_{z,0}$$

$$\rho_{z,+}, G_{z,+}, T_{z,+}$$

$$\rho_{z,-}, G_{z,-}, T_{z,-} \quad .$$

(Throughout this paper, $M+1$ will denote the number of spatial nodes.)

The first subscripts denote differentiation with respect to $z$. The second subscripts (0,+,-) indicate that the direction of the spatial differencing is dictated by the signs of the local characteristics $G/\rho$ , $G/\rho + a$ , $G/\rho - a$, respectively, at the node. Backward differences are used if the sign of the characteristic is positive; otherwise forward differences are used. At each node, there results a system of three linear equations whose solution yields the corresponding time derivatives for the ode solver. (We point out that these linear systems are very badly conditioned. For example, iterative refinement usually will not converge for the systems.) The linear system is given by

$$B \cdot (d\rho_i/dt \ , dG_i/dt \ , dT_i/dt)^T = E$$

where $B$ is evaluated at $z$ . using $\rho_i$ , $G_i$ , and $T_i$ ; and $E = (E_1 , E_2 , E_3)^T$ with

$$E_1 = \sum_{j=1}^{3} B_{1j} \cdot C_j - (G_i/\rho_i) \cdot \{B_{11}\rho_{z,o} + B_{12} G_{z,o} + B_{13} T_{z,o}\} ,$$

$$E_2 = \sum_{j=1}^{3} B_{2j} \cdot C_j - (G_i/\rho_i + a_i) \cdot \{B_{21}\rho_{z,+} + B_{22} G_{z,+} + B_{23} T_{z,+}\} ,$$

$$E_3 = \sum_{j=1}^{3} B_{3j} \cdot C_j - (G_i/\rho_i - a_i) \cdot \{B_{31}\rho_{z,-} + B_{32} G_{z,-} + B_{33} T_{z,-}\} .$$

Arbitrary values may be defined for the initial values of $\rho_i$ , $G_i$ , and $T_i$. The problem of interest is then to integrate until the steady-state solution of the discretized solution is obtained. The initial conditions used in this report were obtained by using a linear rise for the $T_i$'s , calculating the corresponding values of $\rho_i$ using a constant pressure, and using $G_i\ (z\ ,0) = G_0$.

It is necessary to distinguish between the steady-state solution of the pdes and the solution of the odes. For the pdes, we have $G = G_0$ constant at steady-state. As will be discussed in Section 3, the system of pdes may thus be reduced in this case to a system of two odes (in $z$) whose solution determines the steady-state spatial profiles for $\rho$ and $T$. The behavior of the discretized system is actually very different from that of the exact solution for the pdes. Each of $\rho_i(t)$, $G_i(t)$, and $T_i(t)$ is damped and oscillatory. For example, the maximum magnitude of the oscillation for $G_1$ is about ten times larger than the actual steady-state solution value. (The discretized solution thus exhibits many of the characteristics commonly observed for transient problems.) When two-point spatial differences are used (as they are in this report), the steady-state spatial values of $\rho_i$, $G_i$, $T_i$ for the discretized system are monotone in $z$.

The discretized system is stiff. One negative eigenvalue has magnitude approximately $O(10^5)$. The other eigenvalues are shifted from the imaginary axis into the left half-plane (due to the damping that is implicitly introduced into the system by the differencing scheme; see [6]). The stiffness of the system roughly doubles each time the number of spatial nodes is doubled.

A block-wise ordering may be used to order the variables, that is,

$$Y = (G_1,\ \cdots,\ G_M,\ T_2,\ \cdots,\ T_{M+1},\ \rho_2,\ \cdots,\ \rho_{M+1})^T \quad .$$

The nonzero structure of the Jacobian matrix is shown in Figure 1. With this ordering, the Jacobian matrix $J = (\partial F / \partial Y)$ has upper and lower bandwidths of $2(M+1)$. The total bandwidth is $4M+5$. The nonzero elements of the Jacobian matrix belong to five tridiagonal strips. The upper diagonals begin in locations $(i\ ,j) = (1,2)$, $(1,M+1)$, $(1,2M+1)$, $(M+1,3)$, and $(2M+1,3)$. Since the number of equations is $N = 3M$, the Jacobian matrix seen by the ode solver for this ordering is nearly a full matrix. (The number of zero elements outside the band is $M \cdot (M-1)$. The percentage of elements within the band is thus $100(8/\ 9 + 1/\ M) \approx 90\%$ for large $M$.)

Since the system of equations is stiff, LSODE must approximate the Jacobian matrix using numerical differences. The number of derivative evaluations required to do this is equal to the total bandwidth of the Jacobian matrix. It is therefore important to reorder the variables to minimize the bandwidth. To do this, the equations may be reordered in a node-wise fashion, that is,

$$Y = (G_1, G_2, T_2, \rho_2, G_3, T_3, \rho_3, \ldots,\ G_m, T_m, \rho_m, T_{m+1}, \rho_{m+1})^T \quad .$$

With this ordering, the nonzero elements of the Jacobian matrix all belong to a smaller band about the main diagonal. In fact, the Jacobian matrix for this ordering has upper and lower bandwidths of 5 and a total bandwidth of 11. Since the bandwidth is reduced from $4M+5$ to 11, LSODE will have to do much less work for the second ordering.

Although it is more efficient to use the node-wise ordering, it is nevertheless more convenient to think in terms of blocks of variables in the actual computer program. Consequently, each time LSODE passes a node-wise ordered solution to the derivative subroutine and requests that the corresponding derivatives be calculated, the solution array is first copied into a local array and reordered using the block-wise ordering. The block-wise ordered derivatives are then calculated. They are then node-wise reordered before passing them back to LSODE. The manner in which this is done is described in the next section.

We point out that for large values of $M$, it is desirable to exploit the system sparsity. In such cases, it is more appropriate to use sparse variants of LSODE. The LSODES [3] solver and the LSOD28 [1] solver are two such integrators available at ORNL. Use of LSODES and LSOD28 for the solution of the present problem are discussed in [5]. The results for several available integrators are discussed in [6].

## 3. CORLIB IMPLEMENTATION

This section describes the manner in which the various CORLIB modules are used in the solution of the problem in question. Appendix B contains a FORTRAN listing of the computer program used to solve the problem. An abbreviated flowchart of the program is depicted in Figure 2. FLOSLV is the name of the main program. FLOSLV performs the necessary initializations, calls LSODE to perform the ode integration, and generates output.

Let us consider the manner in which LSODE is used. The call sequence for LSODE as implemented in FLOSLV is as follows:

```
      CALL LSODE (DERIVS, NEQ, Y, TIN, TOUT, ITOL, RTOL, ATOL,
     *            ITASK, ISTATE, IOPT, WORK, LWORK, IWORK,
     *            LIWORK, DERIVS, MF)
```

The parameters in the call to LSODE are as follows:

LSODE requires the user to supply a subroutine that calculates system derivatives. The name of this subroutine in the present program is DERIVS. DERIVS must be declared in an EXTERNAL statement in the calling program. It has the following form:

```
      DERIVS (NEQ, T, Y, YDOT)
```

Given the number of odes NEQ, the current value of the independent variable T, and an approximation to the solution Y, DERIVS must calculate the corresponding derivatives YDOT. DERIVS must not change NEQ, T, or Y. (A common mistake is to change Y.) Observe that Y and YDOT are each vectors of length NEQ. The manner in which DERIVS calculates the system derivatives for the present problem will be described in more detail after the other parameters have been described.

NEQ is the number of odes. Since there are $M+1$ spatial nodes, three discretized equations at each node, and three boundary conditions, there are NEQ = $3M$ odes in the system to be solved.

Y is the vector with components Y(1), ... ,Y(NEQ). Before LSODE is called the first time, the initial values of the solution variables must be loaded into this vector. The initial values are defined in subroutine INITAL in the present program. On return from

LSODE, Y will contain the solution at the current value of the independent variable.

TIN is the initial value of the independent variable and TOUT is the next value of the independent variable at which the solution is to be calculated. Observe that LSODE is called in a loop in FLOSLV. TOUT is updated to the next desired value each pass through the loop.

ITOL, RTOL, and ATOL are error tolerance parameters used to indicate to LSODE the type of error control that is desired. LSODE uses a mixed error test in which the estimated per step error in component I is controlled relative to the quantity

RTOL*ABS(Y(I)) + ATOL.

Thus if Y(I) is near zero, the test provides absolute error control and for larger values of ABS(Y(I)), the test provides relative error control. Observe that if ATOL = 0.0, pure relative error control is used and if RTOL = 0.0, pure absolute error control is used. Normally both ATOL and RTOL are scalars in which case the user must define ITOL = 1. However, LSODE also allows ATOL to be a vector in which case the calling program must set ITOL = 2 and define the values ATOL(1), ... ,ATOL(NEQ). This allows different absolute error tolerances to be used for individual components of the solution. In the present program, both tolerances are scalars and are set equal to EPS = 1.0D-5.

ITASK is a flag used to instruct LSODE what it is supposed to do. Before the first call to LSODE, FLOSLV defines ITASK = 1 to initialize LSODE and does not change ITASK on any succeeding call. This simply instructs LSODE to integrate past the next output point and interpolate to obtain the solution at that point. LSODE allows various other options such as integrating exactly to the output point or not integrating past certain critical points. The interested reader is referred to the complete documentation for LSODE for further information regarding ITASK. Normally ITASK = 1 is used since LSODE is more efficient in this case.

ISTATE is a flag used to initialize LSODE and to indicate the status of the integration. Before calling LSODE the first time or on any call for which it is desirable to re-start the integration, the calling program must define ISTATE = 1. If LSODE is able to integrate successfully to TOUT, it will return a value of ISTATE = 2. LSODE checks for various abnormal conditions such as improper input (for example, NEQ less than 1) and for difficulties that arise during the integration (for example, non-convergence of the corrector iteration, or failure to satisfy the error test even after several reductions of the step-size). If an abnormal condition is detected, LSODE returns an appropriate value of ISTATE. The CORLIB documentation for LSODE contains a complete description of the conditions flagged by LSODE and the corresponding values of ISTATE. It is a good idea to check the value of ISTATE after returns from LSODE and to take appropriate action if the normal value of 2 is not returned. In the present case, FLOSLV terminates the integration if this occurs.

LSODE allows the user to input certain optional parameters. IOPT is a flag that is used to indicate if this is desired. If no optional parameters are input to LSODE, the calling program must set IOPT = 0. If any other value is input for IOPT, LSODE assumes that the user has also defined the following optional inputs:

WORK(5) = step-size to be attempted on the first step. Normally, LSODE calculates this value.

WORK(6) = maximum allowable step-size.

WORK(7) = minimum allowable step-size.

IWORK(1) = lower bandwidth ML of the Jacobian matrix. ML is 5 for the present problem.

IWORK(2) = upper bandwidth MU of the Jacobian matrix. MU is 5 for the present problem.

IWORK(5) = the maximum allowable integration order. This value is normally 5 for a stiff problem.

IWORK(6) = the maximum allowable number of steps LSODE can take before returning to the calling program. This value is normally 500.

IWORK(7) = maximum number of printed error messages that can be generated by LSODE. There is normally no limit placed on this value.

The user thus has a great degree of optional control over the integration. If the user does not want to change the default value for a given parameter, he can simply set the corresponding component of WORK to zero or the corresponding component of IWORK to zero. For the present problem, FLOSLV sets the bandwidth parameters ML and MU. It inputs an artificially high value for the maximum number of steps to force the integrator not to return before reaching TOUT. It essentially instructs LSODE to generate messages for any errors that are detected. In addition, it instructs LSODE to use an initial step-size of 1.0D-8.

The amount of work space required by LSODE depends on the size of the problem being solved. A double precision work array WORK with length at least

$$\text{LWORK} = 22 + 10*\text{NEQ} + (2*\text{ML} + \text{MU})*\text{NEQ}$$

is required for a banded problem that is stiff. For the present problem, the length of WORK must be at least

$$\text{LWORK} = 22 + 25*\text{NEQ} = 22 + 75*M .$$

In addition, an integer work array IWORK with length at least

$$\text{LIWORK} = 20 + \text{NEQ}$$

is required. The complete documentation for LSODE contains a description of the necessary lengths of WORK and IWORK for other types of problems and solution options.

The user has the option to input an analytic Jacobian matrix if the exact Jacobian matrix is known. In this case the user must provide a subroutine JAC to calculate the exact Jacobian matrix. For the present problem, FLOSLV instructs LSODE to use an internally generated approximation to the banded Jacobian matrix by setting the flag MF = 25. Since JAC is not used by LSODE for this value of MF, FLOSLV simply supplies a dummy name to LSODE for JAC, namely the name of the derivative subroutine DERIVS.

Let us now consider the manner in which subroutine DERIVS calculates the system derivatives for LSODE. The coding for the calculation of system derivatives should look as much like the written equations as possible. Consequently, DERIVS is merely an interface which copies the vector Y into local storage with names that resemble those in the written equations. For this problem, DERIVS first reorders the vector Y using the block-wise ordering described in the last section. It then calls another routine DERVAL that receives the various portions of the Y array using the local solution names AG, AT, and AR, and the local derivative arrays AGD, ATD, and ARD. After the return to DERIVS from DERVAL, the derivatives are reordered using the node-wise ordering described in the last section and are then returned to LSODE.

DERVAL directs the actual calculation of the system derivatives. It calls VARSET to copy AG, AT, and AR into the local storage arrays G, TF, and RHO. Note that these arrays are each of length $M+1$ since they contain the boundary condition values that are being used. If non-constant boundary conditions were being used, they would next be loaded into these local arrays appropriately. Water property routines would next be called to calculate the local properties. In the present program, the initial interpolated properties are used (that is, the properties are not a function of time); therefore it is not necessary to re-calculate them. Subroutine SPATEL is next called to calculate the finite difference approximations for the spatial derivatives. (SPATEL calls an upwind difference routine to calculate the necessary forward and backward differences.) Subroutine PSEUDO is next called to assemble and solve the linear equations defined in the last section. The advantage of coding the solution in the above modular fashion may be seen by inspection of the coding in subroutine PSEUDO: the "equations" in that subroutine are identical to the defining equations described in the last section. On return to DERVAL from PSEUDO, subroutine DYSET is called to load the local derivative arrays DG, DTF, and DRHO into the appropriate portions of the system derivative vector.

Subroutine PSEUDO contains a loop. It successively sets up the 3 by 3 linear equation at each node and calls the linear equation subroutines DECOMP and SOLVE to solve the equation. Note that after this is done for each spatial node, PSEUDO then overrides the calculated derivatives that correspond to the given boundary conditions.

Let us consider the manner in which DECOMP and SOLVE are used to solve the 3 by 3 linear systems. DECOMP is first called to factor the matrix into a product of simpler triangular matrices. SOLVE is then called to solve the linear system using the factorization computed by DECOMP. The call to DECOMP from subroutine PSEUDO is as follows:

```
CALL DECOMP (IF, NF, F, COND, IPVT, WORK)
```

The parameters in the call to DECOMP are as follows:

IF is the row index in the DIMENSION statement for F in the calling program. Here, a value of IF = 3 is used. If it were desirable to solve systems of different sizes using the same work space for the matrix F, this could be done by allocating enough space in F to

accommodate the largest system to be solved and setting IF to this value. For example, suppose we wished to solve a 3 by 3 system in one call and a 10 by 10 system in another call. We would dimension F(10,10) and set IF = 10. When the 3 by 3 case was solved, we would load F in the first three rows and columns of F and set the system dimension size NF equal to 3.

NF is the size of the linear system to be solved. Here, NF = 3. Confusion regarding IF and NF is common. All one must remember is that NF refers to the dimension of the coefficient matrix of the mathematical linear system while IF refers to the actual first index in the FORTRAN DIMENSION statement for this matrix.

On input, F is the NF by NF coefficient matrix of the linear system. On output, F contains information that describes the factorization of F. This information will be used in the subsequent call to SOLVE and must not be altered before that call.

On output, COND is an estimate of the condition number of F. It provides an estimate of how poorly the matrix F is conditioned. For the linear system F*X = E, changes in F and E may cause changes that are COND times as large in the solution X. Roughly speaking, COND thus provides a measure of how sensitive the solution is to changes in the coefficient matrix and in the right-hand side vector. For the present problem, F is very poorly conditioned in some cases, as indicated by values of COND ranging from 1.0D4 to 1.0D7.

IPVT is an integer vector that must be dimensioned in the calling program for size at least NF. DECOMP performs row interchanges to maintain numerical stability. IPVT is used to record these interchanges. When SOLVE is subsequently called, IPVT is used to perform the same interchanges in the solution vector. The contents of IPVT must not be altered between the calls to DECOMP and SOLVE.

WORK is a scratch array that must be dimensioned in the calling program for size at least NF.

Now let us consider the second step of the linear solution. The call to SOLVE from subroutine PSEUDO is as follows:

```
CALL SOLVE (IF, NF, F, E, IPVT)
```

The parameters in the call to SOLVE are as follows:

IF and NF are the same values that were input to DECOMP.

F contains the factorization information calculated by DECOMP.

IPVT contains the row-interchange information recorded by DECOMP during the factorization of the original matrix F.

On input, E contains the right-hand side vector of the linear system F*X = E. On output, E contains the solution X.

The monotone cubic spline routines PCHIM and PCHFE (which were taken from the PCHIP spline package) are used to approximate water properties in the computer program given in Appendix B. Very accurate properties were previously calculated using 71 spatial

points and a water-property package. The resulting values were loaded in DATA statements in subroutine SETIC of the present program. Initial values for temperature were calculated using a linear rise. The resulting temperature profile was used along with a constant pressure to calculate the initial densities. The resulting temperature and density values were also loaded into DATA statements in SETIC. During the initialization of the problem, SETIC is called by the main program. SETIC in turn calls PCHIM to calculate the coefficients of spline fits for each of the water properties and for the initial values of the dependent variables. It then calls PCHFE to evaluate the splines at each spatial location. The resulting property values are then held constant during the remainder of the integration. In the program given in Appendix B, it is also possible to use water properties that are constant in both space and time. Of course, in an actual production computer code, available property tables or routines would be used to calculate the time-dependent properties. This is not done in the present program in order to avoid the necessity of using a water property package.

Let us consider the use of the spline routines in more detail. For example, consider the first call to PCHIM in SETIC. The call list is as follows:

```
CALL PCHIM (NZ, ZIC, RONIT, D1, INCFD, IER)
```

The parameters in the call to PCHIM are as follows:

NZ is the number of data points. For this problem, NZ = 71.

ZIC is an array of size NZ. On input to PCHIM, it contains the abscissae for the data points. For this problem, ZIC(I) is the location of the spatial node corresponding to the ordinate RONIT(I) .

RONIT is an array of size NZ. On input to PCHIM, it contains the ordinates for the data points. For this problem, RONIT(I) is the value of density at the spatial location ZIC(I).

D1 is an array that must be dimensioned at least NZ in the calling program. The coefficients for the spline fit are stored in D1 for later use by PCHFE.

INCFD is the increment between data points to be used in the fit. In certain applications such as two-dimensional plotting, it is sometimes convenient to force PCHIM to use, say, every other data point. INCFD would be 2 in this case. For the present problem, INCFD is 1.

IER is a flag returned by PCHIM to indicate to the user whether or not an abnormal condition (e.g., not enough data points, or the abscissae not supplied in increasing order) is detected. The CORLIB documentation for PCHIP contains a complete description of the possible values for IER.

Now consider the call to PCHFE. The call list is as follows:

```
CALL PCHFE (NZ, ZIC, RONIT, D1, INCFD,
*           SKIP, MP1, Z, RHO, IER)
```

The parameters in the call to PCHFE are as follows:

NZ, ZIC, RONIT, and INCFD are the parameters that were input to PCHIM.

D1 contains the spline coefficients that were calculated by PCHIM.

SKIP is a logical variable. If SKIP = .FALSE., PCHIM will check the validity of the preceding parameters. If the user is confident that each of the parameters is valid, he can input SKIP = .TRUE., in which case PCHIM will bypass the validity checks. (This is very convenient when PCHFE will be used more than once for the same set of data.)

On input to PCHIM, MP1 is the size of the Z array and of the RHO array. For this problem, MP1 = $M+1$ is the number of spatial nodes used in the discretization of the pdes.

On input to PCHIM, Z is the array of abscissa values at which PCHIM is to calculate the spline fit. For this problem, Z contains the $M+1$ spatial nodes used in the discretization of the pdes.

On output from PCHIM, RHO is the array of spline values corresponding to Z. For this problem, these values will be used for the initial densities at the $M+1$ spatial nodes used in the discretization of the pdes.

IN SETIC, the next calls to PCHIM and PCHFE are used to calculate the initial temperatures in a similar fashion. Spline fits are then calculated for each of the relevant water properties. Observe that the coefficients are saved for each of these fits. This is the usual manner in which PCHIM and PCHFE are used. PCHIM is called once to calculate the spline coefficients for a given set of data. These coefficients are saved for possible later use any time it is necessary to evaluate the corresponding spline fit. For the present problem, the property-related spline coefficients are later used in subroutine PRSPL, in a manner to be described below.

It is possible to calculate the exact steady-state solution for the pdes defined in (2.1). This is due to the fact that the continuity equation embedded in (2.1) implies that $G$ is constant at steady-state. To see this, observe that at steady-state, $\partial\rho/\partial t = 0$ implies $\partial G/\partial z = 0$ so $G(z)$ is a constant, say $G_0$. In this case, the defining partial differential equation may be reduced to:

$$\begin{pmatrix} \frac{1}{\rho\kappa} - \frac{G_0^2}{\rho^2} & \frac{\beta}{\kappa} \\ \frac{-a^2\beta\,\bar{T}G_0}{\rho^2 C_p} & \frac{G_0}{\rho} \end{pmatrix} \begin{pmatrix} d\rho/dz \\ dT/dz \end{pmatrix} = \begin{pmatrix} -KG_o|G_o/\rho| - \rho g_a \sin\theta \\ \frac{a^2\Phi P_H \kappa}{C_p A_f} \end{pmatrix}$$

This constitutes a system of two first-order odes with independent variable $z$. For given values of $\rho$ and $T$, the corresponding linear system may be solved to obtain $d\rho/dz$ and $dT/dz$. Given $\rho(0)$ and $T(0)$, the system may, therefore, be integrated in $z$ (i.e., in space) to obtain the steady-state spatial profiles of $\rho$ and $T$. The three pdes can thus be reduced to a system of 2 odes where the independent variable is $z$. This system of odes can be integrated to determine the steady-state spatial profiles for the solution variables. The system of odes is not stiff. In order to illustrate the use of the non-stiff ode solver ODE in CORLIB and to illustrate how far the solution for the discretized system of odes deviates from the steady-state solution for the pdes, the computer program in Appendix B

also integrates the above two odes.

Let us now consider the manner in which ODE is used to solve the above system of two odes. The main program calls subroutine PRSPL which in turn calls ODE. The call to ODE is as follows:

```
CALL ODE (FODE, NODE, YODE, T, TOUT, RELERR, ABSERR,
*         IFLAG, WORK, IWORK)
```

The parameters in the call to ODE are as follows:

FODE is the name of the subroutine that ODE will call to calculate the system derivatives. FODE must be declared in an EXTERNAL statement in the calling program. FODE has the form:

```
SUBROUTINE FODE (T, RHOTF, DRHOTF)
```

Here, T represents the independent variable (which is now $z$, the original spatial variable). RHOTF is an array of length 2 which contains the solution values for temperature and density for this value of the independent variable. Given T and RHOTF, FODE must calculate the corresponding system derivatives.

NODE is the number of odes in the system to be solved. In this case, NODE = 2.

Since the independent variable is now $z$, the original spatial variable, ODE will thus integrate from the $z = 0.0$ to $z = L$. On input to ODE, YODE is a vector of length 2 that contains the initial values for temperature and density, that is, the values at $z = 0.0$. These values were loaded into the arrays TPDE and RPDE before the call to TRGDIF.

T is the initial value of the independent variable, in this case, $z = 0.0$.

TOUT is the next value of the independent variable at which the solution is desired. In TRGDIF, ODE is called in a loop. Each time through the loop, TOUT is set to the value of $z$ corresponding to the next spatial node.

RELERR and ABSERR are error tolerance parameters. ODE uses a mixed error test in which the per step error in component I is controlled relative to the quantity

```
RELERR*ABS(YODE(I)) + ABSERR.
```

Thus if YODE(I) is near zero, the test provides absolute error control and for larger values of ABS(YODE(I)), the test provides relative error control. Observe that if ABSERR = 0.0, pure relative error control is used and if RELERR = 0.0, pure absolute error control is used. In the present program, a value of 1.0D-12 is used for both ABSERR and RELERR to insure the "exact" solution is computed very accurately.

IFLAG is a flag used to communicate the status of the integration. Before calling ODE for the first time, the calling program must set IFLAG = 1 in order to initialize ODE. The normal output value of IFLAG is 2. This value indicates that ODE successfully completed the integration to TOUT. If an abnormal condition is detected by ODE, an appropriate value of IFLAG is returned. For example, ODE attempts to diagnose stiffness. If it determines the system is stiff, it returns a value of IFLAG = 5. ODE also attempts to determine

if the error tolerances are too small or if too many integration steps are required to solve the problem. The CORLIB documentation for ODE contains a complete description of the possible values of IFLAG.

WORK is a scratch work array that will be used by ODE. It must be dimensioned for at least 100 + 21*NODE = 142 in the calling program. IWORK is an integer scratch array that must be dimensioned for at least 5 in the calling program.

Since ODE will request that derivatives be evaluated at any point between $z = 0$ and $z = L$, and each such evaluation will require water property values, it is necessary for FODE to be able to approximate the water properties at all intermediate values of $z$. To do this, FODE calls subroutine PRSPL. PRSPL in turn calls the spline evaluation routine PCHFE to approximate these properties. Recall that the coefficient calculation routine PCHIM was used in subroutine SETIC to calculate the coefficients of each of the property-related spline fits. The resulting coefficients were saved in arrays D3, D4, D5, and D6 and these arrays were passed to PRSPL through a labeled COMMON block. Hence, it is not necessary to call PCHIM each time ODE requests a derivative evaluation. Since most of the computing time for a spline fit is spent in the calculation of the coefficients, this is a significant savings.

The calculation of the derivatives in FODE requires the solution of a 2 by 2 linear system. DECOMP and SOLVE are used to solve this system in a manner similar to the linear solution in PSEUDO.

## 4. DISCUSSION OF NUMERICAL RESULTS

In this section, we will briefly discuss selected results obtained for the solution of the model problem. The problem was solved for values of $M = 5$, 10, 20, and 40 on a VAX 8600. Table 1 contains a summary of selected integrator results. In particular, it includes the total number of derivative evaluations, the number of Jacobian evaluations, and the number of integration steps required to solve the problem for each value of $M$. In addition, it includes the maximum derivative magnitude in the computed steady-state for the discretized system of odes. The values indicate that LSODE did indeed integrate to the steady-state solution of the discretized system in each case.

Tables 2-4 contain a summary of the calculated steady-state values at six spatial points for mass flux ($G$), temperature ($T$), and density ($\rho$), respectively. The tables also include the exact solution to the pdes at these spatial points. The results illustrate the manner in which the solution of the discretized system of odes can differ from the exact solution of the original pdes. Recall that the derivative magnitudes in Table 1 indicate that the solutions in Tables 2-4 are indeed the steady-state solutions for the corresponding discretized equations. The difference between these solutions and the exact solution for the pdes is due to the spatial discretization error and not to an error by LSODE in the time integration. For each of the variables, the discretized solutions are converging (with an order of convergence equal to 1) to the exact solution as the mesh-size is successively halved. However, a relatively large number of nodes is required for a solution with a small spatial discretization error. For example, 41 spatial nodes are required to reduce the error in the calculated inlet mass flux to about 10%. (81 nodes are required for an error of about 5% in the calculated inlet mass flux.)

It is possible to use fewer nodes by increasing the parameter NPT in the computer program to 3 or 4. This amounts to using higher order spatial difference approximations. (The order of the spatial difference approximations using NPT points in the differences is NPT-1.) However, the bandwidth of the Jacobian matrix also increases in this case. The model problem is actually a mockup of a portion of the subcooled liquid region of a three-region steam generator model. In the full model, it is necessary to link the solutions for the three regions at the boundaries of the second region. Increasing the number of points used in the spatial differences increases the complexity of linking the regions appropriately. For example, if NPT = 3 or 4, the solutions tend not to be spatially monotone while they tend to be monotone for NPT = 2. (For NPT = 3 or 4, the solution for the mass flux tends to have a "kink" near the upper boundary $z = L$.) This compounds the difficulty of linking the models for the different regions at their common boundary. These considerations illustrate some of the kinds of tradeoffs in efficiency versus accuracy that must be made in solving problems of this type.

## 5. OTHER PROBLEMS

The boundary conditions specified for this problem ($T$ and $\rho$ specified at $z = 0$ and $G$ specified at $z = L$) are not the only ones of interest. For example, suppose we wish to specify $G$ and $\rho$ at $z = 0$ and $T$ at $z = L$. Due to the modular structure of the program, it is straightforward to modify the program in Appendix B to accommodate the new boundary conditions. If one performs these modifications (an interesting and worthwhile exercise) and runs the resulting program, a steady-state solution is not obtained. This comes as no surprise since, in general, there is no reason to assume that one can integrate to a steady-state from an arbitrary initial guess. In particular, for the present case, there is not enough damping introduced by the spatial difference scheme used. Reflections in $G$ at $z = L$ are propagated back into the interior of the domain making it impossible for the ode solver to integrate to the steady-state solution.

This problem is an example that illustrates the need for ingenuity when faced with the task of solving such problems. One technique that works for this problem is to perform a continuation-like solution on the heat flux $\Phi$. For the present problem, a constant value of 1.1D5 was specified for $\Phi$. The steady-state solution can be obtained by starting with $\Phi = 0.0$, obtaining the corresponding solution, and increasing $\Phi$ incrementally until the desired value of 1.1D5 is obtained. For the initial guess for the solution, we can set all densities equal to the inlet value at $z = 0$ and all temperatures equal to the outlet temperature at $z = L$. Since $\Phi = 0.0$ corresponds to no heat addition into the region, this guess is very near the solution for $\Phi = 0.0$ and the ode solver can easily integrate to the solution - which then provides a good guess for the solution corresponding to the next value of $\Phi$.

It is possible to obtain a more efficient solution for this problem by using a nonlinear equation solver in a similar fashion. We are interested in finding the steady-state solution of an initial value problem

$$\begin{aligned} dy/dt &= f(t,y) \\ y(t_o) &= y_o \ . \end{aligned}$$

We may do this by solving the equivalent nonlinear system

$$f(t,y) = 0.$$

A nonlinear equation solver such as the CORLIB modules, DNSQE or HYBRD1 may be used in order to solve this system. However, if we try to solve this system in one pass, we again discover that the initial guess is too far from the steady-state solution and available nonlinear solvers will not converge to the solution. We can again step incrementally to the solution starting from $\Phi = 0$. Nonlinear equation solvers tend to have difficulty solving the resulting systems of equations and a relatively small $\Phi$ increment is required. The technique that we have found to work best for this problem is to take a few backward Euler steps and use a nonlinear equation solver to solve the necessary nonlinear corrector equations at each step. This amounts to replacing the above nonlinear system with a sequence (usually three or four) of systems that have the form

$$0 = ynew - yold - h f\ (t\ ,ynew\ ).$$

Here, *ynew* is the solution for the current value of $\Phi$, *yold* is the solution for the old value of $\Phi$, and $h$ is a very large fixed value. What the backward Euler steps effectively accomplish is to avoid the overhead of an adaptive ode solver such as LSODE. This approach is feasible only if we are not interested in tracking the intermediate solution to the discretized system of odes. This approach is not generally feasible for an actual time-dependent transient problem. However, it does demonstrate that with a bit of ingenuity, it is possible to use standard software to solve problems that do not appear to be directly amenable to a straightforward solution.

## 6. SUMMARY

This report illustrated the manner in which subroutines from the CORLIB core mathematical subroutine library may be used for the solution of a model fluid flow problem. The Euler fluid flow equations were spatially discretized using the method of pseudo-characteristics. The stiff ordinary differential equation solver LSODE was used to integrate the resulting system of ordinary differential equations. The non-stiff solver ODE was used to integrate a related system of ordinary differential equations. The linear equation solver subroutines DECOMP and SOLVE were used to solve linear systems whose solutions were required in the calculation of the system time derivatives. The monotone cubic spline interpolation subroutines PCHIM and PCHFE were used to approximate water properties. The use of other CORLIB modules for the solution of similar problems was next discussed. The report thus illustrates the manner in which modules from a standard mathematical software library such as CORLIB can be used as building blocks in the solution of complex problems of practical interest.

Figure 1

Jacobian Structure For M = 10

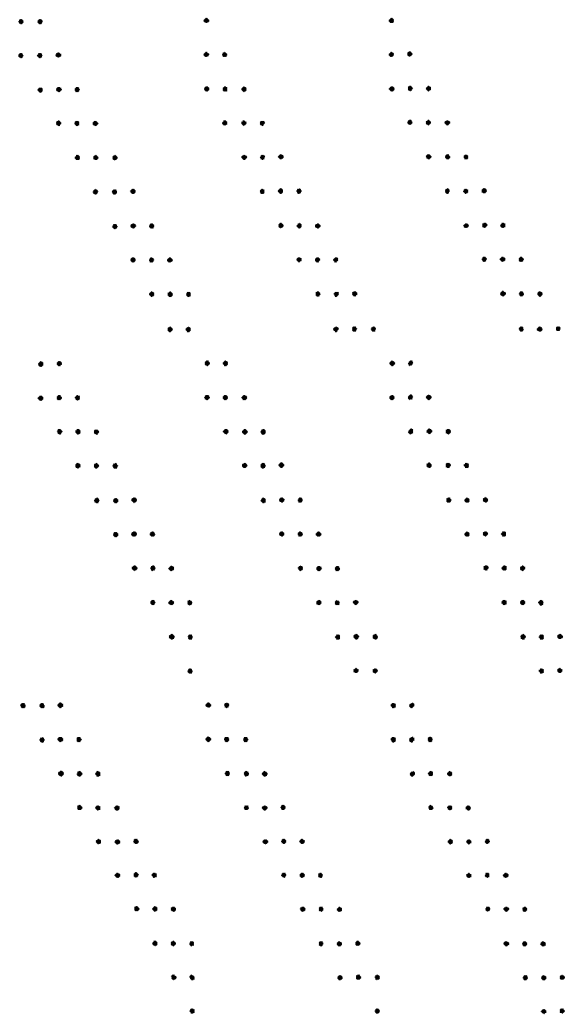

Figure 2

Flowchart for the Computer Program

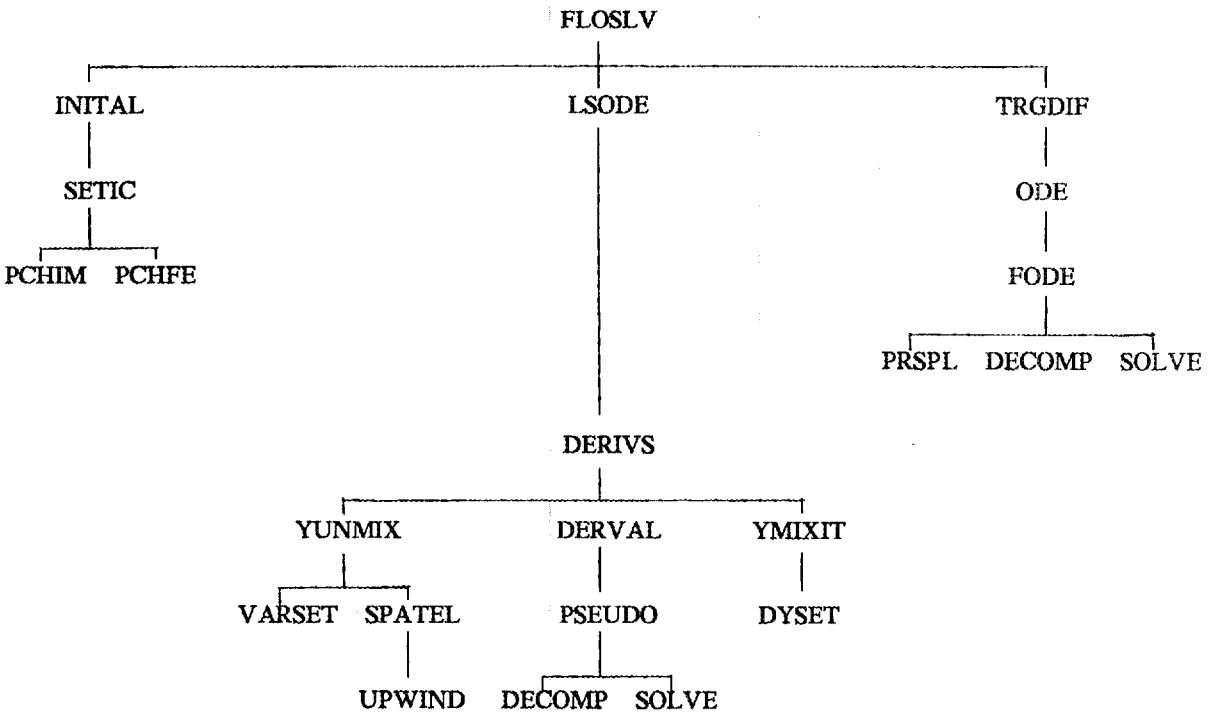

Table 1

Summary of Integrator Statistics

| M | Derivative Evaluations | Jacobian Evaluations | Integration Steps | Maximum Derivative Magnitude |
|---|---|---|---|---|
| 5 | 865 | 46 | 278 | .737D-7 |
| 10 | 2019 | 115 | 547 | .152D-6 |
| 20 | 6581 | 352 | 1911 | .887D-7 |
| 40 | 5776 | 299 | 1829 | .674D-6 |

Table 2

Steady-State Mass Flux Values

| | M | | | | |
|---|---|---|---|---|---|
| Z | 5 | 10 | 20 | 40 | exact |
| 0.0 | 468.13 | 377.10 | 327.08 | 299.97 | 270.90 |
| 0.2 | 401.71 | 349.60 | 314.61 | 294.04 | 270.90 |
| 0.4 | 364.18 | 329.60 | 303.97 | 288.50 | 270.90 |
| 0.6 | 321.33 | 307.80 | 292.65 | 282.68 | 270.90 |
| 0.8 | 270.91 | 283.85 | 280.85 | 276.55 | 270.90 |
| 1.0 | 270.90 | 270.90 | 270.90 | 270.90 | 270.90 |

Table 3

Steady-State Temperature Values

| | M | | | | |
|---|---|---|---|---|---|
| Z | 5 | 10 | 20 | 40 | exact |
| 0.0 | 255.00 | 255.00 | 255.00 | 255.00 | 255.00 |
| 0.2 | 257.35 | 257.66 | 257.95 | 258.17 | 258.46 |
| 0.4 | 259.92 | 260.45 | 260.98 | 261.37 | 261.90 |
| 0.6 | 262.81 | 263.40 | 264.09 | 264.61 | 265.30 |
| 0.8 | 266.20 | 266.57 | 267.30 | 267.88 | 268.67 |
| 1.0 | 269.58 | 269.93 | 270.62 | 271.20 | 272.01 |

Table 4

Steady-State Density Values

| | M | | | | |
|---|---|---|---|---|---|
| Z | 5 | 10 | 20 | 40 | exact |
| 0.0 | 795.52 | 795.52 | 795.52 | 795.52 | 795.52 |
| 0.2 | 791.83 | 791.32 | 790.84 | 790.50 | 790.03 |
| 0.4 | 787.65 | 786.78 | 785.93 | 785.30 | 784.46 |
| 0.6 | 782.83 | 781.85 | 780.74 | 779.91 | 778.80 |
| 0.8 | 777.02 | 776.43 | 775.25 | 774.32 | 773.06 |
| 1.0 | 771.16 | 770.55 | 769.43 | 768.51 | 767.23 |

# REFERENCES

[1] Cox, R. L., *LSOD28 - A Variant of LSODE for Systems Having Large Sparse Jacobian Matrix*, K/CSD-16, Oak Ridge National Laboratory, Oak Ridge, Tennessee, in preparation.

[2] Hindmarsh, A. C., "LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers", *ACM-SIGNUM Newsletter*, Vol. 15, No. 4, 1980, pp. 10-11.

[3] Hindmarsh, A. C., "Toward a Systematized Collection of ODE Solvers", *Proceedings, Vol. 1, 10th IMACS Congress on System Simulation and Scientific Computation, Montreal, Canada*, August 1982, pp. 427-432.

[4] Shampine, L. F. and H. A. Watts, *DEPAC - Design of a User Oriented Package of ODE Solvers*, SAND 79-2374, Sandia Laboratories, Albuquerque, New Mexico, 1980.

[5] Thompson, S., *A Quantitative Assessment of the Effect of Partial Pivoting In Sparse Ordinary Differential Equality Solvers*, ORNL-6207, Oak Ridge National Laboratory, Oak Ridge, Tennessee.

[6] Thompson, S., *On the Choice of an Ordinary Differential Equation Solver*, ORNL-6189, Oak Ridge National Laboratory, Oak Ridge, Tennessee.

[7] Thompson, S. and P. G. Tuttle, "Benchmark Fluid Flow Problems for Continuous Simulation Languages", *Comp. and Math. with Appls.*, to appear.

[8] Thompson, S. and P. G. Tuttle, "The Solution of Several Representative Stiff Problems in an Industrial Environment: The Evolution of an O.D.E. Solver", *Proceedings, Vol. 3, International Conference on Stiff Computation, Park City, Utah*, April 1982.

# APPENDIX A. INTRODUCTION TO THE METHOD OF LINES

The idea involved in the method of lines is to approximate a system of partial differential equations by a larger system of ordinary differential equations. The solution is then generated by integrating along lines in the time-like direction.

To be specific, consider the following simple example:

$$\begin{aligned} &u_t = u_{xx} && 0 < x < 1\ ,\ 0 < t < \infty \\ &u(x,0) = u_I(x) && 0 < x < 1 \\ &u(0,t) = u_L(t) && 0 < t < \infty \\ &u(1,t) = u_R(t) && 0 < t < \infty\ . \end{aligned} \tag{1}$$

Here, $u_I$, $u_L$ and $u_R$ denote the initial condition, the left boundary condition, and the right boundary condition, respectively. Partition the interval [0,1] by defining

$$x_i = i\,\Delta x\ ,\quad i = 0,\ldots,n$$

where

$$\Delta x = 1/n\ .$$

The spatial derivatives may now be replaced by suitable difference approximations. For example, if three-point centered differences are used, $u_{xx}$ may be replaced by

$$u_{xx}\ (x_i,t) \approx \frac{u(x_{i+1},t) - 2u(x_i,t) + u(x_{i-1},t)}{2\Delta x}\ .$$

(1) may now be approximated by the following system of ordinary differential equations.

$$\frac{du_i}{dt} = (u_{i+1} - 2u_i + u_{i-1})\ /\ 2\Delta x\ ,\ i = 1,\ldots,n-1$$

$$u_i(0) = u_I(x_i)\ ,\ i = 1,\ldots,n-1 \tag{2}$$

$$u_0(t) = u_L(t)\ ,\ t > 0$$

$$u_n(t) = u_R(t)\ ,\ t > 0\ .$$

This system of $(n-1)$ ordinary differential equations may now be solved using an appropriate ode solver.

# APPENDIX B. FORTRAN LISTING OF THE COMPUTER TEST PROGRAM

```
      PROGRAM FLOSLV
C
C     PROGRAM TO ILLUSTRATE THE USE OF SELECTED CORLIB ROUTINES.
C
C     PSEUDO-CHARACTERISTIC STEADY-STATE INITIALIZATION
C     OF A SET OF EULER EQUATIONS BY LSODE.
C
C
C            FLOWCHART FOR THE PROGRAM
C
C
C                      FLOSLV
C                        *
C                        *
C      ****************************************
C      *                 *                     *
C      *                 *                     *
C    INITAL            LSODE                 TRGDIF
C      *                 *                     *
C      *                 *                     *
C    SETIC               *                    ODE
C      *                 *                     *
C      *                 *                     *
C   ********             *                     *
C   *      *             *                     *
C   *      *             *                     *
C PCHIM  PCHFE           *                    FODE
C                        *                     *
C                        *                     *
C                        *                ***********
C                        *                *    *    *
C                        *                *    *    *
C               *              PRSPL  DECOMP  SOLVE
C                        *
C                        *
C                      DERIVS
C                        *
C                        *
C         ********************************
C         *              *                *
C         *              *                *
C       YUNMIX         DERVAL          YMIXIT
C         *              *                *
C         *              *                *
C     **********         *                *
C     *        *         *                *
C     *        *         *                *
C   VARSET  SPATEL     PSEUDO           DYSET
C                        *
C                        *
```

```
C               *****************
C               *       *       *
C               *       *       *
C             UPWIND  DECOMP  SOLVE
C
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
C     THE FUNDAT COMMON BLOCK AND THE FOLLOWING DIMENSION
C     STATEMENTS ARE DIMENSIONED FOR A MAXIMUM OF M = 40,
C     WHERE M+1 IS THE NUMBER OF SPATIAL NODES. THIS
C     CORRESPONDS TO A MAXIMUM ON N = 3M = 120 ODES.
C     WORK(22+25N), IWORK(20+N) , Y(N), DY(N), YORIG(N), DYORIG(N)
C     WORK(22+75M), IWORK(20+3M), Y(N), DY(N), YORIG(N), DYORIG(N)
      DIMENSION WORK(3022), IWORK(140), Y(120), DY(120)
C
C
C     GLOSSARY FOR FUNDAT COMMON BLOCK:
C
C     XKFAC     - FRICTIONAL PRESSURE DROP COEFFICIENT
C
C     GA        - GRAVITATIONAL ACCELERATION
C
C     PHI       - HEAT FLUX
C
C     ZMIN      - INLET Z = 0.0
C
C     ZMAX      - OUTLET Z = 1.0
C
C     PH        - HEATED PERIMETER
C
C     AF        - FLOW AREA
C
C     PO        - CONSTANT PRESSURE USED TO CALCULATE INITIAL
C                 DENSITIES AS A FUNCTION OF PRESSURE AND
C                 TEMPERATURE
C
C     NPT       - NUMBER OF POINTS IN UPWIND SPATIAL
C                 DIFFERENCES (-2)
C
C     GO        - OUTLET BOUNDARY VALUE FOR MASS FLUX
```

```
C
C     ABSOLT    - ABSOLUTE TEMPERATURE
C
C     TFO       - INLET BOUNDARY VALUE FOR TEMPERATURE
C
C     TFE       - INITIAL GUESS FOR OUTLET TEMPERATURE
C
C     M,MP1     - MP1 = M+1 = THE NUMBER OF SPATIAL NODES
C
C     BETA1     - COEFFICIENT VOLUME EXPANSION
C
C     XK1       - ISOTHERMAL COMPRESSIBILITY
C
C     SPEED1    - SOUND SPEED
C
C     Z         - SPATIAL NODES
C
C     CSUBP1    - SPECIFIC HEAT
C
C     G         - MASS FLUX
C
C     TF        - TEMPERATURE
C
C     RHO       - DENSITY
C
C     DG        - MASS FLUX TIME DERIVATIVE
C
C     DTF       - TEMPERATURE TIME DERIVATIVE
C
C     DRHO      - DENSITY TIME DERIVATIVE
C
C     VEL       - VELOCITY (MASS FLUX / DENSITY)
C
C     DVEL      - VELOCITY TIME DERIVATIVE
C
C     DZTFO,      SPATIAL DIFFERENCES FOR TEMPERATURE,
C     DZRHOO,     DENSITY, AND VELOCITY DETERMINED BY
C     DZVELO      THE G/RHO CHARACTERISTIC
C
C     DZTFP,      SPATIAL DIFFERENCES FOR TEMPERATURE,
C     DZRHOP,     DENSITY, AND VELOCITY DETERMINED BY
C     DZVELP      THE G/RHO + SPEED1 CHARACTERISTIC
C
C     DZTFM,      SPATIAL DIFFERENCES FOR TEMPERATURE,
C     DZRHOM,     DENSITY, AND VELOCITY DETERMINED BY
C     DZVELM      THE G/RHO - SPEED1 CHARACTERISTIC
C
C     VELPA       G/RHO + SPEED1
C
C     VELMA       G/RHO - SPEED1
```

```
C
C     DZGO,       SPATIAL DIFFERENCES FOR MASS FLUX
C     DZGP,       DETERMINED BY THE G/RHO, G/RHO + SPEED1,
C     DZGM        AND G/RHO - SPEED1 CHARACTERISTICS
C
C     TPDE      - TEMPORARY TEMPERATURE ARRAY
C
C     RPDE      - TEMPORARY DENSITY ARRAY
C
C     GPDE      - TEMPORARY MASS FLUX ARRAY
C
C     YORIG     - REORDERED LSODE SOLUTION ARRAY
C
C     DYORIG    - REORDERED LSODE DERIVATIVE ARRAY
C
C
      EXTERNAL DERIVS
C
    1 FORMAT(13H  SOLUTION - ,/,(2X,I5,3D15.5))
    2 FORMAT(16H  DERIVATIVES - ,/,(2X,I5,3D15.5))
    3 FORMAT(32H  (IFLAG,NFE,NJE,NSTEPS,TIME) = ,I3,3I9,D13.3)
    4 FORMAT(33H  LSODE RETURN FOR STEP NUMBER = ,I5)
    5 FORMAT(21H  ILLEGAL VALUE OF M.)
C
C     OPEN THE INPUT FILE.
      OPEN(UNIT=5,FILE='FLOSLV.DAT',STATUS='OLD')
C
C     OPEN THE OUTPUT FILE.
      OPEN(UNIT=6,FILE='FLOSLV.ANS',STATUS='NEW')
      LOUT = 6
C
C     DEFINE THE NUMBER OF POINTS IN THE
C     SPATIAL MESH.
      READ (5,*) M
      MP1 = M + 1
C
C     CHECK THE NUMBER OF POINTS IN THE
C     SPATIAL MESH.
      IMOK = 0
      IF (M .LT.  5) IMOK = 1
      IF (M .GT. 40) IMOK = 1
      IF (IMOK .NE. 0) WRITE (LOUT,5)
      IF (IMOK .NE. 0) GO TO 200
C
C     DEFINE THE INPUT PARAMETERS.
      NEQ     = 3*M
      TINIT   = 0.0D0
      TIN     = TINIT
      TOUT    = TINIT
      DELTAT  = 1.D-5
```

```
      EPS     = 1.0D-3
C
C     INITIALIZE LSODE.
      ITOL    = 1
      RTOL    = EPS
      ATOL    = EPS
      ITASK   = 1
      ISTATE  = 1
      IOPT    = 1
      LWORK   = 3022
      LIWORK  = 140
      MF      = 25
C     OPTIONAL INPUT.
      DO 20 I=5,10
      WORK(I)  = 0.0D0
      IWORK(I) = 0
   20 CONTINUE
      WORK(5)  = 1.0D-8
      ML       = 5
      MU       = 5
      IWORK(1) = ML
      IWORK(2) = MU
      IWORK(6) = 50000
      IWORK(7) = 1000
C
C     DEFINE THE INITIAL VALUES FOR FLOW-RELATED PARAMETERS.
      CALL INITAL (YORIG)
C
C     REORDER THE SOLUTION TO THE REDUCED BANDWIDTH ORDERING.
      CALL YMIXIT (NEQ, YORIG, Y, M)
C
C     WRITE THE SOLUTION AND DERIVATIVES.
      CALL DERIVS (NEQ, TIN, Y, DY)
      CALL YUNMIX (NEQ, Y, YORIG, M)
      CALL YUNMIX (NEQ, DY, DYORIG, M)
C
C     COMPARE THE SOLUTION OF THE DISCRETIZED
C     ODES AND THE SOLUTION OF THE PDES.
      CALL TRGDIF (LOUT)
C
C     INTEGRATION STEP LOOP.
C
      NSTEP = 10
      DO 100 ISTEP=1,NSTEP
      DELTAT = 10.0D0 * DELTAT
      TIN  = TOUT
      TOUT = TOUT + DELTAT
C
C     SOLUTION BY LSODE.
C
```

```
      CALL LSODE (DERIVS, NEQ, Y, TIN, TOUT, ITOL, RTOL, ATOL,
     *            ITASK, ISTATE, IOPT, WORK, LWORK, IWORK,
     *            LIWORK, DERIVS, MF)
C
      TEXIT  = TIN
      NFE    = IWORK(12)
      NJE    = IWORK(13)
      NSTEPS = IWORK(11)
C
C     WRITE THE TERMINATION FLAG, DERIVATIVE COUNT,
C     JACOBIAN COUNT, AND TIME.
      WRITE (LOUT,4) ISTEP
      WRITE (LOUT,3) ISTATE, NFE, NJE, NSTEPS, TEXIT
C
C     WRITE THE SOLUTION AND DERIVATIVES.
      CALL DERIVS (NEQ, TEXIT, Y, DY)
      CALL YUNMIX (NEQ, Y, YORIG, M)
      CALL YUNMIX (NEQ, DY, DYORIG, M)
      WRITE (LOUT,1) (I,G(I),TF(I),RHO(I),I=1,MP1)
      WRITE (LOUT,2) (I,DG(I),DTF(I),DRHO(I),I=1,MP1)
C
C     COMPARE THE SOLUTION OF THE DISCRETIZED
C     ODES AND THE SOLUTION OF THE PDES.
      CALL TRGDIF (LOUT)
C
C     EXIT THE INTEGRATION LOOP IF
C     LSODE WAS NOT SUCCESSFUL.
      IF (ISTATE .NE. 2) GO TO 200
C
  100 CONTINUE
C
  200 CONTINUE
C
      STOP
      END
```

```
      SUBROUTINE DERIVS (NEQ, T, Y, YDOT)
C
C     DIRECT THE CALCULATION OF THE TIME DERIVATIVES.
C
C     THIS SUBROUTINE SHOULD BE CALLED EACH TIME THE
C     ODE SOLVER REQUESTS THAT SYSTEM DERIVATIVES
C     BE CALCULATED.
C
C     INPUT:
C
C          NEQ  = NUMBER OF ODE*S
C          T    = CURRENT VALUE OF INDEPENDENT VARIABLE
C          Y    = APPROXIMATE SOLUTION FOR THE ODE*S
C
C     OUTPUT:
C
C          YDOT = SYSTEM DERIVATIVES = F(T,Y)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION Y(NEQ), YDOT(NEQ)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
C     RESTORE VARIABLES TO THE ORIGINAL
C     BLOCK-WISE ORDERING.
      CALL YUNMIX (NEQ, Y, YORIG, M)
C
C     CALCULATE THE TIME DERIVATIVES.
      CALL DERVAL (YORIG(1), DYORIG(1), YORIG(M+1),
     *DYORIG(M+1), YORIG(2*M+1), DYORIG(2*M+1), T)
C
C     SHUFFLE THE CALCULATED DERIVATIVES INTO
C     THE REDUCED BANDWIDTH ORDERING.
      CALL YMIXIT (NEQ, DYORIG, YDOT, M)
C
      RETURN
      END
```

```
      SUBROUTINE DERVAL (AG, AGD, AT, ATD, AR, ARD, T)
C
C     PERFORM THE CALCULATION OF THE TIME DERIVATIVES.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION AG(1), AGD(1), AT(1), ATD(1), AR(1), ARD(1)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
C     LOAD THE SOLUTION INTO LOCAL STORAGE.
      CALL VARSET (AG, AT, AR)
C
C     LOAD THE BOUNDARY CONDITIONS AT THIS POINT IF APPLICABLE.
C
C     CALCULATE THE PROPERTIES IF APPLICABLE.
C
C     DEFINE THE SPATIAL DERIVATIVES.
      CALL SPATEL
C
C     DEFINE THE TIME DERIVATIVES.
      CALL PSEUDO
C
C     LOAD THE TIME DERIVATIVES INTO THE INTEGRATOR ARRAY.
      CALL DYSET (AGD, ATD, ARD)
C
      RETURN
      END
```

```
      SUBROUTINE DYSET (AGD, ATD, ARD)
C
C     LOAD THE LOCAL TIME DERIVATIVES
C     INTO THE INTEGRATOR ARRAY.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION AGD(1), ATD(1), ARD(1)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
      DO 20 I=1,M
      ATD(I) = DTF(I+1)
      ARD(I) = DRHO(I+1)
      AGD(I) = DG(I)
   20 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE FODE (T, RHOTF, DRHOTF)
C
C     EVALUATE THE CONTINUOUS_SPACE-DISCRETE-
C     DERIVATIVE (CSDT) DERIVATIVES FOR ODE.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION RHOTF(2), DRHOTF(2)
C
      DIMENSION A(2,2), IPVT(2), WORK(2)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
C     CALCULATE THE PROPERTIES FOR
C     THIS SPATIAL VALUE (T).
      CALL PRSPL (T, XKAPPA, BETA, SPEED, CSUBP)
C
C     SET UP THE 2 BY 2 LINEAR SYSTEM
C     FOR THIS SPATIAL VALUE.
C
      A(1,1) = 1.0D0/(RHOTF(1)*XKAPPA) - (GO/RHOTF(1))**2
C
      A(1,2) = BETA / XKAPPA
C
      A(2,1) = -(SPEED**2 * BETA * (RHOTF(2)+ABSOLT) * GO)
     2         / (CSUBP * RHOTF(1)**2)
C
      A(2,2) = GO / RHOTF(1)
C
      DRHOTF(1) = -XKFAC * GO * ABS(GO/RHOTF(1))
     2            -RHOTF(1) * GA * SINANG
C
      DRHOTF(2) = (SPEED**2 * PHI * PH * XKAPPA)
     2          / (CSUBP * AF)
C
C     SOLVE THE 2 BY 2 SYSTEM OF LINEAR EQUATIONS
C     FOR THIS SPATIAL VALUE.
C
      NL = 2
      IAL = 2
      CALL DECOMP (IAL, NL, A, COND, IPVT, WORK)
      CALL SOLVE (IAL, NL, A, DRHOTF, IPVT)
C
      RETURN
      END
```

```
      SUBROUTINE INITAL (YINIT)
C
C     INITIALIZE THE PROBLEM.
C
C     THIS SUBROUTINE SHOULD BE CALLED BEFORE THE
C     INTEGRATION IS STARTED. WHEN INITAL IS CALLED,
C     THE NECESSARY PROBLEM PARAMETERS WILL BE
C     INITIALIZED. ALSO, THE INITIAL VALUES FOR THE
C     ODE SOLVER WILL BE RETURNED IN THE ARRAY YINIT.
C     BEFORE CALLING INITAL, THE NUMBER OF SPATIAL
C     NODES M MUST BE DEFINED. THE TOTAL NUMBER OF
C     ODE*S WILL BE NEQ = 3*M. YINIT MUST BE
C     DIMENSIONED IN THE CALLING PROGRAM FOR
C     AT LEAST NEQ.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION YINIT(1)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
C     REFER TO THE GLOSSARY IN PROGRAM FLOSLV FOR
C     THE MEANING OF THE FOLLOWING VARIABLES.
C
      NPT    = 2
      XKFAC  = 10.0D0
      GA     = 9.80665D0
      SINANG = 1.0D0
      PHI    = 1.1D5
      ZMIN   = 0.0D0
      ZMAX   = 1.0D0
C     ZMAX   = 2.0953D0
      PH     = 7.97318D+2
      AF     = 3.82760D0
      PO     = 7.4050D+6
      GO     = 270.9D0
      ABSOLT = 273.15D0
      TFO    = 255.0D0
      TFE    = 289.0D0
C
C     DEFINE THE SPATIAL MESH.
      Z(1) = ZMIN
      DELZ   = (ZMAX - ZMIN) / M
```

```
      DO 20 I=2,M
      Z(I) = ZMIN + (I-1) * DELZ
   20 CONTINUE
      Z(MP1) = ZMAX
C
C     DEFINE THE INITIAL GUESSES FOR RHO AND TF, AND
C     LOAD THE PROPERTIES.
      CALL SETIC
C
C     DEFINE THE INITIAL GUESS FOR THE FLOW
C     RATES AND VELOCITIES.
      DO 40 I=1,MP1
      G(I) = G0
      VEL(I) = G(I) / RHO(I)
   40 CONTINUE
C
C     DEFINE THE INITIAL CONDITIONS FOR THE
C     ODE SOLVER.
      DO 60 I=1,M
      YINIT(M+I)   = TF(I+1)
      YINIT(2*M+I) = RHO(I+1)
      YINIT(I)     = G(I)
   60 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE PRSPL (ZVAL, XKAPPA, BETA, SPEED, CSUBP)
C
C     APPROXIMATE THE PROPERTIES AT THE SPATIAL VALUE ZVAL.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL SKIP
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
      COMMON /SPL/
     * RONIT(71), TFNIT(71), XKVAL(71), BEVAL(71), SPVAL(71), CSVAL(71),
     * ZIC(71), D1(71), D2(71), D3(71), D4(71), D5(71), D6(71), NZ
C
      DATA XKAVG /  .1714462720156890-08 /
C
      DATA BEAVG /  .213024626664637D-02 /
C
      DATA SPAVG /  .108595374561510D+04 /
C
      DATA CSAVG /  .496941623289027D+04 /
C
      DATA IPTYPE /1/
C
      SKIP = .FALSE.
      INCFD = 1
      NVAL = 1
C
      GO TO (40,60), IPTYPE
C
   40 CONTINUE
C
C     USE MONOTONE SPLINES FOR THE PROPERTIES.
C
      CALL PCHFE (NZ, ZIC, XKVAL, D3, INCFD,
     2  SKIP, NVAL, ZVAL, XKAPPA, IER)
C
      CALL PCHFE (NZ, ZIC, BEVAL, D4, INCFD,
     2  SKIP, NVAL, ZVAL, BETA ,IER)
C
      CALL PCHFE (NZ, ZIC, SPVAL, D5, INCFD,
     2  SKIP, NVAL, ZVAL, SPEED, IER)
C
```

```
      CALL PCHFE (NZ, ZIC, CSVAL, D6, INCFD,
     2  SKIP, NVAL, ZVAL, CSUBP, IER)
C
      GO TO 100
C
   60 CONTINUE
C
C     PROPERTIES CONSTANT IN SPACE.
C
      XKAPPA = XKAVG
      BETA   = BEAVG
      SPEED  = SPAVG
      CSUBP  = CSAVG
C
  100 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE PSEUDO
C
C     USE GAUSSIAN ELIMINATION TO SOLVE THE EQUATIONS
C     IN CHARACTERISTIC FORM FOR THE TIME DERIVATIVES.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION F(3,3), D(3), E(3), IPVT(3), WORK(3)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
      ILOW  = 1
      IHIGH = MP1
C
      DO 20 I=ILOW,IHIGH
C
C     SET UP THE 3 BY 3 LINEAR SYSTEM
C     THE I-TH SPATIAL NODE.
C
      F(1,1) = SPEED1(I)*SPEED1(I)*BETA1(I)*(TF(I)+ABSOLT)
      F(1,2) = 0.0D0
      F(1,3) = -RHO(I)*CSUBP1(I)
      F(2,1) = -G(I)*XK1(I)*SPEED1(I) + 1.0D0
      F(2,2) = RHO(I)*XK1(I)*SPEED1(I)
      F(2,3) = RHO(I)*BETA1(I)
      F(3,1) = G(I)*XK1(I)*SPEED1(I) + 1.0D0
      F(3,2) = -F(2,2)
      F(3,3) = F(2,3)
C
      D(1) = 0.0D0
      D(2) = -XKFAC*G(I)*DABS(G(I)/RHO(I))
     2       -RHO(I)*GA*SINANG
      D(3) = (SPEED1(I)*SPEED1(I)*PHI*PH*XK1(I))
     2      /(CSUBP1(I)*AF)
C
      E(1) = F(1,1)*D(1) + F(1,2)*D(2)    + F(1,3)*D(3)
     2      -VEL(I) * (F(1,1)*DZRHOO(I)   + F(1,2)*DZGO(I)
     3                                    + F(1,3)*DZTFO(I))
      E(2) = F(2,1)*D(1) + F(2,2)*D(2)    + F(2,3)*D(3)
     2      -VELPA(I) * (F(2,1)*DZRHOP(I) + F(2,2)*DZGP(I)
     3                                    + F(2,3)*DZTFP(I))
      E(3) = F(3,1)*D(1) + F(3,2)*D(2)    + F(3,3)*D(3)
     2      -VELMA(I) * (F(3,1)*DZRHOM(I) + F(3,2)*DZGM(I)
```

```
     3                                          + F(3,3)*DZTFM(I))
C
C      SOLVE THE 3 BY 3 SYSTEM OF LINEAR EQUATIONS
C      FOR THIS SPATIAL NODE.
       IF = 3
       NF = 3
       CALL DECOMP (IF, NF, F, COND, IPVT, WORK)
       CALL SOLVE (IF, NF, F, E, IPVT)
C
C      DEFINE THE TIME DERIVATIVES FOR THIS
C      SPATIAL NODE.
       DTF(I)  = E(3)
       DRHO(I) = E(1)
       DG(I)   = E(2)
    20 CONTINUE
C
C      ZERO THE TIME DERIVATIVES CORRESPONDING
C      TO THE BOUNDARY CONDITIONS.
       DTF(1)  = 0.0D0
       DRHO(1) = 0.0D0
       DG(MP1) = 0.0D0
C
       RETURN
       END
```

```
      SUBROUTINE SETIC
C
C     CALCULATE MONOTONE SPLINE APPROXIMATIONS FOR
C     PROPERTIES AND INITIAL TEMPERATURES AND
C     DENSITIES.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL SKIP
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
      COMMON /SPL/
     * RONIT(71), TFNIT(71), XKVAL(71), BEVAL(71), SPVAL(71), CSVAL(71),
     * ZIC(71), D1(71), D2(71), D3(71), D4(71), D5(71), D6(71), NZ
C
      DIMENSION RONIT1(71), TFNIT1(71), XKVAL1(71),
     2          BEVAL1(71), SPVAL1(71), CSVAL1(71)
C
      DATA (RONIT1(I),I=1,45)
     .   / .7955210863D+03 , .7947589005D+03 , .7939941556D+03 ,
     .     .7932268295D+03 , .7924568994D+03 , .7916843424D+03 ,
     .     .7909091352D+03 , .7901312543D+03 , .7893506754D+03 ,
     .     .7885673744D+03 , .7877813264D+03 , .7869925064D+03 ,
     .     .7862008887D+03 , .7854064475D+03 , .7846091564D+03 ,
     .     .7838089888D+03 , .7830059175D+03 , .7821999148D+03 ,
     .     .7813909527D+03 , .7805790028D+03 , .7797640361D+03 ,
     .     .7789460232D+03 , .7781249342D+03 , .7773007386D+03 ,
     .     .7764734056D+03 , .7756429037D+03 , .7748092010D+03 ,
     .     .7739722649D+03 , .7731320624D+03 , .7722885599D+03 ,
     .     .7714417231D+03 , .7705915173D+03 , .7697379070D+03 ,
     .     .7688808562D+03 , .7680203281D+03 , .7671562856D+03 ,
     .     .7662886904D+03 , .7654175040D+03 , .7645426868D+03 ,
     .     .7636641988D+03 , .7627819990D+03 , .7618960458D+03 ,
     .     .7610062966D+03 , .7601127083D+03 , .7592152367D+03 /
      DATA (RONIT1(I),I=46,71)
     .   /.7583138368D+03 , .7574084629D+03 , .7564990681D+03 ,
     .     .7555856048D+03 , .7546680244D+03 , .7537462772D+03 ,
     .     .7528203126D+03 , .7518900789D+03 , .7509555234D+03 ,
     .     .7500165922D+03 , .7490732303D+03 , .7481253816D+03 ,
     .     .7471729886D+03 , .7462159928D+03 , .7452543342D+03 ,
     .     .7442941759D+03 , .7433220947D+03 , .7423452153D+03 ,
     .     .7413634772D+03 , .7403768189D+03 , .7393851776D+03 ,
```

```
     .     .7383884890D+03 , .7373866878D+03 , .7363797070D+03 ,
     .     .7353674785D+03 , .7343499327D+03 /
C
      DATA (TFNIT1(I),I=1,45)
     .   / .2550000000D+03 , .2554857143D+03 , .2559714286D+03 ,
     .     .2564571429D+03 , .2569428571D+03 , .2574285714D+03 ,
     .     .2579142857D+03 , .2584000000D+03 , .2588857143D+03 ,
     .     .2593714286D+03 , .2598571429D+03 , .2603428571D+03 ,
     .     .2608285714D+03 , .2613142857D+03 , .2618000000D+03 ,
     .     .2622857143D+03 , .2627714286D+03 , .2632571429D+03 ,
     .     .2637428571D+03 , .2642285714D+03 , .2647142857D+03 ,
     .     .2652000000D+03 , .2656857143D+03 , .2661714286D+03 ,
     .     .2666571429D+03 , .2671428571D+03 , .2676285714D+03 ,
     .     .2681142857D+03 , .2686000000D+03 , .2690857143D+03 ,
     .     .2695714286D+03 , .2700571429D+03 , .2705428571D+03 ,
     .     .2710285714D+03 , .2715142857D+03 , .2720000000D+03 ,
     .     .2724857143D+03 , .2729714286D+03 , .2734571429D+03 ,
     .     .2739428571D+03 , .2744285714D+03 , .2749142857D+03 ,
     .     .2754000000D+03 , .2758857143D+03 , .2763714286D+03 /
      DATA (TFNIT1(I),I=46,71)
     .   / .2768571429D+03 , .2773428571D+03 , .2778285714D+03 ,
     .     .2783142857D+03 , .2788000000D+03 , .2792857143D+03 ,
     .     .2797714286D+03 , .2802571429D+03 , .2807428571D+03 ,
     .     .2812285714D+03 , .2817142857D+03 , .2822000000D+03 ,
     .     .2826857143D+03 , .2831714286D+03 , .2836571429D+03 ,
     .     .2841428571D+03 , .2846285714D+03 , .2851142857D+03 ,
     .     .2856000000D+03 , .2860857143D+03 , .2865714286D+03 ,
     .     .2870571429D+03 , .2875428571D+03 , .2880285714D+03 ,
     .     .2885142857D+03 , .2890000000D+03 /
C
      DATA (XKVAL1(I),I=1,45)
     .   / .1522344218D-08 , .1527088808D-08 , .1531865180D-08 ,
     .     .1536669336D-08 , .1541501510D-08 , .1546361938D-08 ,
     .     .1551250861D-08 , .1556168519D-08 , .1561115157D-08 ,
     .     .1566091023D-08 , .1571096368D-08 , .1576131444D-08 ,
     .     .1581196507D-08 , .1586291817D-08 , .1591417636D-08 ,
     .     .1596574230D-08 , .1601761865D-08 , .1606980815D-08 ,
     .     .1612231353D-08 , .1617513758D-08 , .1622828311D-08 ,
     .     .1628175296D-08 , .1633555002D-08 , .1638967719D-08 ,
     .     .1644413743D-08 , .1649893373D-08 , .1655406909D-08 ,
     .     .1660954659D-08 , .1666536932D-08 , .1672154041D-08 ,
     .     .1677806302D-08 , .1683494038D-08 , .1689217573D-08 ,
     .     .1694977237D-08 , .1700773361D-08 , .1706606284D-08 ,
     .     .1712476348D-08 , .1718383897D-08 , .1724329282D-08 ,
     .     .1730312857D-08 , .1736334982D-08 , .1742396021D-08 ,
     .     .1748496341D-08 , .1754636316D-08 , .1760816323D-08 /
      DATA (XKVAL1(I),I=46,71)
     .   / .1767036745D-08 , .1773297970D-08 , .1779600390D-08 ,
     .     .1785944403D-08 , .1792330413D-08 , .1798758827D-08 ,
     .     .1805230060D-08 , .1811744530D-08 , .1818302662D-08 ,
```

```
     .      .1824904886D-08 , .1831551640D-08 , .1838243364D-08 ,
     .      .1844980508D-08 , .1851763524D-08 , .1858592873D-08 ,
     .      .1865469022D-08 , .1872392443D-08 , .1879363616D-08 ,
     .      .1886383028D-08 , .1893451170D-08 , .1900568542D-08 ,
     .      .1907735651D-08 , .1914953010D-08 , .1922221140D-08 ,
     .      .1929540570D-08 , .1936896655D-08 /
C
      DATA (BEVAL1(I),I=1,45)
     .   / .1969311719D-02 , .1973395241D-02 , .1977501859D-02 ,
     .      .1981627933D-02 , .1985773611D-02 , .1989939045D-02 ,
     .      .1994124385D-02 , .1998329785D-02 , .2002555400D-02 ,
     .      .2006801386D-02 , .2011067901D-02 , .2015355105D-02 ,
     .      .2019663159D-02 , .2023992227D-02 , .2028342473D-02 ,
     .      .2032714063D-02 , .2037107167D-02 , .2041521955D-02 ,
     .      .2045958598D-02 , .2050417270D-02 , .2054898147D-02 ,
     .      .2059401407D-02 , .2063927229D-02 , .2068475796D-02 ,
     .      .2073047289D-02 , .2077641896D-02 , .2082259804D-02 ,
     .      .2086901202D-02 , .2091566283D-02 , .2096255240D-02 ,
     .      .2100968270D-02 , .2105705571D-02 , .2110467344D-02 ,
     .      .2115253792D-02 , .2120065119D-02 , .2124901534D-02 ,
     .      .2129763247D-02 , .2134650470D-02 , .2139563418D-02 ,
     .      .2144502308D-02 , .2149467361D-02 , .2154458799D-02 ,
     .      .2159476847D-02 , .2164521733D-02 , .2169593687D-02 /
      DATA (BEVAL1(I),I=46,71)
     .   / .2174692943D-02 , .2179819737D-02 , .2184974307D-02 ,
     .      .2190156896D-02 , .2195367749D-02 , .2200607112D-02 ,
     .      .2205875237D-02 , .2211172378D-02 , .2216498791D-02 ,
     .      .2221854736D-02 , .2227240478D-02 , .2232656282D-02 ,
     .      .2238102418D-02 , .2243579161D-02 , .2249086786D-02 ,
     .      .2254625573D-02 , .2260195808D-02 , .2265797777D-02 ,
     .      .2271431771D-02 , .2277098086D-02 , .2282797020D-02 ,
     .      .2288528876D-02 , .2294293961D-02 , .2300092584D-02 ,
     .      .2305925062D-02 , .2311779325D-02 /
C
      DATA (SPVAL1(I),I=1,45)
     .   / .1125293014D+04 , .1124216065D+04 , .1123136016D+04 ,
     .      .1122054097D+04 , .1120970301D+04 , .1119884622D+04 ,
     .      .1118797053D+04 , .1117707586D+04 , .1116616216D+04 ,
     .      .1115522936D+04 , .1114427738D+04 , .1113330615D+04 ,
     .      .1112231560D+04 , .1111130566D+04 , .1110027625D+04 ,
     .      .1108922731D+04 , .1107815875D+04 , .1106707050D+04 ,
     .      .1105596249D+04 , .1104483463D+04 , .1103368686D+04 ,
     .      .1102251908D+04 , .1101133122D+04 , .1100012320D+04 ,
     .      .1098889494D+04 , .1097764636D+04 , .1096637736D+04 ,
     .      .1095508787D+04 , .1094377781D+04 , .1093244708D+04 ,
     .      .1092109559D+04 , .1090972327D+04 , .1089833002D+04 ,
     .      .1088691574D+04 , .1087548035D+04 , .1086402376D+04 ,
     .      .1085254588D+04 , .1084104660D+04 , .1082952583D+04 ,
     .      .1081798348D+04 , .1080641945D+04 , .1079483363D+04 ,
     .      .1078322594D+04 , .1077159626D+04 , .1075994450D+04 /
```

```
      DATA (SPVAL1(I),I=46,71)
     .   / .1074827056D+04 , .1073657432D+04 , .1072485568D+04 ,
     .     .1071311454D+04 , .1070135078D+04 , .1068956430D+04 ,
     .     .1067775499D+04 , .1066592273D+04 , .1065406741D+04 ,
     .     .1064218891D+04 , .1063028713D+04 , .1061836194D+04 ,
     .     .1060641322D+04 , .1059444085D+04 , .1058244472D+04 ,
     .     .1057042469D+04 , .1055838065D+04 , .1054631246D+04 ,
     .     .1053422000D+04 , .1052210315D+04 , .1050996176D+04 ,
     .     .1049779572D+04 , .1048560487D+04 , .1047338910D+04 ,
     .     .1046114826D+04 , .1044891081D+04 /
C
      DATA (CSVAL1(I),I=1,45)
     .   / .4861255855D+04 , .4863979295D+04 , .4866719332D+04 ,
     .     .4869473379D+04 , .4872241537D+04 , .4875023906D+04 ,
     .     .4877820587D+04 , .4880631683D+04 , .4883457297D+04 ,
     .     .4886297534D+04 , .4889152499D+04 , .4892022299D+04 ,
     .     .4894907040D+04 , .4897806832D+04 , .4900721783D+04 ,
     .     .4903652005D+04 , .4906597610D+04 , .4909558709D+04 ,
     .     .4912535418D+04 , .4915527851D+04 , .4918536124D+04 ,
     .     .4921560355D+04 , .4924600662D+04 , .4927657166D+04 ,
     .     .4930729986D+04 , .4933819246D+04 , .4936925069D+04 ,
     .     .4940047579D+04 , .4943186903D+04 , .4946343168D+04 ,
     .     .4949516503D+04 , .4952707038D+04 , .4955914904D+04 ,
     .     .4959140234D+04 , .4962383162D+04 , .4965643824D+04 ,
     .     .4968922356D+04 , .4972218899D+04 , .4975533590D+04 ,
     .     .4978866573D+04 , .4982217990D+04 , .4985587986D+04 ,
     .     .4988976706D+04 , .4992384300D+04 , .4995810916D+04 /
      DATA (CSVAL1(I),I=46,71)
     .   / .4999256706D+04 , .5002721822D+04 , .5006206419D+04 ,
     .     .5009710653D+04 , .5013234683D+04 , .5016778668D+04 ,
     .     .5020342769D+04 , .5023927152D+04 , .5027531979D+04 ,
     .     .5031157421D+04 , .5034803644D+04 , .5038470821D+04 ,
     .     .5042159125D+04 , .5045868732D+04 , .5049599818D+04 ,
     .     .5053352563D+04 , .5057127149D+04 , .5060923759D+04 ,
     .     .5064742580D+04 , .5068583800D+04 , .5072447609D+04 ,
     .     .5076334200D+04 , .5080243769D+04 , .5084176513D+04 ,
     .     .5088132632D+04 , .5092103860D+04 /
C
      DATA XKAVG /  .171446272015689D-08 /
C
      DATA BEAVG /  .213024626664637D-02 /
C
      DATA SPAVG /  .108595374561510D+04 /
C
      DATA CSAVG /  .496941623289027D+04 /
C
      DATA IPTYPE /1/
C
      NZ = 71
      ZIC(1) = ZMIN
```

```
      NZM1 = NZ - 1
      NZM2 = NZM1 - 1
      DELZC = (ZMAX - ZMIN) / NZM1
      DO 20 I=2,NZM1
      ZIC(I) = ZMIN + (I-1)*DELZC
   20 CONTINUE
      ZIC(NZ) = ZMAX
C
      DO 30 I=1,NZ
      RONIT(I) = RONIT1(I)
      TFNIT(I) = TFNIT1(I)
      XKVAL(I) = XKVAL1(I)
      BEVAL(I) = BEVAL1(I)
      SPVAL(I) = SPVAL1(I)
      CSVAL(I) = CSVAL1(I)
   30 CONTINUE
C
      SKIP = .FALSE.
      INCFD = 1
C
C     USE MONOTONE SPLINE FOR THE INITIAL DENSITIES.
      CALL PCHIM (NZ, ZIC, RONIT, D1, INCFD, IER)
      CALL PCHFE (NZ, ZIC, RONIT, D1, INCFD,
     2 SKIP, MP1, Z, RHO, IER)
C
C     USE MONOTONE SPLINE FOR THE INITIAL TEMPERATURES.
      CALL PCHIM (NZ, ZIC, TFNIT, D2, INCFD, IER)
      CALL PCHFE (NZ, ZIC, TFNIT, D2, INCFD,
     2 SKIP, MP1, Z, TF, IER)
C
      GO TO (40,60), IPTYPE
C
   40 CONTINUE
C
C     USE MONOTONE SPLINES FOR THE PROPERTIES.
C
      CALL PCHIM (NZ, ZIC, XKVAL, D3, INCFD, IER)
      CALL PCHFE (NZ, ZIC, XKVAL, D3, INCFD,
     2 SKIP, MP1, Z, XK1, IER)
C
      CALL PCHIM (NZ, ZIC, BEVAL, D4, INCFD, IER)
      CALL PCHFE (NZ, ZIC, BEVAL, D4, INCFD,
     2            SKIP, MP1, Z, BETA1, IER)
C
      CALL PCHIM (NZ, ZIC, SPVAL, D5, INCFD, IER)
      CALL PCHFE (NZ, ZIC, SPVAL, D5, INCFD,
     2            SKIP, MP1, Z, SPEED1, IER)
C
      CALL PCHIM (NZ, ZIC, CSVAL, D6, INCFD, IER)
      CALL PCHFE (NZ, ZIC, CSVAL, D6, INCFD,
```

```
     2             SKIP, MP1, Z, CSUBP1, IER)
C
      GO TO 100
C
   60 CONTINUE
C
C     PROPERTIES CONSTANT IN SPACE.
C
      DO 80 I=1,MP1
      XK1(I)     = XKAVG
      BETA1(I)   = BEAVG
      SPEED1(I)  = SPAVG
      CSUBP1(I)  = CSAVG
   80 CONTINUE
C
  100 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE SPATEL
C
C     DEFINE THE SPATIAL DERIVATIVES.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
      DO 20 I=1,MP1
      VEL(I)   = G(I) / RHO(I)
      VELPA(I) = VEL(I) + SPEED1(I)
      VELMA(I) = VEL(I) - SPEED1(I)
   20 CONTINUE
C
      DELTAZ = Z(2) - Z(1)
C
C     V CHARACTERISTIC.
      CALL UPWIND (RHO, DZRHOO, VEL, MP1, DELTAZ, NPT)
      CALL UPWIND (G, DZGO, VEL, MP1, DELTAZ, NPT)
      CALL UPWIND (TF, DZTFO, VEL, MP1, DELTAZ, NPT)
C
C     V + A  CHARACTERISTIC.
      CALL UPWIND (RHO, DZRHOP, VELPA, MP1, DELTAZ, NPT)
      CALL UPWIND (G, DZGP, VELPA, MP1, DELTAZ, NPT)
      CALL UPWIND (TF, DZTFP, VELPA, MP1, DELTAZ, NPT)
C
C     V - A CHARACTERISTIC.
      CALL UPWIND (RHO, DZRHOM, VELMA, MP1, DELTAZ, NPT)
      CALL UPWIND (G, DZGM, VELMA, MP1, DELTAZ, NPT)
      CALL UPWIND (TF, DZTFM, VELMA, MP1, DELTAZ, NPT)
C
      RETURN
      END
```

```
      SUBROUTINE TRGDIF (LOUT)
C
C     CALCULATE THE DIFFERENCE BETWEEN THE SOLUTION
C     OF THE DISCRETIZED SYSTEM AND THE EXACT
C     SOLUTION FOR THE PDE SYSTEM.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
      DIMENSION WORK(142), IWORK(5), YODE(2)
C
      EXTERNAL FODE
C
      RPDE(1) = RHO(1)
      TPDE(1) = TFO
      GPDE(1) = GO
C
      T = Z(1)
      ABSERR = 1.0D-12
      RELERR = 1.0D-12
      IFLAG = 1
      NODE = 2
      YODE(1) = RPDE(1)
      YODE(2) = TPDE(1)
C
      DO 100 I=2,MP1
C
      TOUT = Z(I)
C
   40 CONTINUE
C
      CALL ODE (FODE, NODE, YODE, T, TOUT, RELERR, ABSERR,
     *          IFLAG, WORK, IWORK)
C
      IF (IFLAG .EQ. 4) GO TO 40
      IF (IFLAG .EQ. 5) GO TO 40
      IF (IFLAG .NE. 2) GO TO 300
C
      RPDE(I) = YODE(1)
      TPDE(I) = YODE(2)
      GPDE(I) = GO
C
```

```
  100 CONTINUE
C
      WRITE (LOUT,1)
      WRITE (LOUT,3) (I,GPDE(I),TPDE(I),RPDE(I),I=1,MP1)
C
      DO 200 I=1,MP1
      TPDE(I) = TPDE(I) - TF(I)
      RPDE(I) = RPDE(I) - RHO(I)
      GPDE(I) = GPDE(I) - G(I)
  200 CONTINUE
C
      WRITE (LOUT,2)
      WRITE (LOUT,3) (I,GPDE(I),TPDE(I),RPDE(I),I=1,MP1)
C
  300 CONTINUE
C
    1 FORMAT (25H  EXACT SOLUTION OF PDE -)
    2 FORMAT (15H  DIFFERENCES -)
    3 FORMAT ((2X,I5,3D15.5))
C
      RETURN
      END
```

```
      SUBROUTINE UPWIND (U, UX, V, NX, DX, NO)
C
C     UPWIND DIFFERENCING ROUTINE.
C
C     APPROXIMATE DU/DX IN TERMS LIKE V(DU/DX) BY BACKWARD
C     DIFFERENCES IF V IS POSITIVE AND BY FORWARD DIFFERENCES
C     IF V IS NEGATIVE.
C
C     NO MAY BE 2, 3, OR 4  (IN WHICH CASE, 2-POINT, 3-POINT,
C     OR 4-POINT DIFFERENCES WILL BE USED, RESPECTIVELY).
C     U IS THE DEPENDENT VARIABLE TO BE DIFFERENCED.
C     NX IS THE NUMBER OF POINTS IN THE SPATIAL GRID
C     CORRESPONDING TO U.
C     DX IS THE (EQUAL) DISTANCE BETWEEN POINTS IN
C     THE SPATIAL MESH.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION U(NX), UX(NX), V(NX)
C
      N1 = NX-1
      N2 = N1-1
      N3 = N2-1
      NO = NO-1
C
      GO TO (5,15,25), NO
C
C     BACKWARD DIFFERENCES.
C
    5 DO 10 I=2,NX
   10 UX(I) = (U(I)-U(I-1))/DX
      GO TO 35
   15 DO 20 I=3,NX
   20 UX(I) = (1.5D0*U(I)-2.D0*U(I-1)+.5D0*U(I-2))/DX
      GO TO 35
   25 DO 30 I=4,NX
   30 UX(I) = (-2.0D0*U(I-3)+9.0D0*U(I-2)-18.0D0*U(I-1)+11.0D0*U(I))
     2/(6.0D0*DX)
      UX(3) = (1.5D0*U(3)-2.0D0*U(2)+.5D0*U(1))/DX
   35 UX(1) = (U(2)-U(1))/DX
      UX(2) = (U(2)-U(1))/DX
C
C     FORWARD DIFFERENCES (APPLIED ONLY IF V .LT. 0).
C
      GO TO (40,50,60), NO
   40 DO 45 I=1,N1
      IF (V(I) .LT. 0.0D0) UX(I)=(U(I+1)-U(I))/DX
   45 CONTINUE
      GO TO 70
   50 DO 55 I=1,N2
```

```
      IF (V(I) .LT. 0.0D0) UX(I)=(-1.5D0*U(I)+2.0D0*U(I+1)
     2 -.5D0*U(I+2))/DX
   55 CONTINUE
      GO TO 70
   60 DO 65 I=1,N3
      IF (V(I) .LT. 0.0D0)
     2UX(I) = (-11.0D0*U(I)+18.0D0*U(I+1)-9.0D0*U(I+2)+2.0D0*
     3U(I+3))/(6.0D0*DX)
   65 CONTINUE
      IF (V(N2) .LT. 0.0D0)
     2UX(N2)=(-1.5D0*U(N2)+2.0D0*U(N1)-.5D0*U(NX))/DX
   70 IF (V(N1) .LT. 0.0D0)
     2UX(N1)=(U(NX)-U(N1))/DX
      IF (V(NX) .LT. 0.0D0) UX(NX)=UX(N1)
C
      RETURN
      END
```

```
      SUBROUTINE VARSET (AG, AT, AR)
C
C     LOAD THE INTEGRATOR SOLUTION INTO LOCAL STORAGE.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION AG(1), AT(1), AR(1)
C
      COMMON /FUNDAT/
     * XKFAC, GA, SINANG, PHI, ZMIN, ZMAX, PH, AF, PO, NPT, GO, ABSOLT,
     * TFO, TFE, M, MP1, BETA1(41), XK1(41), SPEED1(41), Z(41),
     * CSUBP1(41), G(41), TF(41), RHO(41), DG(41), DTF(41), DRHO(41),
     * VEL(41), DVEL(41), DZTFO(41), DZRHOO(41), DZVELO(41), DZTFP(41),
     * DZRHOP(41), DZVELP(41), DZTFM(41), DZRHOM(41), DZVELM(41),
     * VELPA(41), VELMA(41), DZGO(41), DZGP(41), DZGM(41),
     * TPDE(41), RPDE(41), GPDE(41), YORIG(120), DYORIG(120)
C
      DO 20 I=1,M
      TF(I+1)   = AT(I)
      RHO(I+1)  = AR(I)
      G(I)      = AG(I)
   20 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE YMIXIT (N, Y, Z, M)
C
C     REORDER THE SOLUTION TO REDUCE THE HALF
C     BANDWIDTHS FROM 2*M+2 TO 5.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION Y(N), Z(N)
C
C     G -
C
      I1    = 1
      I2    = 1
      Z(I1) = Y(I2)
      DO 20 I=2,M
      I1    = 2 + 3*(I-2)
      I2    = I
      Z(I1) = Y(I2)
   20 CONTINUE
C
C     TF -
C
      DO 40 I=2,M
      I1    = 3 + 3*(I-2)
      I2    = M + I - 1
      Z(I1) = Y(I2)
   40 CONTINUE
      I1    = 3*M - 1
      I2    = 2*M
      Z(I1) = Y(I2)
C
C     RHO -
C
      DO 60 I=2,M
      I1    = 4 + 3*(I-2)
      I2    = 2*M + I - 1
      Z(I1) = Y(I2)
   60 CONTINUE
      I1    = 3*M
      I2    = 3*M
      Z(I1) = Y(I2)
C
      RETURN
      END
```

```
      SUBROUTINE YUNMIX (N, Y, Z, M)
C
C     RETURN THE SOLUTION TO THE ORIGINAL
C     UNORDERED FORM.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION Y(N), Z(N)
C
C     G -
C
      I1    = 1
      I2    = 1
      Z(I2) = Y(I1)
      DO 20 I=2,M
      I1    = 2 + 3*(I-2)
      I2    = I
      Z(I2) = Y(I1)
   20 CONTINUE
C
C     TF -
C
      DO 40 I=2,M
      I1    = 3 + 3*(I-2)
      I2    = M + I - 1
      Z(I2) = Y(I1)
   40 CONTINUE
      I1    = 3*M - 1
      I2    = 2*M
      Z(I2) = Y(I1)
C
C     RHO -
C
      DO 60 I=2,M
      I1    = 4 + 3*(I-2)
      I2    = 2*M + I - 1
      Z(I2) = Y(I1)
   60 CONTINUE
      I1    = 3*M
      I2    = 3*M
      Z(I2) = Y(I1)
C
      RETURN
      END
```

ORNL/TM-10038

## INTERNAL DISTRIBUTION

1. R. L. Cox
2. T. S. Darland
3. M. V. Denson
4. J. B. Drake
5. G. E. Giles
6. L. J. Gray
7-8. R. F. Harbison
9. M. T. Heath
10. T. L. Hebble
11. J. K. Ingersoll
12. A. A. Khan
13. W. F. Lawkins
14. F. C. Maienschein
15. G. S. McNeilly
16. T. J. Mitchell
17. M. D. Morris
18. J. K. Munro
19. B. D. Murphy
20. E. Ng
21-25. S. Thompson
26. R. C. Ward
27. M. A. Williams
28. A. Zucker
29. P. W. Dickson, Jr. (Consultant)
30. G. H. Golub (Consultant)
31. R. M. Haralick (Consultant)
32. D. Steiner (Consultant)
33. Central Research Library
34. K-25 Plant Library
35. ORNL Patent Office
36. Y-12 Technical Library
    Document Reference Section
37. Laboratory Records--RC
38-39. Laboratory Records Department

## EXTERNAL DISTRIBUTION

40. Dr. Donald M. Austin, Office of Scientific Computing, Office of Energy Research, U.S. Department of Energy, ER-7, Germantown Building, Washington, DC 20545

41. Dr. R. C. Basinger, Lawrence Livermore National Labs., Computing Research and Development Division, L-316, P. O. Box 808, Livermore, California 94550

42. Dr. W. R. Boland, Los Alamos National Labs., C-3, MS B265, Los Alamos, New Mexico 87545

43. Dr. G. D. Bryne, Computing Technology and Services Division, Exxon Research and Engineering Company, Linden, New Jersey 07036

44. Dr. James Corones, Ames Laboratory, Iowa State University, Ames, Iowa 50011

45. Prof. Germund Dahlquist, Royal Institute of Technology, S-100 44 Stockholm 70, Sweden

46. Dr. Kirby Fong, NMFECC, L-560, P. O. Box 5509, Livermore, California 94550

47. Dr. P. W. Gaffney, Christian Michelsen Institute, N-5036 Fantoft, Bergen, Norway

48. Prof. C. W. Gear, Computer Science Department, University of Illinois, Urbana, Illinois 61801

49. Dr. A. C. Hindmarsh, Mathematics and Statistics Division, Lawrence Livermore National Laboratory, Livermore, California 94550

50. Dr. David Kahaner, United States Department of Commerce, National Bureau of Standards, Scientific Computing Division 713, Washington, D.C. 20234

51. Dr. Robert J. Kee, Applied Mathematics Division, 8331, Sandia Laboratories, Livermore, California 94550

52. Prof. Peter D. Lax, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012

53. Dr. Earl Marwill, EG&G - Idaho, Computer Science Center, P. O. Box 1625, Idaho Falls, Idaho 83415

54. Dr. Paul Messina, Argonne National Labs., Computing Services Division, 9700 South Cass Avenue, Argonne, Illinois 60439

55. Dr. E. L. Mitchell, Mitchell & Gauthier, Assoc., Inc., Wayside Square, 801 Main Street, Concord, Massachusetts 01742

56. Dr. Reagan Moore, G. A. Technologies, Inc., P. O. Box 85608, Mail Stop 13/308A, San Diego, California 92138

57. Dr. R. C. Morgan, 4922 Aurora Drive, Kensington, Maryland 20895

58. Dr. Basil Nichols, T-7, Mathematical Modeling and Analysis, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, New Mexico 87545

59. Prof. S. P. Norsett, Institute for Numerisk Mathematics, University of Trondheim (N-7034), Trondheim, Norway

60. Dr. Ronald Peierls, Brookhaven National Labs., Applied Math. Department, Upton, New York 11973

61. Dr. James C. T. Pool, Executive Vice-President, Numerical Algorithms Group, 1101 31st Street, Suite 100, Downers Grove, Illinois 60515-1263

62. Dr. Bill Potratz, IMSL., 2500 Park West Tower One, 2500 City West Boulevard, Houston, Texas 77042-3020

63. Dr. D. J. Rodabaugh, Lockheed Corporation, Dept. 72-30, Bldg. 311, Burbank, California 91520

64. Prof. Diran Sarafyan, Department of Mathematics, University of New Orleans, New Orleans, Louisiana 70122

65. Dr. L. F. Shampine, Sandia Laboratories, Albuquerque, New Mexico 87815

66. Dr. P. G. Tuttle, Babcock and Wilcox, 3315 Old Forest Road, P.O. Box 1260, Lynchburg, Virginia 24505-1260

67. Pseb Vamajoth, c/o Prof. P. C. Jones, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, Illinois 60201

68. Dr. H. A. Watts, Sandia Laboratories, Albuquerque, New Mexico 87185

69. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, Oak Ridge, Tennessee 37830

70-100. Technical Information Center.