RAL 93102
COPY 2 R61 RR
ACCN: 223132.

RAL-93-102

Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-93-102

Implementing a Linear Equation Solver in the CFD Code PHOENICS

M K Boparai

December 1993

Science and Engineering Research Council
"The Science and Engineering Research Council does not accept any responsibility for loss or damage arising from the use of information contained in any of its reports or in any communication about its tests or investigations"

Implementing a Linear Equation Solver in the CFD Code PHOENICS

M.K. BOPARAI

November 1993

Abstract

The CFD code PHOENICS, one of the most widely used commercial codes, is intended to be 'expandable', allowing experienced users to interface their own modelling features to the processing module. This report describes the feasibilty of implementing a user-defined linear equation solver using a simple conjugate gradient algorithm, in the flow-solving module of PHOENICS, without access to the main source code. It shows how the data can be efficiently accessed and stored with reference to the large central storage array of PHOENICS and this can be used as a starting point for incorporating more complex solvers and other features. The problem of heat conduction in a cube which leads to the solution of a system of linear equations is used to test the method. The new user-written routines including a matrix-vector multiplier are given in the appendices, and the way they are interfaced, built and linked with the PHOENICS modules is also shown. Note that the report does not assess the code in any way.

Mathematical Software Group
Computational Modelling Division
Rutherford Appleton Laboratory
Chilton, Didcot
Oxfordshire OX11 0QX

Contents

1 Introduction

This report describes the feasibility of using the computational fluid dynamics (CFD) package PHOENICS, and in particular, the main flow solving module, to implement a linear equation solver. Some preliminary work using the pre-processing module, SATELLITE, is also given. The flexibility and ease of use of the main flow solving module, called EARTH, is assessed by replacing the linear equation solver of PHOENICS by one which uses a conjugate gradient algorithm. The problem used as a basis for testing the solver is that of heat conduction in a cube which leads to the solution of a system of symmetric linear algebraic equations.

Section 2 of this report gives some background to the PHOENICS software. Section 3 describes the preliminaries with the SATELLITE pre-processing module. The user-accessible parts of the processing module EARTH and the *ground.f* program into which the user's own coding is written are described in Section 4. Section 5 describes the implementation of a conjugate gradient routine in EARTH for the heat conduction problem and shows how the data is stored and accessed in EARTH, which is fundamental to being able to write our own features in this module. It also includes the routine for the matrix-vector multiplication which is not provided by the PHOENICS utility routines, and any difficulties encountered during the process. Section 6 shows how to use the exemplary routines provided by PHOENICS to attach the new user-defined routines and how to build and run the EARTH module once this has been done. The structure of the FORTRAN file, *ground.f*, which will contain the new routines and is read by EARTH, is also described here together with any changes made to the Q1 file. The results from the software and the conjugate gradient routine are described in Section 7 and shown at the end of the report. Computer listings of the Q1 and GROUND files, and the Gauss-Seidel routine which is later referred to, are given in the appendices.

2 Background to PHOENICS

PHOENICS is an acronym for Parabolic, Hyperbolic or Elliptic Numerical Integration Code Series. It is a general-purpose code for simulating fluid flow, heat/mass transfer and chemical-reaction phenomena. It can be used to solve problems in one, two or three dimensions, including those exhibiting time-dependence, compressibility, single-phase and multi-phase flow, free surfaces and turbulence. Either polar or curvilinear coordinates can be used on *structured* grids.

The three main modules which make up PHOENICS are:

- SATELLITE, the pre-processor, or the data preparation program, which accepts instructions from the user corresponding to a particular flow simulation, interprets and writes them to a data file which can be read by EARTH.
- EARTH, the processor, which contains the main flow simulation software. It has FORTRAN subroutines which are user-accessible or users may attach their own coding sequences here. EARTH produces an output file to check results and, for graphical display of the results, a file which is to be read by the post-processor.
- PHOTON, which is an interactive post-processing program for displaying grids, contour and vector plots, etc.

PHOENICS uses a finite volume formulation of the differential equations for conservation, which have the form:

$$a_P\phi_P = \sum_{F=W,E,S,N,H,L} a_F\phi_F + a_T\phi_T + b$$

where P is a typical point or node within the cell, F represents the nodes on each of the cell faces, WEST, EAST, SOUTH, NORTH, HIGH and LOW at which the dependent variable ϕ is computed (HIGH is the high neighbour node in the positive z direction and LOW is the low neighbour node in the negative z direction), and a_P, a_F and a_T are coefficients. T is the grid node at earlier time and b represents the source term. This provides an algebraic formula for ϕ involving contributions from all the neighbouring cells.

For the discretisation of the continua, PHOENICS uses the staggered grid system: velocities are computed at the cell walls and all other variables at the cell centres, i.e. cells and nodes for velocity components are 'staggered' relative to those of all other variables.

There are a number of procedures built into PHOENICS for solving the linear equations, including point-by-point, slab-by-slab and whole-field (these terms are described and their use illustrated in Section 5.3). The first type is the Jacobi point-by-point method which updates the nodal values of the variable by simple arithmetic substitution. The slab-by-slab procedure is the default setting and treats the off-slab values as being temporarily known. The whole-field procedure takes all three coordinate direction links into account simultaneously and reaches the solution more rapidly, although it requires more memory than the slab-by-slab procedure [Ref 4].

3 Preliminaries with SATELLITE

The simplest way to begin with PHOENICS is to work with the SATELLITE module. As mentioned above, this is essentially a data preparation program, and the data is set up by means of the input file, Q1, whose structure is described in the following section. The basic steps in the definition of a flow simulation, such as the specification of the grid, variables to be solved and the boundary conditions, are discussed in section 3.2.

3.1 The Q1 file

The purpose of the Q1 ('quick input') file is to transfer the problem definition data from the pre-processor to the processor. It is made up of commands taken from the PHOENICS Input Language (PIL), of which examples are given in the next section. It is read in (or 'compiled') by the SATELLITE and transformed into a data file (EARDAT) for the processing module EARTH. The data is inserted by editing the file in the normal manner, or using the menu system available with the newer versions, or interactively, using the SATELLITE in dialogue mode. The Q1 file is **always** read at the beginning of the SATELLITE run and **must** be always present in the user's directory. It must also contain at least two lines: the first line, which is TALK=T or F, and the last line, which is STOP. TALK=T informs the SATELLITE that the user wants to enter dialogue mode: it will read the Q1 file and then wait for further (interactive) instructions from the user. TALK=F will cause the SATELLITE to read the Q1

file, translate it into the EARDAT file and then stop without any interaction with the user at the terminal.

The data in the Q1 file is arranged in 24 groups, the titles of which are frequently inserted as comments within the file (any lines beginning with blanks in the first two columns are counted as comments). These groups are inserted between the first and last commands given above, and act only as an aid to organise the data in a systematic manner. An empty Q1 file with the group names would be laid out as follows

TALK=T
GROUP 1. Run title and other preliminaries
GROUP 2. Transience; time-step specification
GROUP 3. X-direction grid specification
GROUP 4. Y-direction grid specification
GROUP 5. Z-direction grid specification
GROUP 6. Body-fitted coordinates or grid distortion
GROUP 7. Variables stored, solved & named
GROUP 8. Terms (in differential equations) & devices
GROUP 9. Properties of the medium (or media)
GROUP 10. Inter-phase-transfer processes and properties
GROUP 11. Initialization of variable or porosity fields
GROUP 12. Patchwise adjustment of terms (in differential equations)
GROUP 13. Boundary conditions and special sources
GROUP 14. Downstream pressure for PARAB=.TRUE.
GROUP 15. Termination of sweeps
GROUP 16. Termination of iterations
GROUP 17. Under-relaxation devices
GROUP 18. Limits on variables or increments to them
GROUP 19. Data communicated by satellite to GROUND
GROUP 20. Preliminary print-out
GROUP 21. Print-out of variables
GROUP 22. Spot-value print-out
GROUP 23. Field print-out and plot control
GROUP 24. Dumps for restarts
STOP

PHOENICS has a library of examples of Q1 files written in this format for a variety of problems which can be inspected and changed where necessary. The library is accessible from SATELLITE, by using the command SEELIB. It contains solved problems of many types, for example, free surface flows, flow in porous media, one-phase or two-phase, steady or transient, elliptic or parabolic flows.

3.2 Making Algorithmic Changes in PHOENICS via the Q1 file

In the first instance most changes are made via the Q1 file in which the data settings are made. The following subsections detail any changes to the default settings in PHOENICS which have been made and indicate if these alternatives are available in the program. Where a change has been made, an example is provided.

3.2.1 Boundary Conditions

Boundary conditions are changed in Group 13 of the Q1 file. They are represented as linearised sources of the form $Coefficient\ (Value - \phi)$ in the differential equation, where ϕ is the dependent variable. If no special actions are taken by the user then the EARTH program will assume that the fluid is confined within a container, the boundaries of which are impenetrable to flow of mass, momentum and energy. Boundary conditions are introduced by a number of commands in the Q1 file, e.g. WALL, which specifies where a wall-type boundary operates, COVAL, which indicates the magnitude of the source, i.e. its coefficient and value, and PATCH, to define regions in space and time over which the boundary conditions are to be set by means of the COVAL (COefficient and VAlue) command. They have the following format:

WALL(name, type, fx,lx, fy,ly, fz,lz, ft,lt),
COVAL(name, variable, coefficient, value)

where **name** is a unique identifier of up to 8 characters for the WALL, **type** is used to indicate on which face of the cells involved the wall is (the face options are NORTH, SOUTH, EAST, WEST, HIGH and LOW), **fx** and **lx** are to indicate the first and last cells respectively in the x-direction, etc., **variable** is the dependent variable and **coefficient** and **value** are those in the linearised source above. PATCH has the the same format as WALL, except that the **type** argument may refer to a number of other options as well as the face options for WALL, such as INFLO, for specifying PATCHes at which only inflow is required: thus where a pressure boundary condition prevails at such a PATCH, cell pressures in excess of the outside pressure do not cause outflow. (See the Reference Manual for further detail.) For example, in a fully enclosed flow, it is necessary to fix the pressure at one of the cells in order to determine it uniquely. This is done using the following PATCH command:

PATCH(RELIEF,CELL,NX/2,NX/2,NY/2,NY/2,1,1,1,1)

where RELIEF is the name given to this PATCH, CELL is the PATCH type to indicate that the sources set by the associated COVALs are 'per cell', not per unit area or volume. The remaining arguments indicate the position of the fixed cell in the grid.

However, boundary conditions cannot always be specified by a constant coefficient and a constant value. It may be the case that the coefficient and/or the value are functions of one or more solved-for variables, the value of which cannot be forseen at the stage of data input. To cope with this, PHOENICS provides special flags (GRND, GRND1,...,GRND10) which can be specified as coefficients and/or values in the COVAL command, to activate a set of non-linear sources.

3.2.2 Physical Properties

Density (Kg/m^3)
The default value is 1 which is suitable for air. It is changed in Group 9 of the Q1 file by setting the PIL variable $RHO1$ to the required value.
The user can specify his own coding sequence, also in Group 9 of the GROUND subroutine of the EARTH program, by setting $RHO1 = GRND$ in the Q1 file when there are no built-in

options available or when the density is a function of other variables. The following options are also provided in the subroutine GXRHO, called from GREX in the EARTH program:

Isentropic gas flow:
This is selected by setting $RHO1 = GRND3$, which gives the following formula for the density $density = RHO1A * (p1 + PRESS0) * *RHO1B + RHO1C$, and

Ideal gas flow:
This is selected by setting $RHO1 = GRND5$, which gives the following formula for the density $density = RHO1B * (p1 + PRESS0)/t1$,

where $p1$ is the relative pressure, $t1$ is the temperature determined by the ascription of TMP1 $RHO1A$ is a real parameter used in the first-phase density formulae (default:0), and $RHO1B$ and $RHO1C$ are further parameters of the same kind. $PRESS0$ is a parameter for the reference pressure, to be added to the pressure computed by PHOENICS in order to give the physical pressure needed for calculating the density and other physical properties.

Laminar kinematic viscosity (m^2/s)
This is specified by the PIL variable ENUL. The default is 10^{-5}

Turbulent kinematic viscosity
Specify by changing value of the PIL variable ENUT in the Q1 file. The default is 0.

Temperature
In Version 1.5.2 or greater, temperature is regarded either as a property obtainable from enthalpy through the exression $T = H/CP$, or as a variable for which an equation is solved. It is denoted by TMP1 and its default value is 0.

3.2.3 Dependent Variables

Differential equations for up to 50 dependent variables can be solved simultaneously. Variables for which we solve are specified very simply by the SOLVE (⟨var⟩) command in Group 7 of the Q1 file, where the variable index ⟨var⟩ represents, for example, velocity (U1), enthalpy (H1), temperature (T1).

3.2.4 Grid Generation

As mentioned in Section 2, only structured grids can be used. These are described in terms of either Cartesian, cylindrical polar or curvilinear coordinate systems. The default is Cartesian. The grid may be set up using the following command in Groups 3-5 of the Q1 file:

GRDPWR(grid direction, no. of cells, total length, power)

where the last argument is the power in the mathematical expression used to distribute the cells in the specified direction : a value 1 provides a uniform grid, larger than 1 provides a grid which expands towards the end of the computational domain, and a value less than 1 for a grid

contracting towards the beginning of the domain.

Body-fitted grids can also be generated by use of the PIL command GSET, whose syntax is

GSET(KEYWORD[ARG1,ARG2,ARG3,...]).

The keyword is defined to set, for example, a point, a line, a frame or to transfer grid planes. Detailed information on this command is available on-line, and its use is illustrated in the problem of supersonic flow over a prism in Appendix 1.

3.2.5 Turbulence Models

The command TURMOD, in Group 7 of the Q1 file, is used to activate one of two turbulence models, namely:

TURMOD(KLMODL) selects the KE-LEN1 model, in which KE is the kinetic energy of the turbulence and LEN1 is its length scale, or

TURMOD(KEMODL) selects the KE-EP ($k-\epsilon$ model), in which EP is the rate of dissipation of KE.

3.3 Remarks on SATELLITE

The use of the SATELLITE module, and in particular the use of the Q1 file to set up flow problems, is simplified by the provision of an extensive library of examples. At first it was difficult to run problems which were not closely based on a library case, for example, setting up a non-constant velocity profile at an inlet. In the version used, error messages were not very helpful, e.g. 'Error stop reached. Check data.' These difficulties lessen with practice, however, and allow the user to begin programming more advanced features in the EARTH module, as shown in the following sections.

4 The User-accessible parts of EARTH

EARTH is the main flow-simulating module of PHOENICS. There are three routines in EARTH which are accessible to the user: MAIN, GROSTA and GROUND. These appear in this order to form a single file called *ground.f*, written in standard FORTRAN 77 except for some machine-dependent INCLUDE directives. These three routines are described below and their use in practice will be illustrated in Sections 5 and 6, followed by a description of GREX3, which is an exemplary subroutine supplied with Versions 1.5 and 1.6 of the software.

4.1 The MAIN Program

As in the case of the SATELLITE module, MAIN is accessed only for making changes to the DIMENSION statements. The most common reason for modifying MAIN is to re-dimension the F-array since this is the blank common array that EARTH uses for the storage of all fields and working stores. Other parameters may be reset if it is necessary to increase, for example,

the maximum number of field variables, which is 50 by default, i.e.

```
PARAMETER (NUMPHI=50).
```

This automatically re-dimensions all those arrays that contain data items or flags related to each variable. Another example may be to increase the number of PATCHes, a PIL command to define regions in space and time over which boundary conditions or sources are set.

It should be noted that where associated arrays exist in the SATELLITE module, the dimensions in MAIN of SATELLITE must be exactly the same as those in MAIN of EARTH. However, warnings are given in the places where these changes are made in both the SATELLITE and GROUND files.

4.2 The Subroutine GROSTA

The next subroutine in the *ground.f* file is GROSTA, which is an acronym for 'GROund-STAtion'. It is an interface subroutine which connects the different user-defined subroutines or the existing user-accessible FORTRAN routines GREX3 (see below) to the remainder of EARTH. It is called from within EARTH at several stages in computation. GROSTA then simply determines which GROUND routines are active and calls them sequentially. There are three PIL variables that are used in the Q1 file to activate or deactivate GROUND routines: two logical variables USEGRX and USEGRD, and the character variable NAMGRD. USEGRX must be set to TRUE to activate the use of GREX3. USEGRD must be set to TRUE to activate the use of the GROUND routine (see below), which is originally empty so that the user can insert his own coding. Finally the character variable NAMGRD allows the attachment and activation of routines other than GROUND and GREX3. For example, there are dummy subroutines called SPECGR, SPC1GR, SPC2GR that are provided for users as names for their own special GROUND subroutines which can be linked to the load module as required and activated in any particular run by setting NAMGRD=SPEC.

4.3 The Subroutine GROUND

The subroutine GROUND is called from the subroutine GROSTA (not directly from EARTH) when the variable USEGRD is set to T in the SATELLITE. It is an empty subroutine, organised into 24 parts like the Q1 file in SATELLITE, and performs no function unless the user writes some coding in the spaces provided. The purpose of GROUND-type subroutines is to supplement the data input from SATELLITE by providing those quantities which need to be computed as the flow simulation proceeds. For example, they may be used to provide non-linear fluid property laws or sources of mass, momentum and energy that depend in complex ways upon the the distributions of dependent variables in space and time.

4.4 The Subroutine GREX3

As mentioned above, the GREX3 ('GRound EXample') FORTRAN subroutine is called from GROSTA when USEGRX is set to TRUE in the Q1 file, and it serves two purposes: to provide access to some commonly occurring features such as fluid-property, grid-distortion, linear solver, or boundary conditions, and secondly to serve as an example for users to follow when they are writing their own versions of GROUND coding. It is a long subroutine with

many types of features for flow simulation and solution which needs to be inspected carefully to establish which sections can be used or modified. One such routine which is taken from GREX3 and modified for our purposes is a Gauss-Seidel linear equation solver, and it is listed in Appendix 6 in its original form.

5 On the use of EARTH for Implementing a Linear Equation Solver

In order to determine the feasibility of programming in the EARTH module, we replace the existing linear equation solver in PHOENICS by one which uses a conjugate gradient algorithm. Two implementations of the method were carried out: the first being quite straightforward to test the idea, and the second was for flexibility using only the central storage of PHOENICS. This section gives a brief description of the problem to be solved by the linear equation solver, how the data is stored in PHOENICS, implementation of the conjugate gradient solver using whole-field and full-field storage, and a discussion of some issues which arose during the work.

5.1 The Heat Conduction Problem

This problem is taken from the PHOENICS Library of examples defined in Cases 100-103 and solves for the steady, pure conduction flow in a cube. The cube is heated at one corner and cooled at the diagonally-opposite corner. For the purposes of this work, the main interest in the problem is that its discretisation leads to the solution of a symmetric system of linear algebraic equations, which can be solved by a conjugate gradient algorithm. (The original Library Case 103 uses a Gauss-Seidel routine to solve the linear equations.)

5.2 The Conjugate Gradient Algorithm

The conjugate gradient algorithm [Ref 1,2] is an iterative method which solves a symmetric, positive definite[1] system of linear algebraic equations. Such a system results from the discretisation of many problems of fluid flow, for example, the heat conduction problem given above which is used as a basis for testing the algorithm, or in the solution of potential flow problems.

The algorithm is applied to the following system of n linear equations:

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

where A is a symmetric, positive definite matrix, and $\mathbf{x}$ and $\mathbf{b}$ are vectors. Given an initial guess $\mathbf{x}_0$ (zero, say), a sequence of approximations $\mathbf{x}_i$ to the solution $\mathbf{x}$ is generated as follows:

$$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0 \tag{2}$$

$$\mathbf{p}_0 = \mathbf{r}_0 \tag{3}$$

For $i = 0$ until convergence do (in steps of 1):

$$\alpha_i = (\mathbf{r}_i, \mathbf{r}'_i)/(\mathbf{p}_i, A\mathbf{p}_i) \tag{4}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i \tag{5}$$

[1] A matrix A is positive definite if $\forall x \in \Re^n$ such that $x \neq 0$ we have $x^T Ax > 0$

$$\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i A\mathbf{p}_i \quad (6)$$

$$\beta_i = (\mathbf{r}_{i+1}, \mathbf{r}'_{i+1})/(\mathbf{r}_i, \mathbf{r}'_i) \quad (7)$$

$$\mathbf{p}_{i+1} = \mathbf{r}'_{i+1} + \beta_i \mathbf{p}_i. \quad (8)$$

where $\mathbf{r}$ and $\mathbf{p}$ are known as the residual and search direction vectors respectively. The choice of α_i ensures that $\mathbf{r}_{i+1}$ is orthogonal to $\mathbf{p}_i$, and β_i is chosen so that successive search vectors are conjugate with respect to A, or $(\mathbf{p}_{i+1}, A\mathbf{p}_i) = 0$ for all i. It can be shown that, provided we use exact arithmetic, convergence to the solution is achieved in at most n steps. A preconditioning method was not used as the purpose here was to test the basic idea of interfacing the coded algorithm to the EARTH module.

In order to implement an algorithm such as this, it is first necessary to know how variables are stored and retrieved in PHOENICS. This is discussed in the following 3 sections.

5.3 How the Data is Stored and Accessed

One way of storing and retrieving values of flow variables in PHOENICS, which is the most general way but not the simplest and not advised for beginners, is by directly accessing the large central storage array, called the F array. All access to the flow variables in this array is made by the use of integer indices (e.g. W1, P1, IPHI, ISU) which serve as pointers to where the variables are stored in the F array. The variables are not stored as two-dimensional or three-dimensional arrays. In fact, the cell values of a flow variable at the current *slab*, i.e. those cells having the same value in the z-coordinate direction, are stored in a contiguous block or segment.

In order to access these values, we need the starting location of the segment and the order in which the slab values are stored. All the real variables in EARTH are referred to by use of the F location integer index called LF, in F(LF), defined by:

LF=L0F (variable name) + indicial expression

where L0F is an integer function called the 'zero location index' and is the index of the location just before the segment in which the variables of interest lie. These L0Fs are allocated by PHOENICS at the beginning of the computation and their values depend on the dimensions of the problem, the variables stored and solved for, whether the problem is steady or transient, elliptic or parabolic in nature, etc. The indicial expression is a linear function of the cell-location coordinate indices, IX, IY and IZ, and the total number of cells in each coordinate, NX, NY and NZ. The indicial expressions depend on the type of the variable stored, i.e. whether it is stored slab-wise[2], whole-field[3], etc. For example, the cell (IX,IY) of the current slab of the z-velocity in a single-phase problem, W1, is stored as:

F(L0F(W1) + IY+NY(IX-1))

where L0F(W1) is the location in the F array just before the segment where the current slab of W1 values is stored. Thus the slab of values of W1 is stored in the segment lying from

[2]Slab-wise variables are stored only in segments of NX*NY locations in the F array, and the slabs are overwritten as the equation solving procedure moves through the grid in the z-coordinate direction.

[3]Whole-field-solver variables are stored in segments containing NX*NY*NZ contiguous locations.

F(L0F(W1) +1) to F(L0F(W1) +NX*NY). For this example, the LF of W1 is:

LF=L0F(W1) + IY+NY(IX-1))

Also, the order in which the slab values for each variable are written in each segment is as follows:

First, the NY cells at IX=1,
then, the NY cells at IX=2,
.....
finally, the NY cells at IX=NX.

5.4 Using Whole-field Storage

In using the conjugate gradient algorithm to solve for the linear system of equations $A\mathbf{x} = \mathbf{b}$ we refer to the F array directly for access to **x** and **b** vectors, which reside in the F array as the solution variable, PHI, and the source term, SU, as well as for the off-diagonal elements of the A matrix: AP, AN, AE and AH. They are all stored as whole-field-solver variables, i.e. they occupy segments of the F array containing NX*NY*NZ contiguous locations and are referred to by the pointers IPHI, ISU, IAP, IAN, IAE and IAH as follows:

```
IPHI=L0F(L3PHI)
ISU=L0F(SU)
IAP=L0F(L3AP)
IAN=L0F(L3AN)
IAE=L0F(L3AE)
IAH=L0F(L3AH)
```

Storage for the following vectors is also required: the residual and search vectors, a vector to store the result of the matrix-vector multiplication and a vector to store the main diagonal of the matrix (since PHOENICS does not store this). However, these vectors could not be stored as whole-field variables, and this is discussed in more detail in Section 5.7. We can obtain this storage space by declaring arrays explicitly, although this has obvious limitations when the dimensionality of the problem has to be increased. To obtain more flexibility, we can make use of some of the working space in the F array. Both methods were used for this problem. The first method is straightforward and just involves the standard way of declaring and accessing arrays in FORTRAN 77. The second method is discussed in the next section.

5.5 Using Full-field Storage

The most appropriate way of using parts of the F array as working space for our problem is by making use of the full-field variable storage. Full-field variables are those which have a stored value for each grid cell. Thus there are NX*NY*NZ values stored for each variable of a given kind, but they occupy NZ non-contiguous segments, each containing NX*NY locations. There

are four classes of full-field variables, of which we use two: the whole-field solver variables described above and *dependent* variables. These need not necessarily be the dependent variables of conservation equations, such as velocity, pressure, enthalpy or concentration; they may be variables to which users ascribe a significance of their own. We make use of the dependent variables C1, C3, C5 and C7 (strictly speaking, these are integer indices which refer to the dependent variables as mentioned in Section 3.2, and the odd numbers pertain to the first phase of a problem) for storage of the four vectors listed above, by initially setting:

LR= L0F(C1) for the residual vector,
LPP= L0F(C3) for the search vector,
LAX= L0F(C5) for the result of the matrix-vector multiplication, and
IROW= L0F(C7) for the main diagonal of the A matrix.

Care has to be taken in accessing these variables since they are stored in NZ non-contiguous segments, and these segments are *separated* by segments which pertain to other full-field variables for the same slab. For example, to set the residual vector at the outset of the conjugate gradient algorithm, $\mathbf{r}_0 = \mathbf{b}_0$, it is necessary to set two counters ICEL, say, for the dependent variables and JCEL for the whole-field solver variables, in the double loop as follows:

```
      DO 20 IZ=1,NZ
         DO 10 J=1,NX*NY
                  ICEL= J+ 6*NX*NY*(IZ-1)
                  JCEL= J+ NX*NY*(IZ-1)
                  F(LR+ICEL)= F(ISU+JCEL)
10          CONTINUE
20    CONTINUE
```

where the multiplication by six in the ICEL counter is required since we have six variables that are solved or stored (the four C's described above, as well as two variables used by PHOENICS: TEMP and BLOK) and the *next* segment of NX*NY values for each variable occurs at 6*NX*NY*(IZ-1) locations later. However, we can generalise the solution by introducing an integer, NSEG, to replace the six, whose value is determined from:

```
      NSEG=0
      DO 111 NPHI=1,50
         IF (STORE(NPHI)) NSEG=NSEG+1
111      CONTINUE
```

where NPHI is the number of whole-field variables; it is defaulted to 50. This is not required in the counter for the whole-field solver variables, JCEL, as they are stored in NX*NY*NZ contiguous locations.

Coding sequences of the kind shown above are required throughout the routines, e.g. for updating the iterate and residual, finding the new search vector, calculating the dot-product of two vectors, calculating the main diagonal and for the matrix-vector multiplication routine.

5.6 Algorithm for the Matrix-Vector Multiplier

In the conjugate gradient algorithm, the dominant operation is the matrix-vector multiplication. Since the PHOENICS Utility Library does not provide such a routine, we have written one and it is described below.

The notation (I, J, K) to represent the position of a point maps onto one number L such that L= $(K-1)*NX*NY+(I-1)*NY+J$. If A is a symmetric matrix, AE, AN, AH denote its non-zero diagonal elements and AP is used to apply the boundary conditions as in the Payne-Irons method, then

$A_{L,L}$	corresponds to	AP(L)
$A_{L,L+1}$	corresponds to	AN(L)
$A_{L,L+NY}$	corresponds to	AE(L)
$A_{L,L+NXNY}$	corresponds to	AH(L)
$A_{L,L-1} = A_{L-1,L}$	corresponds to	AN(L-1)
$A_{L,L-NY} = A_{L-NY,L}$	corresponds to	AE(L-NY)
$A_{L,L-NXNY} = A_{L-NXNY,L}$	corresponds to	AH(L-NXNY)

The dependent variables described in the previous section are used to provide storage for the search vector, the main diagonal and the result of the matrix-vector multiplication. Thus the dependent variables C3 (the pointer to the search vector), C5 (the pointer to the resulting vector) and C7 (the pointer to the main diagonal) are passed into the routine, and the starting locations are calculated as follows:

IROW0=L0F(DIAG)
JROW0=L0F(V)
KROW0=L0F(AV).

IAP, IAE, IAN and IAH are the location pointers for matrix elements AP, AE, AN and AH respectively. IX, IY and IZZ are the loop counters.
The algorithm is as follows:

```
Calculate starting locations of the variables AP, AE, AN, AH, IROW, JROW & KROW
    z-loop starts (for NZ iterations, in steps of 1)
        x-loop starts (for NX iterations, in steps of 1)
            y-loop starts (for NY iterations, in steps of 1)
                set ICEL=(IZZ-1)*NX*NY*NSEG +IY+NY*(IX-1)
                set IROW=IROW0+ICEL
                set JROW=JROW0+ICEL
                set KROW=KROW0+ICEL
                set SOLN=F(IROW)*F(JROW)
                if (NX.NE.1) then
                        increment IAE
                        if (IX.NE.1) SOLN=SOLN - F(IAE-NY)*F(JROW-NY)
                        if (IX.NE.NX) SOLN=SOLN - F(IAE)*F(JROW+NY)
                end if
                if (NY.NE.1) then
```

```
                    increment IAN
                    if (IY.NE.1) SOLN=SOLN - F(IAN-1)*F(JROW-1)
                    if (IY.NE.NY) SOLN=SOLN - F(IAN)*F(JROW+1)
                end if
                if (NZ.NE.1) then
                    increment IAH
                    if (IZ.NE.1) SOLN=SOLN - F(IAH-NXNY)*F(JROW-NXNY*NSEG)
                    if (IZ.NE.NZ) SOLN=SOLN - F(IAH)*F(JROW+NXNY*NSEG)
                end if
                F(KROW)=SOLN
            end of y-loop
        end of x-loop
    end of z-loop
```

5.7 Implementation Issues

PHOENICS provides an extensive range of utility subroutines to provide, for example, formulae for turbulent kinematic viscosity and a Gauss-Seidel solver, as well as some service subroutines, e.g. for performing basic operations on vectors such as addition, scalar and vector products, calculation of the unit vector normal to a plane through defined points, on arrays of dimension 3. However, there is no utility routine for a matrix-vector multiplication and hence the need to write our own arose, which is described in Section 5.6. It is based on the Gauss-Seidel solver, GXGAUS (see Appendix 4 for listing), taken from the GREX3 subroutine, but requires many modifications in the references to the matrix elements AP, AE, AN and AH. Also, since the main diagonal of the iteration matrix is not stored explicitly in PHOENICS, this lead to the need to write DIAGON, which uses MATVEC as a framework and combines entries from the off-diagonal elements AE, AN and AH. Also, these off-diagonal elements have positive instead of negative sign, thus references to AE, AN and AH change sign in the MATVEC routine.

The other main point to mention here concerns the access to the full-field variables in the F array in order to make the routines more flexible by not having to declare any arrays explicitly. In particular, there was difficulty in trying to access more than one slab of values and the overwriting of arrays when using the F array to store the work arrays. All the approaches suggested in the documentation were explored but they did not prove fruitful in storing three-dimensional working arrays. Thus, the initial attempt made was at using some of the whole-field solver variables, since these were already being used successfully for storage of the solution variable and the matrix elements. Two ways of accessing the whole-field variables are suggested in the Reference Manual, the L0F and L0FZ functions, both of which were tried without success. The reason may be that there are a very limited number of such variables, which use a considerable amount of memory, and these are reserved for special purposes by PHOENICS. Another type of full-field variable, which was the requirement in order to keep a stored value for each grid cell, called the dependent variables, proved to be adequate. The difficulty with this approach, which is not indicated in the manuals, arises from the fact that these variables are stored in NZ non-contiguous segments of NX*NY locations which are separated by segments pertaining to other full-field variables. Thus all access to these variables has to be modified by multiplying by the number of stored or solved for variables, in order to avoid accessing and

writing over slabs of other variables. The desired flexibility of this approach was then achieved.

6 Writing and Running the Software

6.1 SATELLITE Settings

Firstly, we require the Q1 file for the heat conduction problem from the library. This is obtained by running the SATELLITE module: type *runsat* once in the PHOENICS working directory, and then *load(100-103)*, which also gives us an option for using the Gauss-Seidel solver called from GREX3. Some of the default settings at the end of the Q1 file, which are for using a linear equation solver, need to be set as follows: USEGRX and USEGRD are set to FALSE, since we are attaching our own routine which is called SPECGR (see next section), and thus NAMGRD is set to SPEC. USOLVE is set to TRUE for entry to GROUND sections which contain the linear equation solver. Also, in order to have access to storage for some of the full-field variables in the F array for storing vectors such as the residual and search vectors, the following PIL statements are required:

```
STORE(C1, C3, C5, C7)
SOLUTN(C1, Y, N, Y, N, N, N)
SOLUTN(C3, Y, N, Y, N, N, N)
SOLUTN(C5, Y, N, Y, N, N, N)
SOLUTN(C7, Y, N, Y, N, N, N)
```

where the arguments in the SOLUTN command indicate, respectively, the variable which is to be used, whether it is to be stored, solved for, solved by the whole-field method, solved by the point-by-point method, whether to use an explicit formulation if transient and if the harmonic averaging of exchange coefficients is to be used. The defaults for the last five questions are all N's. The harmonic averaging question relates to how the diffusion coefficients are averaged in order to provide the values used in the finite domain equations (the default setting implies that arithmetic averaging is used).

Finally, if the dimensions of the problem are to be changed, then the number of cells in each coordinate direction needs to be reset, i.e. the values of NX, NY and NZ.

6.2 EARTH Settings

This section shows where to put the new coding and how it fits into the structure of the *ground.f* program. The new subroutines are UUCNGR (which contains the conjugate gradient algorithm), DIAGON and MATVEC. These are called from the subroutine SPECGR which is one of a number of 'SPECial GRound' dummy routines (briefly described in Section 4.2) provided for users to incorporate their coding. SPECGR has been written by using the GREX3 subroutine as a framework: recall from Section 4.4 that GREX3 either serves to provide access to some utility routines provided by PHOENICS (e.g a Gauss-Seidel solver) that can be switched on and off, or it can be used as an example to follow when writing our own features (as we have done here). The original structure of GREX3 has been retained, although much of the coding is not required, for the sake of illustration. The major modifications made to GREX3 are that the Gauss-Seidel solver is replaced by a conjugate gradient solver (called from

Group 8, section 14), the linear equations are solved differently, a matrix-vector multiplication routine and a routine to calculate the main diagonal of the matrix are included. Starting with the original *ground.f* which should reside in the working directory, the conjugate gradient routine is incorporated as follows (the entire listing is given in Appendix 3):

ground.f:

```
PROGRAM MAIN
SUBROUTINE GROSTA
                    ...
                    IF (USEGRX) CALL GREX3
                    ...
                    IF (NAMGRD.EQ.SPEC) CALL SPECGR
                    ...
                    IF (USEGRD) CALL GROUND
                    ...
END
SUBROUTINE GROUND
                    [not used]
END
SUBROUTINE SPECGR
                    ...
                    CALL UUCNGR(NX,NY,NZ)
                    ...
END
SUBROUTINE UUCNGR(NXNY,NZ)
                    ...conjugate gradient routine...
                    CALL DIAGON(C7,NX,NY,NZ,NSEG)
                    CALL MATVEC(C7,C3,C5,NX,NY,NZ,NSEG)
END
SUBROUTINE DIAGON(DIAG,NX,NY,NZ,NSEG)
                    ...calculate main diagonal...
END
SUBROUTINE MATVEC(DIAG,V,AX,NX,NY,NZ,NSEG)
                    ...matrix-vector muliplier...
END
```

Thus GROSTA determines which GROUND routines are active: here it calls the routine, SPECGR, which calls the solver routine, UUCNGR.

Recall from section 3.2 and 3.3.1 that the arguments C3, C5 and C7 in the calls to the subroutines DIAGON and MATVEC are the integer indices which, when used in conjunction with the L0F function, point to where the storage for the working arrays lies in the F array.

6.3 Linking and Building the EARTH Module

Once the *ground.f* file is error-free and compilable, we can build and link the EARTH module by simply typing *bldear* which links the appropriate libraries. Then the SATELLITE program

must be run for the preliminary data settings and pre-processing, by typing *runsat*. If there are any errors at the pre-processing stage in the Q1 file, they will be indicated here and must be dealt with before going any further. Finally, *runpri*, or *runear* in some versions, runs the EARTH module. Thus the sequence of commands is simply:

[compile *ground.f*]
bldear
runsat
runpri /*runear*

The residuals are displayed on the VDU during the run. The flow field values and some plots of the output are sent to the RESULT file in the working directory.

6.4 A Note on PHOENICS Versions 1.5 and 1.6

The work described in this report was completed in 1991 using PHOENICS Version 1.5 and an earlier release of Version 1.6. Both these have clearly been superseded by the current Version 1.6.6, which was released in September 1992. Thus the user-written features in *ground.f* have been interfaced and re-run with Version 1.6.6 without problem (there was some incompatibility with the versions when using the original *ground.f* coding).

The new features supplied by the later versions, for example, the GSET command for grid specification and a menu system for the SATELLITE module and any new EARTH developments, have not appeared to have any influence on the validity or usefulness of the work described here. Although the new EARTH features intended for the *next* version include some conjugate gradient solvers, the implementation methods demonstrated here should still remain useful as a starting point for incorporating further user-defined features.

7 Results

The results from two different problems are given, illustrating the use of both the EARTH and SATELLITE modules respectively.

The results from the heat conduction problem described in Section 5.1 illustrate the use of the EARTH module. The Q1 file which sets up the problem is shown in Appendix 2. After a successful run of the EARTH, the PHOENICS output of flow field variables and plots are automatically written to a file called RESULT in the working directory. The RESULT file for the heat conduction problem using the conjugate gradient solver is given in Appendix 4. Appendix 5 gives some output prompted by the conjugate gradient routine, showing the temperature variation in one plane of the cube, simply for the purposes of comparison with RESULT file. Note that the Y-slab values are generated from $IY = 1$ to $IY = 5$ here, whereas in the PHOENICS RESULT file they are shown from $IY = 5$ to $IY = 1$. The PHOTON plot of the temperature contours at the first Z-slab of the cube is shown in Figure 1, indicating the temperature spread from one corner of the cube to the diagonally-opposite corner.

To illustrate the use of the SATELLITE module, the graphical results (the output from PHOTON) from a separate problem have also been included. This problem arose from the SERC/ERCOFTAC Summer School in 1991 and concerned the computation of high speed

aerodynamic flow over a triangular prism in two dimensions. A plane shock wave travels toward and diffracts around a 40 degree prism. The Q1 file was set up based on the library case 523, which solves for the supersonic flow through a cascade of wedges, and is listed in Appendix 1. The main changes made to the library case included the boundary conditions, physical properties and the use of body-fitted coordinates. A Laplacian-type grid was set up using body-fitted coordinates as shown in Figure 2. The velocity vectors from this problem are shown in Figure 3. The results show the shock wave aligned with the wedge angle as expected (see Figure 4).

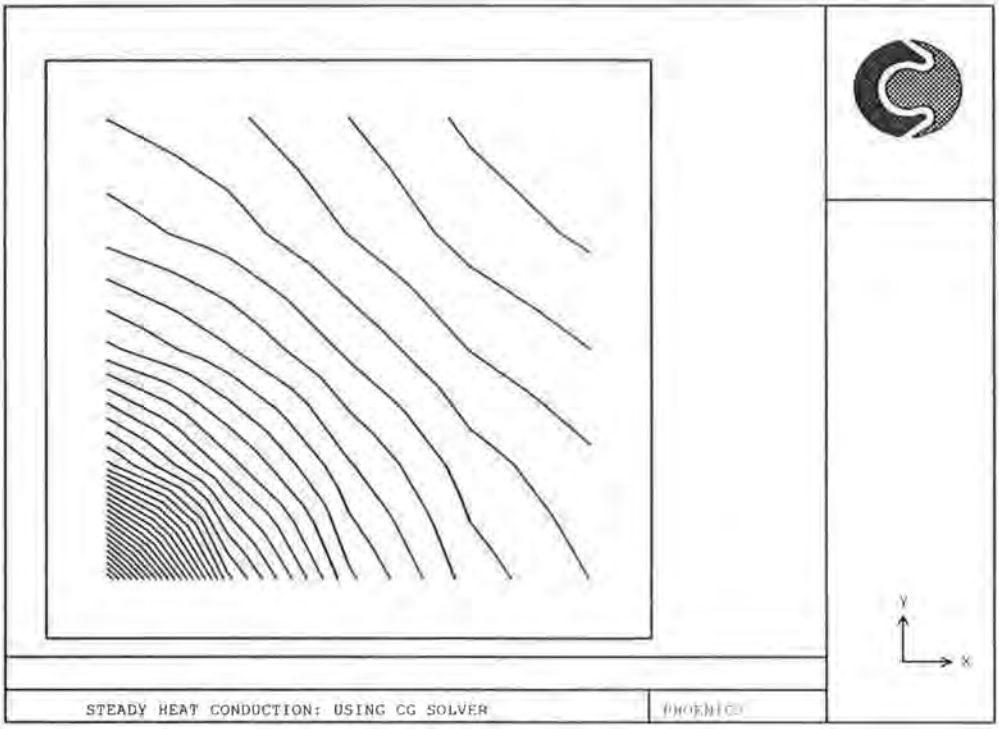


Figure 1: Contour Plot of Temperature at 1st Z-plane

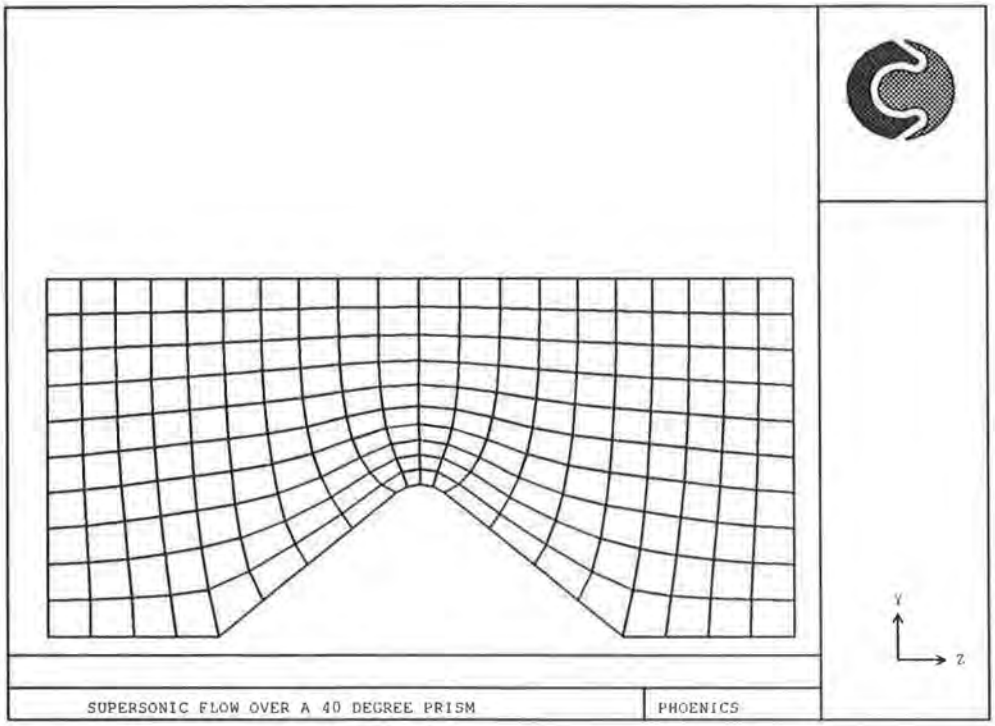


Figure 2: Body-fitted Grid

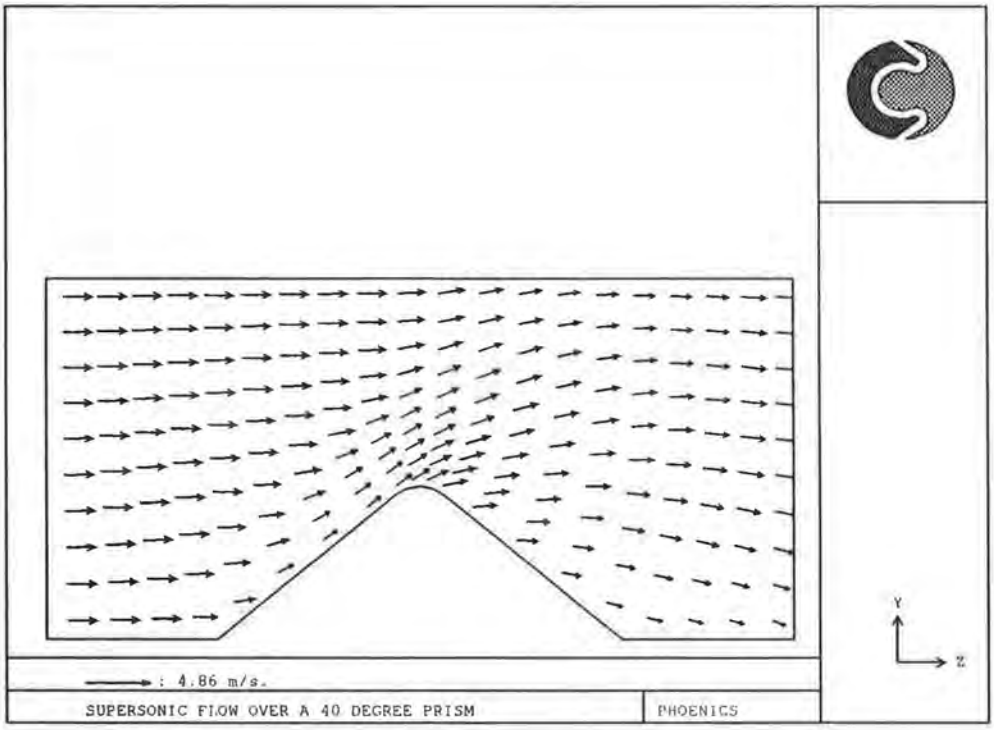


Figure 3: Velocity Components

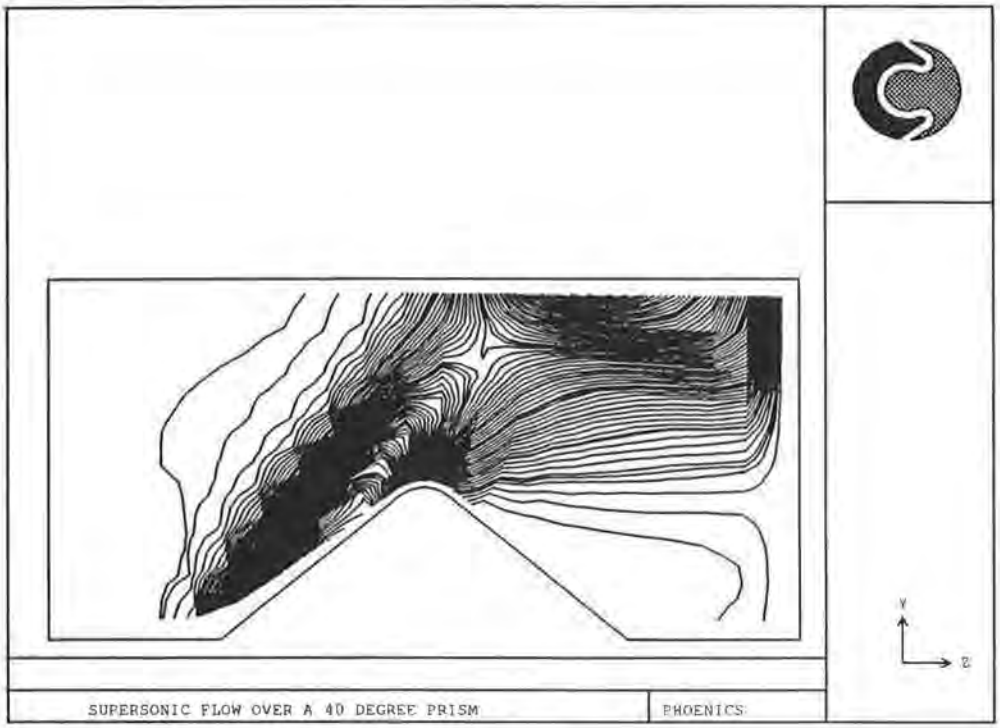


Figure 4: Pressure Contours

8 Summary

This report has described the use of the PHOENICS modules SATELLITE and, in particular, EARTH, with the objective of testing the feasibility of writing user-defined features into the EARTH module. The preliminary work with SATELLITE involved making algorithmic changes in the Q1 file, written in PHOENICS command language (the menu system provided with new versions, ideal for beginners, was not used). Setting up flow problems was much simplified by the provision of the extensive library of examples which could be inspected and carefully modified for many types of problems.

The use of the EARTH module involved the implementation of a conjugate gradient solver to replace the existing linear equation solver. An initial implementation was quite straightforward since it used both explicitly-declared array storage and the F array for storing only the elements of the iteration matrix, which was exemplified in a GREX3 Gauss-Seidel routine. The purpose of the second implementation, which was described in greater detail, was to obtain complete flexibility and used only the F array for storage, that is, both full-field and whole-field variables. However, care needed to be taken to ensure that the vectors were not overwritten, by calculating the locations of the vectors in the F array accordance with the number of such vectors used. The addition of two new routines to calculate the main diagonal of the iteration matrix and to provide a utility for a matrix-vector multiplication operation were also described as these were not available in PHOENICS. With this implementation, the desired flexibility was achieved. Moreover, it provided the valuable experience in understanding how the central storage within EARTH operates, which is fundamental to being able to use the PHOENICS package to our full advantage to solve more complex problems.

The aim in implementing a conjugate gradient solver was not primarily for efficiency, but

it has demonstrated how more complex solvers could be incorporated by the advanced user. The experience gained by such an implemention has been valuable in developing a greater understanding of how PHOENICS operates. Clearly it is easier to use user-declared working space than to access the built-in space in PHOENICS. We would strongly suggest that the users take advantage of all the facilities offered by PHOENICS before writing their own additional modules in practice.

References

[1] **Eisenstat, S.C.** "Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods." SIAM J. Sci. Stat. Comp. Vol 2, No.1, March 1981.

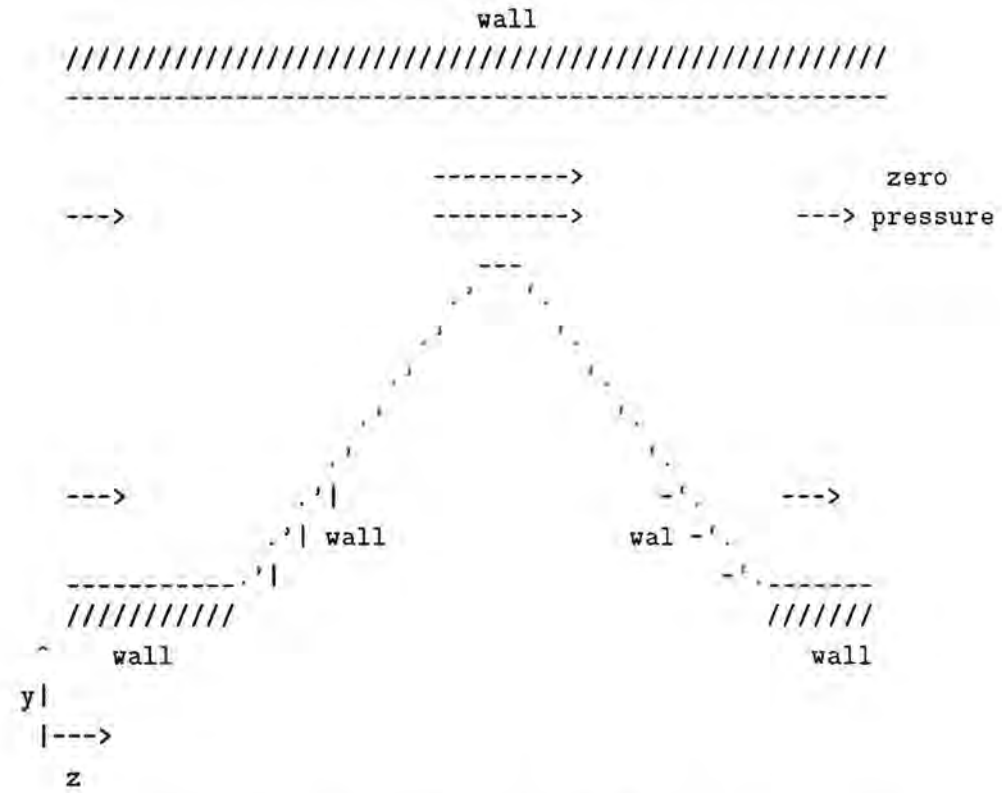
[2] **Meijerink, J.A., van der Vorst, H.A.** "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric *M*-Matrix." Mathematics of Computation, Vol 31, No. 137, January 1977.

[3] **Spalding, D.B.** *The PHOENICS Beginner's Guide (TR/100).* December 1989.

[4] **Spalding, D.B.** *The PHOENICS Reference Manual (TR/200).* December 1989.

APPENDIX 1: The Q1 File for Supersonic Flow over a Prism

The flow considered is supersonic flow over a 40 degree prism with inlet Mach number 3.0 and completely supersonic flow. A leading-edge shock reflects off the pressure surface and should be exactly cancelled at the upstream corner giving a uniform parallel flow through the two surfaces. The flow then expands off the downstream corner and exits through the blade row where two compression waves are formed at the trailing edge. Cyclic boundary conditions are applied upstream and downstream of the cascade. The geometry is as follows:



For simplicity, the flow is treated as isentropic. However, shock theory indicates that there is a significant entropy change across the shocks for the given approach Mach number and wedge angle. Therefore, in future work the isentropic treatment will be replaced with one which allows for entropy changes across shock fronts.

The exit boundary condition is one of fixed pressure according to the post-expansion pressure calculated from gas-dynamic theory; this neglects the presence of trailing-edge shocks. Strictly, the flow is hyperbolic and so the exit boundary condition should be modified accordingly.

```
    GROUP 1. Run title
TEXT(SUPERSONIC FLOW OVER A 40 DEGREE PRISM)
REAL(GASCON,GAMMA,PTOTAL,TTOTAL,RHOTOT,MACHI,PEXRAT,AGAM1,RGAM)
REAL(PIN,TIN,POWER,WIN,RHOIN,PEXIT,CHORD)
REAL(ANGLE1,GZLE,GZBACK,GZFCOR,GZSCOR,GZTE)
INTEGER(IZLE,IZTE,KASE)
GASCON=1.0;GAMMA=1.4;PTOTAL=1.0;TTOTAL=1.0;RHOTOT=1.0;MACHI=3.0
CHORD=4.0;PEXRAT=0.0077
   ** Calculation of inlet velocity
AGAM1=GAMMA-1.;RGAM=1./GAMMA;POWER=GAMMA/AGAM1
PIN=PTOTAL/(1.+AGAM1*MACHI*MACHI/2.)**POWER
RHOIN=RHOTOT/(PTOTAL/PIN)**RGAM
WIN=MACHI*(GAMMA*PIN/RHOIN)**0.5
   ** Calculation of Inlet Temperature
TIN=PIN/(GASCON*RHOIN)
   ** Calculation of exit pressure
PEXIT=PEXRAT*PTOTAL

    GROUP 2. Transience; time-step specification
    GROUP 3. X-direction grid specification
    GROUP 4. Y-direction grid specification
    GROUP 5. Z-direction grid specification

    GROUP 6. Body-fitted coordinates or grid distortion
BFC=T;NONORT=T

    ***set grid dimensions***
GSET(D,1,10,20)
    ***set corner points for the whole frame***
GSET(P,A,0.0,0.0,-1.0)
GSET(P,B,0,0,1)
GSET(P,C,0,1,1)
GSET(P,D,0,1,-1)

REAL(PI,DD,RR,AN,Z1,Y1,Y2)
PI=3.14159
DD=0.54
RR=0.1
AN=40
AN=AN*PI/180.
Z1=RR*SIN(AN)
Y1=DD*TAN(AN)-RR/COS(AN)+RR*COS(AN)
Y2=DD*TAN(AN)-RR/COS(AN)+RR

GSET(P,E1,0,0,-DD)
```

```
GSET(P,E2,0,Y1,-Z1)
GSET(P,E3,0,Y1,Z1)
GSET(P,E4,0,0,DD)

GSET(L,A1,A,E1,4)
GSET(L,12,E1,E2,4)
GSET(L,23,E2,E3,4,ARC,0,Y2,0)
GSET(L,34,E3,E4,4)
GSET(L,4B,E4,B,4)
GSET(L,BC,B,C,10)
GSET(L,CD,C,D,20)
GSET(L,DA,D,A,10)

    ***define frame***
GSET(F,F1,A,E1.E2.E3.E4,B,-,C,-,D,-)
    ***match defined frame onto I1***
GSET(M,F1,+K+J,1,1,1,LAP10.FFFTFF)
    ***copy I1 to I2***
GSET(C,I2,F,I1,+,.1,0,0)
GSET(I)
    VIEW

    GROUP 7. Variables stored, solved & named
SOLVE(P1,V1,W1);STORE(RHO1)
SOLUTN(P1,Y,Y,Y,N,N,N)

    GROUP 8. Terms (in differential equations) & devices
    GROUP 9. Properties of the medium (or media)
ENUL=0.0;ENUT=0.0
   ** Use Isentropic Density Law
RHO1=GRND3;RHO1A=RHOTOT/PTOTAL**RGAM;RHO1B=RGAM;RHO1C=0.;PRESSO=0.
DRH1DP=GRND3

    GROUP 10. Inter-phase-transfer processes and properties
    GROUP 11. Initialization of variable or porosity fields
CONPOR(1.5,NORTH,1,1,NY,NY,1,1)
FIINIT(P1)=PIN;FIINIT(W1)=WIN;FIINIT(RHO1)=RHOIN

    GROUP 12. Unused
    GROUP 13. Boundary conditions and special sources
INLET(INLET,LOW,1,1,1,NY,1,1,1,1)
VALUE(INLET,P1,RHOIN*WIN)
VALUE(INLET,W1,WIN)
PATCH(OUTLET,HIGH,1,1,1,NY,NZ,NZ,1,1)
COVAL(OUTLET,P1,5.E4,PEXIT)
```

```
COVAL(OUTLET,V1,ONLYMS,0.0)
COVAL(OUTLET,W1,ONLYMS,0.0)

PATCH(RELIEF,CELL,1,1,NY,NY,NZ,NZ,1,1)
COVAL(RELIEF,P1,FIXP,0.0)
COVAL(RELIEF,V1,ONLYMS,0.0)
COVAL(RELIEF,W1,ONLYMS,0.0)

    GROUP 14. Downstream pressure for PARAB=.TRUE.
    GROUP 15. Termination of sweeps
LSWEEP=50

    GROUP 16. Termination of iterations
LITER(P1)=15

    GROUP 17. Under-relaxation devices
RELAX(P1,LINRLX,0.8);RELAX(RHO1,LINRLX,1.0)
RELAX(V1,FALSDT,0.5);RELAX(W1,FALSDT,0.5)

    GROUP 18. Limits on variables or increments to them
VARMIN(V1)=-50.;VARMIN(W1)=-50.;VARMAX(V1)=50.;VARMAX(W1)=50.
VARMIN(RHO1)=0.1*RHOIN;VARMAX(RHO1)=RHOTOT
VARMIN(P1)=0.01*PIN;VARMAX(P1)=PTOTAL

    GROUP 19. Data communicated by satellite to GROUND
    GROUP 20. Preliminary print-out
    GROUP 21. Print-out of variables
    GROUP 22. Spot-value print-out
IXMON=2;IZMON=9;NPRMON=LSWEEP

    GROUP 23. Field print-out and plot control
NPRINT=LSWEEP
PATCH(PLOT1,PROFIL,NX/2,NX/2,1,1,1,NZ,1,1)
PLOT(PLOT1,P1,0.0,0.0)
PATCH(CASCADE,CONTUR,1,NX,1,1,1,NZ,1,1)
PLOT(CASCADE,P1,0.0,20.0);PLOT(CASCADE,W1,0.0,20.0)
STOP
```

APPENDIX 2: The Q1 File for the Heat Conduction Problem

```
TALK=T;RUN( 1, 1);VDU=GRAPHICS
** LOAD(100) from the PHOENICS Input Library
    ****** TO LOAD CASE: TYPE LOAD(100) ******
  A cube of uniform-property material is heated at one corner
 and cooled at the diagonally-opposite corner. The steady-state
 temperature distribution is computed by means of the whole-field
 solver, in combination with the x-wise, y-wise and z-wise one-
 dimensional-adjustment features and with the user-selected
 block-correction feature..

  The following diagram illustrates the geometry of the problem
 when NX=NY=NZ=5.

                 /------/------/------/------/------/
                /      /      /      /      /    --/|---temp. fixed
               /------/------/------/------/------/ /   at 1.0
              /      /      /      /      /      /|/|
             /------/------/------/------/------/ / /
            /      /      /      /      /      /|/|/|
           /------/------/------/------/------/ / / /
          /      /      /      /      /      /|/|/|/|
 north   |------|------|------|------|------| / / / /
   ^     |      |      |      |      |      |/|/|/|/|
   |     |------|------|------|------|------| / / / /  high
   |     |      |      |      |      |      |/|/|/|/    ^
  y|     |------|------|------|------|------| / / /    /
   |     |      |      |      |      |      |/|/|/    / z
   |     |------|------|------|------|------| / /    /
   |     |      |      |      |      |      |/|/    /
   |     |------|------|------|------|------| /    /
   |     |  /   |      |      |      |      |/    /
 south   |------|------|------|------|------|   low
          /               x
         /west---------------------->east
        /
  temp. fixed at 0.0

  GROUP 1. Run title and other preliminaries
TEXT(3D STEADY HEAT CONDUCTION IN A CUBE
  Data settings begin below.

REAL(XLENGTH,YLENGTH,ZLENGTH)
XLENGTH=1.0;YLENGTH=1.0;ZLENGTH=1.0
```

```
NX=5;NY=5;NZ=5
    Note that there is no need to declare NX, NY and NZ as
    INTEGERs, because these, unlike XLENGTH, etc
    are already part of the PHOENICS Input Language

    MESG is used for sending messages to the VDU. Note the use of
    colons to ensure that it is the numerical value that is sent,
    not the name.
mesg(3D STEADY HEAT CONDUCTION IN A CUBE
mesg(NX=:nx:; NY=:NY:; NZ=:NZ:
mesg(XLENGTH=:XLENGTH:; YLENGTH=:YLENGTH:; ZLENGTH=:ZLENGTH:

    GROUP 2. Transience; time-step specification
STEADY=T

    GROUP 3. X-direction grid specification
   **Domain is XLENGTH m long in x-direction, with equal intervals
GRDPWR(X,NX,XLENGTH,1.0)

    GROUP 4. Y-direction grid specification
   **Domain is YLENGTH m long in y-direction, with equal intervals
GRDPWR(Y,NY,YLENGTH,1.0)

    GROUP 5. Z-direction grid specification
   **Domain is ZLENGTH m long in z-direction, with equal intervals
GRDPWR(Z,NZ,ZLENGTH,1.0)

    GROUP 6. Body-fitted coordinates or grid distortion
    GROUP 7. Variables stored, solved & named
   **Choose first-phase enthalpy (H1) as dependent variable
     and activate the whole-field elliptic solver
SOLUTN(H1,Y,Y,Y,N,N,N);NAME(H1)=TEMP
   ** Store BLOK, the marker variable needed for the activation
      of the block-correction feature in the linear-equation
      solver.
STORE(BLOK)
   ** Indicate that is is to variable 14 that the block-
      correction feature will be applied
IVARBK=14

    GROUP 8. Terms (in differential equations) & devices
   **For pure conduction, cut out built-in source and convection
     terms
TERMS(TEMP,N,N,Y,Y,Y,N)
```

```
    GROUP 9. Properties of the medium (or media)
   **Thermal conductivity will be ENUL*RHO1/PRNDTL(TEMP), so :
ENUL=1.0;RHO1=1.0;PRNDTL(TEMP)=1.0

    GROUP 10. Inter-phase-transfer processes and properties
    GROUP 11. Initialization of variable or porosity fields

   ** Define eight block-correction blocks by ascribing BLK
     values to chosen locations by way of:-
     FIINIT, for block 1, and PATCH and INIT for the other
     blocks
     Note that INIADD is = F, so that the values set by INIT
     replace the value set by FIINIT. The only region which
     retains that value is that in the east, north, high
     corner.

INIADD=F;FIINIT(BLOK)=1.0
  West, south, low corner
PATCH(BLOK2,INIVAL,1,NX/2,1,NY/2,1,NZ/2,1,LSTEP)
INIT(BLOK2,BLOK,0.0,2.0)
  East, north, low corner
PATCH(BLOK3,INIVAL,NX/2+1,NX,1,NY/2,1,NZ/2,1,LSTEP)
INIT(BLOK3,BLOK,0.0,3.0)
  West, south, low corner
PATCH(BLOK4,INIVAL,1,NX/2,NY/2+1,NY,1,NZ/2,1,LSTEP)
INIT(BLOK4,BLOK,0.0,4.0)
  East, north, low corner
PATCH(BLOK5,INIVAL,NX/2+1,NX,NY/2+1,NY,1,NZ/2,1,LSTEP)
INIT(BLOK5,BLOK,0.0,5.0)
  West, south, high corner
PATCH(BLOK6,INIVAL,1,NX/2,1,NY/2,NZ/2+1,NZ,1,LSTEP)
INIT(BLOK6,BLOK,0.0,6.0)
  East, south, high corner
PATCH(BLOK7,INIVAL,NX/2+1,NX,1,NY/2,NZ/2+1,NZ,1,LSTEP)
INIT(BLOK7,BLOK,0.0,7.0)
  West, north, high corner
PATCH(BLOK8,INIVAL,1,NX/2,NY/2+1,NY,NZ/2+1,NZ,1,LSTEP)
INIT(BLOK8,BLOK,0.0,8.0)

    GROUP 12. Unused
    GROUP 13. Boundary conditions and special sources
   **Corner at IX=IY=IZ=1
PATCH(COLD,CELL,1,1,1,1,1,1,1,1)
   **Fix temperature to zero
COVAL(COLD,TEMP,1.E2,0.0)
```

```
   **Corner at IX=NX, IY=NY, IZ=NZ
PATCH(HOT,CELL,NX,NX,NY,NY,NZ,NZ,1,1)
   **Fix temperature to 1.0
COVAL(HOT,TEMP,1.E2,1.0)

    GROUP 14. Downstream pressure for PARAB=.TRUE.
    GROUP 15. Termination of sweeps
RESREF(TEMP)=1.E-3;LSWEEP=1

    GROUP 16. Termination of iterations
   **Terminate iterations when average correction falls to
     5.E-7 or when 100 iterations have been performed. (The
     minus sign is a signal to activate progress-of-solution
     print-out on the screen at a sweep frequency dictated
     by the value of TSTSWP.)
ENDIT(TEMP)=5.E-7;LITER(TEMP)=-100
  ** Set the over-relaxation factor for the linear-equation
     solver.
OVRRLX=1.7
  ** Set the frequency of application of the block=correction
     feature to once per iteration
ISOLBK=1
  ** Set the frequencies of application of the one-dimensional
     correction features in the linear-equation solver to once
     per itration for each direction.
ISOLX=1;ISOLY=1;ISOLZ=1

    GROUP 17. Under-relaxation devices
    GROUP 18. Limits on variables or increments to them
    GROUP 19. Data communicated by satellite to GROUND
    GROUP 20. Preliminary print-out
ECHO=F

    GROUP 21. Print-out of variables
   **Print fields of temperature
OUTPUT(TEMP,Y,N,N,N,N,N)

    GROUP 22. Spot-value print-out
IXMON=NX/2+1;IYMON=NY/2+1;IZMON=NZ/2+1

    GROUP 23. Field print-out and plot control
NXPRIN=NX/5;NYPRIN=NY/5;NZPRIN=NZ/5
   **Plot a profile along the line IX=nx/2,IZ=nz/2
PATCH(YLINE,PROFIL,NX/2,NX/2,1,NY,NZ/2,NZ/2,1,1)
PLOT(YLINE,TEMP,0.0,0.0);PLOT(YLINE,BLOK,0.0,0.0)
```

```
   **Plot a contour diagram for the plane IX=nx/2
PATCH(XPLANE,CONTUR,NX/2,NX/2,1,NY,1,NZ,1,1)
   **Let the diagram have 20 temperature intervals
PLOT(XPLANE,TEMP,0.0,20.0);PLOT(XPLANE,BLOK,0.0,20.0);ICHR=2

    GROUP 24. Dumps for restarts
NOWIPE=T

       Interesting variants include the following:-
    1. By putting FIINIT(TEMP)= ... in GROUP 11, see how the
       initial guess influences solution time, final result etc.
    2. By changing NX, NY and NZ, see how increasing or decreasing
       the fineness of spatial subdivision influences solution
       time, results, etc.
    3. By changing XLENGTH, YLENGTH and ZLENGTH, see how the
       solution changes when the block is made long and thin,
       short and fat, etc.
    4. By changing PRNDTL(TEMP), see how the value of the
       conductivity changes the solution.
    5. By changing the third, fourth, and further  arguments of
       PATCH(HOT,... and PATCH(COLD,..., explore how changing the
       boundary-condition locations influence the solution.
       Also, introduce more PATCHes at which boundary conditions
       are provided; and display the results in different ways by
       introducing more PATCHes for profile and contour plots.
    6. By changing the COVAL coefficient from 1.E2 to FIXFLU,
       explore the effect of having fixed-flux rather than fixed-
       value boundary conditions; but remember that if all the
       boundary conditions are of fixed-flux type, the solution is
       not uniquely defined.
    7. By changing the values of OVRRLX, ISOLBK, ISOLX, ISOLY,
       ISOLZ, see how these solution-control parameters
       influence the solution and the computer time and number
       of iterations needed to attain it.
    8. By changing the number and location of the blocks used in
       the block-correction feature, establish the influences of
       factors on the accuracy and computer time.
 **END OF LIBRARY CASE 100
     ****** TO LOAD CASE: TYPE LOAD(100-101) ******
    GROUP 1. Run title and other preliminaries
TEXT(3D HEAT COND. WHOLE-FIELD OUT-OF-CORE
    The case is identical to the previous case, except that:
  (a) the block-correction feature is switched off; and
  (b) the calculation is run using out-of-core (ie disk)
```

```
  storage for the temperature.

    The purpose is to check the correct working of the
  disk-storage option: the results should be identical,
  irrespective of the storage medium in use, provided that a
  sufficient number of sweeps is made.

    EARTH normally starts working out-of-core when it finds that
  the dimension of the F array (see MAIN of EARTH) is insufficient
  for wholly in-core operation. However, to test the option without
  having to re-set the F-array dimension, recompile & relink, the
  variables DEBUG and NFMAX are set: they make EARTH act
  as though it has an F-array dimension of NFMAX.

SOLUTN(BLOK,N,P,P,P,P,P)
DEBUG=T;NFMAX=52000
LSWEEP=10

     ****** TO LOAD CASE: TYPE LOAD(100-102) ******
TEXT(3D HEAT COND. SLAB-WISE OUT-OF CORE
    On further reduction of the F-array dimension, EARTH reports
  that it has insufficient space to use the whole-field solver.
  In this circumstance, in order to reduce the storage needs, EARTH
  de-activates the whole-field solver, and activates the slab-by-
  slab solver. Consequently, many sweeps are now needed to get to
  the solution, but the final solution is the same as before.
    The long execution time of this run, in comparison with
  the previous one, is a good illustration of the desirability
  of solving conduction-only problems by means of a whole-field
  linear-equation solver.
DEBUG=T
NFMAX=52000
LSWEEP=300;RESREF(14)=1.E-10;TSTSWP=25
OUTPUT(TEMP,Y,Y,Y,Y,Y,Y)
ITABL=2;NPLT=25



     ****** TO LOAD CASE: TYPE LOAD(100-103) ******
TEXT(STEADY HEAT CONDUCTION: USING CG SOLVER)
    Here, the linear-equation solver provided in EARTH is
  replaced by a conjugate gradient solver written in GROUND.F
  (or the Gauss-Seidel solver provided in GXGAUS), which are
  called from GREX3, in Group 8. Either can be selected by
  setting USOLVE=T and CSG3=CNGR (for the conjugate gradient
```

```
  solver) or CSG3=GAUS (for the Gauss-Seidel solver).

STORE(C1,C3,C5,C7)

SOLUTN(C1,Y,N,Y,N,N,N)
SOLUTN(C3,Y,N,Y,N,N,N)
SOLUTN(C5,Y,N,Y,N,N,N)
SOLUTN(C7,Y,N,Y,N,N,N)
USEGRX=F
USEGRD=F
USOLVE=T;CSG3=CNGR;LITER(14)=-125
NAMGRD=SPEC
OUTPUT(TEMP,Y,Y,Y,Y,Y,Y)
OUTPUT(C1,N,N,N,N,N,N)
OUTPUT(C3,N,N,N,N,N,N)
OUTPUT(C5,N,N,N,N,N,N)
OUTPUT(C7,N,N,N,N,N,N)
DEBUG=T
NFMAX=52000
LSWEEP=1;NOWIPE=F
 LIBREF=103
 **END OF LIBRARY CASE 103
STOP
```

APPENDIX 3: The GROUND.F Program to Implement the Conjugate Gradient Solver

```
C FILE NAME GROUND.FTN--------------------------------161092
C   THIS IS THE MAIN PROGRAM OF EARTH
C
C   (C) COPYRIGHT 1984, 1985,1986,1987,1988,1989,1990,1991,1992,
C   CONCENTRATION HEAT AND MOMENTUM LTD. ALL RIGHTS RESERVED.
C   This subroutine and the remainder of the PHOENICS code are
C   proprietary software owned by Concentration Heat and Momentum
C   Limited, 40 High Street, Wimbledon, London SW19 5AU, England.
C
C
      PROGRAM MAIN
C
C 1   The following  COMMON's, which appear identically in the
C     satellite MAIN program, allow up to 50 dependent variables to
C     be solved for (or their storage spaces to be occupied by
C     other variables, such as density). If a larger number is
C     required, the PARAMETER NUMPHI should be reset to the required
C     larger number. Numbers smaller than 50 are not permitted.
C
      PARAMETER (NUMPHI=50,   NM=NUMPHI,NM4=NM*4)
C
      COMMON/LGE4/L4(NM)
     1/LDB1/L5(NM)/IDA1/I1(NM)/IDA2/I2(NM)/IDA3/I3(NM)/IDA4/I4(NM)
     1/IDA5/I5(NM)/IDA6/I6(NM)/GI1/I7(NM)/GI2/I8(NM)/HDA1/IH1(NM)
     1/GH1/IH2(NM)/RDA1/R1(NM)/RDA2/R2(NM)/RDA3/R3(NM)/RDA4/R4(NM)
     1/RDA5/R5(NM)/RDA6/R6(NM)/RDA7/R7(NM)/RDA8/R8(NM)/RDA9/R9(NM)
     1/RDA10/R10(NM)/RDA11/R11(NM)
     1/GR1/R12(NM)/GR2/R13(NM)/GR3/R14(NM)/GR4/R15(NM)
     1/IPIP1/IP1(NM)/HPIP2/IHP2(NM)/RPIP1/RVAL(NM)/LPIP1/LVAL(NM)
     1/IFPL/IPLO(NM)/RFPL1/ORPRIN(NM)/RFPL2/ORMAX(NM)
     1/RFPL3/ORMIN(NM)/IDA7/ID7(NM)/IDA8/ID8(NM)
      LOGICAL L4,L5,DBGFIL,LVAL
      CHARACTER*4 IH1,IH2,IHP2
C
      COMMON/FO1/I9(NM4)
      COMMON/DISC/DBGFIL
      COMMON/LUNITS/LUNIT(60)
C
      EXTERNAL WAYOUT
C
C 2   Set dimensions of data-for-GROUND arrays here. WARNING: the
C     corresponding arrays in the MAIN program of the satellite
```

```
C     (see SATLIT) must have the same dimensions.
      PARAMETER (NLG=20, NIG=20, NRG=100, NCG=10)
C
      COMMON/LGRND/LG(NLG)/IGRND/IG(NIG)/RGRND/RG(NRG)/CGRND/CG(NCG)
      LOGICAL LG
      CHARACTER*4 CG
C
C 3   Set dimensions of data-for-GREX arrays here. WARNING: the
C     corresponding arrays in the MAIN program of the satellite
C     (see SATLIT) must have the same dimensions.
      PARAMETER(NLSG=30, NISG=40, NRSG=100,NCSG=10)
C
      COMMON/LSG/LSGD(NLSG)/ISG/ISGD(NISG)/RSG/RSGD(NRSG)/CSG/CSGD(NCSG)
      LOGICAL LSGD
      CHARACTER*4 CSGD
C
C 4   Set dimension of patch-name array here. WARNING: the array
C     NAMPAT in the MAIN program of the satellite must have the
C     same dimension.
      PARAMETER (NPNAM=200)
C
      COMMON/NPAT/NAMPAT(NPNAM)
      COMMON/LWFUN1/DOSKIN(NPNAM)
      COMMON/LWFUN2/DHCHKD(NPNAM)
      CHARACTER*8 NAMPAT
      LOGICAL DOSKIN,DHCHKD
C
C     CONFIG FILE name declaration.
      COMMON/CNFG/CNFIG
      CHARACTER CNFIG*48
C
C 5   The numbers in the next statement indicates how much computer
C     memory is to be set aside for storing the main and auxiliary
C     variables. The user may alter them if he wishes, to accord
C     with the number of grid nodes and dependent variables he is
C     concerned with.
      PARAMETER (NFDIM=600000)
C
      COMMON F(NFDIM)
C
C 6   The following three statements concern storage for the PATCH-wise
C     variables. If more than 30 PATCH-wise variables are required
C     NPVDM should be increased and the common block /LBPV/ in the
C     INCLUDE 'file GRDLOC should be lengthened.
      PARAMETER (NPVDM=30)
```

```
      COMMON/INDPV/NPVMX,NIMAX,NITOT,LOPV(NPVDM)
C
      CALL SUB2(NPVMX,NPVDM,NIMAX,NPVDM)
C
      CALL CNFGZZ(2)
      CALL EARSET(1)
      CALL OPENFL(6)
C
      CALL MAIN1(NFDIM,NUMPHI,NLSG,NISG,NRSG,NCSG,NLG,NIG,NRG,NCG)
      CALL WAYOUT(0)
      STOP
      END
C*****************************************************************
      SUBROUTINE GROSTA
      INCLUDE 'lp16/d_earth/SATEAR'
      INCLUDE 'lp16/d_earth/GRDLOC'
      INCLUDE 'lp16/d_earth/GRDEAR'
C
C.... This subroutine acts as a junction-box, directing control to
C     the GROUNDs selected by the SATELLITE settings of USEGRX,
C     NAMGRD & USEGRD.
C
      IF(USEGRX) CALL GREX3
C
C.... SPECGR, SPC1GR, SPC2GR and SPC3GR are names which the user may
C     give to "special GROUNDs" of his own.
C
      IF(NAMGRD.NE.'NONE') THEN
        IF(NAMGRD.EQ.'SPEC') THEN
          CALL SPECGR
C*
C* NOTE THE FOLLOWING CALLS HAVE BEEN COMMENTED OUT SINCE WE ARE ONLY
C* USING 1 NAMED GROUND SUBROUTINE, SPECGR, AS STATED IN THE Q1 FILE.
C* THESE SUBROUTINES, WHICH ARE USUALLY ATTACHED AT THE END, HAVE
C* THUS BEEN DELETED
C*
C*        ELSEIF(NAMGRD.EQ.'SPC1') THEN
C*          CALL SPC1GR
C*        ELSEIF(NAMGRD.EQ.'SPC2') THEN
C*          CALL SPC2GR
C*        ELSEIF(NAMGRD.EQ.'SPC3') THEN
C*          CALL SPC3GR
C*        ELSEIF(NAMGRD.EQ.'STRA') THEN
C*          CALL STRAGR
C*
```

```
        ELSE
          CALL WRITBL
          CALL WRITST
          CALL WRIT40('NAMGRD set but no CALL made, ie.        ')
          CALL WRIT1A('NAMGRD  ',NAMGRD)
          CALL WRIT40('sections 9 and 10 added to Group 19     ')
          CALL WRIT40('Permissible calls for this GROSTA are:- ')
          CALL WRIT40('SPEC, SPC1, SPC2, SPC3,                 ')
          CALL WRIT40('STRA,COAL                               ')
          CALL WRIT40('Use upper-case names only               ')
          CALL WRITST
          CALL WRITBL
          CALL WAYOUT(2)
        ENDIF
      ENDIF
C
C.... The subroutine GROUND attached to the bottom of this file is
C     an unallocated blank form into which the user can insert his
C     own FORTRAN sequences. The PIL parameter USEGRD governs entry
C     to it.
C
C* NOTE THE FOLLOWING CALL HAS BEEN COMMENTED AND THE STANDARD CODING
C* FOR THIS SUBROUTINE WHICH IS NORMALLY ATTACHED HAS BEEN DELETED,
C* SINCE WE HAVE SET USEGRD=F IN THE Q1 FILE
C*
C*     IF(USEGRD) CALL GROUND
C*
C
      IF(IGR.EQ.20) THEN
        IF(ECHO) THEN
          CALL DATPRN(Y,Y,Y,Y,  Y,Y,Y,Y,  Y,Y,Y,N,  Y,Y,Y,Y,
     1                Y,Y,Y,Y,  Y,Y,Y,Y)
        ELSE
          CALL DATPRN(Y,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N,N)
        ENDIF
      ENDIF
      END

C*************************************************************************

      SUBROUTINE SPECGR
C***
C*** NOTE THAT IN CODING THIS SUBROUTINE WE HAVE TAKEN THE STANDARD GREX3
C*** SUBROUTINE AND MODIFIED IT. THUS MOST OF THE ORIGINAL LINES FROM GREX3
C*** ARE LEFT HERE IN ORDER DEMONSTRATE HOW SOME OF THE GREX SUBROUTINES
```

```
C*** PROVIDED BY PHOENICS CAN BE USED AS EXAMPLES OF WAYS OF IMPLEMENTING
C*** OUR OWN FEATURES.
C*** THE MAIN POINT TO NOTE IS WHERE THE USER-DEFINED SUBROUTINE HAS BEEN
C*** CALLED FROM,  I.E.  IN  GROUP 8, SECTION 14 -- USER'S SOLVER
C***
C**************************************************************************

C     FILE NAME GREX3.FTN---------------- 120290
C     This subroutine is called by PHOENICS subroutine GROSTA
C     (in file GROUND.FTN) when the PIL parameter USEGRX=T .
C
C----------------- Explanatory Notes ----------------------------
C
C     Subroutine GREX3, which is supplied with Version 1.5 of
C     PHOENICS, performs all the functions of GREX2, which was
C     supplied with Versions 1.4, and several additional ones.
C
C     It makes extensive use of subsidiary subroutines, especially
C     those in the file GX.FTN. For example, GXRHO supplies the
C     coding for the density options activated by means of
C     RHO1=GRND1, GRND2,...etc. The functions of these subroutines
C     are explained briefly at the points in GREX3 at which they
C     are called, and more extensively by way of comments at the
C     head of each subroutine.
C
C     The GX subroutines that are called in group 13 are activated
C     by settings in the SATELLITE, mainly by way of PATCH names
C     and of setting material properties or other auxiliary
C     quantities used in calculations to GRND(n) options.
C     They are useful both in themselves and as examples, to the
C     advanced user of PHOENICS, of ways in which he can introduce
C     subroutines of his own.
C
C     The FN... subroutines perform various arithmetic operations on
C     segments of the F-array corresponding to variables of like
C     kind, indicated by the arguments. The FUNCT entry of the
C     SATELLITE help file and Appendix 4 of TR200 provide full
C     explanation.
C
C***      SUBROUTINE GREX3
C------------------------------------- include COMMON BLOCK files
      INCLUDE 'lp16/d_earth/SATEAR'

      INCLUDE 'lp16/d_earth/GRDLOC'
```

```
      INCLUDE 'lp16/d_earth/GRDEAR'

C---------------------------------------- additional declarations
      COMMON/RAKEEP/RADIAT
      LOGICAL RADIAT,SOR
      LOGICAL EQZ,NEZ,LTZ,GTZ,GEZ,LEZ,GRN
      LOGICAL QEQ,QNE,QGT,QLT,QGE,QLE,WALL,LWFIL
      COMMON/LWFUN/WALL,LWFIL(202)
      COMMON/NPAT/NAMPAT(100)
      CHARACTER*8 NAMPAT
      LOGICAL NOCALC,DRAGON,DRG1ST,PBFC,DFAIL
      COMMON/DRAGCM/NOCALC,DRAGON,DRG1ST,PBFC,DFAIL
C
C--------------------------------- GO TO the group indicated by IGR
C
      IF(IGR.EQ.19) GO TO 19
      IF(IGR.EQ.13) GO TO 13
      GO TO (1,2,3,4,5,6,25,8,9,10,11,12,13,14,25,25,25,25,19,
     1         20,25,25,23,24) IGR
   25 CONTINUE
C
      RETURN
C*********************************************************************
C
C---- GROUP 1. Run title and other preliminaries
C
    1 GO TO (1001,1002),ISC
 1001 CALL SUB2(IX,1,IY,1)
C---------------------------------- Identification of GREX3 to VDU
      CALL WRYT40(' SPECGR OF 17.01.91 HAS BEEN CALLED       ')

C
C----------- Provision of special EARTH arrays by means of the
C     storage-setting subroutine MAKE used thus:
C     CALL MAKE(variable name)
C     provides storage in EARTH for the named variable.
C
      if(bfc) then
        call sub4(dxg2d,dxgpe,dxu2d,dhxpe,dyg2d,dygpn,dyv2d,dhypn)
        call sub2(xg2d,xp,yg2d,yp)
      endif
C-----------------------------------------------------------XG2D
C.....The following provision relates to Group 9 sections
C     2 and 3, where the length scale is related to
C     x-direction distance.
```

```
      IF (GRNDNO(1,EL1).and..not.bfc) CALL MAKE(XG2D)
C.....The following provision relates to Group 13 section
C     13, where the Boussinesq approximation for buoyancy
C     is illustrated. It is also used in GXPOLR for
C     setting parallel flow at the circumferential boundary
C     of a polar domain.
c
      IF((.NOT.CARTES.AND.((NX.GT.1).OR.NEZ(RSG22))).and..not.bfc)
     1                                                  CALL MAKE(XG2D)
C
C-------------------------------------------------------XU2D
C.....The following provision relates to Group 13 section
C     13, where the Boussinesq approximation for buoyancy
C     is illustrated. It is also used in GXPOLR for
C     setting parallel flow at the circumferential
C     boundary of a polar domain.
      IF(.NOT.CARTES.AND.((ISG1.EQ.14.AND.NY.GT.1).OR.
     1                                  NEZ(RSG22))) CALL MAKE(XU2D)
C
C-------------------------------------------------------YG2D
C.....The following provision relates to Group 9 sections
C     2 and 3, where the length scale is related to
C     y-direction distance.
      IF((GRNDNO(2,EL1).OR.GRNDNO(4,EL1).OR.
     1    GRNDNO(6,EL1).OR.GRNDNO(7,EL1).OR.
     1    GRNDNO(8,EL1).OR.GRNDNO(9,EL1)).and..not.bfc) CALL MAKE(YG2D)
C.....The following provision relates to Group 11 section
C     3, Group 13 section 15, (ie value=GRND3) both of
C     which relate to power-law inlet conditions.
      IF((NEZ(RSG15).AND.NEZ(RSG16).AND.NEZ(RSG19)).and..not.bfc)
     1                                                  CALL MAKE(YG2D)
C
C-------------------------------------------------------DXG2D
      IF((LSG10.AND.STORE(U1)).and..not.bfc) CALL MAKE(DXG2D)
C
C-------------------------------------------------------DYG2D
C.....Provision for calculating velocities in potential
C     flows
      IF((LSG2.OR.(LSG10.AND.STORE(V1))).and..not.bfc) CALL MAKE(DYG2D)
C
C.....The following provision is useful when EPOR is
C     being used to represent the narrowing of the flow
C     space in a lubrication problem:-
      IF(LSG1.and..not.bfc) CALL MAKE(DYG2D)
C
```

```
C-----------------------------------------------------RV2D
C.....Provision of storage for cylindrical-polar rotation
C     terms
      IF(.NOT.CARTES.AND.NEZ(RSG21)) CALL MAKE(RV2D)
C
C-----------------------------------------------------LGEN1
C.....Provision of storage for square-of-velocity-gradient
C     expressions for viscous-dissipation or turbulence-
C     energy source.
C
        GENK = GENK.OR.(SOLVE(H1).AND.SOR(H1)).OR.SOLVE(KE).OR.
     1         (SOLVE(H2).AND.SOR(H2))
        IF (GRNDNO(4,ENUT).OR.GENK.OR.DUDX.OR.DVDX.OR.DWDX.OR.
     1      DUDY.OR.DVDY.OR.DWDY.OR.DUDZ.OR.DVDZ.OR.DWDZ) THEN
          CALL MAKE(EASP5)
          CALL MAKE(EASP6)
          CALL MAKE(EASP11)
          CALL MAKE(EASP12)
          CALL MAKE(EASP13)
          CALL MAKE(EASP14)
          CALL MAKE(EASP15)
          CALL MAKE(EASP16)
          CALL MAKE(EASP17)
          CALL MAKE(EASP18)
          CALL MAKE(EASP19)
          CALL MAKE(LGEN1)
          IF(.NOT.ONEPHS) CALL MAKE(LGEN2)
          NCRT=1
        END IF
        IF((SOLVE(H1).AND.(GRNDNO(5,TMP1).OR.GRNDNO(6,TMP1)))
     1  .OR.(SOLVE(H2).AND.(GRNDNO(5,TMP2).OR.GRNDNO(6,TMP2))))
     1    NCRT=1
C
C-----------------------------------------------------EASP1
C.....Provide storage for use in GXPOTC
        IF (LSG9.AND.LSG10) CALL MAKE(EASP1)
C
C-----------------------------------------------------EASP2
C-----------------------------------------------------EASP3
C-----EASP2 and EASP are always available, for use as local
C     temporary stores (See for example theiruse in GXBUOY)
        CALL MAKE(EASP2)
        CALL MAKE(EASP3)
C
C-----------------------------------------------------EASP4
```

```
        IF(NPOR.NE.0.AND.ISG12.NE.0) CALL MAKE(EASP4)
C.....EASP4 is used for buoyancy sources when BFC=T
C     (see GXBUOY)
        IF (BFC) CALL MAKE(EASP4)
C
C-------------------------------------------------------EASP5
        IF (HPOR.NE.0.AND.ISG13.NE.0) CALL MAKE(EASP5)
C
C-------------------------------------------------------EASP6
        IF (VPOR.NE.0.AND.ISG14.NE.0) CALL MAKE(EASP6)
C.....Provision of storage for cylindrical-polar rotation
C     terms
        IF (.NOT.CARTES.AND.NEZ(RSG21)) CALL MAKE(EASP6)
C
C-------------------------------------------------------EASP7
C.....Store for k-eps source-term linearisation
        IF (BFC.OR.KELIN.EQ.1) CALL MAKE(EASP7)
C
C-------------------------------------------------------EASP8
C.....Store for normal distance for use in Wall Functions.
C     (Also used in subroutine GXSQUR)
        CALL MAKE(EASP8)
C
C-------------------------------------------------------EASP9
C.....Store for resultant velocity in Wall Functions
C     (Also used in subroutine GXSQUR)
        CALL MAKE(EASP9)
C
C------------------------------------------------------EASP10
C.....Store for reciprocal of Reynolds number in group 13.
        CALL MAKE(EASP10)
C
C---------------------------------  Turbulence-model constants
        CALL SUB4R(CMU,0.5478,CD,0.1643,CMUCD,0.09,C1E,1.44)
        CALL SUB3R(C2E,1.92,AK,0.435,EWAL,9.0)
        CALL GXSQUR(0,PARAB,NX,NY,NZ,0)
C
C---------------------------------  Radiation initializations
        RADIAT = GTZ ((RSG23+RSG26))
        IF (RADIAT) CALL GXRADI(CARTES,NX,NY,NZ,TEMP1,DEN1,RHO1)
C------------------------------ Set up PATCH-wise storage for GXBFC
        IF (BFC) CALL GXBFC(NX,NY,NZ)
C
C***** MOD.- KT 15.01,90 **************
C----------------------- PHI and XYZ files for parabolic cases
```

```
C
        IF(PARAB) THEN
          CALL MAKE (YV2D)
          CALL MAKE (XU2D)
          IF (.NOT.CARTES) CALL MAKE (RV2D)
          IF (((AZXU.NE.0).OR.(AZYV.NE.0)).AND.(.NOT.CARTES)
     1              .AND.(NX.GT.1)) THEN
C----- EASP2 is used in this case, but MAKE(EASP2) has been
C----- called above.
            CALL MAKE (XG2D)
          END IF
          IF(LSG3) CALL GXPARA
        ENDIF
C--------------------------------- Wall-function initializations
        WALL = .FALSE.
        DO 10011 I = 1,NUMREG
          CALL GETPAT(I,IDUM,TYPE,IX1,IX2,IY1,IY2,IZ1,IZ2,IT1,IT2)
          IF (QGE(TYPE,17.0).AND.QLE(TYPE,22.0)) THEN
            CALL GXWFUN
            WALL=.TRUE.
            RETURN
          ENDIF
10011   CONTINUE
        RETURN
C
C---- Section 2 of group 1 is used for special overlay practices.
 1002   CONTINUE
        RETURN
C*********************************************************************
C
C--- GROUP 2. Transience; time-step specification
C
    2   CONTINUE
C   * Set DT if TLAST has been made .LE.GRND in satellite
        RETURN
C*********************************************************************
C
C--- GROUP 3. X-direction grid specification
C
    3   CONTINUE
C   * Set XRAT if AZXU has been made .LE.GRND in satellite
        RETURN
C*********************************************************************
C
C--- GROUP 4. Y-direction grid specification
```

```
C
    4   CONTINUE
C   * Set YRAT if AZYV has been made .LE.GRND in satellite
        RETURN
C*********************************************************************
C
C--- GROUP 5. Z-direction grid specification
C
    5   CONTINUE
C   * Set DZ if AZDZ has been made .LE.GRND in satellite
        IF (GRNDNO(1,AZDZ)) DZ = RSG10*XULAST
        IF (GRNDNO(2,AZDZ)) DZ = RSG10*YVLAST
C
        RETURN
C*********************************************************************
C
C--- GROUP 6. Body-fitted coordinates or grid distortion
C   This group is visited when UGEOM is set .TRUE.. The visitations
C   occur at the start of each z slab after its
C   geometry has been computed. Hence, this is the right place
C   to access AEAST, AHIGH,.. etc.
    6   CONTINUE
C
        RETURN
C*********************************************************************
C   * Make changes for this group only in group 19.
C--- GROUP 7. Variables stored, solved & named
C*********************************************************************
C
C--- GROUP 8. Terms (in differential equations) & devices
C
    8   GO TO (81,82,83,84,85,86,87,88,89,810,811,812,813,814,815)
     1         ISC
C   * Add velocities relative to grid for phase 1 and/or phase 2
C   * -----------GROUP 8  SECTION  1 --------------------------
C--- for U1AD.LE.GRND--- phase 1 additional velocity (VELAD).
   81   CONTINUE
C
        RETURN
C   * -----------GROUP 8  SECTION  2 --------------------------
C--- for U2AD.LE.GRND--- phase 2 additional velocity (VELAD).
   82   CONTINUE
C
        RETURN
C   * -----------GROUP 8  SECTION  3 --------------------------
```

```
C--- for V1AD.LE.GRND--- phase 1 additional velocity (VELAD).
   83   CONTINUE
C
        RETURN
C   * -----------GROUP 8  SECTION  4 --------------------------
C--- for V2AD.LE.GRND--- phase 2 additional velocity (VELAD).
   84   CONTINUE
C
        RETURN
C   * -----------GROUP 8  SECTION  5 --------------------------
C--- for W1AD.LE.GRND--- phase 1 additional velocity (VELAD).
   85   CONTINUE
C
        RETURN
C   *  ----------GROUP 8  SECTION  6 --------------------------
C--- for W2AD.LE.GRND--- phase 2 additional velocity (VELAD).
   86   CONTINUE
C
        RETURN
C   * -----------GROUP 8  SECTION 7 ---- VOLUMETRIC SOURCE FOR GALA
C---- Entered when GALA =.TRUE.; block-location index is LSU or LSU2
   87   CONTINUE
C
        RETURN
C   * -----------GROUP 8  SECTION 8 --- CONVECTION COEFFICIENTS
C--- Entered when UCONV =.TRUE.
C     Correction applicable to velocities in compressible flow
   88   IF (LSG8.AND.INDVAR.GE.U1.AND.INDVAR.LE.W2) THEN
          IF ((INDVAR.EQ.U1.OR.INDVAR.EQ.U2).AND.NDIREC.EQ.3) THEN
              NXDASH = NX
            IF (.NOT.XCYCLE) NXDASH = NX - 1
            CALL GXCMPR(LD11,EAST(P1),P1,1.4,NXDASH,NY)
            CALL GXCMPR(LD12,P1,EAST(P1),1.4,NXDASH,NY)
          ELSE IF ((INDVAR.EQ.V1.OR.INDVAR.EQ.V2).AND.NDIREC.EQ.1)
     1                                                        THEN
            CALL GXCMPR(LD11,NORTH(P1),P1,1.4,NX,NY)
            CALL GXCMPR(LD12,P1,NORTH(P1),1.4,NX,NY)
          ELSE IF (INDVAR.EQ.W1.OR.INDVAR.EQ.W2) THEN
            IF (NDIREC.EQ.5) CALL GXCMPR(LD2,P1,HIGH(P1),1.4,NX,NY)
            IF (NDIREC.EQ.6) CALL GXCMPR(LD2,HIGH(P1),P1,1.4,NX,NY)
          END IF
        END IF
C
        RETURN
C   * -----------GROUP 8  SECTION 9 --- DIFFUSION COEFFICIENTS
```

```
C--- Entered when UDIFF =.TRUE.; block-location indices are LAE
C    for east, LAW for west, LAN for north, LAS for
C    south, LD11 for high, and LD11 for low.
C    User should provide INDVAR and NDIREC IF's as above.
C
C    Cut out lateral diffusive links in radiation equations
   89   CONTINUE
        IF (RADIAT) CALL GXRADI(CARTES,NX,NY,NZ,TEMP1,DEN1,RHO1)
C
        RETURN
C   * -----------GROUP 8  SECTION 10 --- CONVECTION NEIGHBOURS
C--- Entered when UCONNE =.TRUE.; block-location indices are LD7
  810   CONTINUE
C
        RETURN
CC   * -----------GROUP 8  SECTION 11 --- DIFFUSION NEIGHBOURS
C--- Entered when UDIFNE =.TRUE.; block-location indices are LD7
C    for south, west, high or low, and LD8 for north or east.
C    User should provide INDVAR and NDIREC IF's as above.
  811   CONTINUE
C
        RETURN
C   * -----------GROUP 8  SECTION 12 --- LINEARISED SOURCES
C--- Entered when USOURC =.TRUE.
  812   CONTINUE
C
        RETURN
C   * -----------GROUP 8  SECTION 13 --- CORRECTION COEFFICIENTS
C--- Entered when UCORCO =.TRUE.
  813   CONTINUE
        RETURN
C   * -----------GROUP 8  SECTION 14 --- USER'S SOLVER
C--- Entered when USOLVE =.TRUE.;
C
C-------------------------              Call Gauss-Seidel solver
  814   CONTINUE

IF (CSG3.EQ.'GAUS'.AND..NOT.SLBSOL)  THEN

         CALL GXGAUS(OVRRLX,LITER(INDVAR),NX,NY,NZ,IXMON,
     1    IYMON,IZMON,ENDIT(INDVAR),XCYCLE,LUPR1,LUPR3)

C*** -------------------- Call user-defined Conjugate Gradient solver

 ELSE IF (CSG3.EQ.'CNGR'.AND..NOT.SLBSOL) THEN
```

```
         CALL UUCNGR(NX,NY,NZ)

        ENDIF

          USER = .NOT.SLBSOL
C
        RETURN

C   * -----------GROUP 8  SECTION 15 --- CHANGE SOLUTION RESULT
C--- Entered when UCORR =.TRUE.; block-location indices are as above.
C       IF (SLBSOL.AND.INDVAR.EQ.U1) CALL PRN('RSLT',LD7)
  815   CONTINUE
C
        RETURN
C   * Make all other group-8 changes in group 19.
C*****************************************************************
C
C--- GROUP 9. Properties of the medium (or media)
C
C   The sections in this group are arranged sequentially in their
C   order of calling from EARTH. Thus, as can be seen from below,
C   the temperature sections (10 and 11) precede the density
C   sections (1 and 3); so, density formulae can refer to
C   temperature stores already set.
    9 GO TO (91,92,93,94,95,96,97,98,99,900,901,902,903,904,905),ISC
C*****************************************************************
C
C GROUP 9 SECTION 10 for TMP1.LE.GRND phase-1 temperature TEMP1
  900 CALL GXTEMP(TEMP1,TMP1,TMP1A,TMP1B,TMP1C,H1,C1,C3,1)
C
      RETURN
C
C GROUP 9 SECTION 11 for TMP2.LE.GRND phase-2 temperature TEMP2
  901 CALL GXTEMP(TEMP2,TMP2,TMP2A,TMP2B,TMP2C,H2,C2,C4,2)
C
       RETURN
C
C GROUP 9 SECTION 12 for EL1.LE.GRND phase-1 length scale  LEN1
  902 CALL GXLEN(LEN1,EL1,EL1A,EL1B,EL1C,ZW,CARTES,NY,YVLAST)
C
      RETURN
C
C GROUP 9 SECTION 13 for EL2.LE.GRND  phase-2 length scale LEN2
  903 CALL GXLEN(LEN2,EL2,EL2A,EL2B,EL2C,ZW,CARTES,NY,YVLAST)
```

```
C
      RETURN
c
C GROUP 9 SECTION 14   for SOLVE(TEMP1)------ phase-1 specic heat
  904 CONTINUE
       CALL GXPRPS(3,-11)
C
      RETURN
C GROUP 9 SECTION 15   for SOLVE(TEMP2)------ phase-2 specic heat
  905 CONTINUE
      CALL GXPRPS(3,-12)
C
      RETURN
C
C GROUP 9 SECTION  1 for RHO1.LE.GRND density for phase 1  DEN1
   91   CALL GXRHO(DEN1,RHO1,RHO1A,RHO1B,RHO1C,H1,RHO2,TEMP1,ONEPHS)
C---- Default settings for compressible flow.
      IF (GRNDNO(3,RHO1)) DRH1DP=GRND3
      IF (GRNDNO(5,RHO1)) DRH1DP=GRND5
      IF (GRNDNO(5,DRH1DP).AND.EQZ(RHO1C)) RHO1C=1.0/1.4
C
      RETURN
CC GROUP 9 SECTION  2 for DRH1DP.LE.GRND: D(LN(DEN))/DP for phase 1
C                                                              (D1DP)
   92 CALL GXDRDP(D1DP,DRH1DP,RHO1B,RHO1C,RHO1,RHO2,ONEPHS)
C
      RETURN
C
C GROUP 9 SECTION 3 for RHO2.LE.GRND: density for phase 2  DEN2
   93 CALL GXRHO(DEN2,RHO2,RHO2A,RHO2B,RHO2C,H2,RHO2,TEMP2,ONEPHS)
C
      RETURN
C
C GROUP 9 SECTION  4 for DRH2DP.LE.GRND: D(LN(DEN))/DP for phase 2
C                                                              (D2DP)
   94 CALL GXDRDP(D2DP,DRH2DP,RHO2B,RHO2C,RHO1,RHO2,ONEPHS)
C
      RETURN
C
C GROUP 9 SECTION  5 for ENUT.LE.GRND: reference turbulent kinematic
C                                                  viscosity (VIST)
   95 CALL GXENUT(VIST,ENUT,LEN1)
C
      RETURN
C
```

```
C GROUP 9 SECTION  6 for ENUL.LE.GRND: reference laminar kinematic
C                                                    viscosity (VISL)
   96 CALL GXENUL(VISL,ENUL,TEMP1)
C
      RETURN
C
C GROUP 9 SECTION  7 for PRNDTL( ).LE.GRND: laminar PRANDTL nos.
CC                                                     or diffusivity
   97 IF (.NOT.RADIAT) THEN
        CALL GXPRL(LAMPR,PRNDTL(INDVAR),LBNAME('PRL '))
      ELSE
        CALL GXRADI(CARTES,NX,NY,NZ,TEMP1,DEN1,RHO1)
       END IF
C
      RETURN
C
C GROUP 9 SECTION  8 for PHINT(H1).LE.GRND: interface value, phase 1
C                                                                (FII1)
   98 IF (GRNDNO(1,PHINT(H1))) CALL FN8(FII1,P1,PHNH1A*HUNIT,PRESSO,
     1                                   PHNH1B,PHNH1C*HUNIT)
C
      RETURN
C
C GROUP 9 SECTION  9 for PHINT(H2).LE.GRND: interface value, phase 1
C                                                                (FII2)
   99 IF (GRNDNO(1,PHINT(H2))) CALL FN8(FII2,P1,PHNH2A*HUNIT,PRESSO,
     1                                   PHNH2B,PHNH2C*HUNIT)
      IF (GRNDNO(2,PHINT(H2))) CALL FN2(FII2,FII1,PHNH2A,PHNH2B)
C
      RETURN
C*****************************************************************
C
C--- GROUP 10. Inter-phase-transfer processes and properties
C
   10 GO TO (101,102,103,104) ISC
C*****************************************************************
C
C GROUP 10 SECTION  1 for CFIPS.LE.GRND: inter-phase friction coeff.
C                                                          INTFRC
  101   IF (QLT(CFIPS,GRND)) CALL FN99(INTFRC,LEN1,CFIPS,
     1                                CFIPA,CFIPB,CFIPC,CFIPD)
C
C---- account for droplet-size variation
        IF (LSG4.AND.SOLVE(RS).AND.SOLVE(R2)) CALL GXDROP(INTFRC)
C
```

```
        RETURN
C
C GROUP 10 SECTION  2 for CMDOT.LE.GRND: inter-phase mass transfer
C                                                          INTMDT.
  102   IF (QLT(CMDOT,GRND)) CALL FN98(INTMDT,CMDOT,CMDTA,
     1                                 CMDTB,CMDTC)
C
        RETURN
C
C GROUP 10  SECTION  3 for CINT( ).LE.GRND: phase1-to-interface
C                                        transfer coefficients (COI1)
  103   IF (GRNDNO(1,CINT(H1))) CALL FN8(COI1,H1,CINH1A,0.0,CINH1B,
     1                                  CINH1C)
C
        RETURN
C
C GROUP 10  SECTION  4 for CINT( ).LE.GRND: phase2-to-interface
C                                        transfer coefficients (COI2)
  104   IF (GRNDNO(1,CINT(H2))) CALL FN3(COI2,H2,CINH2A,CINH2B,
     1                                  CINH2C)
C
        RETURN
C*********************************************************************
C
C--- GROUP 11. Initialization of variable or porosity fields
C
C---- Initial field of uniform flow in a curvilinear grid GXBFC
   11   IF (NPATCH(1:4).EQ.'IBFC') CALL GXBFC(NX,NY,NZ)
C
        RETURN
C*********************************************************************
C
C--- GROUP 12. Convection and diffusion adjustments
C
   12   CONTINUE
C
        RETURN
C*********************************************************************
C
C--- GROUP 13. Boundary conditions and special sources
C
C   * XCYCLE may be changed in group 19.
C
C     Sections 1  to 11 for CO
C     Sections 12 to 22 for VAL
```

```
   13   NPAT=NPATCH(1:4)
C
C---- Velocity resolutes at a curved inlet boundary        GXBFC
        IF (NPAT(1:3).EQ.'BFC') CALL GXBFC(NX,NY,NZ)
C
C-----Sources of momentum caused by buoyancy              GXBUOY
        IF (NPAT.EQ.'BUOY') CALL GXBUOY(BFC,CARTES,PARAB,NX,NY)
C
C---- Chemical-reaction source                            GXCHSO
        IF (NPAT.EQ.'CHSO') CALL GXCHSO(TEMP1,NX,NY)
C
C-----Exit boundary condition for Supersonic flow Z-direction.
        IF (INDVAR.EQ.P1.AND.NPAT.EQ.'OUTL'.AND.ISC.EQ.13)
     1    CALL FN21(VAL,LOW(W1),LOW(DEN1),0.0,-1.0)
C
C-----Sources providing for upwinding of the interphase
C     transport terms                                     GXIPST
        IF (NPAT(1:4).EQ.'IPST')
     1    CALL GXIPST(NX,NY,INTFRC,CINT(INDVAR),NPATCH)
C
C---- Sources of turbulence energy and dissipation rate   GXKESO
        IF (NPAT.EQ.'KESO') CALL GXKESO(VIST,LEN1)
C
C     Momentum source caused by lateral gravity
C     in 2-phase flow                                     GXLATG
        IF (NPAT.EQ.'LATG') CALL GXLATG
C
C---- Tentative length-scale source for two-fluid
C     turbulence model                                    GXLESO
        IF (NPAT.EQ.'LESO') CALL GXLESO('SOUTH')
C
C---- The following call to                               GXNEPA
C     allows the "values" used in
C     a COVAL command to be chosen as the "neighbour-
C     values" of the cell in specified space, time or
C     "variable-space" directions. The condition for
C     the call is that the PATCH name should begin with
C     the characters NE.
        IF (NPAT(1:2).EQ.'NE') CALL GXNEPA(NPAT)
C
C---- The following call to                               GXPOLR
C     fixes the u- and v-velocities at the circumferential
C     boundary of a cylindrical-polar domain for uniform
C     flow at speed RSG22
        IF (NPAT(2:4).EQ.'POL') CALL GXPOLR(NPAT(1:2))
```

```
C
C----Sources used to set profiles                                GXPROF
       IF (NPAT.EQ.'PROF') CALL GXPROF(DEN1,RHO1,FIINIT(14))
C
C---- Calculate radiation sources .                              GXRADI
       IF (NPAT.EQ.'RADI')
     1    CALL GXRADI(CARTES,NX,NY,NZ,TEMP1,DEN1,RHO1)
C
C----- Centrifugal & Coriolis forces due to rotation of
C      coordinate system about axis of cylindrical-polar
C      system.                                                   GXROTA
C      RSG21= rotation rate of coordinate system.
       IF (NPAT.EQ.'ROTA'.AND.(.NOT.CARTES.OR.BFC))
     1    CALL GXROTA(BFC,NX)
C
C----Shear source of lateral velocity for the 2-fluid
C    turbulence model                                            GXSHSO
       IF (NPAT.EQ.'SHSO') CALL GXSHSO(INTFRC,LEN1)
C
C---- Wall functions                                             GXWFUN
       IF (WALLTY) THEN
         CALL GXWFUN
       END IF
C
       RETURN
C*****************************************************************
C
C
C--- GROUP 14. Downstream pressure for PARAB=.TRUE.
C
   14  CONTINUE
C
       RETURN
C*****************************************************************
C   * Make changes for these groups only in group 19.
C--- GROUP 15. Termination of sweeps
C--- GROUP 16. Termination of iterations
C--- GROUP 17. Under-relaxation devices
C--- GROUP 18. Limits on variables or increments to them
C*****************************************************************
C
C--- GROUP 19. Special calls to GROUND from EARTH
C
   19  GO TO (191,192,193,194,195,196,197,198),ISC
C
```

```
C   * ----------GROUP 19  SECTION 1 ---- START OF TIME STEP.
C---- Multiply DYG2D by EPOR, for thin-film lubrication
  191   CONTINUE
        IF (LSG1) CALL FN26(DYG2D,EPOR)
C---- Call GXPIST to expand and contract the grid in accordance
C     with the kinematics of crankshaft-connecting-rod mechanisms.
        IF (NEZ(W1AD).AND.NEZ(AZW1))
     1    CALL GXPIST(ISTEP,NZ,TIM,DT,ISWEEP,LSWEEP,NPRINT,NTPRIN)
        IF (LSG5) CALL GXRSET
        IF (WALL) CALL GXWFUN
C
        RETURN
C   * ----------GROUP 19  SECTION 2 ---- START OF SWEEP.
  192   CONTINUE
C
        RETURN
C   * ----------GROUP 19  SECTION 3 ---- START OF IZ SLAB.
C---- Modify porosities as functions of pressure
  193   CONTINUE
        IF (ISG11.GE.IZ) CALL GXPORA(EPOR,EASP3,RSG11,RSG12,SETPOR,1)
        IF (ISG12.GE.IZ) CALL GXPORA(NPOR,EASP4,RSG13,RSG14,SETPOR,1)
        IF (ISG13.GE.IZ) CALL GXPORA(HPOR,EASP5,RSG15,RSG16,SETPOR,1)
        IF (ISG14.GE.IZ) CALL GXPORA(VPOR,EASP6,RSG17,RSG18,SETPOR,1)
C
        RETURN
C
C GROUP 19  SECTION 4  START OF ITERATION.
C---- Calculation of square-of-velocity-gradient expressions for
C     viscous-dissipation or turbulence-energy source, subject to
C     the allocation of storage for these quantities. The
C     result is put in EARTH store LGEN1, for use in GROUP 13.
C     BFC's is called from GXGENK.
C
  194  IF (STORE(LGEN1)) CALL GXGENK(0,PARAB,NX,NY,NZ,BFC)
       IF (STORE(LGEN2)) CALL GXGENK(1,PARAB,NX,NY,NZ,BFC)
C
C---- Overwrite wall-cell generation rate with integrated value
C     deduced from the wall functions...
        IF (WALL) CALL GXWFUN
C
        RETURN
C
C   * ----------GROUP 19  SECTION 5 ---- FINISH OF ITERATION.
C---- Compute velocities from potential differences.
  195   IF (LSG10.AND.ISWEEP.NE.1) CALL GXPOTV(NZ,DZG,NPOR,EPOR,
```

```
     1                                                          HPOR,bfc)
C
C---- Allow for compressibility in a potential flow
        IF (LSG10.AND.LSG9.AND.ISWEEP.NE.1)
     1    CALL GXPOTC(EPOR,NPOR,HPOR,RSG3,RSG4,NZ)
C
        RETURN
C
C GROUP 19  SECTION 6  FINISH OF IZ SLAB
C---- Reset the nominal porosities
  196   IF (ISG11.GE.IZ) CALL GXPORA(EPOR,EASP3,RSG11,RSG12,SETPOR,2)
        IF (ISG12.GE.IZ) CALL GXPORA(NPOR,EASP4,RSG13,RSG14,SETPOR,2)
        IF (ISG13.GE.IZ) CALL GXPORA(HPOR,EASP5,RSG15,RSG16,SETPOR,2)
        IF (ISG14.GE.IZ) CALL GXPORA(VPOR,EASP6,RSG17,RSG18,SETPOR,2)
C
C***** MOD.- KT 15.01.90 **************
        IF(PARAB.AND.LSG3) CALL GXPARA
        RETURN
C
C GROUP 19  SECTION 7  FINISH OF SWEEP.
  197   CONTINUE
C
        RETURN
C
C GROUP 19  SECTION 8  FINISH OF TIME STEP.
  198   CONTINUE
C.... Save fields and cell-corner coordinates to a series of files,
C     for examination via PHOTON
C.... CSG1 and CSG2 are the names of the field and (if BFC)
C     grid files, and ISG2 is the frequency of dumping.
C     They are set in the satellite.
C     They are set in the satellite.
        CALL DUMPS(CSG1(1:1),CSG2(1:1),ISTEP,ISG2)
        RETURN
C*************************************************************
C
C--- GROUP 20. Preliminary print-out
C
   20   CONTINUE
        RETURN
C*************************************************************
C   * Make changes for these groups only in group 19.
C--- GROUP 21. Print-out of variables
C--- GROUP 22. Spot-value print-out
C--- GROUP 23. Field print-out and plot control
```

```
   23   CONTINUE
        RETURN
C*****************************************************************
C
C--- GROUP 24. Dumps for restarts
C
   24   CONTINUE
C   * Insert CALL DUMP where required in group 19. See the SAVE
C    entry in TR 200 for further information.

        END

C*** ----------------------------------------------------------------
C***
C***      ... END OF GREX3 SUBROUTINE
C***
C*** ----------------------------------------------------------------
C***
C***     START OF USER-DEFINED SOLVER ...
C***
C*** ----------------------------------------------------------------

        SUBROUTINE UUCNGR(NX,NY,NZ)

C*** This routine uses the Conjugate Gradient algorithm to solve
C*** the system Ax=b, as an alternative to the built-in linear-
C*** equation solver of PHOENICS. The central storage (F-) array of
C*** PHOENICS, where the x and b vectors reside, is accessed directly
C*** using the LOF function. This is an integer function which points
C*** to the starting location in the F-array of the segment containing
C*** these particular variables (PHI is the solution variable and SU
C*** is the source). R and P are the arrays containing the residual and
C*** search vectors respectively, and AX stores the result of the
C*** matrix-vector multiplication.

        INCLUDE 'lp16/d_earth/GRDLOC'

COMMON /NAMFN/NAMFUN,NAMSUB
CHARACTER*6 NAMFUN,NAMSUB
        INTEGER NX,NY,NZ,NXNY,LEN,J,ITER,ITMAX,LOF,NSEG
INTEGER IPHI,ISU,LR,LPP,LAX,C1,C3,C5,C7,ICEL,NPHI
        REAL EPS,ANUM,ADEN,BNUM,BDEN,DOTPR,ALPHA,BETA

        EXTERNAL DIAGON, MATVEC, DOTPR, LOF
PARAMETER (EPS=1.0E-7)
```

```
        CALL WRIT40(' ** subroutine UUCNGR  called ** ')

        NAMSUB= 'UUCNGR'
NXNY=NX*NY
        LEN=NX*NY*NZ
        ITMAX = LEN

        NSEG=0
        DO 111 NPHI=1,50
           IF (STORE(NPHI)) NSEG=NSEG+1
 111    CONTINUE

LR=   LOF(C1)
        LPP=  LOF(C3)
LAX=  LOF(C5)
        CALL ZERNUM(LR,LEN)
CALL ZERNUM(LPP,LEN)
        CALL ZERNUM(LAX,LEN)

        IPHI= LOF(L3PHI)
    ISU=  LOF(L3SU)
        CALL ZERNUM(IPHI,LEN)

        WRITE(8,*) ' LR=',LR,' LPP=',LPP,' LAX=',LAX,' IPHI=',IPHI,
     +             ' ISU=',ISU

CALL DIAGON(C7,NX,NY,NZ,NSEG)

C*********SET RESIDUAL AND SEARCH VECTORS*********************

 DO 10 IZ=1,NZ
              DO 12 J=1,NXNY
                   ICEL=NXNY*(IZ-1)*NSEG+J
                   JCEL=NXNY*(IZ-1)+J
                   F(LR+ICEL)=F(ISU+JCEL)
                   F(LPP+ICEL)=F(LR+ICEL)
12            CONTINUE
10       CONTINUE

        BNUM = DOTPR(C1,C1,NZ,NX,NY,NSEG)
write(8,*) ' BNUM= ',BNUM

C******************** <<<  MAIN LOOP >>>  *******************
```

```
        DO 100 ITER=1,LEN
            write(8,*)' Iter=',iter
            CALL MATVEC(C7,C3,C5,NX,NY,NZ,NSEG)

C**********CALCULATE ALPHA************************************

                ANUM=BNUM
                ADEN=DOTPR(C3,C5,NZ,NX,NY,NSEG)
                ALPHA= ANUM / ADEN
c WRITE(8,*)' ALPHA=',ALPHA, '    ADEN=',ADEN

C********UPDATE ITERATE ************************************

           DO 20 IZ=1,NZ
       DO 18 J=1,NXNY
                ICEL=(IZ-1)*NXNY*NSEG+J
                JCEL=(IZ-1)*NXNY+J
                F(IPHI+JCEL)=F(IPHI+JCEL)+ ALPHA*F(LPP+ICEL)
18         CONTINUE
20          CONTINUE

C********UPDATE RESIDUAL*************************************

           DO 40 IZ=1,NZ
       DO 38 J=1,NXNY
              ICEL=(IZ-1)*NXNY*NSEG+J
                  F(LR+ICEL)=F(LR+ICEL) - ALPHA*F(LAX+ICEL)
38         CONTINUE
40          CONTINUE

C********CALCULATE BETA*************************************

                BNUM=DOTPR(C1,C1,NZ,NX,NY,NSEG)
write(8,*) ' <<<**** BNUM ****>>>=',BNUM

C*****test for convergence**************************************

                IF (BNUM.LE.EPS) THEN
                     WRITE(8,*)' F(IPHI)='

                     DO 26 IZ=1,NZ
                         WRITE(8,*)' IZ=',IZ
                         J=IPHI+(IZ-1)*NXNY
                         DO 24 IX=1,NX
                             WRITE(8,29) (F(J+I),I=1,NY)
```

```
                              J=J+NY
24                          CONTINUE
26                        CONTINUE
29                        FORMAT(1X,5F)
                          RETURN
                     ELSE
                          CONTINUE
                     END IF

                     BDEN=ANUM
                     BETA=BNUM / BDEN
c write(8,*) ' BETA=',BETA

C*******NEW SEARCH VECTOR***************************************

c           WRITE(*,*)' Residual F(LR)','    Search vector F(LPP)'

   DO 50 IZ=1,NZ
      DO 48 J=1,NXNY
               ICEL=(IZ-1)*NXNY*NSEG+J
                    F(LPP+ICEL)=F(LR+ICEL) + BETA*F(LPP+ICEL)
c                    WRITE(*,46) F(LR+ICEL),F(LPP+ICEL)
48       CONTINUE
50         CONTINUE

c46 FORMAT(1X,2F)

100     CONTINUE

C********************END OF MAIN LOOP*****************************

        CALL WAYOUT(2)
        RETURN
        END

C********************dot-product function************************

        REAL FUNCTION DOTPR(KKK,LLL,NZ,NX,NY,NSEG)
        INCLUDE 'lp16/d_earth/GRDLOC'
        INCLUDE 'lp16/d_earth/GRDEAR'
        INTEGER NX,NY,NZ,J,IROW,JROW,KKK,LLL,LOF,NSEG
        EXTERNAL LOF

        DOTPR=0.0
IROW=LOF(KKK)
```

```
JROW=LOF(LLL)

        DO 24 IZ=1,NZ
           DO 22 J=1,NX*NY
              ICEL=(IZ-1)*NX*NY*NSEG+J
              DOTPR=DOTPR+F(IROW+ICEL)*F(JROW+ICEL)
c              write(*,*)' DOTPR= ',DOTPR
22         CONTINUE
24      CONTINUE
        RETURN
        END
C******************calculate main diagonal**********************

        SUBROUTINE DIAGON(DIAG,NX,NY,NZ,NSEG)
        INCLUDE 'lp16/d_earth/GRDLOC'
        INTEGER NX,NY,NZ,NXNY,IX,IY,IZZ,LOF,NSEG
        INTEGER IAP,IAE,IAN,IAH,IROW,ICEL
        INTEGER DIAG
        REAL SOLN
        EXTERNAL LOF

IAP=LOF(L3AP)
IAE=LOF(L3AE)
IAN=LOF(L3AN)
IAH=LOF(L3AH)

        IROW= 0
        NXNY=NX*NY
        IROW=LOF(DIAG)

        WRITE(8,*)' IAP=',IAP,' IAE=',IAE,' IAN=',IAN,' IAH=',IAH,
     +            ' DIAG=',IROW

        DO 30 IZZ = 1,NZ
          DO 40 IX = 1,NX
            DO 50 IY = 1,NY
              ICEL=(IZZ-1)*NXNY*NSEG+IY+NY*(IX-1)
      IAP = IAP+1
      SOLN = 0.0
      SOLN = F(IAP)
              IF (NX.NE.1) THEN
                IAE = IAE + 1
                IF (IX.NE.1) THEN
                  SOLN = SOLN + F(IAE-NY)
                END IF
```

```
              IF (IX.NE.NX) THEN
                SOLN = SOLN + F(IAE)
              END IF
            END IF
            IF (NY.NE.1) THEN
              IAN = IAN + 1
              IF (IY.NE.1) THEN
                 SOLN = SOLN + F(IAN-1)
              END IF
              IF (IY.NE.NY) THEN
                SOLN = SOLN + F(IAN)
              END IF
            END IF
            IF (NZ.NE.1) THEN
              IAH = IAH + 1
              IF (IZZ.NE.1) THEN
                SOLN = SOLN + F(IAH-NXNY)
              END IF
              IF (IZZ.NE.NZ) THEN
                SOLN = SOLN + F(IAH)
              END IF
            END IF
            F(IROW+ICEL)=SOLN
c            write(8,*) ' DIAG= ',F(IROW+ICEL)
50         CONTINUE
40       CONTINUE
30     CONTINUE

       RETURN
       END

C******************matrix-vector multiplier*****************

       SUBROUTINE MATVEC(DIAG,V,AX,NX,NY,NZ,NSEG)
       INCLUDE 'lp16/d_earth/GRDLOC'
       INTEGER NX,NY,NZ,NXNY,IX,IY,IZZ,LOF,NSEG
       INTEGER IAE,IAN,IAH,IROW,JROW,KROW
       INTEGER DIAG,V,AX,J,IZ,ICEL,IROWO,JROWO,KROWO
REAL SOLN
       EXTERNAL LOF

IAP=LOF(L3AP)
IAE=LOF(L3AE)
IAN=LOF(L3AN)
IAH=LOF(L3AH)
```

```
    NXNY= NX*NY
IROWO=LOF(DIAG)
JROWO=LOF(V)
KROWO=LOF(AX)

c        WRITE(8,*)' DIAG=',IROWO,'    V =',JROWO,'   AP=',KROWO
c        WRITE(8,366)' (DIAG)',' (P)', '(AP)'
366     FORMAT(1X,3A14)

        ICEL=0
        DO 379 IZ=1,NZ
          DO 377 J=1,NXNY
             ICEL=(IZ-1)*NXNY*NSEG+J
c            WRITE(8,367) F(IROWO+ICEL),F(JROWO+ICEL),F(KROWO+ICEL)
367          FORMAT(1X,3F)
377       CONTINUE
379     CONTINUE

        DO 30 IZZ = 1,NZ
          DO 40 IX = 1,NX
              DO 50 IY = 1,NY
              ICEL=(IZZ-1)*NXNY*NSEG+IY+NY*(IX-1)
              IROW = IROWO + ICEL
      JROW = JROWO + ICEL
      KROW = KROWO + ICEL
      SOLN = 0.0
      SOLN = F(IROW)*F(JROW)
             IF (NX.NE.1) THEN
               IAE = IAE + 1
               IF (IX.NE.1) THEN
                   SOLN = SOLN - F(IAE-NY)*F(JROW-NY)
               END IF
               IF (IX.NE.NX) THEN
                   SOLN = SOLN - F(IAE)*F(JROW+NY)
              END IF
             END IF
             IF (NY.NE.1) THEN
               IAN = IAN + 1
               IF (IY.NE.1) THEN
                    SOLN = SOLN - F(IAN-1)*F(JROW-1)
               END IF
               IF (IY.NE.NY) THEN
                    SOLN = SOLN - F(IAN)*F(JROW+1)
               END IF
```

```
            END IF
            IF (NZ.NE.1) THEN
              IAH = IAH + 1
              IF (IZZ.NE.1) THEN
                   SOLN = SOLN - F(IAH-NXNY)*F(JROW-NXNY*NSEG)
              END IF
              IF (IZZ.NE.NZ) THEN
                   SOLN = SOLN - F(IAH)*F(JROW+NXNY*NSEG)
              END IF
            END IF

            F(KROW)=SOLN
C            write(8,*) ' A*P= ', F(KROW)
50          CONTINUE
40        CONTINUE
30      CONTINUE
      write(8,*) ' --------------'
      RETURN
      END
```

APPENDIX 4: The PHOENICS RESULT File

```
************************************************************


   ----------------------------------------------------
        CCCC HHH         PHOENICS - EARTH     Version 1.6.6
      CCCCCCCC HHHHH     (C) Copyright 1992
    CCCCCCC HHHHHHHHHH   Concentration Heat and Momentum Ltd
   CCCCCCC HHHHHHHHHHHH  All rights reserved.
   CCCCCC HHHHHHHHHHHHH  Address:  Bakery House, 40 High St
   CCCCCCC HHHHHHHHHHHH  Wimbledon, London, SW19 5AU
    CCCCCCC HHHHHHHHHH   Tel: 081-947-7651
      CCCCCCCC HHHHH     Facsimile: 081-879-3497
        CCCC HHH         The option level is   -18
   ----------------------------------------------------
  This program forms part of the PHOENICS installation for:
              UNIX Installation
   ----------------------------------------------------
  This code may be used only under the terms and conditions
  of a licence from Concentration, Heat and Momentum Ltd.
  The code expiry date is the end of : Dec 1994
   ----------------------------------------------------
  Replication of this code is prohibited unless
  specifically authorised in writing by
  Concentration, Heat, and Momentum Ltd.
   ----------------------------------------------------

 ************************************************************

Number of F-array location available is       52000
Number used before BFC allowance (if any) is  3669
 ************************************************************
  Group 1. Run Title and Number
 ************************************************************
 ************************************************************

 TEXT(STEADY HEAT CONDUCTION: USING CG SOLVER )

 ************************************************************
 ************************************************************

 IRUNN    =       1 ;LIBREF =     103

 *** grid-geometry information ***
```

```
X-coordinates of the cell centres
   1.000E-01   3.000E-01   5.000E-01   7.000E-01   9.000E-01

Y-coordinates of the cell centres
   1.000E-01   3.000E-01   5.000E-01   7.000E-01   9.000E-01

Z-coordinates of the cell centres
   1.000E-01   3.000E-01   5.000E-01   7.000E-01   9.000E-01


--- INTEGRATION OF EQUATIONS BEGINS ---

 ** subroutine UUCNGR  called **

************************************************************
TIME STP=     1 SWEEP NO=     2 ZSLAB NO=     1 ITERN NO=     1


FLOW FIELD AT ITHYD=   1, IZ=   1, ISWEEP=   2, ISTEP=    1
FIELD VALUES OF TEMP
IY=   5   4.717E-01   4.827E-01   5.000E-01   5.173E-01   5.284E-01
IY=   4   4.495E-01   4.649E-01   4.862E-01   5.056E-01   5.173E-01
IY=   3   3.965E-01   4.272E-01   4.607E-01   4.862E-01   5.000E-01
IY=   2   2.821E-01   3.651E-01   4.272E-01   4.649E-01   4.827E-01
IY=   1   1.682E-03   2.821E-01   3.965E-01   4.495E-01   4.717E-01
 IX=       1           2           3           4           5

************************************************************
TIME STP=     1 SWEEP NO=     2 ZSLAB NO=     2 ITERN NO=     1


FLOW FIELD AT ITHYD=   1, IZ=   2, ISWEEP=   2, ISTEP=    1
FIELD VALUES OF TEMP
IY=   5   4.827E-01   4.944E-01   5.138E-01   5.351E-01   5.505E-01
IY=   4   4.649E-01   4.788E-01   5.000E-01   5.212E-01   5.351E-01
IY=   3   4.272E-01   4.488E-01   4.764E-01   5.000E-01   5.138E-01
IY=   2   3.651E-01   4.070E-01   4.488E-01   4.788E-01   4.944E-01
IY=   1   2.821E-01   3.651E-01   4.272E-01   4.649E-01   4.827E-01
 IX=       1           2           3           4           5

************************************************************
TIME STP=     1 SWEEP NO=     2 ZSLAB NO=     3 ITERN NO=     1


FLOW FIELD AT ITHYD=   1, IZ=   3, ISWEEP=   2, ISTEP=    1
```

```
 FIELD VALUES OF TEMP
 IY=   5   5.000E-01   5.138E-01   5.394E-01   5.728E-01   6.035E-01
 IY=   4   4.862E-01   5.000E-01   5.236E-01   5.512E-01   5.728E-01
 IY=   3   4.607E-01   4.764E-01   5.000E-01   5.236E-01   5.394E-01
 IY=   2   4.272E-01   4.488E-01   4.764E-01   5.000E-01   5.138E-01
 IY=   1   3.965E-01   4.272E-01   4.607E-01   4.862E-01   5.000E-01
  IX=       1           2           3           4           5

 ************************************************************
 TIME STP=     1 SWEEP NO=     2 ZSLAB NO=     4 ITERN NO=     1


 FLOW FIELD AT ITHYD=   1, IZ=   4, ISWEEP=   2, ISTEP=    1
 FIELD VALUES OF TEMP
 IY=   5   5.173E-01   5.351E-01   5.728E-01   6.349E-01   7.179E-01
 IY=   4   5.056E-01   5.212E-01   5.512E-01   5.930E-01   6.349E-01
 IY=   3   4.862E-01   5.000E-01   5.236E-01   5.512E-01   5.728E-01
 IY=   2   4.649E-01   4.788E-01   5.000E-01   5.212E-01   5.351E-01
 IY=   1   4.495E-01   4.649E-01   4.862E-01   5.056E-01   5.173E-01
  IX=       1           2           3           4           5

 ************************************************************
 TIME STP=     1 SWEEP NO=     2 ZSLAB NO=     5 ITERN NO=     1


 FLOW FIELD AT ITHYD=   1, IZ=   5, ISWEEP=   2, ISTEP=    1
 FIELD VALUES OF TEMP
 IY=   5   5.284E-01   5.505E-01   6.035E-01   7.179E-01   9.983E-01
 IY=   4   5.173E-01   5.351E-01   5.728E-01   6.349E-01   7.179E-01
 IY=   3   5.000E-01   5.138E-01   5.394E-01   5.728E-01   6.035E-01
 IY=   2   4.827E-01   4.944E-01   5.138E-01   5.351E-01   5.505E-01
 IY=   1   4.717E-01   4.827E-01   5.000E-01   5.173E-01   5.284E-01
  IX=       1           2           3           4           5

 ************************************************************
 TIME STP=     1 SWEEP NO=     2 ZSLAB NO=     1 ITERN NO=     1


Whole-field residual sum(s) before solution
Resref values determined by SATELLITE
 variable   resref  (res sum)/resref
     TEMP   1.000E-10   1.000E+12


 Net source of TEMP at patch named: COLD      =-1.000E-08
```

```
Net source of TEMP at patch named: HOT      = 1.000E+02

************************************************************
spot values vs sweep or iteration number
   IXMON=     3    IYMON=     3    IZMON=     3 TIMESTEP=     1

Tabulation of abscissa and ordinates...
 ISWP         TEMP
    1  1.000E-10

************************************************************
residuals vs sweep or iteration number

Tabulation of abscissa and ordinates...
 ISWP         TEMP
    1  1.000E+12
************************************************************
PATCH(YLINE   ,PROFIL,   2,   2,   1,   5,   2,   2,   1,   1)
PLOT(YLINE   ,TEMP, 0.000E+00, 0.000E+00)
  VARIABLE        TEMP
    MINVAL=  3.651E-01
    MAXVAL=  4.944E-01
    CELLAV=  4.388E-01

1.00 +....+....+....+....+....+....+....+....+....+....T
     .                                                  .
0.90 +                                        T         +
     .                                                  .
0.80 +                                                  +
     .                                                  .
0.70 +                                                  +
     .                         T                        .
0.60 +                                                  +
     .                                                  .
0.50 +                                                  +
     .                                                  .
0.40 +                                                  +
     .                                                  .
0.30 +            T                                     +
     .                                                  .
0.20 +                                                  +
     .                                                  .
0.10 +                                                  +
     .                                                  .
0.00 T....+....+....+....+....+....+....+....+....+....+
```

```
    0   .1   .2   .3   .4   .5   .6   .7   .8   .9  1.0
the abscissa is       Y  .  min= 1.00E-01 max= 9.00E-01

************************************************************
PATCH(XPLANE  ,CONTUR,   2,   2,   1,   5,   1,   5,   1,   1)
PLOT(XPLANE  ,TEMP, 0.000E+00, 2.000E+01)
CONTOUR PLOT OF TEMP
  X-PLANE; Y HORIZONTAL, Z VERTICAL
CELL AVE= 4.663E-01
MIN VALU= 2.821E-01 MAX VALU= 5.505E-01
PHOENICS RECTANGULAR CONTOUR PLOT
N,S,E,W... DENOTE: NORTH, SOUTH, EAST ..

 0 HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
 1 S4            66666666       8888888          ON
 2 S44444          6666666       88888888         N
 3 S44444444       66666666      88888888         N
 4 S44444444444      6666666      8888888888      N
 5 S  44444444444      6666666      8888888888 N
 6 S     444444444      666666       888888888N
 7 S22         444444      6666666         88888N
 8 S222222       444444     6666666          88N
 9 S   2222222      44444      6666666         N
10 S       222222     444444     666666666     N
11 S00000      22222     444444     6666666666  N
12 S    00000     22222    444444      666666666N
13 S888     0000     2222     44444       666666N
14 S   8888    0000    2222     44444          66N
15 S666   8888   0000    2222     444444         N
16 S   666   888   0000   2222     4444444       N
17 S 44   666  888   000   22222    44444444     N
18 S2   44  66   888  000    2222    444444444   N
19 S  22  44  66  888   00    2222     44444444  N
20 S00  22  4  66   88   00    2222      44444444N
20  LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL

20   2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4
************************************************************

************************************************************
SATLIT RUN NUMBER =   1 ; LIBRARY REF.=  103
RUN COMPLETED AT 15:50:15 ON MONDAY, 15 NOVEMBER 1993
MACHINE-CLOCK TIME OF RUN =       2 SECONDS.
TIME/(VARIABLES*CELLS*TSTEPS*SWEEPS*ITS) = 8.000E-03
************************************************************
```

APPENDIX 5: Output from the Conjugate Gradient Solver

The field values of temperature at the z-direction slabs in the cube are shown below.

IZ= 1:

	IX=1	IX=2	IX=3	IX=4	IX=5
IY=1:	0.0016823	0.2820732	0.3964866	0.4494858	0.4716741
IY=2:	0.2820732	0.3650928	0.4271933	0.4648995	0.4827205
IY=3:	0.3964865	0.4271934	0.4606636	0.4862333	0.4999893
IY=4:	0.4494859	0.4648995	0.4862333	0.5056289	0.5172616
IY=5:	0.4716741	0.4827206	0.4999893	0.5172615	0.5284057

IZ= 2:

0.2820732	0.3650928	0.4271934	0.4648996	0.4827206
0.3650928	0.4069684	0.4488342	0.4788359	0.4943536
0.4271934	0.4488341	0.4764080	0.5000004	0.5137562
0.4648996	0.4788359	0.5000004	0.5211856	0.5350998
0.4827206	0.4943537	0.5137562	0.5350997	0.5505069

IZ= 3:

0.3964865	0.4271935	0.4606637	0.4862334	0.4999894
0.4271935	0.4488342	0.4764081	0.5000005	0.5137562
0.4606636	0.4764081	0.5000488	0.5236227	0.5393596
0.4862335	0.5000004	0.5236228	0.5511805	0.5728081
0.4999895	0.5137564	0.5393596	0.5728081	0.6035045

IZ= 4:

0.4494860	0.4648996	0.4862334	0.5056291	0.5172618
0.4648997	0.4788361	0.5000005	0.5211858	0.5351000
0.4862335	0.5000004	0.5236228	0.5511804	0.5728082
0.5056292	0.5211859	0.5511804	0.5930440	0.6348937
0.5172617	0.5350998	0.5728081	0.6348937	0.7179024

IZ= 5:

0.4716742	0.4827207	0.4999895	0.5172617	0.5284058
0.4827207	0.4943538	0.5137564	0.5350999	0.5505070
0.4999894	0.5137563	0.5393597	0.5728083	0.6035046
0.5172617	0.5350999	0.5728082	0.6348937	0.7179025
0.5284059	0.5505069	0.6035047	0.7179025	0.9983179

APPENDIX 6: The GREX3 Gauss-Seidel Subroutine

```
C file name .................... gxutil.ftn .............041290
C file name .................... gxutil.ftn .............041290
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C     SUBROUTINE GXGAUS replaces the built-in linear-equation solver
C     of PHOENICS by a Gauss-Seidel solver. The built-in solver is
C     appreciably more efficient than the Gauss-Seidel solver in
C     most circumstances; so subroutine GXGAUS is of interest mainly
C     as an example of how users may replace the built-in solver by
C     solvers of their own.
C
C***  SUBROUTINE GXGAUS is called from GREX3, group 8, section 14;
C     and it is entered when USOLVE is set TRUE in
C     the SATELLITE, and also CSG3 is set equal to 'GAUS'.
C
C.....The dummy OVRRLX is an over-relaxation factor for use in
C     the whole-field linear-equation solvers; LITDSH is the maxmum
C     number of iterations which are to be performed by the linear-
C     equation solver for variable phi; IXMON, IYMON and IZMON are
C     IX-, IY- and IZ-value of spot-value location respectively;
C     ENDIT is the iteration-termination criterion; XCYCLE is an
C     logical for setting cyclic boundary conditions along the
C     east and west boundaries of the integration domain; LUPR1 and
C     LUPR3 are the logical units.
C
C.....The library case 103 exemplifies its use.
C
      SUBROUTINE GXGAUS(OVRRLX,LITDSH,NX,NY,NZ,IXMON,IYMON,IZMON,ENDIT,
     1                 XCYCLE,LUPR1,LUPR3)
      INCLUDE 'GRDLOC'
      LOGICAL EQZ,NEZ,LTZ,GTZ,GEZ,LEZ,GRN
      COMMON /IGE/IXF,IXL,IYF,IYL,IREG,NZSTEP,IGR,ISC,IRUN,IZSTEP,ITHYD,
     1       ISWEEP,ISTEP,INDVAR,VAL,CO,NDIREC,WALDIS,PATGEO,IGES20(6)
      INTEGER VAL,CO,WALDIS,PATGEO
      LOGICAL XCYCLE,DONE,MON,MONZ,MONX,MONY
      SAVE DONE
      COMMON /NAMFN/NAMFUN,NAMSUB
      CHARACTER*6 NAMFUN,NAMSUB
      DATA DONE/.FALSE./
C
      NAMSUB = 'GXGAUS'
C    Initialize the variable solved on the first visit...
      NXNY = NX*NY
      IF(.NOT.DONE) THEN
```

```
      CALL ZERNUM(LOF(L3PHI),NXNY*NZ)
      DONE = .TRUE.
    ENDIF
    RLX = OVRRLX
    IF(EQZ(OVRRLX)) RLX = 1.0
    LUPRST = LUPR1
    LIT = IABS(LITDSH)
    IF(LITDSH.LT.0) THEN
      LUPR1 = LUPR3
      CALL WRIT40('OUTPUT FROM SOLVER SUBROUTINE GXGAUS    ')
      CALL WRIT3I('INDVAR  ',INDVAR,'ISWEEP  ',ISWEEP,'ISTEP   ',
     1             ISTEP)
    ENDIF
    IPHIO = LOF(L3PHI)
    ISUO = LOF(L3SU)
    IAPO = LOF(L3AP)
    IF(NX.GT.1) IAEO = LOF(L3AE)
    IF(NY.GT.1) IANO = LOF(L3AN)
    IF(NZ.GT.1) IAHO = LOF(L3AH)
    MON = LITDSH .LT. 0
    DO 20 ITER = 1,LIT
      IPHI = IPHIO
      ISU = ISUO
      IAP = IAPO
      IAE = IAEO
      IAN = IANO
      IAH = IAHO
      DO 30 IZZ = 1,NZ
        MONZ = MON .AND. IZZ .EQ. IZMON
        DO 40 IX = 1,NX
          MONX = IX .EQ. IXMON .AND. MONZ
          DO 50 IY = 1,NY
            MONY = IY .EQ. IYMON .AND. MONX
            SNUMER = 0.0
            SDENOM = 0.0
            IPHI = IPHI + 1
            IF(NX.NE.1) THEN
              IAE = IAE + 1
              IF(IX.NE.1) THEN
                SNUMER = F(IAE-NY)*F(IPHI-NY)
                SDENOM = SDENOM + F(IAE-NY)
              ENDIF
              IF(IX.NE.NX) THEN
                SNUMER = SNUMER + F(IAE)*F(IPHI+NY)
                SDENOM = SDENOM + F(IAE)
```

```
            ENDIF
          ENDIF
          IF(NY.NE.1) THEN
            IAN = IAN + 1
            IF(IY.NE.1) THEN
              SNUMER = SNUMER + F(IAN-1)*F(IPHI-1)
              SDENOM = SDENOM + F(IAN-1)
            ENDIF
            IF(IY.NE.NY) THEN
              SNUMER = SNUMER + F(IAN)*F(IPHI+1)
              SDENOM = SDENOM + F(IAN)
            ENDIF
          ENDIF
          IF(NZ.NE.1) THEN
            IAH = IAH + 1
            IF(IZZ.NE.1) THEN
              SNUMER = SNUMER + F(IAH-NXNY)*F(IPHI-NXNY)
              SDENOM = SDENOM + F(IAH-NXNY)
            ENDIF
            IF(IZZ.NE.NZ) THEN
              SNUMER = SNUMER + F(IAH)*F(IPHI+NXNY)
              SDENOM = SDENOM + F(IAH)
            ENDIF
          ENDIF
          ISU = ISU + 1
          IAP = IAP + 1
          PHINEW = (SNUMER+F(ISU))/ (SDENOM+F(IAP))
          PHIOLD = F(IPHI)
          F(IPHI) = PHIOLD + RLX* (PHINEW-PHIOLD)
          IF(MONY) THEN
            IF(MOD(ITER,10).EQ.0) CALL WRIT2I('ISWEEP  ',ISWEEP,
     1           'ITER.NO.',ITER)
            CALL WRIT2R('SPOT VAL',F(IPHI),'SPOT DIF',
     1                  F(IPHI)-PHIOLD)
            IF(ITER.GT.2*NZ .AND. ABS(F(IPHI)-PHIOLD).LE.
     1           ENDIT) GO TO 60
          ENDIF
   50     CONTINUE
   40   CONTINUE
   30 CONTINUE
   20 CONTINUE
C------------------------------------------- end of loop
   60 LUPR1 = LUPRST
      END
```