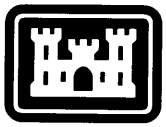
Special Report 87-26

December 1987



US Army Corps of Engineers

Cold Regions Research & Engineering Laboratory

XYFREZ.4 User's manual

Kevin O'Neill

Prepared for
OFFICE OF THE CHIEF OF ENGINEERS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Special Report 87-26	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION U.S. Army Cold Regions Research and Engineering Laboratory 6b. OFFICE SYMBOL (If applicable) CECRL	7a. NAME OF MONITORING ORGANIZATION Office of the Chief of Engineers
6c. ADDRESS (City, State, and ZIP Code) 72 Lyme Road Hanover, New Hampshire 03755-1290	7b. ADDRESS (City, State, and ZIP Code) Washington, D.C. 20314-1000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. 4A762730 AT42 / TASK NO. BS / WORK UNIT ACCESSION NO. 001

11. TITLE (Include Security Classification)
XYFREZ.4 User's Manual

12. PERSONAL AUTHOR(S)
Kevin O'Neill

13a. TYPE OF REPORT	13b. TIME COVERED FROM ____ TO ____	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
		December 1987	60

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Computer modeling; Finite elements; Freeze/thaw; Heat conduction; Phase change; User's manual

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Using the program XYFREZ, version 4, one may simulate two-dimensional conduction of heat, with or without phase change. The mathematical method employed uses finite elements in space and finite differences in time, and includes latent heat effects through a singularity in the heat capacity. The user need have no real familiarity with either the underlying equations or the numerical procedures. He must only specify material properties, geometrical features, initial and boundary conditions, and information on the desired manner and duration of simulation through time. Heterogeneous material properties may be specified. Boundary conditions currently implemented allow one to specify 1) temperature values which vary arbitrarily in space and time, 2) convective conditions, via a heat transfer coefficient and an ambient temperature, and 3) a no-flux or symmetry condition. The program outputs computed temperature values at numerical mesh points, as well as information for later plotting. From the latter one may see the mesh configuration as well as the phase change isotherm location on it over time.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Kevin O'Neill	22b. TELEPHONE (Include Area Code) 603-646-4100 22c. OFFICE SYMBOL CECRL-EG

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

PREFACE

This report contains a user's manual for running the computer program XYFREZ, version 4. The basic mathematical method incorporated in it was devised, developed, and programmed by Dr. Kevin O'Neill, of the Geotechnical Research Branch, Experimental Engineering Division, U.S. Army Cold Regions Research and Engineering Laboratory. The initial program and manual were expanded and tested further by Dr. O'Neill and Capt. Richard Jardine of the U.S. Military Academy at West Point. Support for this work came primarily from the Department of Army In-House Laboratory Independent Research Program (ILIR). Documentation and work to make the program more "user friendly" was supported in part by DA Project 4A762730AT42, Task BS, Work Unit 001, *Deformation of Soils and Pavements Due to Freeze/Thaw*, under the leadership of Dr. Richard Berg.

The author is thoroughly grateful for forward-looking support from the two sources mentioned, which has enabled the development of a program from innovative mathematical concepts, all the way through to a documented and reasonably user-friendly program. The diligent and insightful assistance of Capt. Jardine is also appreciated. Terry Rogers and Nancy Humiston reviewed the report with great care, struggling through all its directions in detail, as representative users. Their reactions were much appreciated by the author, and numerous changes were made in response to their experiences. The author also acknowledges the patient editorial assistance of Steve Bowen and Sandy Smith, and the most accommodating illustrating of Ed Perkins.

CONTENTS

ILLUSTRATIONS

XYFREZ.4 User's Manual

KEVIN O'NEILL

INTRODUCTION

The program uses linear triangular finite elements to compute two-dimensional heat conduction with or without phase change. Convection may enter through convective boundary conditions, as explained below, but is not considered in the domain interior. Heterogeneous material properties may be specified. The current version is the latest in a developing series, each with various added efficiencies, options, and greater idiot-proofing than the previous. The user should note that, while the current version has been tested carefully in a variety of problems with known solutions, it has not yet been widely used. This causes us to muse on an old adage—at least as old as the young field of digital computing—

> *There are no bug-free programs, only programs in which all the bugs have not yet been discovered.*

A parallel set of programs is being developed, the RZFREZ series, for solving axisymmetric heat conduction and phase change problems in cylindrical coordinates.

This manual describes the use of the program XYFREZ, version 4. Although the program utilizes innovative, state-of-the-art methods (O'Neill 1983) both the program and this manual have been constructed as much as possible to accommodate users with minimal knowledge of computational methods. Ancillary programs for automatic generation of input data and for plotting input data and computed results are also available at CRREL. Contact Kevin O'Neill with any questions or comments.

The mathematical basis of the methods used and their numerical implementations are described in the paper by Kevin O'Neill (1983). In general, an explanation of the finite element method and its use in the program is beyond the scope of this manual. However, the interested reader may consult any of a variety of introductory finite element books to learn more than enough to grasp this program, (e.g. Pinder and Gray 1977, and Becker et al. 1981). Suffice it to say for what follows that one approaches problem solution by dividing the overall physical domain up into a mesh, i.e. smaller subsections, each of which is known as a finite element. In the particular variant of

the method employed here, each of these elements must be a triangle. Based on linear interpolation over the interior of each element, the method solves for temperature values at the vertices ("nodes") at the corner of each element. The program steps through time to obtain transient solutions, and the nodal temperature values are printed at whatever intervals the user specifies. The number of elements and nodes that are used in a given problem is up to the user, who inputs the mesh. On the whole, more elements and/or smaller elements can be used where one wishes relatively finer resolution to be applied in the calculations. An example mesh appears in the section entitled *Sample Problem*.

In principle, you may use as many nodes, elements, and time steps as you please (or can afford) in a given problem. The program is set up so that specific computer memory requirements are derived from input quantities, while a large block of space is potentially set aside in the introductory routine for later use within the program. If you are not running on the CRREL Prime computer system or do not have a virtual machine at your disposal, the potential memory space initially set aside may be larger than your machine can handle. For example, some arrays are assigned potential sizes of up to 2000k words. If you cannot live with that kind of space requirement in "COMMON," call Kevin O'Neill.

In the following sections the required input data are described, followed by explanatory comments on the meaning of and options within that data. *It is strongly recommended that you study the explanatory comments carefully.*

INPUT DATA SPECIFICATION

For the most part, data for this program are read in from an input file composed and stored by the user before running the program. On the CRREL Prime system, the program will ask you to enter the name of your data file through your terminal, after you have given the command to run the program. On other systems one may have to alter this in accordance with the prevailing practices and available options, noting that the data are read in on unit 5. Unless otherwise noted, everything is read in under open format. That is, any format may be used in the data list—exponential notation or not, commas or any number of spaces to separate entries, etc. *The usual Fortran variable type conventions must be observed*, keyed to the names as spelled below: variable names beginning with the letters "I" through "N" are integers and must not contain a decimal point. Study the example data in the section entitled *Sample Problem* for more guidance. The choice of unit system is up to the user, but must be strictly consistent.

The data are:

Item No.	Variable	Meaning
0.	NBND, NMBC1, NELS, NTS, MEDIA	Number of nodes, number of "first type" boundary nodes, number of elements, number of time steps, number of different media in region simulated, respectively
1.	IO	Unit number for writing all output except for plotting information
2.	TITLE	Alphanumeric output title, 80 chararacters
3.	ITRLM, KPRTVL	Iteration limit, printing interval respectively
4.	IZDPLT, MSHPRT, KBVPRT	Integer flags for printing control of output for isotherm plotting, for detailed mesh description, and for time step list and boundary condition details, respectively
5.	KPAUSE	Flag controlling pause option
6.	(NODE, NDBC, XG, YG)	List of node numbers, boundary condition indicator, X and Y coordinates, respectively
7.	(NL, MTYPE, N1, N2, N3)	List of element numbers, material type for each element, and the global node numbers at each element's vertices
8.	(HETF, HETU, CNDF, CNDU, EL, TF)	List of thermal properties for each MTYPE.
9.	TIMLIM, AMP, THETA	Time duration of simulation, numerical control parameters
10.	KIC	Integer flag for initial condition choice system
11.	TEMPIC or (NODE, TEMPIC)	Initial conditions (see explanatory comments)
12.	KBCT	Choice indicator for BC and time step system in item 13
13.	(NODE, TBCVAL), and KDT or (TIMBD, NUMCHG, NTSBD, (NODE, TBCVAL))	List of boundary nodes and their temperatures, followed by time step system indicator (See explanatory comments)
14.	H, TA	Boundary heat transfer coefficient, ambient temperature

EXPLANATORY COMMENTS

Item 0

NBND = number of nodes in the mesh, including the boundary; NELS = number of elements in the mesh; NTS = the number of time steps to be taken; MEDIA = the number of different media (material types) within the region covered by the element mesh. The program functions by solving for the temperature distribution in space at sequential points in time. It proceeds incrementally from each point in time to the next, hence taking "time steps" until the simulation is complete. The value of NTS only indicates the total number of such steps (points in time) you want to have. The manner in which those steps are taken is determined by more detailed information in other items in the input.

NMBC1 = the number of boundary nodes with prescribed temperatures, whether constant or time-varying (as opposed to boundary nodes with flux or convective boundary conditions). These are "first type" boundary condition nodes. If you cannot easily determine this number or in any case want to have the program do so, you can begin a run using any integer here. The program determines the correct value, and if your input is wrong, it will complain mildly, tell you the correct number, and then halt. You can then put this correct number in your data and start over.

Item 1

The user supplies an integer value for IO to determine where the output will be sent. IO is the number of the unit on which all non-plotting output will be printed. On the CRREL Prime system, IO = 1 causes the output to appear on your CRT screen. Any other number will cause writing to be done into a file. If you specify IO not equal to 1, the program will ask you to enter an output file name through your terminal. This output file name may correspond to a new file, which will be created, or an old file which will be overwritten. To ensure that you do not collide with Fortran preempted file numbers or others defined elsewhere in the program, use IO = 6 to write output into a file.

Users on systems other than CRREL's Prime may have to rewrite file definitions, openings, closings, etc., in accordance with the conventions built into their operating systems. As the program is currently set up, all non-interactive input reading is done on unit 5. Unit 33 is selected for writing output information to be used in plotting programs designed for the CRREL system.

Item 2

TITLE is an 80-character-long label for your problem. In other words, just specify on one line a title you want to appear at the top of the output.

Item 3

ITRLM is the iteration limit. Because phase change problems are inherently nonlinear and because we wish to use implicit computational schemes, in principle some iteration is required within each time step for a fully consistent solution. Experience has shown however that this is usually not necessary to obtain acceptable answers, and the user is advised to specify ITRLM = 0, at least for starters. If time-step sizes are reasonable (see below) but bumpiness still develops in the solution in either space or time, higher values of ITRLM may be tried. The value of this parameter decides how many times all the equations will be set up with improved parameter evaluations and solved over again (after the first time) within a single time step. Thus specifying higher values of ITRLM will increase computing expenses, assuming that time step sizes are kept the same as ITRLM is increased. Note that excessive "bumpiness" in the shape of the phase change isotherm can occur when it is curved and you did not specify a fine enough mesh to resolve its shape.

KPRTVL is the printing interval. That is, the temperature solution for all nodes will be printed out every KPRTVL time steps, beginning with time step number KPRTVL.

Item 4

IZDPLT is an integer which controls printing of information on unit 33 for later use in plotting. As of this writing, the plotting program to do this has only been implemented on the CRREL Prime computer system.

If IZDPLT is set to zero, then no such printing is done; if you are not set up to print on unit 33 or to do any of this plotting, set IZDPLT = 0.

If IZDPLT = -1, then only your input data for the mesh will be stored and no information for isotherm plotting will be saved. This option allows you to plot your input mesh and check it before proceeding with actual solution of the problem over time. See explanation below of the KPAUSE parameter. The necessary information is stored before the PAUSE in the program occurs. Thus you can halt a run at the PAUSE point, and the mesh plotting information will not be lost.

If IZDPLT is greater than zero, then information will be saved for plotting both the mesh and freezing isotherms on it at specified points in time. Every IZDPLT time steps, the results necessary to perform that plotting are dumped into a file you have indicated, beginning with time step number IZDPLT.

If you specify any nonzero value for IZDPLT, the program will ask you to enter through your terminal a file name under which plotting information for the current run is to be stored for later use (see the following section). The name may be chosen arbitrarily, to allow you to distinguish between results from different runs. Later, if you use the program KOPLOT to plot the contents of the file, then that program will request the file name.

MSHPRT is a flag which controls printing (on unit IO) of mesh information which you have input. If you specify MSHPRT = 1, then node locations together with NDBC values will be written, as well as the element incidence list (which nodes go with which elements). If MSHPRT = 0, this information will not be printed.

The value of KBVPRT controls the printing of information on the times and boundary values associated with each time step. If you do not want any of this information printed, specify KBVPRT = 0, and move on.

If KBVPRT = 1, then at the beginning of the run the program will print a list of the times and time step sizes for each time step, without indicating the associated boundary temperature values.

If KBVPRT = 2, then the same information as for KBVPRT = 1 will be printed, together with the temperatures of all type 1 boundary nodes at each time step. (See explanation of "type 1" boundary nodes under item 6 below.) These options allow you to check the boundary and time step information to be used in the solution procedures before they are performed (before the PAUSE). This is especially worth checking when you have had the program interpolate boundary values and associated times from values you have given it at only a few points in time (see items 12–13).

Item 5

This item concerns the PAUSE option. If KPAUSE = 1, the program will pause after all input to the solution procedures has been either input by the user or calculated by the program; you will then be asked on the CRT screen (i.e., unit 1) whether you wish to proceed. If KPAUSE = 0, the program will assume that you do not want to pause, and this question will be skipped. This option is intended to allow you to check over all the input data before grinding up a lot of computing time marching ahead through the solution in time.

Item 6

Input a sequence of lines, one for each node, in which the node number (integer), the boundary condition indicator NDBC (integer), and the X and Y coordinates (floating point) for each node are given.

The boundary condition indicator NDBC should equal 1 if the associated node is on the boundary and temperature values are to be specified there ("type 1" boundary conditions). The specified temperature may be constant or may vary with time. If a node is on the boundary and NDBC = 0 is specified, the program will assume that the node is on a zero flux boundary or line of symmetry. At present one cannot specify nonzero boundary flux except in the sense that applies when NDBC equals 2, explained below. General nonzero flux boundary conditions are under development.

If the node is not on the boundary, specify NDBC = 0. In principle, one could specify the temperature values of node points *not* on the boundary, as explained below (items 12–13) by simply tagging an interior node with a value of 1 for NDBC. Then one would provide temperature information through time as if this were a boundary node. The user should note however that the effect of constraining interior temperature values has not been investigated in the development of this program.

If NDBC is set equal to 2 for a node, this means that a convective, radiation, or transfer-type boundary condition applies there. This means that heat flux through the boundary around that node is controlled by a relation of the form

$$QN = H * (T - TA)$$

where QN is the heat flux normal to the boundary (positive *outward*), H is a heat transfer coefficient (heat flow rate/area/degree), T is the unknown boundary temperature, and TA is the ambient temperature. Thus, when the ambient temperature is lower than the boundary temperature, heat will flow out of the region. Values for H and TA are given by the user in item 14.

Typical input for item 6 might look like:

```
1, 0, 0.0, 1.0
8, 0, 10.0, 0.0
2, 1, 3.0, 4.0
```

These example numbers say that node number 2 has a "type 1" boundary condition (its temperature value will be specified), and it is located at $X = 3.0$, $Y = 4.0$. Nodes number 1 and 8 will not have their temperatures specified and thus NDBC = 0 for each. They are either interior nodes or are on boundaries where the normal flux of heat through the boundary (temperature gradient normal to the boundary) is zero. In either case the program will solve for the temperatures at nodes 1 and 8 and they are located at $X = 0.0$, $Y = 1.0$ and $X = 10.0$, $Y = 0.0$, respectively.

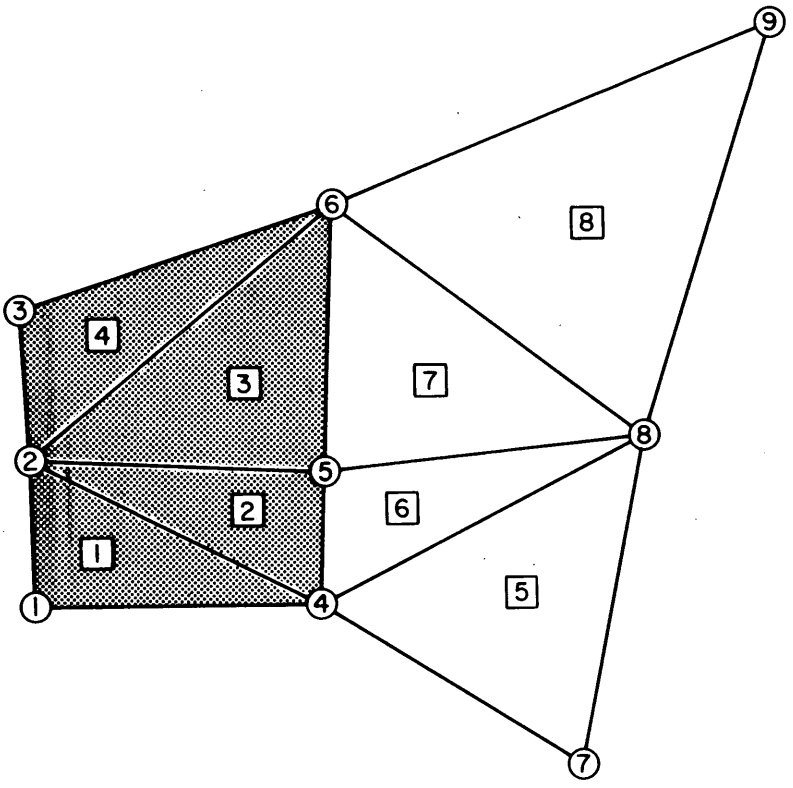


Figure 1. Example finite element mesh constructed of linear triangles. Shaded elements 1 through 4 consist of material number 2, and the rest are material number 1.

Item 7

Specify a list of element numbers, with each one followed by the material type within the element, in turn followed by the node numbers of each vertex of the triangular element. For example, for the mesh in Figure 1 the user should input:

1, 2, 1, 4, 2
2, 2, 2, 4, 5
3, 2, 2, 5, 6
4, 2, 2, 6, 3
5, 1, 8, 4, 7
6, 1, 5, 4, 8
7, 1, 5, 8, 6
8, 1, 8, 9, 6

This means that element number 1 contains material number 2 and has nodes 1, 4, and 2 at its vertices, while element number 5 contains material type 1 and has nodes 8, 4, and 7, etc. These numbers are just labels, in effect, which means that you may number the elements in any order that you like. Similarly, you can number the nodes any way you like. The relation between node and element numbers is arbitrary, as long as you are consistent in the manner shown in the example (i.e., make sure that you associate the correct nodes with each element).

The efficiency of the algorithm for ultimately solving the equations depends greatly on the compactness of the information it receives. This means that you should utilize your freedom in node numbering *to minimize the difference between node numbers across any one element.* If you have used the automatic mesh generator program, then your node numbering scheme will already have been optimized.

If you have the program print out detailed mesh information (MSHPRT = 1), you may notice that the nodes associated with some element or other are not listed in the same order in which you input them. That is because the program automatically reorders them when necessary to preserve right-handedness in the local coordinate systems in which it operates. (In other words, nevermind.)

Item 8

This item consists of a sequence of lines, one for each material type present, which provide thermal properties to be assigned to the various elements.

Information on the first line of input characterizes material number 1, and all elements assigned MTYPE number 1 will be assigned the listed characteristics. The second line characterizes material number 2, and so forth. If the domain considered is entirely homogeneous, then all elements would have been assigned MTYPE = 1 in item 7, and item 8 would consist only of one line.

HETF and HETU are the heat capacities of the frozen and unfrozen material, respectively, *per unit volume*; CNDF and CNDU are the frozen and unfrozen thermal conductivities, respectively.

EL is the latent heat, meaning the latent heat released *per unit volume of material frozen*. Thus for pure water EL would be the latent heat per unit mass of water times the density of *ice*, or 80 cal/g * 0.917 g/cc = 73.36 cal/cc ice. For soil, this figure would be multiplied times a factor equal to the volume of ice per volume of soil. For a rigid, saturated soil, this factor equals the porosity.

TF is the phase change temperature.

For these parameters, as for everything else in the program, any *consistent* system of units may be used.

Note that the program assigns a single material type to each element, but allows for a discontinuity in material (thermal) properties within a single element when phase change occurs.

Item 9

TIMLIM is the length of time to be simulated. Its value is ignored and may be specified arbitrarily when boundary condition information is given for specific points in time (see explanation of items 12 and 13).

AMP and THETA are numerical control parameters which have to do with treatment of the nonlinearity and the degree of implicitness in the solution scheme, respectively. Recommended values are 0.65 for each, and it's probably best not to mess with these. If stability problems develop, you might increase THETA to 0.75 or to 1.0. AMP is an interpolation parameter, which determines the point within a time step at which nonlinear coefficients are to be evaluated. This will be explained in a forthcoming paper. In general, unless you know what you're doing, leave these quantities alone. The value of AMP only matters when ITRLM is greater than zero.

Item 10

Here the program is told how the initial conditions are going to be specified.

If KIC equals zero, then all nodes have the same initial temperature, to be specified in item 11. If KIC equals 1, then the initial temperature for each individual node will be specified in item 11.

Item 11

If KIC was specified as zero, then a single value (TEMPIC) is input here and all nodes will be assigned that initial temperature.

If KIC was specified as 1, then this item contains a list of *all* the nodes with the initial temperature of each, with one node per line. For example the list

1, 10.0
3, 4.0
2, 9.55

means that node 1 has an initial temperature of 10.0, node 3 an initial temperature of 4.0, and node 2 an initial temperature of 9.55.

Item 12

In this item you provide the program with a value for KBCT, which indicates which option in item 13 will be used to specify the boundary conditions and to determine the time steps. KBCT may equal either 1 or 2, the significance of each choice being explained below in connection with item 13. While the numerical system used here has been designed to be highly stable, it is always possible to take time steps that are so large that they cause the solution to degenerate. A conservative rule of thumb is:

> *Do not take time steps that cause the phase change isotherm to pass through more than about one third of an element in a single time step.*

If phase change is not occurring, you might apply this to some other reference isotherm, an appropriate alternative being the leading edge of an imposed change. You can attempt to adhere to this by using some general foreknowledge of the problem at hand and then choosing among the time step options accordingly. Alternatively you can try some crude preliminary runs, and subsequently adjust the time step sizes.

Item 13

There are two systems of boundary condition and time step specification under this item, each of which pertains only to setting values for "type 1" boundary nodes. That is, we are only talking about setting temperature values for certain nodes, in particular those for which NDBC was specified as 1 in item 6.

BC, DT System 1

If KBCT was set to 1, this means that all type 1 boundary nodes will have temperatures which will be fixed through all time. Thus item 13 contains a list, with one node per line, of all type 1 nodes, each with its assigned temperature.

This list is followed by a value for KDT, which indicates what time step (DT) system will be used. If KDT is assigned the value 1, then uniform time steps are taken between time zero and TIMLIM. If KDT is set equal to 2, then time steps are taken which are equivalent to equal increments in the square root of elapsed time between time zero and TIMLIM. The reason for this second option is that many heat flow/phase change problems with constant boundary conditions start up quickly and then settle down, with changes occurring at rates more or less proportional to the square root of time. In any case, the KDT = 2 option allows you to take small time steps in early time with increasingly large steps as time proceeds.

Note again the danger of taking time steps which are too large, which can creep up on you unawares if you just let the program project DT over long times. The implicit solution system used is designed to be robust in the face of large time steps, even when DT is so large that accuracy degenerates. But any car can be driven off the road. Experiments with the program have also suggested that the numerical method used is much more tolerant of large time steps than some moving boundary approaches. Nevertheless, the nonlinearity of the problem undercuts the guarantees of stability derived for otherwise comparable implicit linear problems. Investigations of moving boundary phase change systems suggest that excessively large time steps can cause instability even when there is very little action in the solution. The most common, seemingly innocent situation where one is likely to slip into unwise DT sizes occurs near a steady state. As time runs on, little happens and the user (or the program on his behalf) is tempted to take larger and larger time steps. If instability should develop in the solution, check this item.

The KDT time step options are summarized as follows:

KDT = 1 Uniform time steps
KDT = 2 Uniform steps in square root of time

If there were only two boundary nodes (numbered 3 and 12), both at 8 degrees through all time, and you wanted to take uniform time steps, you would specify the following for item 13, assuming you had specified KBCT = 1 in item 12:

3, 8.0
12, 8.0
1

BC, DT System 2

If KBCT = 2, you must specify values for each of the type 1 boundary nodes at various times. When you specify the time at which some boundary values apply, the program will also record how many time steps you want to be taken since the last such specification. Thus in addition to boundary values you will be indirectly specifying time step sizes and overall time duration. This option for boundary condition and time step specification may seem complicated in its verbal description. However, with a little familiarity you will see that it is designed to give you the greatest flexibility in assigning time and space dependent boundary temperatures with arbitrary time steps, by specifying a minimum of information. The system is most easily comprehended through an example, which will be followed by a more general statement of the system.

Suppose you had only two type 1 boundary nodes, numbered 1 and 4, and you wanted to assign their temperatures and take time steps as shown in Figure 2. Information need only be specified for the points indicated by the arrows on the horizontal axis in the figure. The values at time zero should not be specified here—they have already been specified via the initial conditions. For the total elapsed time TIMLIM of 30 you would input the information in Figure 2 as

10.0, 2, 3
1, 5.0
4, 1.5
30.0, 1, 2
1, 5.0

This means that the first boundary information is to be specified for an elapsed time of (TIMBD =) 10.0. Two nodal temperatures will be specified (NUMCHG = 2), and (NTSBD =) three time steps will be taken since the last information was given (in this case, since the initial condition shown on the left axis of the figure). The next two lines contain the specific boundary temperature information for time 10.0: the values 5.0

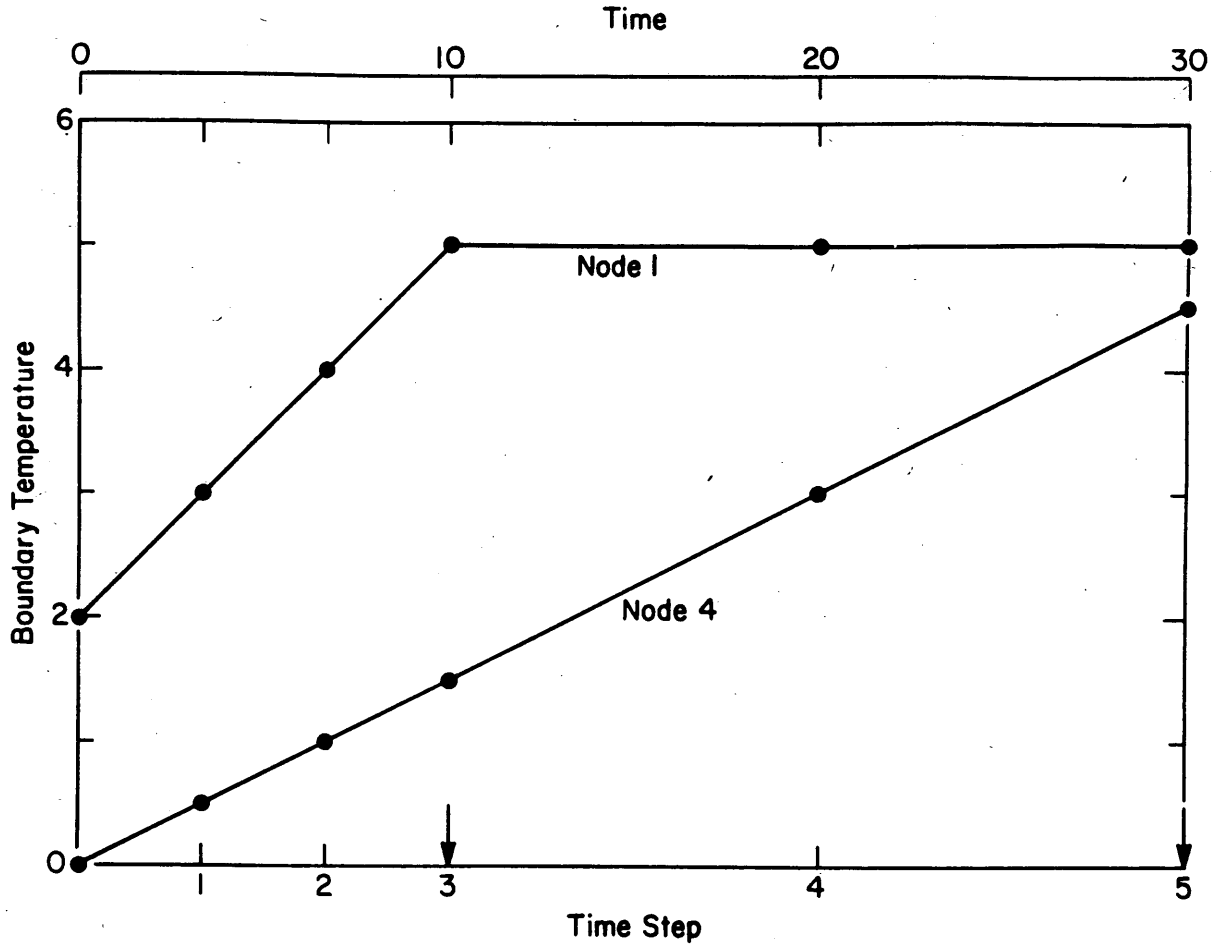


Figure 2. Example boundary condition temperatures over time. Vertical arrows indicate points in time where information is given in the input data.

and 1.5 are assigned to nodes 1 and 4, respectively. Because three time steps have been indicated, the program will take three time steps after time zero, each of equal length (10.0/3 = 3.33 = DT) and all boundary temperatures will be linearly interpolated over the intermediate time steps since the initial condition. This is indicated on Figure 2 between time zero and the first vertical arrow.

The next line gives information for an elapsed time of 30.0. Only the temperature of one node is to be assigned, and 2 time steps are indicated since the last specification. Boundary values are given in the next line for time 30.0, the temperature value of node number 1 is specified as 5.0. The program then determines uniform time steps between time 10.0 and time 30.0, and linearly interpolates the temperature values over intermediate time steps.

Note that for the second point in time *only* the temperature of node 1 had to be specified. *Temperature information need only be given when the rate of change of a boundary temperature is to be altered.* Because the rate of change of node 4 remained unaltered after time 10.0, nothing was indicated for it at time 30.0, and the program extrapolated along the same line begun over the first three time steps. In effect, a linear interpolation is accomplished between time zero and time 30.0. This system allows you to hold the temperature of a boundary node constant by assigning it the same value in two sequential specifications, and then giving no more information about it. So, for example, if you wanted to continue holding node 1 at 5.0 degrees during time steps after time step 5 (after time 30.0), you could simply omit any information about it in subsequent specifications after those shown above.

The temperature of a type 1 boundary node will be kept at its initial condition until and unless some information to the contrary appears in this section of the input. For example, suppose there were a type 1 boundary node which you wanted to remain at its initial temperature while nodes 1 and 4 changed as above. In this case you would input exactly the same information for times 10.0 and 30.0 as appears above. If you wanted the temperature of this additional boundary node to *begin* changing at time 30.0, you would specify a new value for it for some time *after* 30.0.

At first it may seem strange that boundary temperatures must be indicated when a nodal temperature's *rate of change* is altered, and not when its value is altered. This is to allow efficient assignment of boundary temperature changes which are smooth (linear) over relatively long stretches (e.g. that of node 4 in the example) without the user having to indicate every single change in value. If the program were set up to allow specification of values by node (i.e. not by time) using only beginning and ending values over arbitrary time intervals, that would require the user to coordinate the information in a potentially complex way, so that the same time steps fit the changes in boundary values for all nodes.

One always retains the option of inputting values for all type 1 boundary nodes at all time steps (or every so many time steps). This is in fact likely to be suitable for laboratory or field data, in which data are often obtained in (long) columns of parallel measurements over the entire duration of an operation. While it might be possible to achieve the same results by specifying less information, there is no compelling computational reason for doing so.

If you only want to change the time step size but let the program continue with the previously specified boundary values, simply specify NUMCHG = 0 at a new time, and provide no further information. That is, in the example above, if after time 30.0 you wanted node 1 to continue at 5 degrees and node 4 to continue changing at the

same rate, but you wanted to take four more time steps of DT = 25.0, you would follow the specifications above with this one:

130.0, 0, 4

The overall system when KBCT = 2 can be stated briefly as follows:

Boundary temperature values for at least some type 1 nodes (NDBC = 1) must be read in for some time or times after the initial condition. Thus when KBCT = 2, item 13 consists of a sequence of boundary information specifications. Each specification consists of a value for TIMBD (the time for which information is being given), a value for NUMCHG (the number of nodes whose rates of change or initial values are to be altered), and a value for NTSBD (the number of time steps to be taken since the last specification). This information is followed as part of the same specification by a list NUMCHG-long of nodes each followed by its temperature at TIMBD. The temperature of a type 1 boundary node will remain at its initial condition through all points in time for which no other information is provided.

Note that *no value of KDT is to be provided when KBCT = 2.*

Item 14

In this item one specifies values for a heat transfer coefficient and ambient temperature for those boundary nodes flagged with NDBC = 2. If NDBC does not equal 2 for any node, then skip this item.

At present, the program is set up so that only uniform constant values for *H* and for *TA* apply for all such boundary nodes. This should be generalized shortly so that variable *H* and *TA* can be used, and so that *H* can be a function of the temperature.

A potential problem arises at nodes where different types of boundary conditions intersect. To keep the rules straight here, one might think of a hierarchy of boundary conditions, with the highest being a type 1 condition (NDBC = 1), the second highest a heat transfer/radiative/convective type (NDBC = 2), and the lowest being a specified (zero) flux (NDBC = 0). The higher priority BC type always takes precedence. For example, with reference to Figure 1, suppose that you wanted the temperature to be specified along the boundary line containing nodes 1, 2, and 3, and you wanted a convective-type condition over the line between nodes 3, 6, and 9. In this case node 3 would be a type 1 boundary node, and you would specify its temperature along with that of nodes 2 and 1. Alternatively, if the side 1, 2, 3 was a zero-flux boundary, and the side 2, 6, 9 a convective boundary, then node 3 would be treated as a convective-boundary node. If one observes these conventions, then the program is always capable of translating boundary-node information correctly into information pertaining to boundary areas between the nodes.

There is currently one constraint on the use of convective-type boundary conditions: *Do not specify all three nodes of a single element as being convective-boundary nodes.* Thus, if one wanted a convective condition to apply over the boundary between nodes 4, 7, and 8, in Figure 1 he would have to use a different mesh arrangement. The program is set up to alert you if you have violated this rule.

RUNNING THE PROGRAM AND PLOTTING ROUTINES

If you are not computing on the CRREL Prime system, you will have to compile the source code, and make sure in the process that all the conventions of your system are satisfied, including the availability of input/output files. The remaining paragraphs in this section refer only to operations on the CRREL system.

When you want to run XYFREZ on the CRREL Prime system, first make sure that you have stored all necessary input in a file in your current user file directory. The file may have any name. To execute the program simply enter the command

XYFREZ

as if it were a Prime system command. The program will begin executing as soon as that command has registered, and you will promptly be asked the name of your data file. Enter that name and the program will proceed, writing information either on the screen or in a file, depending upon what you have specified for IO.

During program execution, a plotting output file may be generated, depending upon the value you have specified for IZDPLT. At CRREL, this file will reside in your user file directory until you remove it. You can use the plotting software described below to obtain mesh and phase change isotherm plots for the run which generated the file in question.

To plot the mesh only (for IZDPLT = -1 in your input) or to plot both the mesh and a number of phase change isotherm locations (IZDPLT greater than zero), you may run the program KOPLOT. The program resides on the CRREL system and is designed for it. It may also work for comparable systems, or for individual office setups, but that must be determined on a case-by-case basis.

To run the program KOPLOT on the CRREL Prime system, simply assign the HP plotter in the terminal room to yourself by entering the command

ASHP

After the system has reassured you ("OK"), then enter

KOPLOT

When that command has registered, the plotting program will begin. It will ask you to enter the name of the file in which your plotting data are stored, i.e., you will have to enter the name you supplied in response to the question asked by XYFREZ. KOPLOT is currently set up so that isotherms are labelled by time step, rather than by time, and the graphs are $8^1/2$ by 11 inches in size. An example is provided in the next section.

SAMPLE PROBLEM

This section presents a sample problem to demonstrate the input, execution, and graphics associated with XYFREZ. The example is a simple case for which there exists a known solution. The numerical solution of the problem using XYFREZ was originally presented in the reference by O'Neill (1983). A listing of the input data is provided in Appendix A.

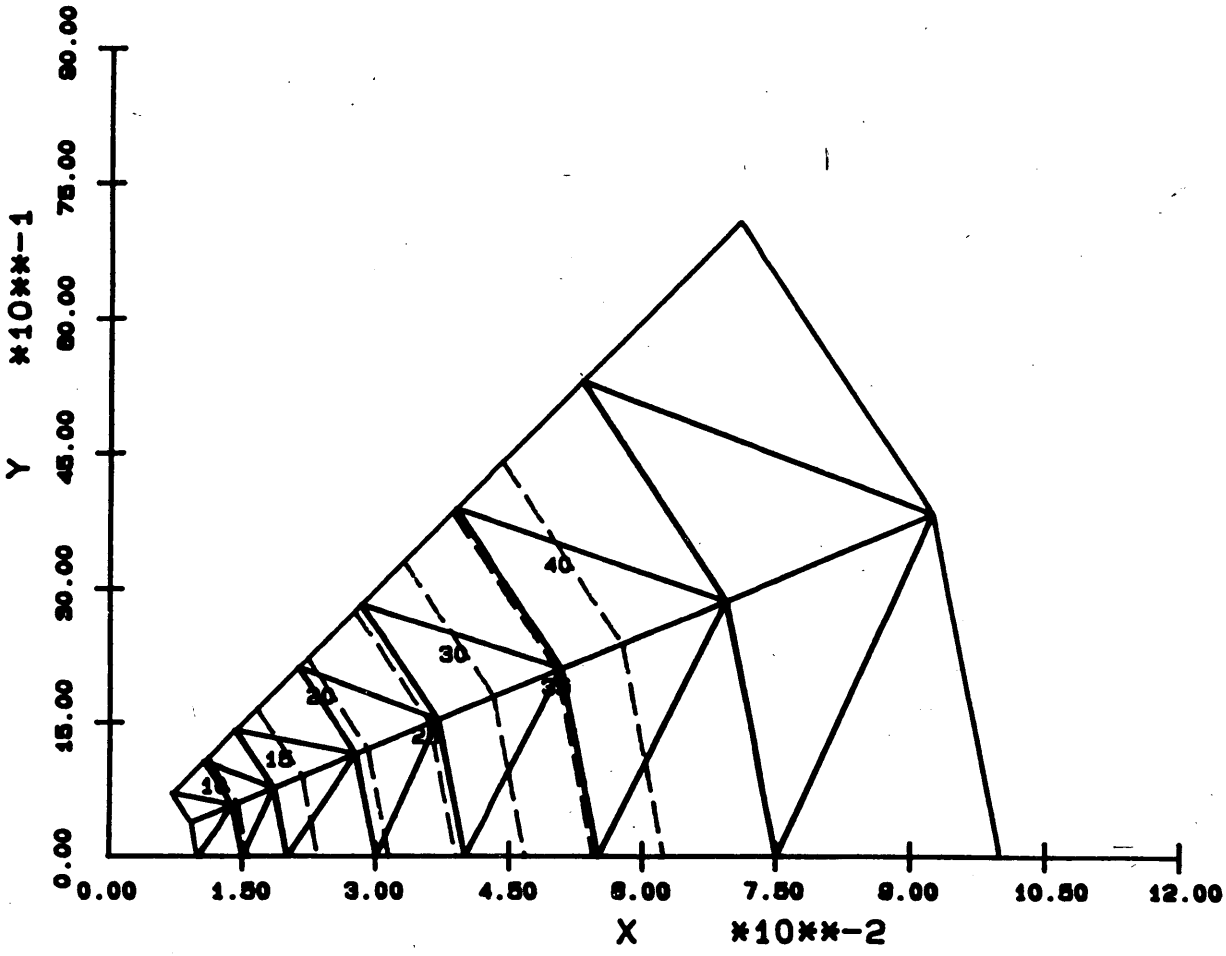


Figure 3. Wedge-shaped domain of the sample problem, divided into linear triangles as plotted by KOPLOT. Dashed lines show freezing front locations, with associated time step numbers.

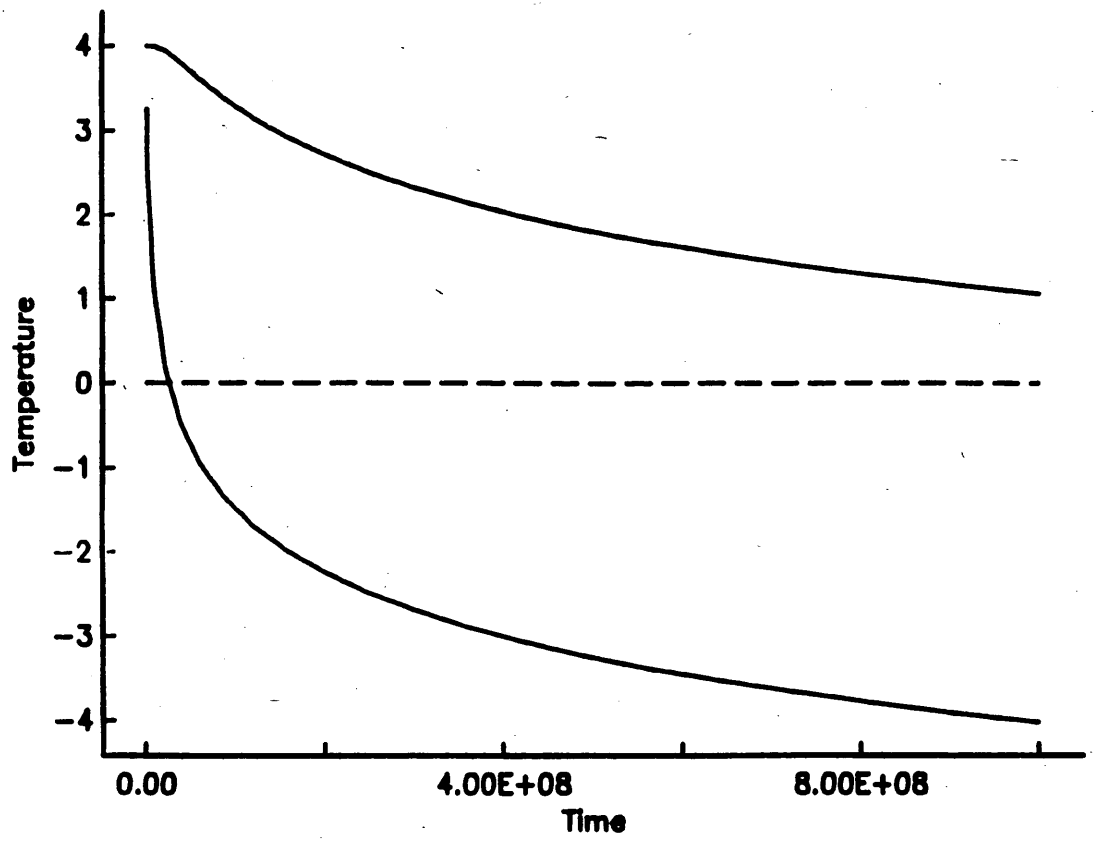


Figure 4. Cold and warm side boundary temperatures over time for the sample problem.

Graphical description of the boundary conditions, mesh geometry, and the location of the phase transition in time are contained in Figures 3 and 4. Note that the domain in question is the shape of a wedge, in which the straight boundaries are formed by radial lines of symmetry. The normal flux of heat is zero along those boundaries, and NDBC is set to zero there. The nodes on the left and right ends of the wedge have NDBC = 1, and their temperatures will be specified as part of the boundary condition input as per Figure 4. The domain is described in terms of two material types, so that one may see the input operations which that entails. Both materials are given the same characteristics, however, so that this corresponds to the original problem for homogeneous material, as reported by O'Neill (1983).

Note in the data that boundary temperatures are given below for nodes 1–3 and 22–24 every other time step. The program interpolates the times, time steps, and boundary values in between. The output listed below can be made more compact if one specifies zero for MSHPRT and KBVPRT. The computed results in Appendix B below correspond to those reported by O'Neill (1983) and, as may be seen in that paper, are quite accurate. Freezing front locations are shown on the mesh in Figure 3, as plotted by the KOPLOT routine, the use of which is described above.

It is recommended that the user experiment with XYFREZ using this sample data before proceeding to a real problem. If you have obtained this program from Kevin O'Neill, you probably also have this input data on tape or in a computer file. Try altering input temperatures or time steps, change the material properties of patches of elements, etc., and note the effects of doing so.

BLIND SAMPLE PROBLEM

Information is included in this section for a "blind" sample problem, to help the user gain experience in using this program. That is, data are given in somewhat the same form in which they might be available for a real problem, in terms of physical parameter values, length of time to be simulated, boundary conditions, etc. However, the user must decide on and specify the particulars of mesh, time step size, and other matters of judgment or individual strategy required to run the program. The problem posed is a simple one, with an exact solution, which is also provided.

The problem consists of one-dimensional freezing in a semi-infinite medium. The initial temperature is 4.0 degrees, the cold-side boundary temperature is constant at -10.0 degrees, and the freezing temperature is 0.0 degrees. Volumetric heat capacities in the frozen and unfrozen zones are 0.49 and 0.62, respectively. Frozen and unfrozen thermal conductivities are 9.6E-3 and 6.9E-3 respectively. The latent heat content is 17.68 per unit volume frozen material. These values correspond roughly to parameters which might characterize a moist soil, in cal-cgs-deg C. The exact solution to this problem is given by

$$Xf = B * \sqrt{t}$$

where t is time, Xf is the location of the freezing front relative to the cold side, and B is a constant equal to 8.8865E-2.

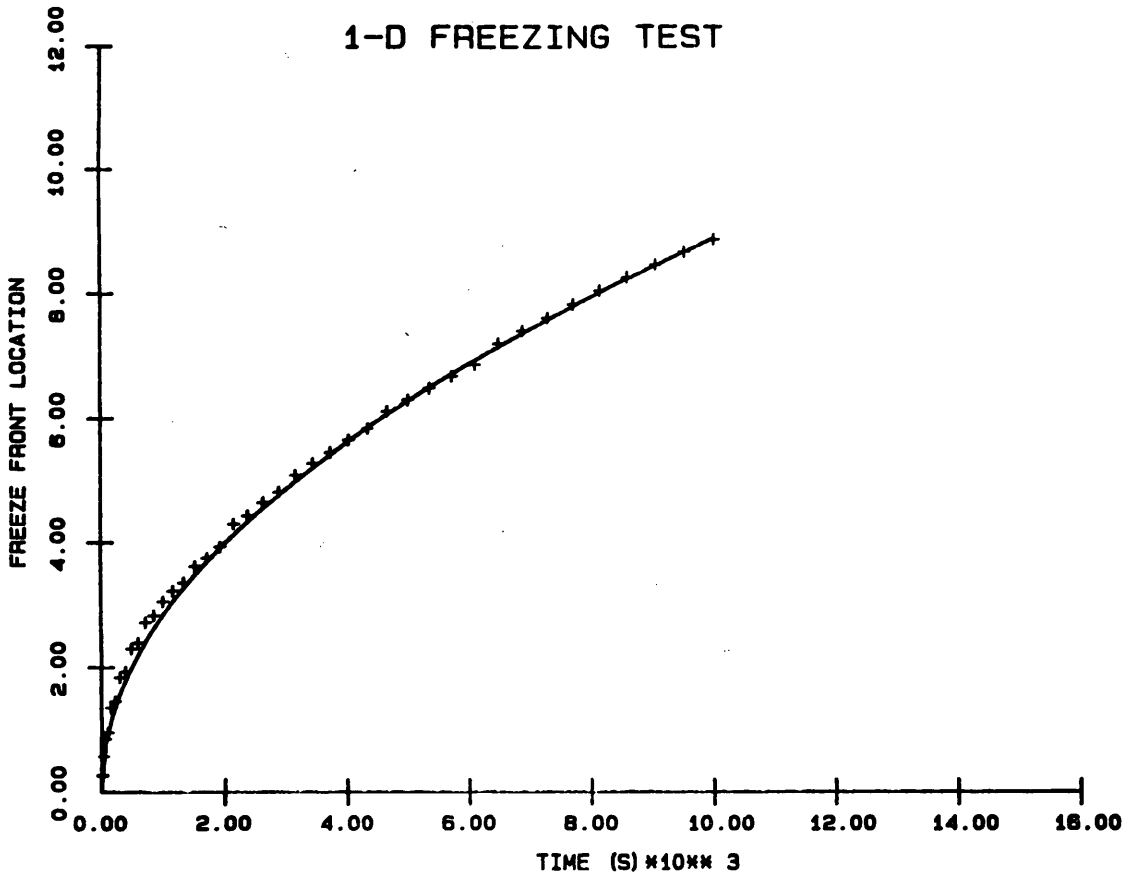


Figure 5. Computed points for freezing front location and continuous analytical solution for blind sample problem.

Figure 5 shows numerical locations of the freezing front versus the continuous analytical solution, over 40 time steps, up to a time of 1.0E4. Of course, your own answers will differ somewhat, depending on your choices for time step, mesh characteristics, and ITRLM value. You will have to obtain freezing front locations by interpolating between nodes on either side of the front, based on their temperatures.

LITERATURE CITED

Becker, E.B., G.F. Carey, and J.T. Oden (1981) *Finite Elements, An Introduction.* New York: Prentice-Hall.

O'Neill, K. (1983) Fixed Mesh Finite Element Solution For Cartesian Two-Dimensional Phase Change. *Journal of Energy Resources Technology*, **105**, 436–441.

Pinder, G.F. and W.G. Gray (1977) *Finite Element Simulation in Surface and Subsurface Hydrology*. New York: Academic Press.

APPENDIX A: SAMPLE PROBLEM INPUT DATA

```
24, 6, 28, 40, 2
6
      XYFREZ Sample Problem: Freezing of a Wedge
0, 10
5, 1, 2
1
   1   1   1.000E+02   0.000E -01
   2   1   9.239E+01   3.827E+01
   3   1   7.071E+01   7.071E+01
   4   0   1.500E+02   0.000E -01
   5   0   1.386E+02   5.740E+01
   6   0   1.061E+02   1.061E+02
   7   0   2.000E+02   0.000E -01
   8   0   1.848E+02   7.654E+01
   9   0   1.414E+02   1.414E+02
  10   0   3.000E+02   0.000E -01
  11   0   2.772E+02   1.148E+02
  12   0   2.121E+02   2.121E+02
  13   0   4.000E+02   0.000E -01
  14   0   3.696E+02   1.531E+02
  15   0   2.828E+02   2.828E+02
  16   0   5.500E+02   0.000E -01
  17   0   5.081E+02   2.105E+02
  18   0   3.889E+02   3.889E+02
  19   0   7.500E+02   0.000E -01
  20   0   6.929E+02   2.870E+02
  21   0   5.303E+02   5.303E+02
  22   1   1.000E+03   0.000E -01
  23   1   9.239E+02   3.827E+02
  24   1   7.071E+02   7.071E+02
   1   1    1   4   5
   2   1    2   1   5
   3   1    3   2   5
   4   1    3   5   6
   5   1    4   7   8
   6   1    5   4   8
   7   1    6   5   8
   8   1    6   8   9
   9   1    7  10  11
  10   1    8   7  11
  11   1    9   8  11
  12   1    9  11  12
  13   1   10  14  13
  14   1   11  10  14
  15   1   12  11  14
  16   1   12  14  15
  17   2   13  16  17
```

```
   18   2   14  13  17
   19   2   15  17  14
   20   2   15  17  18
   21   2   16  19  20
   22   2   17  16  20
   23   2   18  17  20
   24   2   18  20  21
   25   2   19  22  23
   26   2   20  19  23
   27   2   21  20  23
   28   2   21  23  24
0.5083, 0.7559, 7.2E-3, 4.2E-3, 36.0, 0.0
0.5083, 0.7559, 7.2E-3, 4.2E-3, 36.0, 0.0
1.0E9, 0.65, 0.65
0
4.0
2
 0.2500E+07   6  2
  1  2.501E+00
  2  2.501E+00
  3  2.501E+00
 22  4.000E+00
 23  4.000E+00
 24  4.000E+00
 0.1000E+08   6  2
  1  1.060E+00
  2  1.060E+00
  3  1.060E+00
 22  3.998E+00
 23  3.998E+00
 24  3.998E+00
 0.2250E+08   6  2
  1  1.605E-01
  2  1.605E-01
  3  1.605E-01
 22  3.944E+00
 23  3.944E+00
 24  3.944E+00
 0.4000E+08   6  2
  1 -4.754E-01
  2 -4.754E-01
  3 -4.754E-01
 22  3.796E+00
 23  3.796E+00
 24  3.796E+00
 0.6250E+08   6  2
  1 -9.669E-01
  2 -9.669E-01
  3 -9.669E-01
 22  3.588E+00
 23  3.588E+00
 24  3.588E+00
 0.9000E+08   6  2
  1 -1.369E+00
  2 -1.369E+00
  3 -1.369E+00
```

```
 22  3.359E+00
 23  3.359E+00
 24  3.359E+00
0.1225E+09   6  2
  1 -1.709E+00
  2 -1.709E+00
  3 -1.709E+00
 22  3.130E+00
 23  3.130E+00
 24  3.130E+00
0.1600E+09   6  2
  1 -2.004E+00
  2 -2.004E+00
  3 -2.004E+00
 22  2.908E+00
 23  2.908E+00
 24  2.908E+00
0.2025E+09   6  2
  1 -2.264E+00
  2 -2.264E+00
  3 -2.264E+00
 22  2.698E+00
 23  2.698E+00
 24  2.698E+00
0.2500E+09   6  2
  1 -2.497E+00
  2 -2.497E+00
  3 -2.497E+00
 22  2.501E+00
 23  2.501E+00
 24  2.501E+00
0.3025E+09   6  2
  1 -2.707E+00
  2 -2.707E+00
  3 -2.707E+00
 22  2.316E+00
 23  2.316E+00
 24  2.316E+00
0.3600E+09   6  2
  1 -2.899E+00
  2 -2.899E+00
  3 -2.899E+00
 22  2.142E+00
 23  2.142E+00
 24  2.142E+00
0.4225E+09   6  2
  1 -3.076E+00
  2 -3.076E+00
  3 -3.076E+00
 22  1.979E+00
 23  1.979E+00
 24  1.979E+00
0.4900E+09   6  2
  1 -3.240E+00
  2 -3.240E+00
  3 -3.240E+00
```

```
22  1.825E+00
23  1.825E+00
24  1.825E+00
0.5625E+09   6  2
 1 -3.393E+00
 2 -3.393E+00
 3 -3.393E+00
22  1.680E+00
23  1.680E+00
24  1.680E+00
0.6400E+09   6  2
 1 -3.535E+00
 2 -3.535E+00
 3 -3.535E+00
22  1.543E+00
23  1.543E+00
24  1.543E+00
0.7225E+09   6  2
 1 -3.669E+00
 2 -3.669E+00
 3 -3.669E+00
22  1.413E+00
23  1.413E+00
24  1.413E+00
0.8100E+09   6  2
 1 -3.795E+00
 2 -3.795E+00
 3 -3.795E+00
22  1.290E+00
23  1.290E+00
24  1.290E+00
0.9025E+09   6  2
 1 -3.915E+00
 2 -3.915E+00
 3 -3.915E+00
22  1.172E+00
23  1.172E+00
24  1.172E+00
0.1000E+10   6  2
 1 -4.028E+00
 2 -4.028E+00
 3 -4.028E+00
22  1.060E+00
23  1.060E+00
24  1.060E+00
```

APPENDIX B: SAMPLE PROBLEM OUTPUT

XYFREZ Sample Problem: Freezing of a Wedge

NBND	NMBCl	NELS	NTS	ITRLM	KPRTVL
24	6	28	40	0	10

IZDPLT	MSHPRT	KBVPRT
5	1	2

KPAUSE = 1

Node	BC	X	Y
1	1	1.00E+02	0.00E-01
2	1	9.24E+01	3.83E+01
3	1	7.07E+01	7.07E+01
4	0	1.50E+02	0.00E-01
5	0	1.39E+02	5.74E+01
6	0	1.06E+02	1.06E+02
7	0	2.00E+02	0.00E-01
8	0	1.85E+02	7.65E+01
9	0	1.41E+02	1.41E+02
10	0	3.00E+02	0.00E-01
11	0	2.77E+02	1.15E+02
12	0	2.12E+02	2.12E+02
13	0	4.00E+02	0.00E-01
14	0	3.70E+02	1.53E+02
15	0	2.83E+02	2.83E+02
16	0	5.50E+02	0.00E-01
17	0	5.08E+02	2.10E+02
18	0	3.89E+02	3.89E+02
19	0	7.50E+02	0.00E-01
20	0	6.93E+02	2.87E+02
21	0	5.30E+02	5.30E+02

22	1	1.00E+03	0.00E-01
23	1	9.24E+02	3.83E+02
24	1	7.07E+02	7.07E+02

Element	Material	Nodes			Element	Material	Nodes		
1	1	1	4	5	2	1	2	1	5
3	1	3	2	5	4	1	3	5	6
5	1	4	7	8	6	1	5	4	8
7	1	6	5	8	8	1	6	8	9
9	1	7	10	11	10	1	8	7	11
11	1	9	8	11	12	1	9	11	12
13	1	10	13	14	14	1	11	10	14
15	1	12	11	14	16	1	12	14	15
17	2	13	16	17	18	2	14	13	17
19	2	15	14	17	20	2	15	17	18
21	2	16	19	20	22	2	17	16	20
23	2	18	17	20	24	2	18	20	21
25	2	19	22	23	26	2	20	19	23
27	2	21	20	23	28	2	21	23	24

Thermal Properties of Material 1

HETF	HETU	CNDF	CNDU	EL	TF
5.08E-01	7.56E-01	7.20E-03	4.20E-03	3.60E+01	0.00E-01

Thermal Properties of Material 2

HETF	HETU	CNDF	CNDU	EL	TF
5.08E-01	7.56E-01	7.20E-03	4.20E-03	3.60E+01	0.00E-01

Solution Control Parameters

TIMLIM	AMP	THETA
1.000E+09	6.500E-01	6.500E-01

Initial Temperatures at Each Node

1	4.000E+00	2	4.000E+00	3	4.000E+00
4	4.000E+00	5	4.000E+00	6	4.000E+00
7	4.000E+00	8	4.000E+00	9	4.000E+00
10	4.000E+00	11	4.000E+00	12	4.000E+00
13	4.000E+00	14	4.000E+00	15	4.000E+00

16	4.000E+00	17	4.000E+00	18	4.000E+00
19	4.000E+00	20	4.000E+00	21	4.000E+00
22	4.000E+00	23	4.000E+00	24	4.000E+00

Time and DT at Each Time Step, with
Boundary Values at Each Node

TIME STEP = 1 TIME = 1.250E+06 DT = 1.250E+06

Nodes and their temperatures

1	3.250E+00	2	3.250E+00	3	3.250E+00
22	4.000E+00	23	4.000E+00	24	4.000E+00

TIME STEP = 2 TIME = 2.500E+06 DT = 1.250E+06

Nodes and their temperatures

1	2.501E+00	2	2.501E+00	3	2.501E+00
22	4.000E+00	23	4.000E+00	24	4.000E+00

TIME STEP = 3 TIME = 6.250E+06 DT = 3.750E+06

Nodes and their temperatures

1	1.780E+00	2	1.780E+00	3	1.780E+00
22	3.999E+00	23	3.999E+00	24	3.999E+00

TIME STEP = 4 TIME = 1.000E+07 DT = 3.750E+06

Nodes and their temperatures

1	1.060E+00	2	1.060E+00	3	1.060E+00
22	3.998E+00	23	3.998E+00	24	3.998E+00

TIME STEP = 5 TIME = 1.625E+07 DT = 6.250E+06

Nodes and their temperatures

1	6.102E-01	2	6.102E-01	3	6.102E-01

```
22   3.971E+00        23   3.971E+00        24   3.971E+00


 TIME STEP =   6     TIME =   2.250E+07     DT =   6.250E+06

           Nodes and their temperatures
 1   1.605E-01        2   1.605E-01        3   1.605E-01
22   3.944E+00       23   3.944E+00       24   3.944E+00


 TIME STEP =   7     TIME =   3.125E+07     DT =   8.750E+06

           Nodes and their temperatures
 1  -1.575E-01        2  -1.575E-01        3  -1.575E-01
22   3.870E+00       23   3.870E+00       24   3.870E+00


 TIME STEP =   8     TIME =   4.000E+07     DT =   8.750E+06

           Nodes and their temperatures
 1  -4.754E-01        2  -4.754E-01        3  -4.754E-01
22   3.796E+00       23   3.796E+00       24   3.796E+00


 TIME STEP =   9     TIME =   5.125E+07     DT =   1.125E+07

           Nodes and their temperatures
 1  -7.212E-01        2  -7.212E-01        3  -7.212E-01
22   3.692E+00       23   3.692E+00       24   3.692E+00


 TIME STEP =  10     TIME =   6.250E+07     DT =   1.125E+07

           Nodes and their temperatures
 1  -9.669E-01        2  -9.669E-01        3  -9.669E-01
22   3.588E+00       23   3.588E+00       24   3.588E+00


 TIME STEP =  11     TIME =   7.625E+07     DT =   1.375E+07

           Nodes and their temperatures
 1  -1.168E+00        2  -1.168E+00        3  -1.168E+00
```

```
22   3.473E+00        23   3.473E+00        24   3.473E+00


  TIME STEP =  12     TIME =   9.000E+07     DT =   1.375E+07

           Nodes and their temperatures
 1  -1.369E+00        2  -1.369E+00        3  -1.369E+00
22   3.359E+00       23   3.359E+00       24   3.359E+00


  TIME STEP =  13     TIME =   1.062E+08     DT =   1.625E+07

           Nodes and their temperatures
 1  -1.539E+00        2  -1.539E+00        3  -1.539E+00
22   3.245E+00       23   3.245E+00       24   3.245E+00


  TIME STEP =  14     TIME =   1.225E+08     DT =   1.625E+07

           Nodes and their temperatures
 1  -1.709E+00        2  -1.709E+00        3  -1.709E+00
22   3.130E+00       23   3.130E+00       24   3.130E+00


  TIME STEP =  15     TIME =   1.412E+08     DT =   1.875E+07

           Nodes and their temperatures
 1  -1.857E+00        2  -1.857E+00        3  -1.857E+00
22   3.019E+00       23   3.019E+00       24   3.019E+00


  TIME STEP =  16     TIME =   1.600E+08     DT =   1.875E+07

           Nodes and their temperatures
 1  -2.004E+00        2  -2.004E+00        3  -2.004E+00
22   2.908E+00       23   2.908E+00       24   2.908E+00


  TIME STEP =  17     TIME =   1.812E+08     DT =   2.125E+07

           Nodes and their temperatures
 1  -2.134E+00        2  -2.134E+00        3  -2.134E+00
```

```
22    2.803E+00          23    2.803E+00          24    2.803E+00


  TIME STEP =  18      TIME =    2.025E+08      DT =    2.125E+07

            Nodes and their temperatures
 1  -2.264E+00           2  -2.264E+00           3  -2.264E+00
22   2.698E+00          23   2.698E+00          24   2.698E+00


  TIME STEP =  19      TIME =    2.262E+08      DT =    2.375E+07

            Nodes and their temperatures
 1  -2.381E+00           2  -2.381E+00           3  -2.381E+00
22   2.599E+00          23   2.599E+00          24   2.599E+00


  TIME STEP =  20      TIME =    2.500E+08      DT =    2.375E+07

            Nodes and their temperatures
 1  -2.497E+00           2  -2.497E+00           3  -2.497E+00
22   2.501E+00          23   2.501E+00          24   2.501E+00


  TIME STEP =  21      TIME =    2.762E+08      DT =    2.625E+07

            Nodes and their temperatures
 1  -2.602E+00           2  -2.602E+00           3  -2.602E+00
22   2.408E+00          23   2.408E+00          24   2.408E+00


  TIME STEP =  22      TIME =    3.025E+08      DT =    2.625E+07

            Nodes and their temperatures
 1  -2.707E+00           2  -2.707E+00           3  -2.707E+00
22   2.316E+00          23   2.316E+00          24   2.316E+00


  TIME STEP =  23      TIME =    3.312E+08      DT =    2.875E+07

            Nodes and their temperatures
 1  -2.803E+00           2  -2.803E+00           3  -2.803E+00
```

```
22   2.229E+00        23   2.229E+00        24   2.229E+00


  TIME STEP =  24     TIME =   3.600E+08     DT =   2.875E+07

           Nodes and their temperatures
 1  -2.899E+00         2  -2.899E+00         3  -2.899E+00
22   2.142E+00        23   2.142E+00        24   2.142E+00


  TIME STEP =  25     TIME =   3.912E+08     DT =   3.125E+07

           Nodes and their temperatures
 1  -2.988E+00         2  -2.988E+00         3  -2.988E+00
22   2.061E+00        23   2.061E+00        24   2.061E+00


  TIME STEP =  26     TIME =   4.225E+08     DT =   3.125E+07

           Nodes and their temperatures
 1  -3.076E+00         2  -3.076E+00         3  -3.076E+00
22   1.979E+00        23   1.979E+00        24   1.979E+00


  TIME STEP =  27     TIME =   4.562E+08     DT =   3.375E+07

           Nodes and their temperatures
 1  -3.158E+00         2  -3.158E+00         3  -3.158E+00
22   1.902E+00        23   1.902E+00        24   1.902E+00


  TIME STEP =  28     TIME =   4.900E+08     DT =   3.375E+07

           Nodes and their temperatures
 1  -3.240E+00         2  -3.240E+00         3  -3.240E+00
22   1.825E+00        23   1.825E+00        24   1.825E+00


  TIME STEP =  29     TIME =   5.262E+08     DT =   3.625E+07

           Nodes and their temperatures
 1  -3.317E+00         2  -3.317E+00         3  -3.317E+00
```

```
22   1.753E+00        23   1.753E+00        24   1.753E+00


  TIME STEP =  30     TIME =   5.625E+08     DT =   3.625E+07

           Nodes and their temperatures
 1  -3.393E+00        2  -3.393E+00        3  -3.393E+00
22   1.680E+00       23   1.680E+00       24   1.680E+00


  TIME STEP =  31     TIME =   6.012E+08     DT =   3.875E+07

           Nodes and their temperatures
 1  -3.464E+00        2  -3.464E+00        3  -3.464E+00
22   1.612E+00       23   1.612E+00       24   1.612E+00


  TIME STEP =  32     TIME =   6.400E+08     DT =   3.875E+07

           Nodes and their temperatures
 1  -3.535E+00        2  -3.535E+00        3  -3.535E+00
22   1.543E+00       23   1.543E+00       24   1.543E+00


  TIME STEP =  33     TIME =   6.812E+08     DT =   4.125E+07

           Nodes and their temperatures
 1  -3.602E+00        2  -3.602E+00        3  -3.602E+00
22   1.478E+00       23   1.478E+00       24   1.478E+00


  TIME STEP =  34     TIME =   7.225E+08     DT =   4.125E+07

           Nodes and their temperatures
 1  -3.669E+00        2  -3.669E+00        3  -3.669E+00
22   1.413E+00       23   1.413E+00       24   1.413E+00


  TIME STEP =  35     TIME =   7.662E+08     DT =   4.375E+07

           Nodes and their temperatures
 1  -3.732E+00        2  -3.732E+00        3  -3.732E+00
```

```
22    1.352E+00          23    1.352E+00          24    1.352E+00


  TIME STEP =  36      TIME =    8.100E+08      DT =    4.375E+07

            Nodes and their temperatures
 1  -3.795E+00           2  -3.795E+00           3  -3.795E+00
22   1.290E+00          23   1.290E+00          24   1.290E+00


  TIME STEP =  37      TIME =    8.562E+08      DT =    4.625E+07

            Nodes and their temperatures
 1  -3.855E+00           2  -3.855E+00           3  -3.855E+00
22   1.231E+00          23   1.231E+00          24   1.231E+00


  TIME STEP =  38      TIME =    9.025E+08      DT =    4.625E+07

            Nodes and their temperatures
 1  -3.915E+00           2  -3.915E+00           3  -3.915E+00
22   1.172E+00          23   1.172E+00          24   1.172E+00


  TIME STEP =  39      TIME =    9.512E+08      DT =    4.875E+07

            Nodes and their temperatures
 1  -3.971E+00           2  -3.971E+00           3  -3.971E+00
22   1.116E+00          23   1.116E+00          24   1.116E+00


  TIME STEP =  40      TIME =    1.000E+09      DT =    4.875E+07

            Nodes and their temperatures
 1  -4.028E+00           2  -4.028E+00           3  -4.028E+00
22   1.060E+00          23   1.060E+00          24   1.060E+00
```

Calculated Temperature Solution

Plotting data stored for TIME = 1.625E+07

TIME = 6.250E+07 TIME STEP = 10

NODE	TEMP	NODE	TEMP	NODE	TEMP
1	-9.669E-01	2	-9.669E-01	3	-9.669E-01
4	-8.441E-02	5	-8.567E-02	6	-8.364E-02
7	5.637E-01	8	5.616E-01	9	5.634E-01
10	1.450E+00	11	1.448E+00	12	1.450E+00
13	2.058E+00	14	2.053E+00	15	2.058E+00
16	2.681E+00	17	2.674E+00	18	2.681E+00
19	3.207E+00	20	3.200E+00	21	3.207E+00
22	3.588E+00	23	3.588E+00	24	3.588E+00

Plotting data stored for TIME = 6.250E+07

Plotting data stored for TIME = 1.412E+08

TIME = 2.500E+08 TIME STEP = 20

NODE	TEMP	NODE	TEMP	NODE	TEMP
1	-2.497E+00	2	-2.497E+00	3	-2.497E+00
4	-1.612E+00	5	-1.612E+00	6	-1.611E+00
7	-9.806E-01	8	-9.813E-01	9	-9.809E-01
10	-1.017E-01	11	-1.025E-01	12	-1.021E-01
13	5.515E-01	14	5.500E-01	15	5.511E-01
16	1.259E+00	17	1.257E+00	18	1.259E+00
19	1.924E+00	20	1.921E+00	21	1.924E+00
22	2.501E+00	23	2.501E+00	24	2.501E+00

Plotting data stored for TIME = 2.500E+08

Plotting data stored for TIME = 3.912E+08

TIME = 5.625E+08 TIME STEP = 30

NODE	TEMP	NODE	TEMP	NODE	TEMP
1	-3.393E+00	2	-3.393E+00	3	-3.393E+00
4	-2.498E+00	5	-2.498E+00	6	-2.497E+00
7	-1.859E+00	8	-1.859E+00	9	-1.860E+00
10	-9.658E-01	11	-9.658E-01	12	-9.662E-01
13	-3.284E-01	14	-3.289E-01	15	-3.287E-01
16	3.907E-01	17	3.894E-01	18	3.906E-01
19	1.068E+00	20	1.067E+00	21	1.068E+00
22	1.680E+00	23	1.680E+00	24	1.680E+00

Plotting data stored for TIME = 5.625E+08

Plotting data stored for TIME = 7.662E+08

TIME = 1.000E+09 TIME STEP = 40

NODE	TEMP	NODE	TEMP	NODE	TEMP
1	-4.028E+00	2	-4.028E+00	3	-4.028E+00
4	-3.134E+00	5	-3.134E+00	6	-3.133E+00
7	-2.495E+00	8	-2.495E+00	9	-2.495E+00
10	-1.601E+00	11	-1.601E+00	12	-1.602E+00
13	-9.636E-01	14	-9.638E-01	15	-9.639E-01
16	-2.594E-01	17	-2.615E-01	18	-2.594E-01
19	4.356E-01	20	4.357E-01	21	4.355E-01
22	1.060E+00	23	1.060E+00	24	1.060E+00

Plotting data stored for TIME = 1.000E+09

APPENDIX C: LISTING OF THE PROGRAM

```
00001:C          PROGRAM XYFREZ.4.PUBLIC3
00002:C
00003:C
00004:C    This program solves the 2-D heat equation, including phase change
00005:C     effects through the heat capacity.
00006:C
00007:C          Main Program
00008:C
00009:C
00010:C    This routine obtains input for determining array dimensions.
00011:C      The "real" main program follows in SUBROUTINE MAIN
00012:C
00013:       COMMON  NDBC(2000), XG(2000), YG(2000),
00014:      &    R(2000), T(2000), TOLD(2000), TLAST(2000),NGLNBC(2000),
00015:      &    ANS(2000), MTYPE(4000), NBCFLG(1000), NBCNGL(1000),
00016:      &    NODEL(20000),DT(2000),HETU(10),HETF(10),
00017:      &    CNDU(10),CNDF(10),TF(10),EL(10), A(200000),TBC(2000000)
00018:C
00019:C
00020:       CHARACTER*32 DATFIL
00021:C
00022:C
00023:       WRITE(1,701)
00024:   701 FORMAT(///////,5X,' WELCOME TO XYFREZ,  VERSION 4',/////,
00025:      &    'Enter name of data file')
00026:       READ(1,7777) DATFIL
00027:  7777 FORMAT(A)
00028:       OPEN(5,FILE=DATFIL)
00029:C
00030:C
00031:C
00032:       READ(5,*) NBND, NBC1IN, NELS, NTS, MEDIA
00033:C
00034:C
00035:C
00036:       IF(NBND.LE.2000) GO TO 110
00037:       WRITE(1,7001) NBND
00038:  7001 FORMAT(/////,'NBND value of',I10,'is greater than 2000!')
00039:       GO TO 1000
00040:   110 IF(NBC1IN.LE.1000) GO TO 120
00041:       WRITE(1,7002) NBC1IN
00042:  7002 FORMAT(/////,'NMBC1 value of',I10,'is greater than 1000!')
00043:       GO TO 1000
00044:   120 IF(NELS.LE.4000) GO TO 130
00045:       WRITE(1,7003) NELS
00046:  7003 FORMAT(/////,'NELS value of',I10,'is greater than 4000!')
00047:       GO TO 1000
00048:   130 IF(NTS.LE.2000) GO TO 140
00049:       WRITE(1,7004) NTS
00050:  7004 FORMAT(/////,'NTS value of',I10,'is greater than 2000!')
00051:       GO TO 1000
00052:   140 IF(MEDIA.LE.10) GO TO 150
00053:       WRITE(1,7005) MEDIA
00054:  7005 FORMAT(/////,'MEDIA value of',I10,'is greater than 10!')
```

```
00055: 1000 WRITE(1,7999)
00056: 7999 FORMAT(/////////,' EXECUTION HALTED')
00057:      CLOSE(5)
00058:      STOP
00059:C
00060:C
00061:  150 CONTINUE
00062:      NTSP = NTS + 1
00063:C
00064:C    The following is a dummy value of NBW, for use in the
00065:C      dimension statement in SUBROUTINE MAIN
00066:C
00067:      NBW = 1
00068:C
00069:      CALL MAIN(A, NBND, NBW, NDBC, XG, YG, R, T, TOLD,TLAST,
00070:     &    NGLNBC, ANS, MTYPE, NBCFLG, NBC1IN, NBCNGL, DT,
00071:     &    NTSP, TBC, NODEL, NELS, MEDIA)
00072:C
00073:C
00074:      STOP
00075:      END
00076:C
00077:C
00078:      SUBROUTINE MAIN(A, NBND, NBW, NDBC, XG, YG, R, T, TOLD,
00079:     &  TLAST,NGLNBC, ANS, MTYPE, NBCFLG, NBC1IN, NBCNGL, DT, NTSP,
00080:     &    TBC, NODEL, NELS, MEDIA)
00081:C
00082:C
00083:C
00084:      DIMENSION A(NBND,NBW), NDBC(NBND), XG(NBND), YG(NBND)
00085:      DIMENSION R(NBND), T(NBND), TOLD(NBND), TLAST(NBND)
00086:      DIMENSION NGLNBC(NBND), ANS(NBND), MTYPE(NELS), NBCFLG(NBC1IN)
00087:      DIMENSION NBCNGL(NBC1IN),DT(NTSP),TBC(NTSP,NBC1IN),NODEL(NELS,3)
00088:      DIMENSION TITLE(20), TF(10), EL(10)
00089:      DIMENSION HETU(10),HETF(10),CNDU(10),CNDF(10)
00090:C
00091:C
00092:      CHARACTER*32 OUTFIL, PLTFIL
00093:C
00094:      NTS = NTSP -1
00095:C
00096:C
00097:C
00098:C        INPUT
00099:C
00100:      READ(5,*) IO
00101:      IF(IO.EQ.1) GO TO 12
00102:      WRITE(1,773)
00103:  773 FORMAT(///,' Enter file name for general output',///)
00104:      READ(1,7777) OUTFIL
00105: 7777 FORMAT(A)
00106:      OPEN(IO,FILE=OUTFIL)
00107:   12 CONTINUE
00108:C
```

```
00109:       READ(5,1791) (TITLE(K),K=1,20)
00110: 1791   FORMAT(20A4)
00111:       WRITE(IO,791) (TITLE(K),K=1,20)
00112:  791   FORMAT(/////20A4//////////////)
00113:C
00114:       READ(5, *)  ITRLM,  KPRTVL
00115:       WRITE(IO,707) NBND, NBC1IN, NELS, NTS, ITRLM, KPRTVL
00116:  707 FORMAT(/,10X,'NBND',5X,'NMBC1',6X,'NELS',7X,'NTS',
00117:     &    5X,'ITRLM',4X,'KPRTVL',/,4X,6I10)
00118:C
00119:       READ(5,*) IZDPLT,MSHPRT,KBVPRT
00120:       WRITE(IO,719) IZDPLT,MSHPRT,KBVPRT
00121:  719   FORMAT(//,21X,'IZDPLT',6X,'MSHPRT',6X,
00122:     &    'KBVPRT',/,15X,4I12)
00123:C
00124:       IF(IZDPLT.EQ.0) GO TO 112
00125:       WRITE(1,774)
00126:  774 FORMAT(////,' Enter file name for plotting output',///)
00127:       READ(1,7777) PLTFIL
00128:       OPEN(33,FILE=PLTFIL)
00129:  112 CONTINUE
00130:C
00131:        WRITE(1,8001)
00132:8001    FORMAT(///////,20X,' Preparing Finite Element Mesh...',/////)
00133:       READ(5,*) KPAUSE
00134:       WRITE(IO,709) KPAUSE
00135:  709 FORMAT(//,30X,'KPAUSE =',I2)
00136:C
00137:C
00138:C The next loop reads each node, the type bc at the node
00139:C and the coordinates of the node.  Also sets bc flags.
00140:C
00141:       NMBC1 = 0
00142:       KBC2 = 0
00143:       DO 100 NODE = 1,NBND
00144:          READ(5,*) ND,NDBCV,XGV,YGV
00145:          IF(NDBCV.EQ.2) KBC2 = 1
00146:          NGLNBC(ND) = 0
00147:          IF(NDBCV.NE.1) GO TO 95
00148:          NMBC1 = NMBC1 + 1
00149:          NGLNBC(ND) = NMBC1
00150:          NBCNGL(NMBC1) = ND
00151:    95    NDBC(ND) = NDBCV
00152:          XG(ND) = XGV
00153:          YG(ND) = YGV
00154:  100 CONTINUE
00155:C
00156:       IF(NMBC1.EQ.NBC1IN) GO TO 111
00157:       WRITE(1,7770) NBC1IN, NMBC1
00158: 7770 FORMAT(///,10X,'NMBC1 value of',I6,' input is different from'
00159:     &    /,10X,' actual value of',I10,
00160:     &    ///////,20X,'EXECUTION HALTED!')
00161:       CLOSE(5)
00162:       CLOSE(IO)
```

```
00163:       CLOSE(33)
00164:       STOP
00165:   111 CONTINUE
00166:C
00167:C
00168:       IF(MSHPRT.EQ.1)
00169:      & WRITE(IO,701) (K,NDBC(K),XG(K),YG(K),K=1,NBND)
00170:   701 FORMAT(//,(12X,'Node',4X,' BC ',12X,'X',
00171:      &     10X,'Y'),/,(9X,2I7,3X,1P2E11.2) )
00172:C
00173:C The following loop reads the connectivity of the nodes
00174:C and determines the material type of the element
00175:C
00176:       DO 110 NLMT = 1,NELS
00177:          READ(5,*) NL,LTYPE,N1,N2,N3
00178:          MTYPE(NLMT) = LTYPE
00179:          NODEL(NL,1) = N1
00180:          NODEL(NL,2) = N2
00181:          NODEL(NL,3) = N3
00182:   110 CONTINUE
00183:C
00184:C
00185:C      Check for flipped element: is X1*Y2 - Y1*X2 .LT. 0
00186:C
00187:C           Also determine NDG and NBW
00188:C
00189:       NNDG = 0
00190:       DO 250 L = 1,NELS
00191:       KNTERR = 0
00192:   249    N1 = NODEL(L,1)
00193:          N2 = NODEL(L,2)
00194:          N3 = NODEL(L,3)
00195:            NLBC2 = NDBC(N1) + NDBC(N2) + NDBC(N3)
00196:            IF(NLBC2.NE.6) GO TO 2249
00197:            WRITE(IO,7249) L
00198: 7249       FORMAT(////,' Element no',I5,' has three type 2 BCs',//,
00199:      &    'FIX IT!')
00200:            GO TO 1111
00201: 2249       NDIF = IABS(N2-N1)
00202:            IF(NDIF.GT.NNDG) NNDG = NDIF
00203:            NDIF = IABS(N3-N2)
00204:            IF(NDIF.GT.NNDG) NNDG = NDIF
00205:            NDIF = IABS(N1-N3)
00206:            IF(NDIF.GT.NNDG) NNDG = NDIF
00207:          X1 = XG(N2) - XG(N1)
00208:          X2 = XG(N3) - XG(N1)
00209:          Y1 = YG(N2) - YG(N1)
00210:          Y2 = YG(N3) - YG(N1)
00211:          PROD = X1*Y2 - Y1*X2
00212:          IF(PROD.GT.0.0) GO TO 250
00213:            KNTERR = KNTERR + 1
00214:            IF(KNTERR.GT.1) THEN
00215:               WRITE(IO,7111) L
00216: 7111          FORMAT(////,' Incurable mesh problem in element no',I6)
```

```
00217:                GO TO 1111
00218:       ELSE
00219:       ENDIF
00220:          NHOLD = NODEL(L,3)
00221:          NODEL(L,3) = NODEL(L,2)
00222:          NODEL(L,2) = NHOLD
00223:       GO TO 249
00224:   250 CONTINUE
00225:C
00226:C Arguments to be passed for later calculations involving the matrix A,
00227:C e.g. NBW is the bandwidth of the banded matrix.
00228:C
00229:       NBW = 2*NNDG + 1
00230:       NDG = NNDG + 1
00231:C
00232:C
00233:C
00234:       IF(MSHPRT.NE.1) GO TO 1250
00235:       WRITE(IO,702) (K,MTYPE(K),(NODEL(K,J),J=1,3),K=1,NELS)
00236:   702 FORMAT(//,2(6X,'Element',3X,'Material',6X,' Nodes',2X),/,
00237:      &    2(9X,I4,5X,I2,6X,3I4) )
00238:C
00239:  1250 CONTINUE
00240:C
00241:C    Write mesh plotting info on disk
00242:C
00243:       IF(IZDPLT.EQ.0) GO TO 1108
00244:       WRITE(33,7010) NELS
00245:  7010 FORMAT(I7)
00246:       DO 107 NL = 1,NELS
00247:          WRITE(33,7011) NL, MTYPE(NL), (NODEL(NL,J),J=1,3)
00248:  7011    FORMAT(5I7)
00249:   107 CONTINUE
00250:       WRITE(33,7010) NBND
00251:       DO 108 ND = 1,NBND
00252:          WRITE(33,7012) ND, NDBC(ND), XG(ND), YG(ND)
00253:  7012    FORMAT(2I7,1P2E14.5)
00254:   108 CONTINUE
00255:  1108 CONTINUE
00256:C
00257:C Input thermal properties
00258:C
00259:       DO 1109 I=1,MEDIA
00260:       READ(5,*) HETF(I),HETU(I),CNDF(I),CNDU(I),EL(I),TF(I)
00261:       WRITE(IO,705) I,HETF(I),HETU(I),CNDF(I),CNDU(I),EL(I),TF(I)
00262:  705  FORMAT(///,15X,'Thermal Properties of Material',I3,//,9X,
00263:      &    'HETF',6X,'HETU',6X,'CNDF',6X,'CNDU',8X,'EL',8X,'TF',
00264:      &    3X,/,(3X,1P6E10.2),2X)
00265:1109  CONTINUE
00266:C
00267:C
00268:C    THETA and AMP control implicitness in time stepping and iteration
00269:C
00270:       READ(5,*) TIMLIM, AMP, THETA
```

```
00271:       WRITE(IO,706)  TIMLIM, AMP, THETA
00272:   706 FORMAT(////,15X,'Solution Control Parameters',//,21X,
00273:      &    'TIMLIM',9X,'AMP',7X,'THETA',/,15X,1P3E12.3)
00274:C
00275:C           Determine IC'S
00276:C
00277:       READ(5,*) KIC
00278:       IF(KIC.EQ.1) GO TO 60
00279:       READ(5,*) TEMPIC
00280:C
00281:C  Initialize TOLD
00282:C
00283:       DO 50 ND = 1,NBND
00284:          TOLD(ND) = TEMPIC
00285:    50 CONTINUE
00286:C
00287:       GO TO 99
00288:    60 DO 70 ND = 1,NBND
00289:          READ(5,*) NODE, TEMPIC
00290:          TOLD(NODE) = TEMPIC
00291:    70 CONTINUE
00292:C
00293:    99 CONTINUE
00294:       WRITE(IO,733) (K,TOLD(K),K=1,NBND)
00295:   733  FORMAT(/////,15X,'Initial Temperatures at Each Node',//,
00296:      &    3(I10,1P1E12.3) )
00297:C
00298:C
00299:C
00300:C
00301:C    Set BC'S and time step values
00302:C
00303:       READ(5,*) KBCT
00304:C
00305:C    For KBCT = 1, bc values are constant over time, with
00306:C           KDT = 1 for uniform time steps, or
00307:C           KDT = 2 for uniform steps in SQRT(TIME)
00308:C
00309:C    for KBCT = 2 bc values over time and time steps are input
00310:C
00311:       IF(KBCT.EQ.2) GO TO 360
00312:C
00313:C    Constant BC's, initialize the RHS of the nonlinear system at bc's
00314:C
00315:       DO 301 ND = 1,NMBC1
00316:          READ(5,*) NODE, TBCVAL
00317:          R(NODE) = TBCVAL
00318:   301 CONTINUE
00319:C
00320:       READ(5,*) KDT
00321:       IF(KDT.EQ.2) GO TO 310
00322:C
00323:C    Uniform time steps, calculate DT(nts)
00324:C
```

```
00325:       DTCNST = TIMLIM/NTS
00326:       DO 305 IDT = 1,NTS
00327:   305 DT(IDT) = DTCNST
00328:       GO TO 399
00329:C
00330:C    Uniform steps in SQRT(TIME), calculate DT(nts)
00331:C
00332:   310 DTSQT = SQRT(TIMLIM)/NTS
00333:       DT(1) = DTSQT**2
00334:       TIM2 = DT(1)
00335:       DO 320 IDT = 2,NTS
00336:          TIM1 = TIM2
00337:          TSQT = SQRT(TIM2)
00338:          TNWSQT = TSQT + DTSQT
00339:          TIM2 = TNWSQT**2
00340:          DT(IDT) =  TIM2 - TIM1
00341:   320 CONTINUE
00342:       GO TO 399
00343:C
00344:C
00345:C    Read in bc values and associated times, with number of intervening
00346:C       time steps.  Set flags and interpolate between nodes.
00347:C
00348:   360 CONTINUE
00349:       IT = 0
00350:       TIMOLD = 0.0
00351:C
00352:       DO 390 IBV = 1,10000
00353:          READ(5,*) TIMBD, NUMCHG, NTSBD
00354:          DTIM = (TIMBD-TIMOLD)/NTSBD
00355:          TIMOLD = TIMBD
00356:          DO 365 ND = 1,NMBC1
00357:   365    NBCFLG(ND) = -1
00358:          N1 = 1
00359:C
00360:          DO 380 ND = 1,NMBC1
00361:            IF(ND.GT.NUMCHG) GO TO 366
00362:            READ(5,*) NODE, TBCVAL
00363:            TBCVLP = TOLD(NODE)
00364:            NM = NGLNBC(NODE)
00365:            IF(IT.NE.0) TBCVLP = TBC(IT,NM)
00366:            DTMP = (TBCVAL-TBCVLP)/NTSBD
00367:            GO TO 377
00368:   366      CONTINUE
00369:            DO 370 NM = N1,NMBC1
00370:               IF(NBCFLG(NM).LT.0) GO TO 371
00371:   370      CONTINUE
00372:   371      N1 = NM+1
00373:            NODE = NBCNGL(NM)
00374:            IF(IT.EQ.0) GO TO 374
00375:            TBCVLP = TBC(IT,NM)
00376:            ITM = IT-1
00377:            IF(IT.EQ.1) TBCVPP = TOLD(NODE)
00378:            IF(IT.GT.1) TBCVPP = TBC(ITM,NM)
```

```
00379:            DTMP = (TBCVLP-TBCVPP) * DTIM/DT(IT)
00380:            GO TO 377
00381:  374       TBCVLP = TOLD(NODE)
00382:            DTMP = 0.0
00383:  377       NBCFLG(NM) = 1
00384:            DO 378 ITB = 1,NTSBD
00385:              ITC = IT + ITB
00386:              DT(ITC) = DTIM
00387:              TBC(ITC,NM) = TBCVLP + ITB*DTMP
00388:  378       CONTINUE
00389:  380 CONTINUE
00390:C
00391:         IT = IT + NTSBD
00392:         IF(IT.GE.NTS) GO TO 399
00393:  390 CONTINUE
00394:C
00395:  399 CONTINUE
00396:C
00397:C
00398:       IF(KBVPRT.EQ.2) GO TO 440
00399:       IF(KBVPRT.NE.1) GO TO 499
00400:       WRITE(IO,741)
00401:  741 FORMAT(////,15X,'Time and DT values at Each Time Step',//,
00402:      &    11X,'TIME STEP',8X,'TIME',10X,'DT')
00403:       TIMCK = 0.0
00404:       DO 401 IT = 1,NTS
00405:          TIMCK = TIMCK + DT(IT)
00406:          WRITE(IO,742) IT,TIMCK,DT(IT)
00407:  742     FORMAT(I20,1P2E12.3)
00408:  401 CONTINUE
00409:       GO TO 499
00410:C
00411:  440 CONTINUE
00412:       WRITE(IO,1743)
00413: 1743  FORMAT(/////,15X,'Time and DT at Each Time Step, with ',/,
00414:      &    20X,'Boundary Values at Each Node')
00415:       TIMCK = 0.0
00416:       DO 460 IT = 1,NTS
00417:          TIMCK = TIMCK + DT(IT)
00418:          WRITE(IO,743) IT,TIMCK,DT(IT)
00419:  743     FORMAT(////,10X,'TIME STEP =',I4,5X,'TIME =',1P1E12.3,5X,
00420:      &    'DT =',1P1E12.3)
00421:          WRITE(IO,745)
00422:  745     FORMAT(/,20X,'Nodes and their temperatures')
00423:          IF(KBCT.EQ.2)
00424:      &    WRITE(IO,744) (NBCNGL(J),TBC(IT,J),J=1,NMBC1)
00425:  744     FORMAT(3(I10,1P1E12.3))
00426:          IF(KBCT.EQ.1)
00427:      &    WRITE(IO,744) (NBCNGL(J),R(NBCNGL(J)),J=1,NMBC1)
00428:  460 CONTINUE
00429:C
00430:  499 CONTINUE
00431:C
00432:C
```

```
00433:C     Input convective BC stuff
00434:C
00435:      IF(KBC2.EQ.0) GO TO 44
00436:      READ(5,*) HH, TA
00437:      WRITE(IO,7744) HH, TA
00438: 7744 FORMAT(//,20X,'HH =',1P1E12.3,4X,'TA =',1P1E12.3)
00439:   44 CONTINUE
00440:C
00441:      IF(KPAUSE.EQ.0) GO TO 998
00442:      WRITE(1,7998)
00443: 7998 FORMAT(///////4X,'OK to proceed ( 1 = YES)')
00444:      READ(1,*) KOK
00445:       WRITE(1,8000)
00446:8000   FORMAT(//////,20X,'Computing Finite Element Solution...',/////)
00447:      IF(KOK.EQ.1) GO TO 998
00448:      WRITE(1,7999)
00449: 7999 FORMAT(//////////,10X,'User mandated STOP!')
00450:C
00451:C          Close all files and stop if user has so commanded
00452:C
00453:      CLOSE(5)
00454:      CLOSE(IO)
00455:      CLOSE(33)
00456:C
00457:      STOP
00458:C
00459:  998 CONTINUE
00460:C
00461:C
00462:C    Initialize time loop parameters
00463:C
00464:      TIME = 0.0
00465:      THETAM = THETA - 1.0
00466:      AMPM = 1.0 - AMP
00467:      INEW = 0
00468:      ITRLMP = ITRLM+1
00469:      KTPRNT = 0
00470:      KTPLOT = -1
00471:C
00472:C Adjust dt and tbc
00473:C
00474:      NTSP = NTS + 1
00475:      DT(NTSP) = DT(NTS)
00476:      DO 999 NM=1,NMBC1
00477:  999 TBC(NTSP,NM) = TBC(NTS,NM)
00478:C
00479:      IF(IZDPLT.NE.0) WRITE(33,7099) IZDPLT
00480:      IF(IZDPLT.EQ.0) IZDPLT = -1
00481: 7099 FORMAT(I5)
00482:C
00483:C          Initialize ANS, TLAST
00484:C
00485:      DO 989 ND = 1,NBND
00486:         ANS(ND) = TOLD(ND)
```

```
00487:          TLAST(ND) = TOLD(ND)
00488:   989 CONTINUE
00489:C
00490:       WRITE(IO,7001)
00491:  7001 FORMAT(5(/////),20X,'Calculated Temperature Solution',/////)
00492:C
00493:C
00494:C
00495:C
00496:C           Time Loop
00497:C
00498:C
00499:       DO 1000 IT = 1,NTSP
00500:C
00501:          TIMOL = TIME
00502:          TIME = TIME + DT(IT)
00503:          ITER = 0
00504:          KTPRNT = KTPRNT + 1
00505:          KTPLOT = KTPLOT + 1
00506:C
00507:          IF(KBCT.EQ.1) GO TO 930
00508:          DO 920 NM = 1,NMBC1
00509:            NODE = NBCNGL(NM)
00510:            R(NODE) = TBC(IT,NM)
00511:   920    CONTINUE
00512:   930    CONTINUE
00513:C
00514:C    Call to routine which performs element by element integrations and
00515:C           returns coefficient matrix A and r.h.s vector R
00516:C
00517:   901    CALL TRINT(A,NBND,NBW,NDBC,NODEL,NELS,XG,YG,IO,NDG,DT(IT),R,
00518:      &   CNDF,CNDU,HETF,HETU,TF,THETA,THETAM,T,TOLD,TLAST,EL,ITER,
00519:      &   ITRLMP,TIMOL,AMP,AMPM,IZDPLT,KTPLOT,ANS,IT,MEDIA,MTYPE,HH,TA)
00520:C
00521:C    Solve algebraic system of equns
00522:C
00523:          CALL BANSAL(A,NBW,NBND,NBND,R,ANS,INEW,KERR,IO)
00524:C
00525:          IF(ITER.EQ.ITRLM) GO TO 950
00526:          ITER = ITER+1
00527:          GO TO 901
00528:C
00529:   950    CONTINUE
00530:C
00531:          IF(KTPRNT.NE.KPRTVL) GO TO 990
00532:          KTPRNT = 0
00533:C
00534:C           Print results
00535:C
00536:          WRITE(IO,716) TIME,IT
00537:   716    FORMAT(//,25X, 'TIME =',1P1E12.3,4X,'TIME STEP =',I6)
00538:          WRITE(IO,713) (K,ANS(K),K=1,NBND)
00539:   713    FORMAT(//,3(6X,'NODE',8X,'TEMP'),/,3(5X,I5,1P1E12.3))
00540:   990     CONTINUE
```

```
00541:C
00542:C    Update values of TLAST and TOLD
00543:C
00544:          DO 800 ND = 1,NBND
00545:          TLAST(ND) = TOLD(ND)
00546:  800     TOLD(ND) = ANS(ND)
00547:C
00548:C
00549: 1000 CONTINUE
00550:C
00551:       FIL = 1.0
00552:       VRYNEG = -1.1E30
00553:       IF(IZDPLT.GT.0) WRITE(33,7007) VRYNEG,FIL,FIL,FIL
00554: 7007 FORMAT(1P4E11.2)
00555:C
00556:C
00557: 1111 CONTINUE
00558:C
00559:C          Close all files and stop
00560:C
00561:C
00562:       CLOSE(5)
00563:       CLOSE(IO)
00564:       CLOSE(33)
00565:C
00566:C
00567:       RETURN
00568:       END
00569:C
00570:C
00571:C
00572:C
00573:C
00574:C
00575:        SUBROUTINE TRINT(A,NBND,NBW,NDBC,NODEL,NELS,XG,YG,IO,NDG,DT,
00576:      &    R,CNDF,CNDU,HETF,HETU,TF,THETA,THETAM,T,TOLD,TLAST,EL,ITER,
00577:      &    ITRLMP,TIMOL,AMP,AMPM,IZDPLT,KTPLOT,ANS,IT,MEDIA,MTYPE,HH,TA)
00578:C
00579:C    Subroutine to perform finite element integrations and assemble banded
00580:C          global matrix for linear triangles
00581:C
00582:C
00583:       COMMON /SMALL/ ZERO
00584:C
00585:       DIMENSION A(NBND,NBW), NDBC(NBND), NODEL(NELS,3), XG(NBND)
00586:       DIMENSION YG(NBND), X(3), Y(3), R(NBND), T(NBND), TOLD(NBND)
00587:       DIMENSION TLAST(NBND), ELA(3), ELB(3), TE(3), ANS(NBND), EL(10)
00588:       DIMENSION TF(10), HETU(10),HETF(10)
00589:       DIMENSION CNDF(10),CNDU(10),MTYPE(NELS)
00590:C
00591:C Identifies a first pass
00592:C
00593:       IF(IT*(ITER+1) .NE. 1) GO TO 101
00594:C
```

```
00595:C The next computations eliminate the problems that could
00596:C start if the maximum initial temperature was zero
00597:C
00598:       TMAX = TOLD(1)
00599:       TMIN = TOLD(1)
00600:       DO 100 ND = 2,NBND
00601:          TM = TOLD(ND)
00602:          IF(TM .GT.TMAX) TMAX = TM
00603:          IF(TM .LT.TMIN) TMIN = TM
00604:   100 CONTINUE
00605:       DELTF = 0.0025*( TMAX - TMIN )
00606:       TMAX = ABS(TMAX)
00607:       IF(ABS(TMIN).GT.TMAX) TMAX = ABS(TMIN)
00608:       IF(DELTF.LT. 0.0025*TMAX) DELTF = 0.0025*TMAX
00609:   101 CONTINUE
00610:C
00611:C Update the T(ND) used to calculate the matrix A
00612:C
00613:       DO 105 ND = 1,NBND
00614:          T(ND) = AMP*ANS(ND) + AMPM*TOLD(ND)
00615:   105 CONTINUE
00616:C
00617:       DO 10 ND = 1,NBND
00618:C
00619:C Initialize the matrix A, the r.h.s. vector R
00620:C
00621:          DO 5 J = 1,NBW
00622:     5    A(ND,J) = 0.0
00623:          IF(NDBC(ND).NE.1) GO TO 9
00624:          A(ND,NDG) = 1.0
00625:          GO TO 10
00626:     9    R(ND) = 0.0
00627:    10  CONTINUE
00628:C
00629:C
00630:C   ELEMENT LOOP.  This is the one that does the quadrature,
00631:C and assembles A and R
00632:C
00633:        DO 1000 NLMT = 1,NELS
00634:C
00635:C
00636:C Initialize flags
00637:C    KPHS = -3 : Element frozen
00638:C    KPHS = +3 : Element unfrozen
00639:C    KFRZ = 1  : Phase change element
00640:          KFRZ = 3
00641:          KPHS = 0
00642:          RF = 1.0
00643:C
00644:C    Identify material type of this element
00645:C
00646:          I1 = MTYPE(NLMT)
00647:C
00648:C The next loop works with each element to determine if it is
```

```
00649:C a phase change element or not and then does necessary
00650:C computations to enable calculation of phase change effect
00651:C
00652:C
00653:          DO 400 K = 1,3
00654:            NK = NODEL(NLMT,K)
00655:            X(K) = XG(NK)
00656:            Y(K) = YG(NK)
00657:            TE(K) = T(NK)
00658:C
00659:C This section accounts for the possibility that the node
00660:C temperature equals the phase change temperature.  If
00661:C it does, it is shifted in the direction of its last recorded value
00662:C
00663:            TFMZ = TF(I1) - DELTF
00664:            TFPZ = TF(I1) + DELTF
00665:            IF ((TE(K).GT.TFMZ).AND.(TE(K).LT.TFPZ)) THEN
00666:               IF (TLAST(NK).GT.TF(I1)) THEN
00667:                  TE(K) = TF(I1)+DELTF
00668:                ELSE
00669:                  TE(K) = TF(I1) - DELTF
00670:               ENDIF
00671:            ENDIF
00672:C
00673:C    Record frozen or unfrozen status of node
00674:C
00675:            IF(TE(K).LT.TF(I1)) KPHS = KPHS - 1
00676:            IF(TE(K).GT.TF(I1)) KPHS = KPHS + 1
00677:C
00678:C    Identify next node around the element
00679:C
00680:            KP = K+1
00681:            IF(KP.EQ.4) KP = 1
00682:            NP = NODEL(NLMT,KP)
00683:C
00684:C Bypass the rest of the loop if no phase change
00685:C
00686:            IF(((T(NP).GT.TF(I1)).AND.(T(NK).GT.TF(I1))).OR.
00687:     &        ((T(NP).LT.TF(I1)).AND.(T(NK).LT.TF(I1)))) GO TO 400
00688:C
00689:            IF(KFRZ.EQ.3) NPL = NP
00690:            IF(KFRZ.EQ.3) GO TO 380
00691:            NDF = NPL - NK
00692:            IF(NDF.NE.0) GO TO 360
00693:C
00694:C    (XC,YC) is vertex of triangular sub-element
00695:C
00696:            ND1 = NK
00697:            XC = X(K)
00698:            YC = Y(K)
00699:            GO TO 380
00700:  360       ND1 = NP
00701:            XC = X(KP)
00702:            YC = Y(KP)
```

```
00703:  380       CONTINUE
00704:C
00705:            KFRZ = KFRZ-1
00706:C
00707:C The next computations are for the phase change element,
00708:C computing the coordinates of the phase change isotherm,
00709:C and the values of the basis/weight function at the isotherm
00710:C (ELA and ELB).
00711:C
00712:C  The next few lines interpolate to find the phase
00713:C change isotherm
00714:C
00715:            DLTF = ABS((TF(I1)-T(NK))/(T(NP)-T(NK)))
00716:            DELX = DLTF*(XG(NP)-XG(NK))
00717:            DELY = DLTF*(YG(NP)-YG(NK))
00718:            IF(KFRZ.EQ.1) GO TO 390
00719:            XB = XG(NK) + DELX
00720:            YB = YG(NK) + DELY
00721:            ELB(KP) = DLTF
00722:            ELB(K) = 1.0 - DLTF
00723:            KPP = KP+1
00724:            IF(KPP.EQ.4) KPP = 1
00725:            ELB(KPP) = 0.0
00726:            GO TO 400
00727:  390       XA = XG(NK) + DELX
00728:            YA = YG(NK) + DELY
00729:            ELA(KP) = DLTF
00730:            ELA(K) = 1.0 - DLTF
00731:            KPP = KP + 1
00732:            IF(KPP.EQ.4) KPP = 1
00733:            ELA(KPP) = 0.0
00734:  400     CONTINUE
00735:C Update TLAST
00736:          DO 102 ND = 1,NBND
00737:102       TLAST(ND) = T(ND)
00738:          IF(KPHS.NE.3) GO TO 410
00739:C
00740:C The next assignments are for an unfrozen element
00741:C
00742:          HET = HETU(I1)
00743:          COND = CNDU(I1)
00744:          GO TO 499
00745:  410     IF(KPHS.NE.-3) GO TO 420
00746:C
00747:C The next assignments are for a frozen element
00748:C
00749:          HET = HETF(I1)
00750:          COND = CNDF(I1)
00751:          GO TO 499
00752:C
00753:C    Calc area of triangular sub-element
00754:C
00755:420       A1 = XB*YC - XC*YB - XA*YC + XC*YA + XA*YB -XB*YA
00756:          A1 = ABS(A1/2.0)
```

```
00757:C
00758:C          Assign frozen and unfrozen conductivities
00759:C            to appropriate parts of element
00760:C
00761:          COND = 0.0
00762:          VK1 = CNDU(I1)
00763:          VK2 = CNDF(I1)
00764:          IF(T(ND1).GT.TF(I1)) GO TO 498
00765:          VK1 = CNDF(I1)
00766:          VK2 = CNDU(I1)
00767:  498     CONTINUE
00768:  499     CONTINUE
00769:C
00770:C
00771:C Calculate the area of the element using cross product
00772:C
00773:          AE = X(2)*Y(3) - X(3)*Y(2) - X(1)*Y(3)
00774:     &     + X(3)*Y(1) + X(1)*Y(2) - X(2)*Y(1)
00775:          AE = AE/2.0
00776:C
00777:C
00778:C          Assign conductivity factor for non-phase change element
00779:C
00780:          C4A = 0.25*COND/AE
00781:C
00782:       IF(KFRZ.EQ.1) GO TO 450
00783:       BCF = HET * AE/(12.0*DT)
00784:       GO TO 4450
00785:C
00786:C
00787:C  This next assignment accounts for the different
00788:C  sensible heat in the unfrozen and frozen portion
00789:C  of the phase change element.
00790:C
00791:  450 CONTINUE
00792:            IF (T(ND1).GT.TF(I1)) THEN
00793:                C1 = HETU(I1)
00794:                C2 = HETF(I1)
00795:            ELSE
00796:                C1 = HETF(I1)
00797:                C2 = HETU(I1)
00798:            END IF
00799:                BCF = (C1*A1 + C2*(AE - A1))/(12.0*DT)
00800:C
00801:C Calculations on element with phase change to account for the
00802:C latent heat effect discontinuity
00803:C
00804:C  Calculate length of tau
00805:          TAU = SQRT( (XB-XA)**2 + (YB-YA)**2 )
00806:C  The magnitude of the gradient of T is computed using the x and y
00807:C derivatives of the equation of the plane T(x,y) = a + bx + cy,
00808:C i.e., expressing a and b in terms of x(i), y(i), & T(i) using Cramer's
00809:C rule.  The 2*AE is the determinant used in Cramer's rule.
00810:          SUMDX = TE(1)*(Y(2)-Y(3)) + TE(2)*(Y(3)-Y(1))
```

```
00811:      &    + TE(3)*(Y(1)-Y(2))
00812:           SUMDY = TE(3)*(X(2)-X(1)) + TE(2)*(X(1)-X(3))
00813:      &    + TE(1)*(X(3)-X(2))
00814:           DTDS = SQRT( SUMDX**2 + SUMDY**2)/(2.0*AE)
00815:           ELDSDT = EL(I1)/(DTDS*DT)
00816:C
00817:C     Store z.d.isotherm locations for plotting
00818:C
00819:       IF(KTPLOT.EQ.IZDPLT.AND.ITER.EQ.0) THEN
00820:          WRITE(33,7001) XA,YA,XB,YB
00821: 7001     FORMAT(1P4E13.4)
00822:       ENDIF
00823:C
00824: 4450     CONTINUE
00825:C
00826:C   ELEMENT NODE ROW LOOP
00827:C
00828:           DO 600 I = 1,3
00829:C
00830:             IG = NODEL(NLMT,I)
00831:             IF(NDBC(IG).EQ.1) GO TO 600
00832:C
00833:C            Treat convective BC's
00834:C
00835:       IF(NDBC(IG).NE.2) GO TO 599
00836:       DO 510 J = 1,3
00837:       IF(J.EQ.I) GO TO 510
00838:       JG = NODEL(NLMT,J)
00839:       IF(NDBC(JG).NE.2) GO TO 510
00840:       DDX = X(I) - X(J)
00841:       DDY = Y(I) - Y(J)
00842:       DD = SQRT(DDX**2 + DDY**2)
00843:       R(IG) = R(IG) + HH*TA*DD/2.0
00844:      &    + THETAM*HH*DD * (TOLD(IG)/3.0 + TOLD(JG)/6.0)
00845:       A(IG,NDG) = A(IG,NDG) + THETA*HH*DD/3.0
00846:       JGBD = JG - IG + NDG
00847:       A(IG,JGBD) = A(IG,JGBD) + THETA*HH*DD/6.0
00848:   510 CONTINUE
00849:   599 CONTINUE
00850:C
00851:             IJ = I+1
00852:             IF(IJ.EQ.4) IJ = 1
00853:             IK = IJ+1
00854:             IF(IK.EQ.4) IK = 1
00855:C
00856:C    ELEMENT NODE COLUMN LOOP
00857:C
00858:             DO 500 J = 1,3
00859:               JI = J-1
00860:               IF(JI.EQ.0) JI = 3
00861:               JK = J + 1
00862:               IF(JK.EQ.4) JK = 1
00863:               JG = NODEL(NLMT,J)
00864:               JGBD = JG - IG + NDG
```

```
00865:C This term (DRV) is part of the element quadrature
00866:              DRV =  (Y(IJ)-Y(IK))*(Y(JK)-Y(JI)) + (X(IK)-X(IJ))
00867:     &    *(X(JI)-X(JK))
00868:              BIJ = BCF
00869:              IF(I.EQ.J) BIJ = 2.0*BCF
00870:              IF(KFRZ.NE.1) GO TO 480
00871:C
00872:C For the phase change elements
00873:C
00874:          BIJ = BIJ + ELDSDT*TAU* ( 2.0*(ELB(I)-ELA(I))*(ELB(J)-ELA(J))
00875:     &    + 3.0*ELB(I)*ELA(J) + 3.0*ELB(J)*ELA(I) )/6.0
00876:C
00877:C  The next term distributes the appropriate conductivity
00878:C over the correct proportions in the phase change element.
00879:C
00880:              CIJ = VK2*DRV/(4.0*AE) + (VK1-VK2)*DRV*A1/(4.0*AE*AE)
00881:              GO TO 495
00882:C
00883:C For the other elements
00884:C
00885:  480         CIJ = C4A*DRV
00886:C
00887:C Load the matrix A and the vector R for the system solver
00888:C
00889:C           The THETA parameter controls numerical implicitness
00890:C
00891:  495         A(IG,JGBD) = A(IG,JGBD) + BIJ + THETA*CIJ
00892:              R(IG) = R(IG) + (BIJ + THETAM*CIJ)* TOLD(JG)
00893:C
00894:  500        CONTINUE
00895:  600      CONTINUE
00896:C
00897: 1000  CONTINUE
00898:C
00899:C
00900:       IF(KTPLOT.NE.IZDPLT .OR. ITER.NE.0) GO TO 1010
00901:       WRITE(IO,7070) TIMOL
00902: 7070 FORMAT(//,12X,'Plotting data stored for TIME =',1P1E12.3)
00903:       KTPLOT = 0
00904:       VRYLGE = 1.1E30
00905:       VITM = IT-1.0
00906:       WRITE(33,7701) VRYLGE, VITM, TIMOL, TIMOL
00907: 7701 FORMAT(1P4E11.2)
00908: 1010 CONTINUE
00909:C
00910:C
00911:C    Determine "ZERO", i.e. a magnitude below which significance is lost
00912:C
00913:       IF(IT*(ITER+1) .NE. 1) GO TO 1101
00914:       AMAX = 0.0
00915:       DO 1100 I = 1,NBND
00916:       DO 1100 J = 1,NBW
00917:       ABSVAL = ABS(A(I,J))
00918:       IF(ABSVAL.GT.AMAX) AMAX = ABSVAL
```

```
00919: 1100 CONTINUE
00920:       ZERO = AMAX * 0.0001
00921:C
00922:C
00923: 1101 CONTINUE
00924:       RETURN
00925:C
00926:       END
00927:C
00928:C
00929:C
00930:C The following routine solves the system Ax=B
00931:C
00932:C
00933:       SUBROUTINE BANSAL (A,NBW,NEQ,NMAX,B,X,INEW,KERR,IO)
00934:C
00935:C
00936:C
00937:       COMMON /SMALL/ ZERO
00938:C
00939:       DIMENSION  A(NMAX,NBW),B(NMAX),X(NMAX)
00940:C
00941:C          INITILIZE  COUNTERS
00942:C
00943:       KERR=0
00944:       NBWT=NBW
00945:       N=NEQ
00946:       NB2=  NBWT/2 +1
00947:       NB1= NB2-1
00948:       N1=N-1
00949:       NP1=N+1
00950:       NB1M1= NB1-1
00951:       DO 10 I=1,N
00952:       X(I)=B(I)
00953: 10    CONTINUE
00954:       IF(INEW.NE.0)GO TO 350
00955:C
00956:C          FORWARD ELIMINATION  OF MATRIX
00957:C
00958: 50    K1=NB1
00959:       IM1=1
00960:       DO 300  I=2,N
00961:          K1=MINO(K1+1,N)
00962:          K=NB1
00963:          AII=A(IM1,NB2)
00964:          IF((AII.LT.ZERO).AND.(AII.GT.-ZERO)) GO TO 1850
00965:          DO 200 J=I,K1
00966:            CX= A(J,K)/ AII
00967:            L1=NB2 +1
00968:            KPB= K +NB1
00969:            KP1= K+1
00970:            DO 100 L=KP1,KPB
00971:              A(J,L)=  A(J,L) -CX* A(IM1,L1)
00972: 100          L1=L1+1
```

```
00973:            A(J,K)= CX
00974:            X(J)=X(J)-X(IM1)*CX
00975:            K= K-1
00976: 200      CONTINUE
00977: 300    IM1=I
00978:        GO TO 550
00979:C
00980:C               OPERATE ON THE NEW RHS -IS DONE AFTER THE  FIRST PASS
00981:C
00982: 350    IN=NB1
00983:        IM1=1
00984:        DO 500 I=2,N
00985:           IN=MIN0(IN+1,N)
00986:           J1= NB1
00987:           DO 400 J=I,IN
00988:             X(J)= X(J) -X(IM1)* A(J,K1)
00989:             K1=K1-1
00990: 400       CONTINUE
00991: 500       IM1=I
00992:C
00993:C            BACK WARD SUBSTITUTION
00994:C
00995: 550       X(N)= X(N)/A(N,NB2)
00996:           DO 700 I=1,N1
00997:             SUM=0.0
00998:             L= NB2 +1
00999:             K= N-I
01000:             K1=  MIN0(N1, K+NB1M1)
01001:             DO 600 J=K,K1
01002:               SUM=SUM + A(K,L)* X(J+1)
01003:               L=L+1
01004: 600         CONTINUE
01005:             X(K) = ( X(K)-SUM)/ A(K,NB2)
01006: 700       CONTINUE
01007:        RETURN
01008: 1850   WRITE(IO,9000) IM1
01009:        KERR=1
01010:        RETURN
01011: 9000   FORMAT(1HO,120(1H*)/10X,'ERROR FROM BANSAL-',I5,13H DIAGONAL ELE,
01012:       1 20HMENT REDUCED TO ZERO/1X,120(1H*) )
01013:        END
```

☆ U. S. GOVERNMENT PRINTING OFFICE: 1988--500-050--62067