```
           Y= IYPOS/FLOAT(NNODY-1)
           GO TO 30
        END IF
C                   AT END OF FIRST LEVEL
        IF (TEST.EQ.1.00E0) THEN
           LEV=111111
           INPOS=NOD-ISTRT
           X=       (N1X-1)/FLOAT(N1X-1)
           Y= IYPOS/FLOAT(NNODY-1)
           GO TO 30
        END IF
C                   IN SECOND LEVEL OF NODES
        IF (TEST.GT.1.00E0) THEN
           LEV=2
           INPOS=NOD-ISTRT-N1X
           X=       INPOS/FLOAT(N2X-1)
           Y= (IYPOS+1)/FLOAT(NNODY-1)
           GO TO 30
        END IF
C                   AT END OF SECOND LEVEL
        TEST=(NOD-ISTRT+1)/N3X+1.0
        IF (TEST.EQ.1.00E0) THEN
           LEV=222222
           INPOS=NOD-ISTRT+1
           X=        N3X/FLOAT(N2X-1)
           Y= (IYPOS+1)/FLOAT(NNODY-1)
        END IF
C      X=NOD-ISTRT
30     WRITE(6,*) NOD, X, Y
100     CONTINUE

       DO 35 NELE=1,ITELS
35       WRITE(6,20) NELE,(ILTOP(NELE,J),J=1,8)
C             LIST BOUNDARY NODES
       I=1
       N1=1
       DO 40 J=1,NELY
         N2=N1+N1X
         NBND(I)=N1
         I=I+1
         NBND(I)=N2
         I=I+1
40     N1=N2+N2X
       NBND(I)=NBND(I-1)+N2X
       NEXT=N1X-1
       DO 50 J=1,NEXT
         I=I+1
         NBND(I)=NBND(I-1)+1
50     CONTINUE
       N1=NBND(I)
       DO 60 J=1,NELY
         N2=N1-N1X
         NBND(I)=N1
         I=I+1
         NBND(I)=N2
         I=I+1
60     N1=N2-N2X
       NBND(I)=NBND(I-1)-N2X
       NBOX=N1X-1
```

```
      DO 80 J=1,NBOX
        I=I+1
        NBND(I)=NBND(I-1)-1
80    CONTINUE
      J=1
      WRITE(6,99) (NBND(J),J=1,N1Y)
      IWRT1=N1Y+1
      IWRT2=IWRT1+N1X-2
      WRITE(6,99) (NBND(J),J=IWRT1,IWRT2)
      IWRT1=IWRT2+1
      IWRT2=IWRT1+N1Y-2
      WRITE(6,99) (NBND(J),J=IWRT1,IWRT2)
      IWRT1=IWRT2+1
      IWRT2=IWRT1+N1X-3
      WRITE(6,99) (NBND(J),J=IWRT1,IWRT2)
      ITBND=2*N1Y+2*(N1X-2)
99    FORMAT (3X,15(1X,I3))
      STOP
      END
```

The following program CRUSH FORTRAN allows crushing of a finite element mesh of 8-noded quadrilateral elements.

```
C --- Program to crush a mesh of 8-noded quadrilateral elements
      DIMENSION NLTOP(500,14),COORD(500,3)
      DIMENSION NBNDRY(5,6,80), NBND(5)
      DIMENSION IVXTR(6), IEXLST(6,4,60), NODXTR(6)
      DATA NIN, NOUT / 5, 7/
C --- Choose where to crush mesh
C     1-U 2-BOTH 3-L 4-R 5-? 6-L&M&R
      IMESH=4
C --- Read in data from SNODEQ FORTRAN
      READ (NIN,*) NODTOT, IDIMN, NVAR
      WRITE (NOUT,*) NODTOT, IDIMN, NVAR
      READ (NIN,*) NX, NY
      WRITE (NOUT,*) NX, NY
      DO 10 I=1,NODTOT
        READ (NIN,*) NODNUM, (COORD(NODNUM,J),J=1,IDIMN)
          X=COORD(NODNUM,1)
          COORD(NODNUM,1)=FMESH(X, IMESH)
        WRITE (NOUT,*) NODNUM, (COORD(NODNUM,J),J=1,IDIMN)
10    CONTINUE
      READ (NIN,*) IELTYP, ITELS, NODEL
      WRITE (NOUT,*) IELTYP  ITELS, NODEL
      DO 20 I=1,ITELS
        READ (NIN,*) IELNM, (NLTOP(IELNM,J+2),J=1,NODEL)
        WRITE (NOUT,*) IELNM, (NLTOP(IELNM,J+2),J=1,NODEL)
20    CONTINUE
      DO 30 IEQ=1,NVAR
        READ (NIN,*) NBND(IEQ)
        WRITE (NOUT,*) NBND(IEQ)
      DO 30 I=1,NBND(IEQ)
      READ (NIN,*) NBTYPE,NODSID, NUMNOD, (NBNDRY(IEQ,I,J+3),J=1,NUMNOD)
      WRITE(NOUT,*)NBTYPE,NODSID, NUMNOD, (NBNDRY(IEQ,I,J+3),J=1,NUMNOD)
30    CONTINUE
      READ (NIN,*) NXTRCT
      WRITE (NOUT,*) NXTRCT
      DO 40 I=1,NXTRCT
        READ (NIN,*) IVXTR(I), NODXTR(I),
     *    (IEXLST(I,IVXTR(I),J),J=1,NODXTR(I))
        WRITE (NOUT,*) IVXTR(I), NODXTR(I),
     *    (IEXLST(I,IVXTR(I),J),J=1,NODXTR(I))
40    CONTINUE
      STOP
      END
      REAL FUNCTION FMESH (X, IMESH)
C --- Function to crush mesh
C     1-U 2-BOTH 3-L 4-R 5-? 6-L&M&R
      PI=3.14159
      FMESH=0.00E0
         IF (IMESH.EQ.1) F=X
         IF (IMESH.EQ.2) F=-1.0*(0.5)*COS(PI*X)+(A+B)/2.0
         IF (IMESH.EQ.3) F=1.0-COS(X*PI/2.0)
         IF (IMESH.EQ.4) F=SIN(X/2.0*PI)
         IF (IMESH.EQ.5) F=(4.0/9.0*X**3-6.0/10.0*X**2+X/10.0)
         IF (IMESH.EQ.6) F=(X/2.0-SIN(2.0*X)/4.0)/PI
      FMESH=F
      RETURN
      END
```

Description of 'FEPDE' - a finite element code for the solution of systems of steady-state, two-dimensional non-linear partial differential equations

P. Anderson

Department of Chemical Engineering
University of the Witwatersrand

1986

INTRODUCTION

'FEPDE' has been written to solve systems of nonlinear, linked partial differential equations using finite element methods. The finite element method was chosen for its versatility and ability to handle the complex form of equations and boundary conditions encountered in practice.

Knowledge of finite element theory is required to set up and solve a problem using 'FEPDE'. An adequate introduction to finite element theory may be found in the 'NAG Finite Element Library - Level 1' manual (1983). The data structure and coding of 'FEPDE' is based on the example program 'NAGFE3P2 FORTRAN', which may be found in the Level 1 manual, as well as on the 'ENGINEERING' disk on the Wits mainframe.

Overview of 'FEPDE'

The code may be divided into two 3 major sections:

1) Main section
This section sets up common blocks and reads in all the data required, eg. mesh geometry and topology, experimental measurements, output checking flags and so on. The nodal freedom array 'NF' (which is used in the assembly of element contributions into the final system matrix 'SYSK') is contructed from the user-supplied mesh and boundary condition data (see below). Any model parameters required are read in by the main program.

2) Subroutine 'FELMNT'
This is the section of program that actually calculates the 'element stiffness' matrices 'ELK', which are derived from the derivative terms in the equation(s), as well as the element vector 'XTVEC', which accounts for any other terms in the equation, such as energy generation ('heat source') terms or constants. The abovementioned vector and array are defined in user supplied subroutines (see below). These element contributions are assembled into the system matrix and vector 'SYSK' and 'RHS' by the routines 'ASUSM' and 'ASRHS' respectively, using the 'steering' vector 'NSTER' which is derived from element topology data by the subroutine 'DIRECT'.

Dirichlet and Cauchy boundary conditions are included by the Payne-Irons method, and a boundary integral, respectively.

3) Subroutines defining the equations and boundary conditions. These are divided as follows:

3.1) Subroutine 'EQNS'
This subroutine defines the equations resulting from the finite element analysis of the original partial differential equations. The contributions from each variable and each equation are calculated separately, and may depend on any

variable dependent or independent), or the gradient of any dependent variable. See example below.

3.2) Subroutine 'SOURCT'
This subroutine calculates the conrribution from source terms (which may be distributed or concentrated) or constants. The value returned by 'SOURCT' may depend on any variable (dependent or independent), or the gradient of any dependent variable.

3.3) Subroutine 'BFUNM'
This subroutine calculates the contributions of the dependent variables in the boundary conditions to the system matrix. Again, contributions from each variable and equation are calculated separately. The boundary condition terms may depend on any variable (dependent or independent), or the gradient of any dependent variable

3.4) Subroutine 'BFUNV'
This subroutine calculated the contributions of constants in the boundary conditions to the right-hand side matrix 'RHS'. Contributions from each equation are calculated separately. The boundary conditions may depend on any variable (dependent or independent), or the gradient of any dependent variable.

3.5) Function 'H'
This function calculates the dirichlet condition for the equations. The value may depend on any variable (dependent or independent) or the gradient of any dependant variable. Values for each equation are calculated separately.

3.6) Functions 'FXY' and 'DFXY'
'FXY' ['DFXY'] evaluates the value of [gradient of] any dependent variable given element number and position within the element [and direction in which gradient is required].

4) Service Subroutines
These are as follows:

4.1) Subroutines 'ASMAT' and 'ASVEC'
This assembles equation/variable matrix-contributions into the element matrix. This is best described by an example.

Eg. if the following equations are to be solved

$$P(\Phi) + Q(\Psi) = R(\underline{x})$$

$$S(\Phi) + T(\Psi) = U(\underline{x})$$

Where P, Q, S, and T are differential operators operating in R^2 (i.e. (x,y)).

Ident'fy the contributions from P, Q, S, T with p, q, s, t respectively.

The element matrix contribution, e^1 from Φ in A may be written
as follows (assuming three noded triangular elements for example):

$$e_p = \begin{matrix} p & p & p \\ p & p & p \\ p & p & p \end{matrix} \qquad \text{similarly for } Q,S,T$$

Where the subscript P denotes contribution from that operator

e_p to e_T are then assembled into the overall element matrix as follows:

$$E_{ele} = \begin{vmatrix} \bullet & q & \bullet & q & \bullet & q & \bullet & q \\ s & t & s & t & s & t & s & t \\ \bullet & q & \bullet & q & \bullet & q & \bullet & q \\ s & t & s & t & s & t & s & t \\ \bullet & q & \bullet & q & \bullet & q & \bullet & q \\ s & t & s & t & s & t & s & t \\ \bullet & q & \bullet & q & \bullet & q & \bullet & n \\ s & t & s & t & s & t & s & t \end{vmatrix}$$

The 'p entries have been replaced with a $\bullet$ to clarify the assembly process. It is clear from the above demonstration that the matrices contributed by each variable are 'stretched' out and placed in a nNxnN matrix, where N is the number of variables, and n is the number of nodes per element. The above matrix is contructed by 'ASMAT', given $e_P \ldots e_T$.

Similarly the right hand side contributions are assembled by 'ASVEC' as follows:

$$RHS_{ele} = \begin{vmatrix} r \\ u \\ r \\ u \\ r \\ u \\ r \\ u \end{vmatrix}$$

The boundary condition matrices and vectors as described in section 3 are subjected to the same assembly operations.

4.2) Subroutine 'SEPRT'

The solution of the system of linear equations defined by [SYSK|RHS] is in the form of a vector of length Nxtotal number of nodes. 'SEPRT' separates this lumped vector into arrays FVAL(i,j) where i denotes variable number, and j denotes node number. The value of any particular variable at any node are thus readily availableduring execution. This is useful and economical in cases where iteration is required.

PDFELM

A Finite Element Code for the Solution of a General System of Steady or Unsteady Nonlinear Partial Differential Equations

by:

P. Anderson (MSc(Eng))

and

S.M. Bradshaw (MSc(Eng))

Department of Chemical Engineering

University of the Witwatersrand

1987

CONTENTS

CONTENTS

1 Introduction

1.1 General considerations and background.

The general class of problem soluble by "PDFELM" is as follows:

$$M\ [\Phi_t,\Phi] = K\ [\Phi] \qquad (1/1)$$

Where M is an operator, operating linearly on the *first* time derivative of Φ. The l.h.s. of (1/1) takes the form of a linear sum of first order time-derivatives, whose individual coefficients may be nonlinear in spatial derivatives if necessary.

K is a general, nonlinear operator, operating on spatial derivatives of Φ of any order below that of the shape functions employed.

Note: Application of Green's theorem reduces the order of the spatial derivatives, thus expanding the applicability range of any given shape function.

The current "PDFELM" implementation uses the Galerkin weighted residual method, though only minor changes would be necessary to change this (to a collocation method, for example).

"PDFELM" 's predecessor ("FEPDE", Anderson. 1986), was designed to solve only the steady-state analogue of (1/1). It was therefore a comparatively small task to extend the code and data structure to solve the unsteady-state problem as well.

The preliminary finite element analysis of the left and r.h. sides of (1/1) is discussed in detail in many texts, Norrie and de Vries (1973), for example. Assembly of r.h.s. contributions into a single matrix and a vector is discussed in detail by Anderson (1986). Assembly of l.h.s. contributions (into the matrix 'SYSM', the time-derivative multiplier) is done in a similar fashion.

1.2 Who should use "PDFELM" ? (or 'What is the purpose of this manual? ')

The mathematical sophistication required by finite-element methods exceeds that of the average user; moreover, some preliminary finite-element analysis on a problem is necessary before it can be submitted to "PDFELM". There are also various control options which must be correctly chosen if the program is to operate at maximum efficiency. As a result, this program is *not* a tool for beginners.

Therefore, the purpose of this manual is to:

a) inform those persons *au fait* with finite-element methods of the format of the program and required data, so that they may use it if necessary.

b) inform a wider audience of the type of problem soluble by the program, so that they may request instruction in the setting up of their particular problem, if applicable.

2 Brief Outline of Method of Solution

Finite-element analysis of equation (1/1) yields:

$$\mathrm{SYSM}(\underline{\Phi})\,\frac{\partial \underline{a}}{\partial t} = \mathrm{SYSK}(\underline{\Phi})\underline{\Phi} + \underline{f}(\underline{x},\underline{\Phi}) \qquad (2/1)$$

Where SYSM and SYSK are matrices (both may depend on the dependent variable Φ as shown)

$\underline{a}$ is the time dependent part of Φ (where $\Phi = \underline{N} \cdot \underline{a}$)
$\underline{N}$ are the shape functions, dependent only on $\underline{x}$
$\underline{f}$ is a vector

Details of the composition of the above matrices and vectors are discussed in detail by Anderson (1986).

- for an unsteady-state solution, the system (2/1) is solved for $\partial \underline{a}/\partial t$ (r.h.s evaluated at the current time value) using supplied initial conditions. The solution is then advanced in time by a multistep method or by Gear's method.

- for a steady-state solution, the r.h.s. of (2/1) is equated to zero, and the resulting expression solved iteratively for Φ until successive values satisfy a user-specified convergence criterion.

Note: if the *coefficients* of the time derivatives do not depend on

Φ or t, (this can be seen by inspection) the user can set a flag which ensures that the matrix 'SYSM' is assembled and decomposed (into upper and lower triangular matrices) only *once* during unsteady simulations, right at the start. This has a darmatic effect on run times, since solution of (2/1) during successive time-marching steps requires only 'back-substitution' into the triangular matrices. As the total number of degrees of freedom in a problem increases, reduction of 'SYSM' rapidly begins to dominate overall solution time.

3 Applications

Some sample problems (with solutions) which have been solved using the program are summarised below. Many of the results have been checked against published data; these checks are ommited here for the sake of brevity.

3.1 Simulation of Rivulet Flow - a single, linear partial differential equation.

The equation describing potential flow, in a wall-bound vertical rivulet, for example, is:

$$\frac{\partial v_z}{\partial t} = \frac{\mu}{\rho} \nabla^2 v_z - g_z$$

The boundary conditions are shown on fig. 1. Initial condition is $v_z = 0$ over the domain Ω.

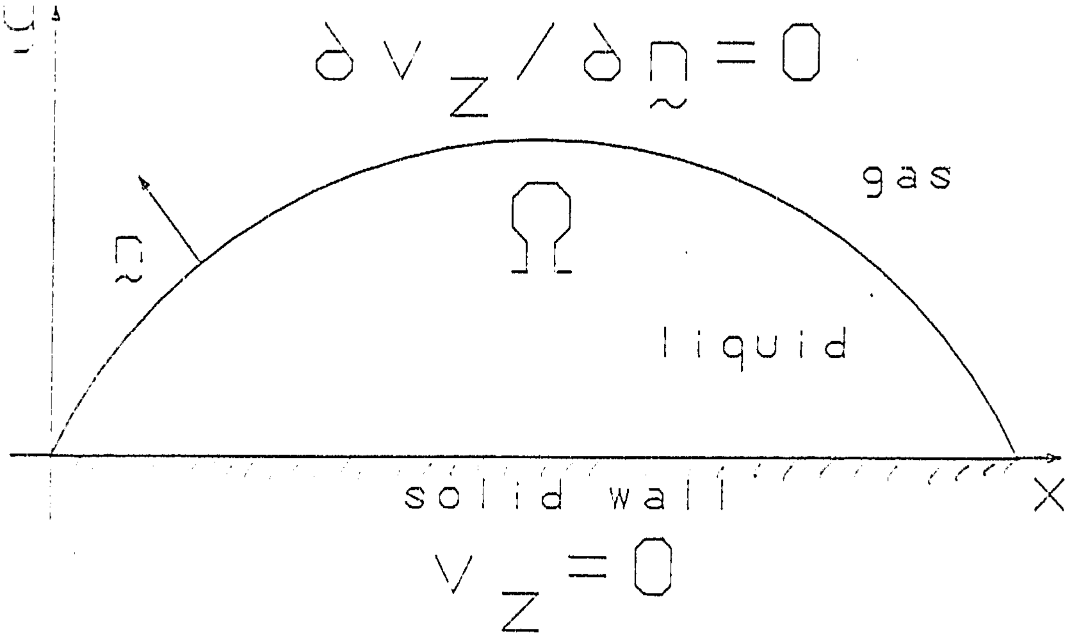
y
$\partial v_z / \partial \tilde{n} = 0$
gas
$\tilde{n}$
Ω
liquid
solid wall
X
$v_z = 0$

Fig. 1

Fig. 2 shows the steady-state velocity profile in a rivulet; note the maximum velocity at the furthermost edge of the rivulet (marked with a ·)

Fig. 3 shows the evolution of the maximum velocity with time.

This kind of analysis is relevant to the design of solid-fluidd chemical reactors, such as 'trickle-beds'.

3.2 Simulation of steady thermal conduction in a composite medium - a single nonlinear partial differential equation.

The equation describing steady thermal conduction in a composite medium whose conductivity depends on both position and temperature is as follows:

$$\nabla . \left[k(x,y,\theta)\nabla\theta \right] = 0$$

Boundary conditions are as shown in fig. 4; variation of thermal conductivity with position is also shown.

A temperature-position surface is shown in fig. 5 for the case where conductivity depends linearly on θ, $k(x,y,\theta) = k(x,y)[1+\theta]$. Fig. 6 shows the temperature along the vertical centreline, i.e. $x=0.5$. The 'step' resulting from the highly conductive core is clearly visible here.

Calculations of this sort are used to estimate thermal energy transfer in glass-fibre reinforced plastics and steel-concrete composites, for example.

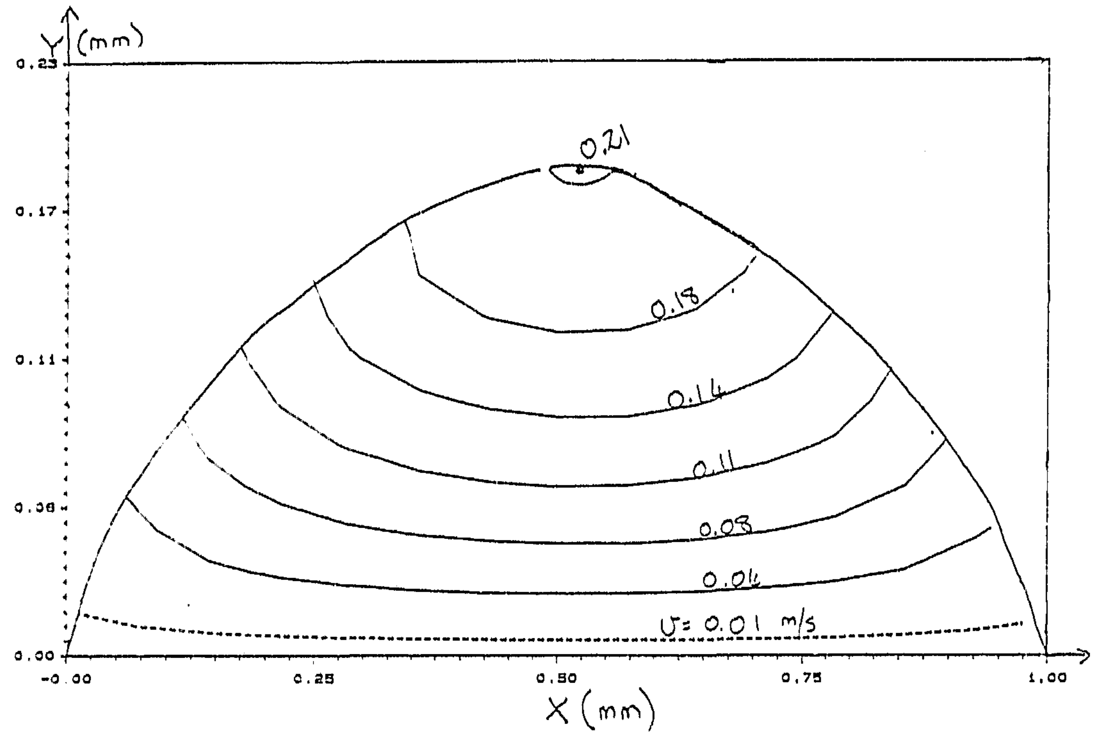
Y (mm)
0.23
0.17
0.11
0.06
0.00
0.21
0.18
0.14
0.11
0.08
0.06
v= 0.01 m/s
-0.00
0.25
0.50
0.75
1.00
X (mm)

Fig. 2

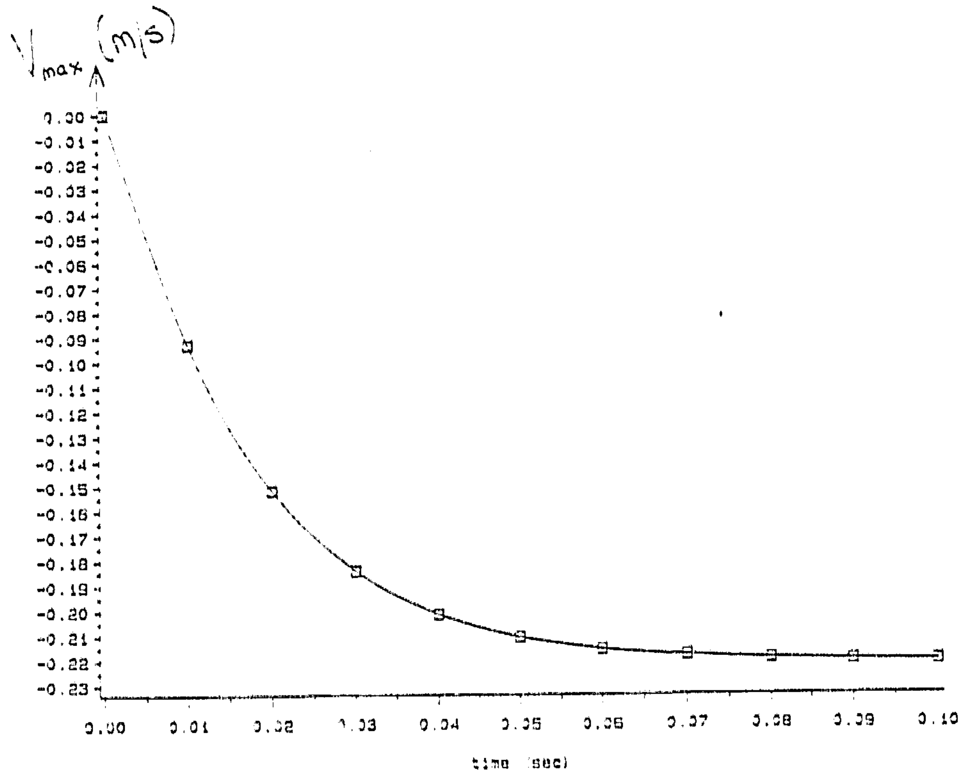
V_max (m/s)
0.00
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
-0.08
-0.09
-0.10
-0.11
-0.12
-0.13
-0.14
-0.15
-0.16
-0.17
-0.18
-0.19
-0.20
-0.21
-0.22
-0.23
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
time (sec)

Fig. 3

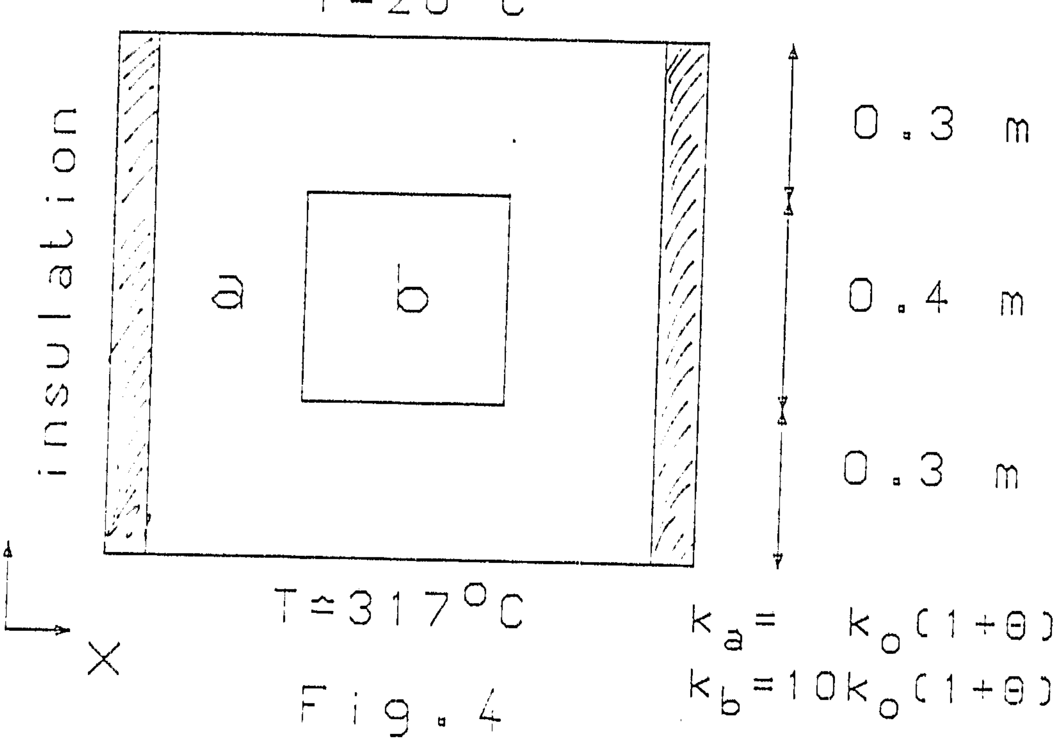
T ≐ 20°C
insulation
a
b
0.3 m
0.4 m
0.3 m
T ≐ 317°C
y
X
$k_a = k_o(1+\theta)$
$k_b = 10k_o(1+\theta)$

Fig. 4

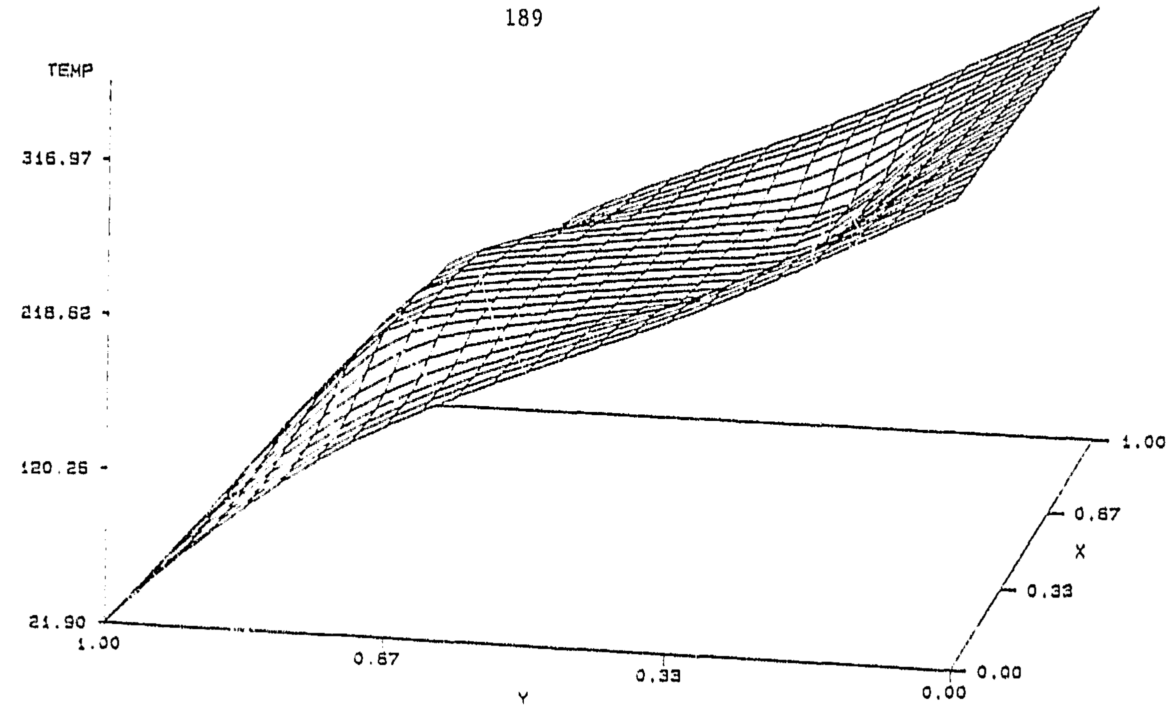
TEMP
316.97
218.62
120.26
21.90
1.00
0.67
0.33
0.00
Y
0.00
0.33
0.67
1.00
X

Fig. 5

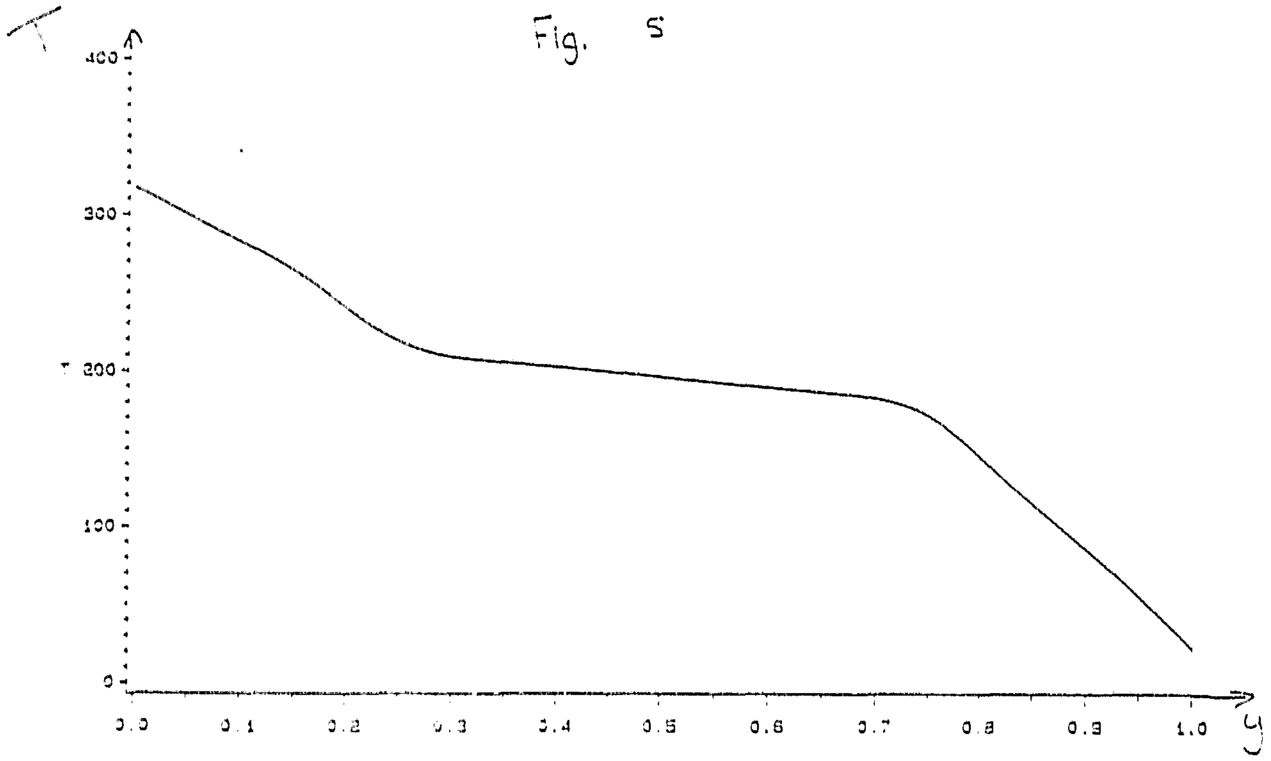
T
400
300
T 200
100
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
y

Fig. 6

3.3 Simulation of natural convection heat transfer in porous annulus - two linked nonlinear partial differential equations

The equations describing free convection in this case are as follows:

$$\frac{\partial^2\Psi}{\partial r^2} - \frac{1}{r}\frac{\partial\Psi}{\partial r} + \frac{\partial^2\Psi}{\partial z^2} = r \, \mathrm{Ra} \, \frac{\partial\theta}{\partial r}$$

$$\left[\frac{1}{r}\frac{\partial\Psi}{\partial z}\right]\frac{\partial\theta}{\partial r} - \left[\frac{1}{r}\frac{\partial\Psi}{\partial r}\right]\frac{\partial\theta}{\partial z} - \frac{\partial^2\theta}{\partial r^2} - \frac{1}{r}\frac{\partial\theta}{\partial r} - \frac{\partial^2\theta}{\partial z^2} = 0$$

Where the streamfunction, Boussinesq approximation and Darcy law have been used to simplify the full equations (see Bejan 1984 for details).

Appropriate boundary conditions and dimensions are shown in fig. 7.

Results for a Rayleigh number (Ra) of 300 are presented. Fig. 8 shows the streamlines, and fig. 9 shows the isotherms.

Such calculations are used to evaluate the average Nusselt number for heat transfer through layers of insulation around steam pipes, and heat-leaks into cryogenic installations, for example.

3.4 Fitting measured temperature data to a model - least-squares estimation of parameters.

The sum of squares of temperature prediction errors for some experimental measurements was minimised by variation of the Rayleigh number and a heat transfer coefficient. For experimental details and numerical results, see Anderson (1987). Fitted results predicted permeabilities (embedded in Ra) of the same order of magnitude as those estimated using known physical quantities (particle size, viscosity, etc.).

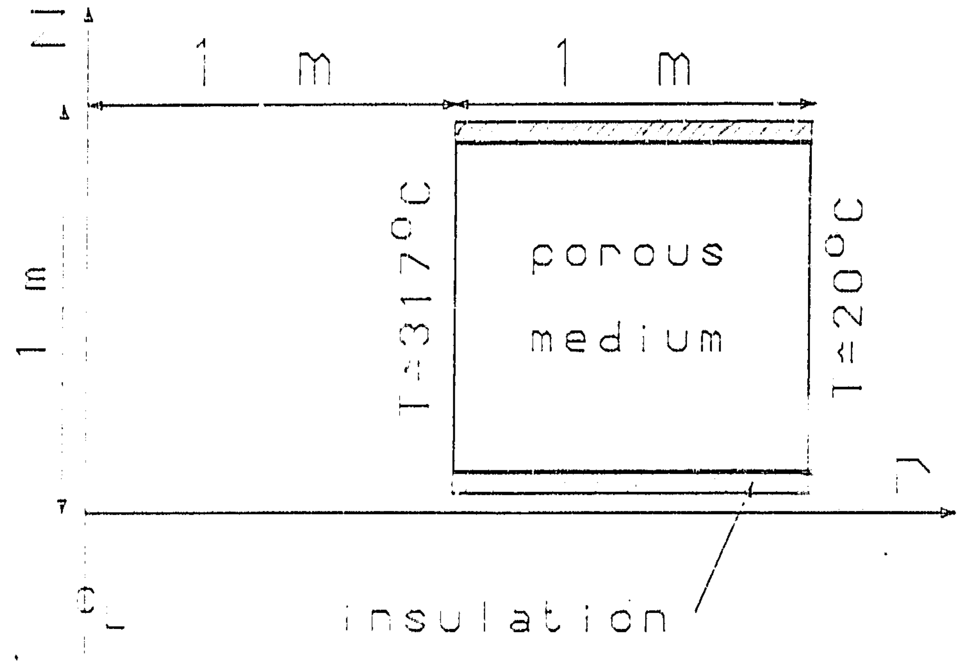

Fig. 7

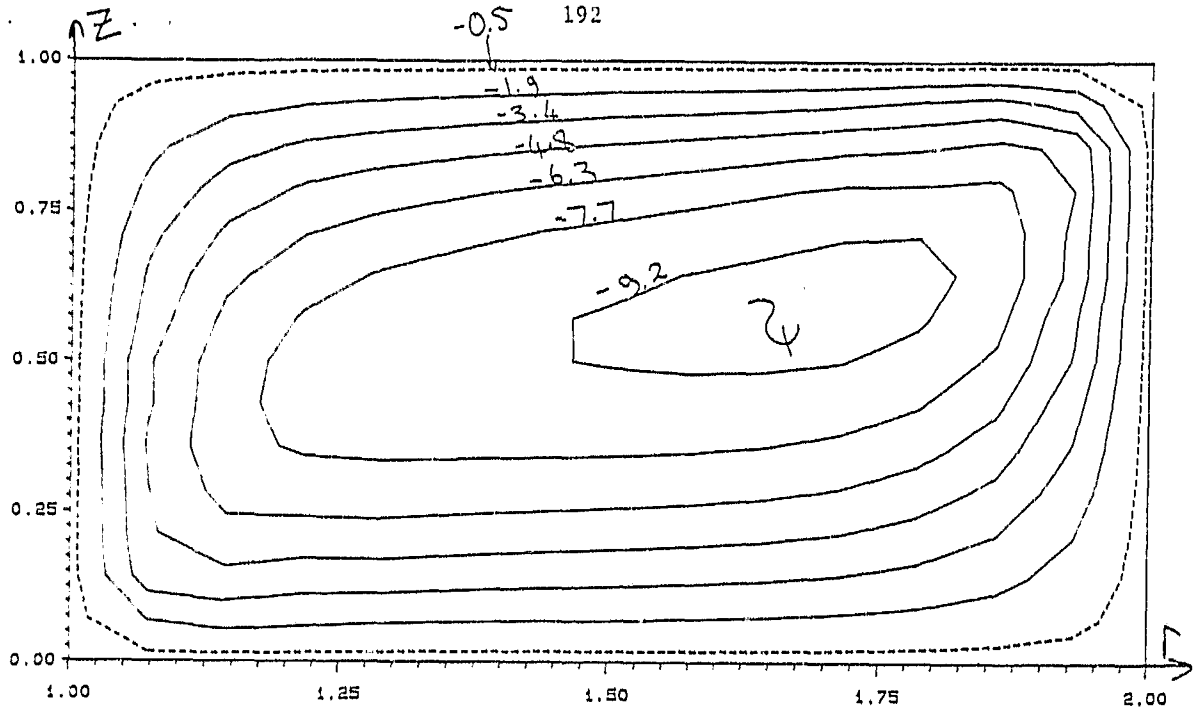
Z
-0.5
-1.9
-3.4
-4.8
-6.3
-7.7
-9.2
ψ
1.00
0.75
0.50
0.25
0.00
1.00
1.25
1.50
1.75
2.00
r

Fig. 8

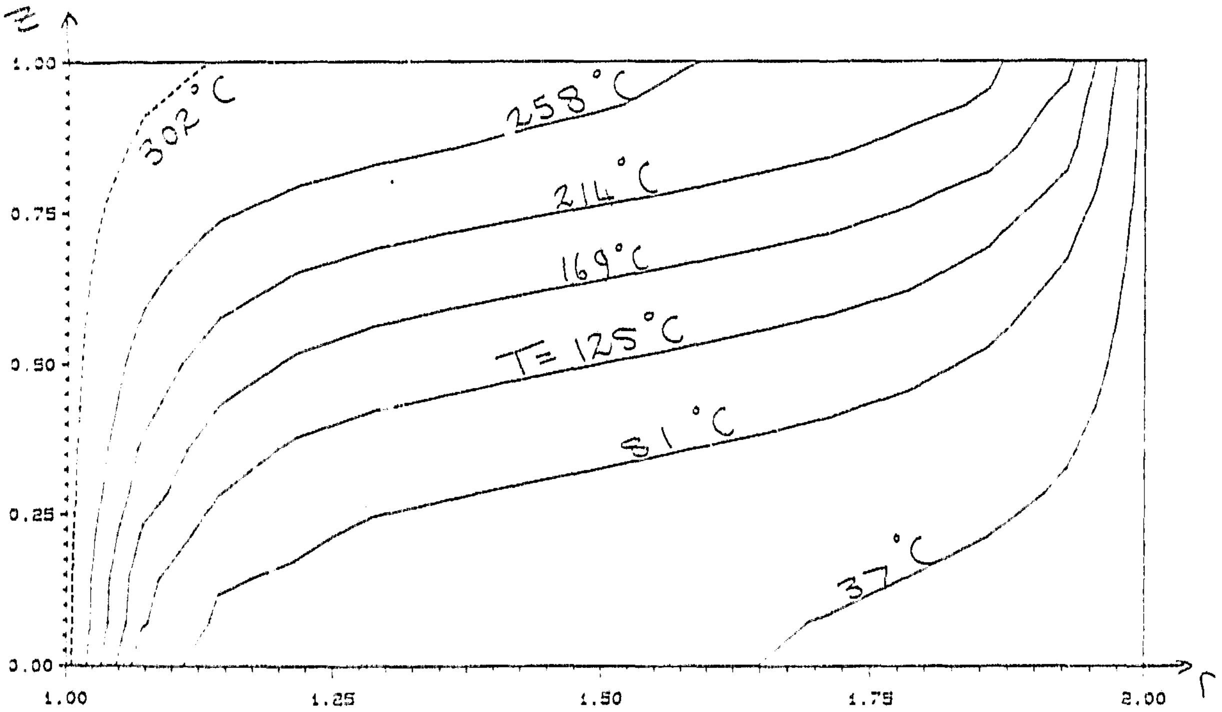
Z
302°C
258°C
214°C
169°C
T= 125°C
81°C
37°C
1.00
0.75
0.50
0.25
0.00
1.00
1.25
1.50
1.75
2.00
r

Fig. 9

3.5 Summary

The four problem types described above are merely representative of the range of problems soluble by 'PDFELM'. A more general description of solvable equations is given in section 1.

4 Operational Details

4.1 Data structure

The data structure of 'PDFELM' has been built up (considerably) from that of a 'NAG' example program. Variable and common-block names are self explanatory, and far too numerous to list here. The significance and format of user supplied data (element geometry, topology etc.) are detailed in section 4.3, in order to assist the user to set up the required data file.

4.2 General program structure

Program, subprogram and function program names are self explanatory. The main program and control subroutines are kept in a file called 'PDFELM FORTRAN', and the service and output control routines are kept in 'PDSUBS FORTRAN'. The reason for this is that normally only service routines (such as that which defines the equations to be solved, and boundary conditions) need to be edited and compiled by the user. This separation saves compilation time.

All subroutines of interest to the general user are listed and described in the 'FEPDE' manual (Anderson, 1986).

4.3 Format of the nodal data file

```
        no. of nodes          spatial dimension       no. of equations
                  (list of nodes - format as follows)
        node number           x-coord    y-coord ...... as reqd.
        element type          no. of elements         nodes per element
                 (list of element topology - format as follows)
        element number        node1  node2  node3  node4  ....
    __
    |   no. of boundary condn. lists for variable 1 *
    |
    |  __
    | | bndry. condn          nodes per side          number of nodes
    | |     type                                      in this list
    | | list of bndry. nodes in groups of ten
    | |__
    |
    |  inner  bracket  repeated  -  see  *
    |__
        outer bracket repeated (per variable as req'd)

        no. of lists of 'extraction' nodes  +
       __
      | extrctn. var.         no. of extrctn. nodes   (list of nodes)
      |__
                              bracket repeated  - see +

        flag for manual check of convergence criteria

        list of parameters req'd by problem.

        ε0     εmax       no. of      flag for     flag for     task
                          contin'n    trace of     contin'n#    flag
                          steps       convgnce.

        start. time           end time                no. of t steps

        flag for SYSM characteristics (time or Φ dependent ?)
```

notes:

- 'Extraction' nodes. Although values at all nodes are printed at each step, it is often desirable to extract values at specific nodes (eg. centreline temperatures) and print them to a separate file. This facility allows such manipulations to be done conveniently

- #Continuation. The continuation method (see de Villiers, 1984) may be implemented if necessary. Parameters ϵ_o, ϵ_{max} and the number of continuation steps are essential data if this option is selected.

- flags set to zero cause bypass of feature

- task flag at present takes on values of 0,1,2,3,4.
  0 ⇒ single steady-state simulation
  1 ⇒ fit parameters to a model by minimization of sum of squared errors. This requires experimental data and means of interpolation.
  4 ⇒ unsteady state simulation.

For purposes of clarification, a sample nodal datafile is given below. This datafile is for (see fig. 10 for details):

- steady state solution (task flag=0) of a single equation over a domain of square cross section, using 9 quadrilateral elements of the 8 noded type.

- a Dirichlet (type 1) boundary condition is applied along the bottom edge (y=0); derivative (type 2) boundary conditions are applied along the remaining edges.

- Function values at the nodes on the positive-slope diagonal (bottom-left to top right) are extracted, hence '1' list.

- Convergence criteria checking is automatic.

- Convergence is traced by screen output.

- Continuation is employed, with ε advanced from 0 to 1 in 7 steps.

- Time stepping data and SYSM characteristics are irrelevant in this case.

SAMPLE DATA FILE

```
     9          2          1
     1          0.0        0.0
     2          0.166      0.0
     3          0.333      0.0
                -
                -
                -
    38          0.666      1.0
    39          0.833      1.0
    40          1.000      1.0

 1      1    8     12    13    14    9     3     2
 2      3    9     14    15    16   10     5     4
 3      5   10     16    17    18   11     7     6
        .
        .
        .
 9     27   32     38    39    40    33   29    28

 2
 2   3   19
             1    8   12   19   23   30   34   35   36   37
             38  39   40   33   29   22   18   11    7
 1   3    5
             2    3    4    5    6
 1
 1   4       1   14   27   40
 param1    param2     param3  ....
 0
 0      1.0  7    1     1    0
  (SYSM characteristics flag)
```

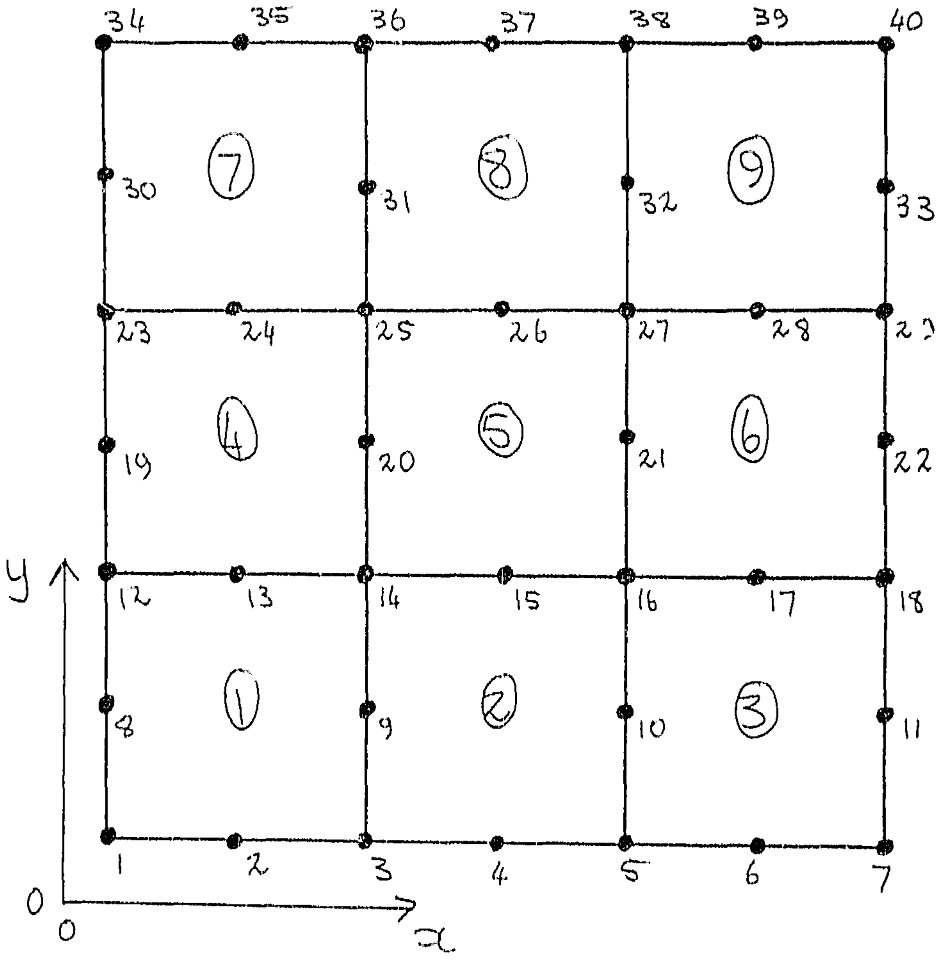
34
35
36
37
38
39
40
30
7
31
8
32
9
33
23
24
25
26
27
28
29
19
4
20
5
21
6
22
y
12
13
14
15
16
17
18
8
1
9
2
10
3
11
1
2
3
4
5
6
7
0
0
x

Fig. 10

REFERENCES

Anderson P., 1986. *Description of 'FEPDE' - A User's Manual.* Dept. of Chemical Engineering, University of the Witwatersrand, Johannesburg.

Anderson P., 1987. *Anisotropic Transfer of Thermal Energy in Porous Media*, PhD progress report, Dept. Of Chemical Engineering, University of the Witwatersrand, Johannesburg.

Bejan A., 1984. *Convection Heat Transfer*, John Wiley & Sons, New York.

De Villiers N., 1984. *Continuation methods for the solution of nonlinear problems in engineering science*, PhD thesis, University of the Witwatersrand.

Norrie D., de Vries G. 1973. *The Finite Element Method*, Academic Press, New York.

APPENDIX C

MATHEMATICAL STATEMENT OF BOUNDARY CONDITION ON INCLINED SURFACE

We consider the symmetrical half of the frustum, in cartesian coordinates r^* and Z, with height H, base length R and top length R_1. The side angle to the horizontal is α. We wish to consider boundary conditions on the inclined surface defined by:

$$Z = \frac{H}{R-R_1} r^* + \left(\frac{-H\,R}{R-R_1} \right) \qquad (C.1)$$

Now let us consider the temperature boundary condition as an example:

$$k_e \frac{\partial T}{\partial n} = h\,(T-T_a) \qquad (C.2)$$

We write this as:

$$\frac{\partial T}{\partial n} = \left[\frac{\partial T}{\partial Z}\frac{d\,Z}{d\,n} + \frac{\partial T}{\partial r^*}\frac{d\,r^*}{d\,n} \right] = \frac{h}{k_e}\,(T-T_a) \qquad (C.3)$$

Now,

$$\frac{d\,Z}{d\,n} = \cos\alpha \qquad \frac{d\,r^*}{d\,n} = \sin\alpha \qquad (C.4)$$

and we express cos α and sin α in terms of R, R_1 and H.

Non-dimensionalizing we obtain:

$$\frac{\partial\theta}{\partial z}\,\frac{(R-R_1)}{\left[(R-R_1)^2+H^2\right]^{1/2}} + \frac{\partial\theta}{\partial r}\,\frac{H}{\left[(R-R_1)^2+H^2\right]^{1/2}} = Bi\,\theta \qquad (C.5)$$

we let:

$$A = \frac{R-R_1}{R} \qquad B = \frac{H}{R} \qquad (C.6)$$

and then we can write our boundary condition in terms of these two geometric parameters:

$$A\frac{\partial\theta}{\partial z} + B\frac{\partial\theta}{\partial r} = \left(A^2 + B^2 \right)^{0.5} Bi\,\theta \qquad (C.7)$$

Eq. (C.7) is valid for the region:

$$\begin{aligned} R - AR \le r \le R \\ 0 \le z \le BR \end{aligned} \qquad (C.8)$$

APPENDIX D

IMPLEMENTATION OF THE GALERKIN METHOD

In this appendix a description is given of how the Galerkin method is implemented in section 3.5. This description complements the formal description of the Galerkin method given in section 3.3. In this appendix a less formal explanation is given, so that the details of the implementation are clearer.

In words the essence of the Galerkin method can be stated: "substitute the suitable trial functions into the differential equations, multiply by the weighting functions and integrate over the domain". In section 3.5 the trial functions have been given. The weighting function is:

$$\exp(- i (rkx + sly)) \qquad (D.1)$$

The general term that arises when the above procedure is applied looks like:

$$\int_{\infty}^{\infty}\int_{\infty}^{\infty}\int_{0}^{1} B_{pqh} \exp(i ([p - r]kx + [q - s]ly)) \\ X \sin (g\pi z) \sin (h\pi z) \, dx \, dy \, dz \qquad (D.2)$$

B_{pqh} is not a function of (x,y,z) and can be taken out of the integral sign. To make matters clearer the integrations over (x,y,z) will be considered separately.

From the orthogonality of the sine function on this domain:

$$\int_{0}^{1} \sin (g\pi z) \sin (h\pi z) \, dz = \frac{1}{2} \qquad h = g$$
$$= 0 \qquad h \neq g \qquad (D.3)$$

Now we will consider integration with respect to x. Obviously we could equally well have considered integration with respect to y. Because p and r are integers one can write:

$$p - r = n \qquad (D.4)$$

where n is also an integer.

The following integral must be considered:

$$\int_{\infty}^{\infty} \exp(inkx)\, dx \qquad \begin{cases} n = 0 \\ n \neq 0 \end{cases} \qquad (D.5)$$

For the case $n \neq 0$ the integral can be expressed as:

$$\int_{\infty}^{\infty} \cos\ (nkx) + i\ \sin\ (nkx)\ dx \qquad (D.6)$$

The sine function is odd, and the integral of such a function over a domain symmetric about 0 is 0, hence what remains is:

$$\int_{\infty}^{\infty} \cos\ (nkx)\ dx \qquad (D.7)$$

Now we must consider our infinite domain more carefully. It is required that there be a whole number of convection cells in the domain, and on the boundary of a cell $\partial\theta/\partial x=0$. Examination of the cosine function shows that this condition is fulfilled for half periods symmetric about 0. Thus without loss of generality one can replace the limits in the integration with:

$$\int_{-\frac{m\pi}{nk}}^{\frac{m\pi}{nk}} \cos\ (nkx)\ dx \qquad (D.8)$$

where m is an integer. This integral is equal to zero, and so we have shown that for $n \neq 0$, i.e. $p \neq r$, the integral Eq. (D.5) vanishes. Now we consider the case $n=0$. In that case we must evaluate:

$$\int_{\infty}^{\infty} 1\ dx \qquad (D.9)$$

If the limits of the integration are replaced as was done for Eq. (D.8), Eq. (D.9) is equal to

$$\int_{-\frac{m\pi}{nk}}^{\frac{m\pi}{nk}} 1\ dx = \frac{2m\pi}{nk} \qquad (D.10)$$

This integral will appear in front of every term in each equation to which the Galerkin method is applied. The situation for the integration over y is analogous. Thus it can be seen that the only terms resulting from the integration which are non-zero are those for which pqh=rsg. That is why the coefficient B_{rsg} appears in the set of algebraic equations resulting from the Galerkin analysis.

Now we consider terms in the differential equation which do not depend on (x,y,z). Such a term appears when the temperature exponential is approximated by a quadratic expansion. In this case one can easily show that the integral vanishes for (r,s)≠0, and that the z integration is:

$$\int_0^1 \sin (g\pi z)\, dz = - \frac{1}{g\pi} \left[(-1)^g - 1 \right] \qquad \text{(D.11)}$$

$$\int_{-\frac{m\pi}{nk}}^{\frac{m\pi}{nk}} \exp(-i\, rkx)\, dz = \begin{cases} 0 & r \neq 0 \\ \frac{2m\pi}{nk} & r = 0 \end{cases} \qquad \text{(D.12)}$$

Because terms like 2mπ/(nk) (and similar terms resulting from the integration over y) appear in front of every term of the algebraic equations they can be divided out. Thus we can see that the term which results from applying the Galerkin method to constant terms in the differential will be given by Eq. (D.11) for r=s=0.

Definitions of functions used in Galerkin analysis in sections 3.3-3.7

F(a,z) used in equation (3.33) is found by solvirg

$$\frac{d^2F(a,z)}{d z^2} - a^2 F(a,z) = Ra^* \sin(\pi z) \qquad \text{(D.13)}$$

One finds:

$$F(a,z) = c_1 \left[e^{a z} - e^{-a z} \right] + c_2 \sin(\pi z) \qquad \text{(D.14)}$$

where:

$$c_1 = \frac{c_2\ \pi}{a\,(e^a + e^{-a})} \qquad \text{(D.15)}$$

$$c_2 = \frac{Ra^*}{\pi^2 + a^2}$$

$F_h(\lambda,z)$ used in equation (3.36) is found by solving:

$$\frac{d^2F_h(\lambda,z)}{d\,z^2} \cdot \lambda^2 F_h(\lambda,z) = Ra^* \sin(h\pi z) \qquad (D.16)$$

For $\lambda \neq 0$:

$$F_h(\lambda,z) = c_1 \left[e^{\lambda z} - e^{-\lambda z} \right] + c_2 \sin(h\pi z) \qquad (D.17)$$

where:

$$c_1 = \frac{c_2\ \pi\ h}{\lambda\ (e^{\lambda} + e^{-\lambda})} \qquad (D.18)$$

$$c_2 = \frac{Ra^*}{\pi^2 h^2 + \lambda^2} \qquad (D.19)$$

For $\lambda = 0$:

$$F_h(\lambda,z) = \frac{Ra^*\ z}{\pi\ h} + \frac{Ra^*}{\pi^2\ h^2} \sin(\pi h z) \qquad (D.21)$$

The integrals $a(f,g,h,\lambda)$-$b(f,g,h,\lambda)$ and $c(f,g,h)$ used in section 3.5 are defined as follows:

$$a\ (f,g,h,\lambda) = \int_0^1 \sin(f\pi z)\ \sin(g\pi z)\ \frac{d\ F_h(\lambda,z)}{d\ z}\ dz \qquad (D.22)$$

$$b\ (f,g,h,\lambda) = \int_0^1 \pi\ f\ \cos(f\pi z)\ \sin(g\pi z)\ F_h(\lambda,z)\ dz \qquad (D.23)$$

$$c\ (f,g,h) = \int_0^1 \sin(h\pi z)\ \sin(f\pi z)\ \sin(g\pi z)\ dz \qquad (D.24)$$

APPENDIX E

ANALYSIS OF THE INTERIOR MODEL WITH A FIRST ORDER REACTION

In this appendix the infinite coal layer model of Chapter 3 is examined, with the reaction considered to be of first order. The system of algebraic equations is derived, from which steady-state solutions can be obtained, and a method for examining the stability of these solutions is presented. As will become clear, solution of the finite layer model with a first reaction is much more difficult than the zero order reaction case, both in terms of programming complexity and computational speed. Because it is known that the zero order reaction approximation is good at ignition, where the main interest of this work lies, little attention was given to the more complex task of solving the first [illegible] case

The model for w[illegible] [illegible]ons are sought in this appendix is exactly the same as [illegible] in Chapter 3, apart from the fact that the chemisorptio[illegible] w assumed to be of first order. Precisely the same form [illegible] [illegible]nalysis is followed as was used in Chapter 3.

We seek solutions of the form:

$$w = \sum_h \sum_{p,q} B_{pqh} \exp \left\{ i(pkx + qly) \right\} \sin (\pi hz) \qquad (E.1)$$

$$\chi = \sum_h \sum_{p,q} B_{pqh} \exp \left\{ i(pkx + qly) \right\} F_h (\lambda, z) \qquad (E.2)$$

$$T = \sum_h \sum_{p,q} C_{pqh} \exp \left\{ i(pkx + qly) \right\} \cos \left[(h - \tfrac{1}{2}) \pi z \right] \qquad (E.3)$$

where

$$\lambda^2 = (pk)^2 + (ql)^2 \qquad (E.4)$$

in which k and l are the x- and y-components of the wave number a.

The limits on the integer summation indices are:

$$-\infty < (p,q) \; [illegible] \; ; \; 1 \le h < \infty \qquad (E.5)$$

The form of $F_h(\lambda,z)$ is found by substituting (E.1) and (E.2) into Eq. (3.12) and solving the resulting ordinary differential equation. $F_h(\lambda,z)$ is given in Eqs. (E.28) and (E.31).

To obtain our system of algebraic equations Eqs. (E.1)-(E.3) are substituted into Eqs. (3.13)-(3.14), each equation is multiplied by its weighting function and integrated over the layer. This results in an infinite set of equations in the coefficients B_{rsg} and C_{rsg}. As discussed section 3.3 the exponential function of temperature is approximated by a second order Taylor expansion about w=0, which is expected to under-estimate the reaction rate dependence on temperature. In steady-state form the infinite set of equations is:

$$0 = -\frac{1}{2} B_{rsg} (g^2\pi^2 + \nu^2) - \sum_{h,f} \sum_{\substack{p+t=r \\ q+u=s}} B_{pqh} B_{tuf} \Big\{ \lambda^2 b (f,g,h,\lambda) -$$

$$(ptk^2 + qul^2) a (f,g,h,\lambda) \Big\} + \frac{FK^*}{\sqrt{Ra^*}} \Big\{ \frac{1}{2} B_{rsg} + \frac{(1 (-1)^g)}{g \pi} \Big|_{r=s=0} +$$

$$\sum_f C_{rsf} a_1 + \sum_{h,f} \sum_{\substack{p+t=r \\ q+u=s}} \Big[B_{pqh} C_{tuf} a_2 + \frac{1}{2} B_{pqh} B_{tuf} a_3 \Big] +$$

$$\frac{1}{2} \sum_{h,f,e} \sum_{\substack{p+t+m=r \\ q+u+n=s}} B_{pqh} B_{tuf} C_{mne} a_4 \Big\} \qquad \text{(E.6)}$$

$$0 = -\frac{Le}{2} C_{rsg} ((g - \frac{1}{2})^2\pi^2 + \nu^2) + \sum_{h,f} \sum_{\substack{p+t=r \\ q+u=s}} B_{pqh} C_{tuf} \Big\{ \lambda^2 d (f,g,h,\lambda)$$

$$+ (ptk^2 + qul^2) c (f,g,h,\lambda) \Big\} - \frac{FK^*}{\beta\gamma\sqrt{Ra^*}} \Big\{ \frac{1}{2} C_{rsg} + \frac{\sin [(h - \frac{1}{2})\pi]}{\pi (h - \frac{1}{2})} \Big|_{r=s=0} +$$

$$\sum_f B_{rsf} a_5 + \sum_{h,f} \sum_{\substack{p+t=r \\ q+u=s}} \Big[B_{pqh} C_{tuf} a_6 + \frac{1}{2} B_{pqh} B_{tuf} a_7 \Big] +$$

$$\frac{1}{2} \sum_{h,f,e} \sum_{\substack{p+t+m=r \\ q+u+n=s}} B_{pqh} C_{tuf} B_{mne} a_8 \Big\} \qquad \text{(E.7)}$$

where

$$\nu^2 = (rk)^2 + (sl)^2 \qquad \text{(E.8)}$$

The integrals a_1-a_8 and $a(f,g,h,\lambda)$-$d(f,g,h,\lambda)$ are given at the end of this appendix. In order to limit the solutions of this set of equations to those which are physically reasonable the infinite set of equations is truncated such that modes are neglected for which:

$$g^2 + \frac{3}{4} r^2 + \frac{1}{4} s^2 > \Lambda^2 + 1 \qquad (E.9)$$

where Λ is an integer truncation number. This is the form of truncation used by Tveitereid,1977 and Tveitereid and Palm,1976. The components of the wave number satisfy:

$$k^2 + l^2 = 4 \; l^2 = a^2 \qquad (E.10)$$

Only real coefficients where

$$B_{rsg} = B_{-r-sg} = B_{r-sg} = B_{-rsg} \qquad (E.11)$$

are considered, and similarly for C_{rsg}.

These restrictions allow hexagons, rolls and squares to be planforms as these are the most likely solutions (Tveitereid,1977, Tveitereid and Palm,1976) and it is known that these are physically reasonable solutions (Chandrasekhar,1961, White,1988, Busse and Frick,1985).

As can be seen, solution of this model requires twice as many equations to be solved as in the case of a zero order reaction. This has a very pronounced effect on the computational times, making the examination of this first order case prohibitively time-consuming with the computer facilities that were available for this study. The solution time for the nonlinear equations is approximately proportional to N^2, where N is the number of equations to be solved.

Stability of the steady-state solutions

In this section a method is described, based on the approach of Kimura et al.,1987, for examining the stability of the steady-state solutions to small perturbations. It was found that the numerical routines used to evaluate the eigenvalues required in this method were very unreliable. We consider small perturbations to the steady-state Fourier coefficients and assume that the perturbations are sufficiently small to allow a linear thoery to be formulated. The perturbations have an exponential time dependence.

In order to perform the stability analysis, the concentration equation, Eq. (E.7) is written in unsteady-state form. The left hand side of the equation is replaced by $\epsilon\ \partial T/\partial\tau$. The stability analysis of the first order reaction case is done in exactly the same way as for the zero order reaction case in section 3.7.

We write

$$B_{pqh} = B^{(o)}_{pqh} + \delta^{B}_{pqh} \exp(\sigma_{pqh}\ \tau) \qquad (E.12)$$

$$C_{pqh} = C^{(o)}_{pqh} + \delta^{C}_{pqh} \exp(\sigma_{pqh}\ \tau) \qquad (E.13)$$

Where

$$\sigma = \sigma_r + i\ \sigma_i \qquad (E.14)$$

$$\zeta^{B}_{pqh} = \delta^{B}_{pqh} \exp(\sigma_{pqh}\ \tau) \qquad (E.15)$$

$$\zeta^{C}_{pqh} = \delta^{C}_{pqh} \exp(\sigma_{pqh}\ \tau) \qquad (E.16)$$

The $(B^{(o)}_{pqh}, C^{(o)}_{pqh})$ are the steady-state coefficients. The trial functions are obtained by substituting Eqs. (E.12)-(E.13) into Eqs. (E.1)-(E.3) to give:

$$w = \sum_{h} \sum_{p,q} \left[B^{(o)}_{pqh} + \zeta^{B}_{pqh} \right] \exp\left\{ i(pkx + qly) \right\} \sin(\pi hz) \qquad (E.17)$$

$$T = \sum_{h} \sum_{p,q} \left[C^{(o)}_{pqh} + \zeta^{C}_{pqh} \right] \exp\left\{ i(pkx + qly) \right\} \cos\left[(h - \tfrac{1}{2})\pi z\right] \qquad (E.18)$$

$$\chi = \sum_{h} \sum_{p,q} \left[B^{(o)}_{pqh} + \zeta^{B}_{pqh} \right] \exp\left\{ i(pkx + qly) \right\} F_h(\lambda, z) \qquad (E.19)$$

The form of $F_h(\lambda,z)$ is found in the same way as in section 3.5. Eqs. (E.17)-(E.19) are then substituted into Eqs. (3.13)-(3.14) and the Galerkin method applied. This yields the following equations in the ζ_{pqh}:

$$\frac{\wp}{2}\,\dot{\zeta}^{B}_{rsg} = -\,\frac{1}{2}\;\zeta^{B}_{rsg}\,(g^{2}\pi^{2}+\nu^{2}) - \sum_{h,f}\;\sum_{\substack{p+t=r\\ q+u=s}} \left(\zeta^{B}_{tuf}B^{(o)}_{pqh} + \zeta^{B}_{pqh}B^{(o)}_{tuf}\right) X$$

$$\left\{\lambda^{2}\;b(f,g,h,\lambda)\;-^{2}\;\;(ptk\;+qul^{2})\;a\;(f,g,h,\lambda)\right\} + \frac{FK^{*}}{\sqrt{Ra^{*}}}\left\{\;\frac{1}{2}\,\zeta^{B}_{rsg}\right.$$

$$+\sum_{f}\zeta^{C}_{rsf}\;a_{1}$$

$$+\sum_{h,f}\;\sum_{\substack{p+t=r\\ q+u=s}}\left[\;\frac{1}{2}\;\left(\zeta_{tuf}B^{(o)}_{pqh} + \zeta_{pqh}B^{(o)}_{tuf}\right)\;a_{3} + \left(\zeta^{B}_{pqh}C^{(o)}_{tuf} + \zeta^{C}_{tuf}B^{(o)}_{pqh}\right)\right.$$

$$\left. X\;a_{2}\;\right]$$

$$+\;\frac{1}{2}\sum_{h,f,l}\;\sum_{\substack{p+t+m=r\\ q+u+n=s}}\left(\zeta^{C}_{mnl}\;B^{(o)}_{pqh}\;\;B^{(o)}_{tuf} + \zeta^{B}_{pqh}\;\;B^{(o)}_{tuf}\;C^{(o)}_{mnl} + \zeta^{B}_{tuf}\;\;B^{(o)}_{pqh}\;C^{(o)}_{mnl}\;\right)$$

$$\left. X\quad a_{4}\;\right\} \tag{E.20}$$

$$\epsilon\frac{\varphi}{2}\,\dot{\zeta}^{C}_{rsg} = -\frac{Le}{2}\;\zeta^{C}_{rsg}\;(g^2\pi^2 + \nu^2) - \sum_{h,f}\;\sum_{\substack{p+t=r\\q+u=s}} \left[\zeta^{C}_{tuf}B^{(o)}_{pqh} + \zeta^{B}_{pqh}C^{(o)}_{tuf}\right] X$$

$$\left\{-\lambda^2 d(f,g,h,\lambda) - \quad (ptk^2 + qul^2)\;c\;(f,g,h,\lambda)\right\} - \frac{FK^*}{\gamma\beta\sqrt{Ra^*}}\left\{\frac{1}{2}\,\zeta^{C}_{rsg}\right.$$

$$+\sum_{f}\zeta^{B}_{rsf}\;a_5$$

$$+\sum_{h,f}\;\sum_{\substack{p+t=r\\q+u=s}}\left[\frac{1}{2}\left[\zeta^{B}_{tuf}B^{(o)}_{pqh} + \zeta^{B}_{pqh}B^{(o)}_{tuf}\right]a_7 + \right.$$

$$\left.\left[\zeta^{B}_{pqh}\;C^{(o)}_{tuf} + \zeta^{C}_{tuf}\;B^{(o)}_{pqh}\right]\;a_6\right\}$$

$$+\frac{1}{2}\sum_{h,f,l}\;\sum_{\substack{p+t+m=r\\q+u+n=s}}\left[\zeta^{C}_{mnl}\;B^{(o)}_{pqh}\;B^{(o)}_{tuf} + \zeta^{B}_{pqh}\;B^{(o)}_{tuf}\;C^{(o)}_{mnl} + \zeta^{B}_{tuf}\;B^{(o)}_{pqh}\;C^{(o)}_{mnl}\right]$$

$$\left. X\quad a_4\right\} \tag{E.21}$$

where:

$$\nu^2 = (rk)^2 + (sl)^2 \tag{E.22}$$

Noting that:

$$\dot{\zeta}_{rsg} = \sigma_{rsg}\;\zeta_{rsg} \tag{E.23}$$

enables us to write the left-hand sides of Eqs. (E.20)-(E.21) as:

$$\frac{\varphi}{2}\,\sigma_{rsg}\zeta^{B}_{rsg} \tag{E.24}$$

$$\frac{\epsilon}{2}\,\sigma_{rsg}\zeta^{C}_{rsg} \tag{E.25}$$

Thus it can be seen that Eqs. (E.20)-(E.21) are an eigenvalue problem of the form

$$\sigma \zeta = A \zeta \qquad (E.26)$$

The behaviour of the eigenvalues σ determines the stability of the steady-state solutions. If any of the eigenvalues has a positive real part, then that steady-state solution is not stable to the small perturbation, as that perturbation will grow in time. If all the real parts of the eigenvalues are negative, then the solution is stable. If the real part of one of the eigenvalues is zero and the imaginary part non-zero then the solution is oscillatory.

As discussed in section 3.5, [illegible] as found that the commercial routines that were used to calculat[illegible] eigenvalues were very unreliable and unfortunately this method [illegible] be used to determine the stability of the steady-state soluti[illegible]

Definitions of functions used in the above Galerkin analysis

$F_h(\lambda,z)$ used in equation (E.2) is found by solving:

$$\frac{d^2F_h(\lambda,z)}{d\,z^2} - \lambda^2 F_h(\lambda,z) = Ra^* \sin(h\pi z) \qquad (E.27)$$

For $\lambda \neq 0$:

$$F_h(\lambda,z) = c_1 \left(e^{\lambda z} - e^{-\lambda z} \right) + c_2 \sin(h\pi z) \qquad (E.28)$$

where:

$$c_1 = \frac{c_2 \ \pi h}{\lambda \ (e^{\lambda} + e^{-\lambda})} \qquad (E.29)$$

$$c_2 = -\frac{Ra^*}{\pi^2 h^2 + \lambda^2} \qquad (E.30)$$

Fo[illegible]:

$$F_h(\lambda,z) = \frac{Ra^* z}{\pi h} + \frac{Ra^*}{\pi^2 h^2} \sin(\pi hz) \qquad (E.31)$$

The integrals a(f,g,h,λ)-d(f,g,h,λ) and a1-a8 are defined as follows.

$$a\ (f,g,h,\lambda) = \int_0^1 \sin\ (f\pi z)\ \sin\ (g\pi z)\ \frac{d\ F_h(\lambda,z)}{d\ z}\ dz \qquad (E.32)$$

$$b\ (f,g,h,\lambda) = \int_0^1 \pi\ f\ \cos\ (f\pi z)\ \sin\ (g\pi z)\ F_h(\lambda,z)\ dz \qquad (E.33)$$

$$c\ (f,g,h,\lambda) = \int_0^1 \cos\ [(f - \tfrac{1}{2})\pi z]\ \cos[(g - \tfrac{1}{2})\pi z]\ \frac{d\ F_h(\lambda,z)}{d\ z}\ dz \qquad (E.34)$$

$$d\ (f,g\ h,\lambda) = \int_0^1 \pi\ (f - \tfrac{1}{2})\ \sin[(f - \tfrac{1}{2})\pi z]\ \cos[(g - \tfrac{1}{2})\pi z)\ F_h(\lambda,z)\ dz \qquad (E.35)$$

$$a_1 = \int_0^1 \sin\ (g\pi z)\ \cos[(f - \tfrac{1}{2})\pi z)\ dz \qquad (E.36)$$

$$a_2 = \int_0^1 \sin\ (h\pi z)\ \cos[(f - \tfrac{1}{2})\pi z]\ \sin\ (g\pi z)\ dz \qquad (E.37)$$

$$a_3 = \int_0^1 \sin\ (h\pi z)\ \sin\ (f\pi z)\ \sin\ (g\pi z)\ dz \qquad (E.38)$$

$$a_4 = \int_0^1 \sin\ (h\pi z)\ \sin\ (f\pi z)\ \mathrm{oos}[(1 - \tfrac{1}{2})\pi z]\ dz \qquad (E.39)$$

$$a_5 = \int_0^1 \sin\ (f\pi z)\ \cos[(g - \tfrac{1}{2})\pi z]\ dz \qquad (E.40)$$

$$a_6 = \int_0^1 \sin\ (n\pi z)\ \cos[(f - \tfrac{1}{2})\pi z]\ \cos[(g - \tfrac{1}{2})\pi z]\ dz \qquad (E.41)$$

$$a_7 = \int_0^1 \sin\ (h\pi z)\ \sin\ (f\pi z)\ \cos\ [(g - \tfrac{1}{2})\pi z)\ dz \qquad (E.42)$$

$$a_8 = \int_0^1 \sin\ (h\pi z)\ \sin\ (1\pi z)\ \cos[(f - \tfrac{1}{2})\pi z]\ \cos[(g - \tfrac{1}{2})\pi z]\ dz \qquad (E.43)$$

The following Fortran 77 program (BED3NEW FORTRAN) calculates the steady or unsteady-state solutions to the first order reaction infinite layer model described in this appendix.

```
**************************************************************************
*  CALCULATE THE FOURIER COEFFICIENTS FOR THE STABILITY                  *
*  ANALYSIS OF FLOW IN AN INFINITE COAL BED                              *
*  ASSUMING A FIRST ORDER REACTION                                       *
*  WITH OPEN TOP AND QUADRATIC EXPANSION OF THE EXPONENTIAL              *
*  [illegible] NUMERICAL INTEGRATIONS                                    *
**************************************************************************
Can use either DIVPAG for the unsteady problem or C05NBF for
the steady problem
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 K, L
REAL*8 WA([illegible]), F(200)
REAL*8 Y(200), YINIT(200)
REAL*8 PARAM(50), AIVPAG(1,1)
COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
COMMON /COEFF/ BDERV(-20:+1,-20:+1,20), B(-20:+1,-20:+1,20),
               CDERV(-20:+1,-20:+1,20), C(-20:+1,-20:+1,20)
COMMON /WAVE/ K, L, M
COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
COMMON /TRC/ ITRC(-20:+1,-20:+1,20)
COMMON /WORKSP/ RWKSP
REAL*8 RWKSP([illegible])
EXTERNAL FCN
EXTERNAL [illegible]
DATA IN /2/, NOUT /[illegible]/, ITERM /7/, ICOEF /8/, [illegible] /9/, INIT /10/
DATA NITMAX /[illegible]/
[illegible] ensure that there is sufficient workspace for DIVPAG
[illegible]
CALL IWKIN([illegible])
[illegible] parameters
  [illegible]
  PI=3.14159265358979D0
  [illegible]
Read truncation parameter, wave number and task flag for [illegible]
READ(IN,*) M, A, ITASK
Read physical parameters
READ(IN,*) DP, EPSI, EA, XK0, XKEQ, XMU
READ(IN,*) T0, RHO, GRAV, HTCAP, DELTAH, CHRL
READ(IN,*) IREAD
Read continuation parameters
READ(IN,*) [illegible], FKEND, NSTEPS
  PERM=DP*DP*EPSI**3/150.0D0/(1.0D0-EPSI)**2
  RA=GRAV*3.67D+5*T0*CHRL*RHO**2*PERM*HTCAP/XKEQ/XMU
  GAMMA=EA/8.314D0/T0
  BETA=DELTAH/0.029D0/T0/HTCAP/4.76D0
  THIELE=6.0D0*XK0*RHO*HTCAP*(1.0D0-EPSI)*CHRL*CHRL
         /(XKEQ*DP)*DEXP(-GAMMA)
  XLE=6.0D0*XK0*RHO**2*CHRL**2.5*HTCAP**1.5*DEXP(-GAMMA)
      /(XKEQ**1.5*(GRAV*T0*3.67D+5*EPSI**3/150.0D0/XMU)**0.5)
Calculate parameters for the positive exponential approximation
  RASTAR=RA/GAMMA
  RA=RASTAR
  FK=BETA*THIELE*GAMMA
  WRITE(NOUT,*)'FK', FK
calculate wave numbers
  L=SQRT(A*A/4.0D0)
  K=SQRT(A*A-L*L)
Maximum and minimum values for the subscripts
      MAXIR=0
      MAXIS=0
      MAXIG=1
      MINIR=0
      MINIS=0
      MINIG=1
    IF(IREAD.EQ.1) THEN
      READ(ICOEF,*)ICOUNT
      DO 11 I=1,ICOUNT
        READ(ICOEF,*)IR, IS, IG, B(IR, IS, IG)
        Y(I)=B(IR,IS,IG)
        YINIT(I)=Y(I)
        ISUB1(I)=IR
        ISUB2(I)=IS
        ISUB3(I)=IG
        IF(IR.GT.MAXIR) MAXIR=IR
        IF(IS.GT.MAXIS) MAXIS=IS
        IF(IG.GT.MAXIG) MAXIG=IG
        IF(IR.LT.MINIR) MINIR=IR
        IF(IS.LT.MINIS) MINIS=IS
        IF(IG.LT.MINIG) MINIG=IG
        IF(I.LE.ICOUNT/2) THEN
          ITRC(IR,IS,IG)=I
        END IF
11    CONTINUE
    ELSE
C --- ICOUNT counts the number of coefficients i.e. the number of
C --- equations  that satisfy the truncation and parity
C --- Set up coefficients for the energy equation
      ICOUNT=0
      NCOUNT=0
      DO 10 IR=-10, 10
        DO 10 IS=-10, 10
          DO 10 IG=1,10
C --- Check parity and truncation
            CALL TRNCHK(IR, IS, IG, M, ITFLG)
             IF(ITFLG.NE.0) GO TO 10
            CALL PARCHK(IR, IS, IPRFLG)
             IF(IPRFLG.NE.0) GO TO 10
            ICOUNT=ICOUNT+1
C --- Initialise B
C           IF((IABS(IS).GE.3).OR.(IABS(IR).GE.2)) THEN
            IF(IR.NE.0) THEN
              B(IR,IS,IG)=0.0D0
            ELSE
              B(IR,IS,IG)=1.0D-2
            END IF
C --- Copy B into dummy variables , Y is a vector
            Y(ICOUNT)=B(IR,IS,IG)
            YINIT(ICOUNT)=Y(ICOUNT)
            ISUB1(ICOUNT)=IR
            ISUB2(ICOUNT)=IS
            ISUB3(ICOUNT)=IG
            IF(IR.GT.MAXIR) MAXIR=IR
            IF(IS.GT.MAXIS) MAXIS=IS
            IF(IG.GT.MAXIG) MAXIG=IG
            IF(IR.LT.MINIR) MINIR=IR
            IF(IS.LT.MINIS) MINIS=IS
            IF(IG.LT.MINIG) MINIG=IG
      ITKG(IR,I   )=ICOUNT
  [illegible]INUE
 Set up coeffic[illegible]ts for the concentration equation
 DO 110 IR=-10
    DO 110 IS=  , 10
       DO 110 IG  ,10
  Check parity and truncation
        CALL TRNCHK(IR, IS, IG, M, ITFLG)
         IF(ITFLG.NE.0) GO TO 110
         CALL PARCHK(IR, IS, IPRFLG)
         IF(IPRFLG.NE.0) GO TO 110
        ICOUNT=ICOUNT+1
 Initialise C
        IF((IABS(IS).GE. ).OR.(IABS(IR).GE.2)) THEN
        IF(IR.NE.0) THEN
          C(IR,IS,IG)=0.0D0
        ELSE
          C(IR,IS,IG)=1.0D-2
        END IF
 Copy C into dummy variables , Y is a vector
        Y(ICOUNT)=C(IR,IS,IG)
        YINIT(ICOUNT)=Y(ICOUNT)
        ISUB1(ICOUNT)=IR
        ISUB2(ICOUNT)=IS
        ISUB3(ICOUNT)=IG
  CONTINUE
   IF
 WRITE(INIT, [illegible]
   DO   I=1,ICOUNT
  WRITE INIT,  ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
   NTINUE
   WRITE(ITERM,*)'NUMBER OF TERMS', ICOUNT
   Integrate the ODES or solve non-linear eqs equations
  IF(ITASK.EQ.0) THEN
  INTEGRATE O D E S USING DIVPAG
  PARAMETERS FOR DIVPAG
        READ(IN,*) TINIT, TFINAL, NTST
        TSTEP=(TFINAL-TINIT)/DFLOAT(NTSTEP)
        TSTART=TINIT
        TEND=TSTEP
        TOLR=1.0D-4
          METH=1 -> ADAMS METHOD    MITER=0 -> FUNCTIONAL ITERN.
          METH=2 -> stiff method
 Set parameters for DIVPAG to defaults
        CALL SSET(50, 0.0D0, PARA, 1)
        INORM=1
        METH=2
        MITER=0
        IDO=1
        PARA(10)=INORM
        PARA(12)=METH
        IF(NTSTEP.EQ.1) THEN
          N1=2
          N2=2
        ELSE
          N1=2
          N2=NTSTEP
        END IF
        DO 20 ITSTEP=N1, N2
 Steps closer at beginning
```

```
C          TEND=TINIT+(TFINAL-TINIT)*(1.0D0-DCOS(PI/2.0D0*
C    #          DFLOAT(ITSTEP-1)/DFLOAT(NTSTEP-1)))
C --- Steps closer at and
           TEND=TINIT+(TFINAL-TINIT)*DSIN(PI/2.0D0*
     #          DFLOAT(ITSTEP-1)/DFLOAT(NTSTEP-1))
           CALL DIVPAG(IDO, ICOUNT, TDERV, PDERV, AIVPAG, TSTART,
     #                 TEND, TOLR, PARA, Y)
           WRITE(4,*)'TEND,  IDO = ',TEND, IDO
           CALL TNEG(Y, ICOUNT)
           REWIND (UNIT=2)
           REWIND (UNIT=9)
           REWIND (UNIT=7)
           WRITE(NOUT,*)'TEND', TEND*CHRL**2/2.0E-4/3600.0/24.0,'DAY'
           WRITE(NOUT,*)'TAU', TEND, 'FK', FK
           WRITE(NOUT,*)'RA', RA*GAMMA, 'THIELE', THIELE
           WRITE(NOUT,*)'D', DD
           WRITE(NOUT,*)'RA*', RA  'FK', FK/DSQRT(RA)
           WRITE(NOUT,*)'WAVE NUMBERS', K, L
           WRITE(ICOUT,*)ICOUNT
           DO 30 I=1,ICOUNT
             WRITE(NOUT,*)'B(R,S,G)' ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
             WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
30         CONTINUE
           CALL PLOT(Y, ICOUNT)
           CALL TMAX(Y, ICOUNT, TEND)
C ---
20      CONTINUE
      END IF
      IF(ITASK.EQ.1) THEN
C ---   SOLVE EQS USING C05NBF
C ---   PARAMETERS FOR C05NBF
        ERRREL=DSQRT(X02AAF(0.0D0))
        LWA=70000
        IFAIL=1
C --- Start continuation taking smaller steps at the end
C       FKIN=FK
       DO 324 ICONT=1, NSTEPS
         FK=FKIN+(FKEND-FKIN)*DSIN(PI/2.0D0*DFLOAT(ICONT-1)
     #      /DFLOAT(NSTEPS-1))
        WRITE(ITERM,*)FK, ICONT
       CALL C05NBF (FUNC, ICOUNT, Y, F, ERRREL, WA, LWA, IFAIL)
       IF (IFAIL.GT.0) THEN
         WRITE(NOUT,*)'IFAIL', IFAIL
         WRITE(ITERM,*)'IFAIL'
         FNORM=0.0D0
         WRITE(ICOUT,*)ICOUNT
         DO 430 I=1,ICOUNT
           FNORM=FNORM+F(I)**2
           WRITE(NOUT,*)'B(R,S,G)',ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
           WRITE(ICOUT,*)ISUB1(I,, ISUB2(I), ISUB3(I), Y(I)
430      CONTINUE
         WRITE(NOUT,*)'L2 NORM', DSQRT(FNORM)
         CALL PLOT(Y, ICOUNT)
         STOP
       END IF
         CALL TNEG(Y, ICOUNT)
         WRITE(NOUT,*)'RA', RA*GAMMA, 'THIELE', THIELE
         WRITE(NOUT,*)'RA*', RA, 'FK', FK/DSQRT(RA)
         WRITE(NOUT,*)'FK*', FK
      WRITE (NOUT  'D', PD
      WRITE(NOUT  'WAVE NUMBERS', K, L
      FNORM=0.0D0
      WRITE(ICOUT,*)ICOUNT
      DO 130 I=1,ICOUNT
        FNORM=FNORM+F(I)**2
        WRITE(NOUT,*)'B(R,S,G) ',ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
        WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
      CONTINUE
      WRITE(NOUT,*)'L2 NORM', DSQRT(FNORM)
      CALL PLOT(Y, ICOUNT)

    CALL IMAN(Y, ICOUNT)

 Re-initialize Y
    DO [illegible] I=1,ICOUNT
      Y(I)=YINIT(I)
    CONTINUE
  CONTINUE

 END IF
 STOP
 END

  SUBROUTINE PARCHK(IR, IS, IPFLG)
  IMPLICIT REAL*8 (A-H,O-Z)
  Check parity
   ISUM=IFLOAT(IR+IS)
   TEST=DMOD(DSUM,2.0D0)
   IF(TEST.NE.0.0D0) THEN
     IPFLG=1
   ELSE
     IPFLG=0
   END IF
   RETURN
 END

 SUBROUTINE TRNCHK(IR, IS, IG, M, ITFLG)
 IMPLICIT REAL*8 (A-H,O-Z)
 DATA ITERM /4/
 Check truncation
   TEST=0.75D0*DFLOAT(IR**2)+DFLOAT(IS**2)*0.25D0+DFLOAT(IG*IG)
   IF(TEST.GT.(DFLOAT(M*M+1))) THEN
     ITFLG=1
   ELSE
     ITFLG=0
   END IF
   RETURN
 END

 SUBROUTINE TDERV(ICOUNT, X, Y, YDERV)
 IMPLICIT REAL*8 (A-H,O-Z)
--- Evaluate the time derivative of the coefficients for DIVPAG
 REAL*8 K, L, NU
 REAL*8 YDERV(200), Y(200)
 COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
 COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
 COMMON /TRPSUM/  SUM5
 COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20),
```

```
     #                 CDERV(-20:41,-20:41,20), C(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      COMMON /SUBS/ IR, IS, IG
      COMMON /WAVE/ K, L, M
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
C     REWIND (UNIT=9)
C     WRITE(ICOUT,*)'Y IN TDERV'
C     DO 15 I=1,ICOUNT
C       WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
C15     CONTINUE
C --- The correct element of Y is assigned on the same basis
C --- as in the main program i.e. using the counter Ncount which is
C --- incremented for every admissible rs pair
      NCOUNT=0
C --- Equation for energy balance
      DO 20 NCOUNT=1,ICOUNT/2
        IR=ISUB1(NCOUNT)
        IS=ISUB2(NCOUNT)
        IG=ISUB3(NCOUNT)
          NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
C --- Set flag for equation
          IFLAG=1
          CALL SUMB(Y, ICOUNT)
          CALL SUMTRP(Y, ICOUNT, IFLAG)
          IF((IR.EQ.0).AND.(IS.EQ.0)) THEN
            TERM=(1.0D0-(-1.0D0)**IG)/DFLOAT(IG)/PI
     *            +0.5D0*Y(NCOUNT)
     *            +SUM1+SUM2+0.5D0*(SUM4+SUM5)
          ELSE
            TERM=0.5D0*Y(NCOUNT)+SUM1+SUM2+0.5D0*(SUM4+SUM5)
          END IF
          BDERV(IR,IS,IG)=(-SUM3-(PI*PI*DFLOAT(IG*IG)+NU*NU)
     *                    *0.5D0*Y(NCOUNT)
     *          +FK*TERM)/SIGMA*2.0D0
          YDERV(NCOUNT)=BDEPV(IR,IS,IG)
20    CONTINUE
C ---
C --- Equation for concentraton Lalance
      DO 30 NCOUNT=ICOUNT/2+1, ICOUNT
        IR=ISUB1(NCOUNT)
        IS=ISUB2(NCOUNT)
        IG=ISUB3(NCOUNT)
          NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
C --- Set flag for equation
          IFLAG=2
          CALL SUMB(Y, ICOUNT)
          CALL SUMTRP(Y, ICOUNT, IFLAG)
          IF((IR.EQ.0).AND.(IS.EQ.0)) THEN
            TERM=DSIN((DFLOAT(IG)-0.5D0)*PI)/(PI*(DFLOAT(IG)-0.5D0))
     *            +0.5D0*Y(NCOUNT)
     *            +SUM1+SUM2+0.5D0*(SUM4+SUM5)
          ELSE
            TERM=0.5D0*Y(NCOUNT)+SUM1+SUM2+0.5D0*(SUM4+SUM5)
          END IF
          CDERV(IR,IS,IG)=(-SUM3-(PI*PI*(DFLOAT(IG)-0.5D0)**2+NU*NU)
     *                    *0.5D0*Y(NCOUNT)*XLE
     *          -FK/BETA/GAMMA*TERM)/0.5D0
          YDERV(NCOUNT)=CDERV(IR,IS,IG)
30    CONTINUE
C ---
```

```
      RETURN
      END
C
      SUBROUTINE FUN (ICOUNT, Y ,F, PAR)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the equation for coefficients, for CONBF
      REAL*8 K, L, N
      REAL*8 F(200), Y(200)
      COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /TRPSUM/  SUM5
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20),
     #               CDERV(-20:41,-20:41,20), C(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      COMMON /SUBS/ IR, IS, IG
      COMMON /WAVE/ K, L, N
      DATA IN /1/, NOUT /2/, ITERM/6/, ICOEF /8/, ICOUT /9/, INIT /10/
      REWIND (UNIT=9)
      WRITE(ICOUT,*)'Y IN FUNC'
      DO 10 I=1,ICOUNT
        WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
10    CONTINUE
C --- The correct element of Y is assigned on the same basis
C --- as in the main program i.e. using the counter Ncount which is
C ---   incremented for every admissible rs pair
C --- ICNT = 
C --- Equation for the energy balance
      DO 20 NCOUNT=1, ICOUNT/2
        IR=ISUB1(NCOUNT)
        IS=ISUB2(NCOUNT)
        IG=ISUB3(NCOUNT)
          NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
C --- Set flag for equation
          IFLAG=1
          CALL SUMB(Y, ICOUNT)
          CALL SUMTRP(Y, ICOUNT, IFLAG)
          IF((IR.EQ.0).AND.(IS.EQ.0)) THEN
            TERM=(1.0D0+(-1.0D0)**IG)/DFLOAT(IG)/PI
     *            +0.5D0*Y(NCOUNT)
     *           +SUM1+SUM2+0.5D0*(SUM4+SUM5)
          ELSE
            TERM=0.5D0*Y(NCOUNT)+SUM1+SUM2+0.5D0*(SUM4+SUM5)
          END IF
          F(NCOUNT)=-SUM3-(PI*PI*DFLOAT(IG*IG)+NU*NU)*0.5D0*Y(NCOUNT)
     *           +FK*TERM
20    CONTINUE
C
C --- Equation for the concentration balance
      DO 30 NCOUNT=ICOUNT/2+1, ICOUNT
        IR=ISUB1(NCOUNT)
        IS=ISUB2(NCOUNT)
        IG=ISUB3(NCOUNT)
          NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
C --- Set flag for equation
          IFLAG=2
          CALL SUMB(Y, ICOUNT)
          CALL SUMTRP(Y, ICOUNT, IFLAG)
          IF((IR.EQ.0).AND.(IS.EQ.0)) THEN
            TERM=DSIN((DFLOAT(IG)-0.5D0)*PI)/(PI*(DFLOAT(IG)-0.5D0))
     *              +0.5D0*Y(NCOUNT)
     *              +SUM1+SUM2+0.5D0*(SUM4+SUM5)
          ELSE
            TERM=0.5D0*Y(NCOUNT)+SUM1+SUM2+0.5D0*(SUM4+SUM5)
          END IF
          F(NCOUNT)=-SUM3-(PI*PI*(DFLOAT(IG)-0.5D0)**2+NU*NU)
     *                         *0.5D0*Y(NCOUNT)*XLE
     *              -FK/BETA/GAMMA*TERM
30    CONTINUE
C ---
      RETURN
      END
C
      SUBROUTINE SUMB(Y, ICOUNT)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the multiple summation arising from non-linear term
      REAL*8 KAPPA, K, L
      REAL*8 Y(200)
      COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20),
     #               CDERV(-20:41,-20:41,20), C(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, N
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      COMMON /SUBS/ IR, IS, IG
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /TRC/ ITRC(-20:41,-20:41,20)
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL F1, F2, F3, F5, F6, F7, FAKZ0, FAKZ1, FBKZ0, FBKZ1,
     #         FCKZ0, FCKZ1, FDKZ0, FDKZ1
      DATA ITERM /3/
      SUM1=0.0D0
      SUM2=0.0D0
      SUM3=0.0D0
      SUM4=0.0D0
      E=DEXP(1.0D0)
C --- Parameters for DQDAG
C --- We split up the region of integration to avoid errors with
C --- periodic functions
      X1=0.0D0
      X2=0.7D0
      X3=1.0D0
      AERR=1.0D-5
      RERR=1.0D-5
C --- Quadrature rule for oscillatory functions
      IRULE=6
C ---
      DO 50 IF=1,MAXIG
        XIF=DFLOAT(IF)
        XIG=DFLOAT(IG)
        CALL DQDAG(F1, X1, X2, AERR, RERR, IRULE, A11, ER)
        CALL DQDAG(F1, X2, X3, AERR, RERR, IRULE, A12, ER)
        A1=A11+A12
        CALL DQDAG(F5, X1, X2, AERR, RERR, IRULE, A51, ER)
        CALL DQDAG(F5, X2, X3, AERR, RERR, IRULE, A52, ER)
        A5=A51+A52
C --- Energy equation
```

```
  IF(IFLAG.EQ.1) THEN
    N=IIRC(IR,IS,IF)
    SUM1=SUM1+A1*(N+ICOUNT/2)
  END IF
Concentration equation
  IF(IFLAG.EQ.  ) THEN
    N=IIRC(IR,  ,IF)
     M1=SUM1+ *Y(N)
  END IF
CONTINUE

IPQCNT and ITUCNT are counters for the pq and tu coutts
DO 10 IPQCNT=1,ICOUNT/2
  IP=ISUB1(IPQCNT)
  IQ=ISUB2(IPQCNT)
  IH=ISUB3(IPQCNT)
  DO 10 ITUCNT=1,ICOUNT/2
    IT=ISUB1(ITUCNT)
    IU=ISUB2(ITUCNT)
    IIF=ISUB3(ITUCNT)
     IF((IP+IT).NE.IR) GO TO 10
     IF((IQ+IU).NE.IS) GO TO 10

[illegible] dummy subscripts that are real
              XIG=DFLOAT(IG)
              XIF=DFLOAT(IIF)
              XIH=DFLOAT(IH)

 [illegible]ulate SUM2

[illegible]rgy equation
          IF(IFLAG.EQ.1) THEN
            CALL DQDAG(F2, X1, X2, AERR, RERR, IRULE, A21, ER)
            CALL DQDAG(F2, X2, X3, AERR, RERR, IRULE, A22, ER)
            A2=A21+A22
            SUM2=SUM2+Y(IPQCNT)*Y(ITUCNT+ICOUNT/2)*A2
          END IF
 [illegible]ncentration equation
          IF(IFLAG.EQ.2) THEN
            CALL DQDAG(F6, X1, X2, AERR, RERR, IRULE, A61, ER)
            CALL DQDAG(F6, X2, X3, AERR, RERR, IRULE, A62, ER)
            A6=A61+A62
            SUM2=SUM2+Y(IPQCNT)*Y(ITUCNT+ICOUNT/2)*A6
          END IF

 Calculate SUM3

 Energy equation

        IF(IFLAG.EQ.1) THEN
          KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
          IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
          THEN
            CALL DQDAG (FAKZ0, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
            CALL DQDAG (FAKZ0, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
            AKZ=AKZ1+AKZ2
            CALL DQDAG (FBKZ0, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
            CALL DQDAG (FBKZ0, X2, X3, AERR, RFRR, IRULE, BKZ2, ER)
            BKZ=BKZ1+BKZ2
```

```
          ELSE
            C1=RA*PI*XIG/KAPPA/(DEXP(KAPP ) +DEXP(-KAPPA))/
     *              (PI**2*XIG**2+KAPPA**_,
            C2=RA/(PI**2*XIG**2+KAPPA**2)
            CALL DQDAG (FAKZ1, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
            CALL DQDAG (FAKZ1, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
            AKZ=AKZ1+AKZ2
            CALL DQDAG (FBKZ1, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
            CALL DQDAG (FBKZ1, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
            BKZ=BKZ1+BKZ2
          END IF
          SUM3=SUM3+Y(IPQCNT)*Y(ITUCNT)*
     *         (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*AKZ
     *          +KAPPA*KAPPA*BKZ)
        END IF
C---
C --- Concentration equarion
C---
        IF(IFLAG.EQ.2) THEN
          KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
          IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *    THEN
            CAEL DQDAG(FCKZ0, X1, X2, AERR, RERR, IRULE, CKZ1, ER)
            CALL DQDAG(FCKZ0, X2, X3, AERR, RERR, IRULE, CKZ2, ER)
            CKZ=CKZ1+CKZ2
            CALL DQDAG(FDKZ0, X1, X2, AERR, RERR, IRULE, DKZ1, ER)
            CALL DQDAG(FDKZ0, X2, X3, AERR, RERR, IRULE, DKZ2, ER)
            DKZ=DKZ1+DKZ2
          ELSE
C --- KAPPA.NE.0
            C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA,)/
     *              (PI**2*XIG**2+KAPPA**2)
            C2=RA/(PI**2*XIO**2+KAPPA**2)
            CALL DQDAG(FCKZ1, X1, X2, AERR, RERR, IRULE, CKZ1, ER)
            CALL DQDAG(FCKZ1, X2, X3, AERR, RERR, IR  E, CKZ2, ER)
            CKZ=CKZ1+CKZ2
            CALL DQDAG(FDKZ1, X1, X2, AERR, RERR, IRULE, DKZ1, ER)
            CALL DQDAG(FDKZ1, X2, X3, AERR, RERR, IRULE, DKZ2, ER)
            DKZ=DKZ1+DKZ2
          END IF
          SUM3=SUM3+Y(IPQCNT)*Y(ITUCNT+ICOUNT/2)*
     *         (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*CKZ
     *          -KAPPA*KAPPA*DKZ)
        END IF
C --- CALCULATE SUM4
C ---
C --- Energy equation
          IF(IFLAG.EQ.1) THEN
            CALL DQDAG(F3, X1, X2, AERR, RERR, IRULE, A31, ER)
            CALL DQDAG(F3, X2, X3, AERR, RERR, IRULE, A32, ER)
            A3=A31+A32
            SUM4=SUM4+Y(IPQCNT)*Y(ITUCNT)*A3
          END IF
C --- Concentration equation
          IF(IFLAG.EQ.2) THEN
            CALL DQDAG(F7, X1, X2, AERR, RERR, IRULE, A71, ER)
            CALL DQDAG(F7, X2, X3, AERR, RERR, IRULE, A72, ER)
            A7=A71+A72
            SUM4=SUM4+Y(IPQCNT)*Y(ITUCNT)*A7
            END IF
   CONTINUE
   RETURN
   END

   SUBROUTINE [illegible](Y, ICOUNT, IFLAG)
   IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the triple summation arising from non-linear term
   REAL*8 KAPPA, K, L
   REAL*8 Y(200)
   COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
   COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20),
  #               CDERV(-20:41,-20:41,20), C(-20:41,-20:41,20)
   COMMON /WAVE/ K, L, M
   COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
   COMMON /SUBS/ IR, IS, IG
   COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
   COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
   COMMON /TRC/ ITRC(-20:41,-20:41,20)
   COMMON /[illegible]/ SUM5
   COMMON /WORKSP/ RWKSP
   REAL*8 RWKSP([illegible])
   EXTERNAL F4, F8
   DATA ITERM /[illegible]/
   [illegible]
   [illegible]
C --- parameters for DQDAG
C --- split up the region of integration to avoid errors with
C --- [illegible] functions
   [illegible]
   [illegible]
   [illegible]
   [illegible]
   [illegible]
C --- quadrature rule for oscillatory functions
   IRULE=6

C --- IPQCNT and ITUCNT IMNCNT are counters for the pq, tu mn coeffs
   DO 10 IPQCNT=1,ICOUNT/2
     IP=ISUB1(IPQCNT)
     IQ=ISUB2(IPQCNT)
     IH=ISUB3(IPQCNT)
     DO 10 ITUCNT=1,ICOUNT/2
       IT=ISUB1(ITUCNT)
       IU=ISUB2(ITUCNT)
       IIF=ISUB3(ITUCNT)
       DO 10 IMNCNT=1,ICOUNT/2
         IM=ISUB1(IMNCNT)
         IN=ISUB2(IMNCNT)
         IL=ISUB3(IMNCNT)
            IF((IP+IT+IM).NE.IR) GO TO 10
            IF((IQ+IU+IN).NE.IS) GO TO 10
C ---
C --- assign dummy subscripts that are real
            XIG=DFLOAT(IG)
            XIF=DFLOAT(IIF)
            XIH=DFLOAT(IH)
            XIL=DFLOAT(IL)
C --- Calculate summation
C ---
C --- Energy equation
```

```
            IF(IFLAG.EQ.1) THEN
              CALL DQDAG(F4, X1, X2, AERR, RERR, IRULE, A41, ER)
              CALL DQDAG(F4, X2, X3, AERR, RERR, IRULE, A42, ER)
              A4=A41+A42
              SUM5=SUM5+Y(IPQCNT)*Y(ITUCNT)*Y(IMNCNT+ICOUNT/2)*A4
            END IF
C --- Concentration equation
            IF(IFLAG.EQ.2) THEN
              CALL DQDAG(F8, X1, X2, AERR, RERR, IRULE, A81, ER)
              CALL DQDAG(F8, X2, X3, AERR, RERR, IRULE, A82, ER)
              A8=A81+A82
              SUM5=SUM5+Y(IPQCNT)*Y(ITUCNT+ICOUNT/2)*Y(IMNCNT)*A8
            END IF
10    CONTINUE
      RETURN
      END
C ---
      SUBROUTINE TNEG(Y, ICOUNT)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Locate any temperature less than ambient
      REAL*8 K, L , KAPPA
      COMMON /WAVE/ K, L, M
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20),
     #               CDERV(-20:41,-20:41,20), C(-20:41,-20:41,20)
      REAL*8 Y(200)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      REAL*8 COEFF(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
      DATA NTMAX /17/
      DO 10 I=1,ICOUNT
        COEFF(ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
10    CONTINUE
        ZMAX=PI/K*2.0D0/DSQRT(3.0D0)
        XMAX=PI/K
        DO 15 Z=0.0D0, ZMAX+1.0D-3, ZMAX/5.0D0
          DO 15 X=0.0D0, XMAX+1.0D-3, XMAX/5.0D0
            DO 15 ZZ=0.0D0,1.0001D0, 2.00D-1
            THETA=0.0D0
              DO 20 NCOUNT=1,ICOUNT
                I=ISUB1(NCOUNT)
                J=ISUB2(NCOUNT)
                KK=ISUB3(NCOUNT)
                SUM=COEFF(I,J,KK)
     *              *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
     *              -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
                THETA=THETA+SUM*DSIN(PI*DFLOAT(KK)*ZZ)
20            CONTINUE
              IF(THETA.LT.0.0D0) THEN
                WRITE(NOUT,*)'******TEMPERATURE NEGATIVE******'
                WRITE(ITERM,*)'******TEMPERATURE NEGATIVE******'
                STOP
              END IF
15      CONTINUE
      RETURN
      END
C ---
```

```
      SUBROUTINE FDERV(N,X,Y,PD)
      IMPLICIT REAL*8 (A-H,O-Z)
      RETURN
      END


      SUBROUTINE PLOT( Y, ICOUNT)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Results for plotting
      REAL*8 K, L , KAPPA
      COMMON /WAVE/ K, L, M
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20),
     #               CDERV(-20:41,-20:41,20), C(-20:41,-20:41,20)
      REAL*8 Y(200)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      REAL*8 COEFF(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      DATA ITERM /4/
C --- Energy equation
      DO 10 I=1,ICOUNT/2
        COEFF(ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
        COEFF(-ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
        COEFF(ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
        COEFF(-ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
10    CONTINUE
C --- Concentration equation
      DO 20 I=ICOUNT/2+1,ICOUNT
        C(ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
        C(-ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
        C(ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
        C(-ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
20    CONTINUE
C --- Choose the height ZZ
      [illegible]
        DO 5 X=0.0D0,10.0001D0, 1.0D-1
        DO 15 ZZ=0.0D0,1.0001D0, 1.00D-1
          THETA=0.0D0
          VELZ=0.0D0
            CONC=0.0D0
C --- Energy equation
          DO 20 NCOUNT=1,ICOUNT/2
            I=ISUB1(NCOUNT)
            J=ISUB2(NCOUNT)
            KK=ISUB3(NCOUNT)
                SUM=COEFF(I,J,KK)
     #               *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
     #               -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
              KAPPA=SQRT(DFLOAT(I)**2*K**2+DFLOAT(J)**2*L**2)
              IF((I.EQ.0).AND.(J.EQ.0)) THEN
                F=RA*ZZ/PI/KK+RA/PI**2/KK**2*DSIN(KK*PI*ZZ)
              ELSE
                A=RA/(PI**2*KK**2+KAPPA**2)
                C1=A*PI*KK/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))
                F=C1*(DEXP(KAPPA*ZZ)-DEXP(-KAPPA*ZZ))+A*DSIN(KK*PI*ZZ)
              END IF
              THETA=THETA+SUM*DSIN(PI*DFLOAT(KK)*ZZ)
              VELZ=VELZ+SUM*KAPPA**2*F
20        CONTINUE
C
C --- Concentration equation
          DO 30 NCOUNT=1,ICOUNT/2
            I=ISUB1(NCOUNT)
            J=ISUB2(NCOUNT)
            KK=ISUB3(NCOUNT)
                SUM=C(I,J,KK)
     #               *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
     #               -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
              CONC=CONC+SUM*DCOS(PI*(DFLOAT(KK)-0.5D0)*ZZ)
30        CONTINUE
C ---
          THETA=THETA/GAMMA
          CONC=0.21D0*(CONC+1.0D0)
          WRITE(7,*)293.0*(THETA+1.0), CONC,
     #              VELZ*2.0E-4/CHRL, X*CHRL, Z*CHRL
15      CONTINUE
5     CONTINUE
      RETURN
      END
C ---
C
      SUBROUTINE TMAX(Y, ICOUNT, TEND)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Results for plotting
      REAL*8 K, L , KAPPA
      COMMON /WAVE/ K, L, M
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20),
     #               CDERV(-20:41,-20:41,20), C(-20:41,-20:41,20)
      REAL*8 Y(200)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      REAL*8 COEFF(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
      DATA NTMAX /17/
C --- Energy equation
      DO 10 I=1,ICOUNT/2
        COEFF(ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
10    CONTINUE
C --- Choose the height X and Y first check at hexagon centre
      X=0.0D0
      Z=0.0D0
      THTMAX=0.0D0
30      DO 15 ZZ=0.0D0,1.0001D0, 1.00D-1
          THETA=0.0D0
          VELZ=0.0D0
C --- Energy equation
          DO 20 NCOUNT=1,ICOUNT/2
            I=ISUB1(NCOUNT)
            J=ISUB2(NCOUNT)
            KK=ISUB3(NCOUNT)
              SUM=COEFF(I,J,KK)
     #               *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
     #               -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
              KAPPA=SQRT(DFLOAT(I)**2*K**2+DFLOAT(J)**2*L**2)
              IF((I.EQ.0).AND.(J.EQ.0)) THEN
                F=RA*ZZ/PI/KK+RA/PI**2/KK**2*DSIN(KK*PI*ZZ)
              ELSE
            A=RA/(PI**2*KK**2+KAPPA**2)
            C1=A*PI*KK/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))
            F=C1*(DEXP(KAPPA*ZZ)-DEXP(-KAPPA*ZZ))+A*DSIN(KK*PI*ZZ)
          END
          THETA=THETA+SUM*DSIN(PI*DFLOAT(KK)*ZZ)
          VEL=VEL+SUM*KAPPA**2*F
      CONTINUE
C ---
C ---
      THETA=THETA/GAMMA
      IF(THETA.GT.THTMAX) THEN
        THTMAX=THETA
        HEIGHT=ZZ
        YCOORD=Z
      END IF
     CONTINUE
C -- Now check at vertex
      IF(Z.EQ.0.0D0) THEN
        Z=PI*K*2.0D0/DSQRT(3.0D0)
        GO TO 20
      END IF
      WRITE(17,1123) [illegible](THTMAX+1.0), THTMAX, YCOORD*CHRL, HEIGHT*CHRL
     #            , TEND, TEND*CHRL**2/2.2[illegible]/3600.0
 1123 FORMAT(6F12.5)
      RETURN
      END
C
C The following functions contain the integrands for DQDAG
C
      DOUBLE PRECISION FUNCTION F1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357793D0
      F1=DSIN(XIG*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)
      RETURN
      END
C --
      DOUBLE PRECISION FUNCTION F2(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357793D0
      F2=DSIN(XIH*PI*Z)*DSIN(XIG*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)
      RETURN
      END
C --
      DOUBLE PRECISION FUNCTION F3(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357793D0
      F3=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F4(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357793D0
      F4=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)*DCOS((XIL-0.5D0)*
     #   PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F5(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357793D0
      F5=DSIN(XIF*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F6(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357793D0
      F6=DSIN(XIH*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F7(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357793D0
      F7=DSIN(XIH*PI*Z)*DSIN(XIF*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F8(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357793D0
      F8=DSIN(XIH*PI*Z)*DSIN(XIL*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)*DCO
     #   0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FAKZ0(Z)
      IMPLICIT REAE*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FAKZ0=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #       RA/PI/XIH*(1.0D0+DCOS(PI*XIH*Z))
      RETURN
      END
C ---
      DOUBLE PRFCISION FUNCTION FAKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      RFAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FAK21=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*C1*KAPPA*(EXP(KAPPA*Z)+
     #      EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION FBK20(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FBK20=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
      RETURN
      END

      DOUBLE PRECISION FUNCTION FBK21(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FBK21=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION FCK20(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FCK20=DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)*
     #      RA/PI/XIH*(1.D0+DCOS(PI*XIH*Z))
      RETURN
      END

      DOUBLE PRECISION FUNCTION FCK21(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FCK21=DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
     #                                   *C1*KAPPA*(EXP(KAPPA*Z)+
     #      EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION FDK20(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FDK20=PI*(XIF-0.5D0)*DSIN((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
     #      *RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
      RETURN
      END

      DOUBLE PRECISION FUNCTION FDK21(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FDK21=PI*(XIF-0.5D0)*DSIN((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
     #      *C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
      RETURN
      END
```

The following program written in Fortran 77 (STAB3 FORTRAN) is used to calculate the eigenvalues for the stability analysis of the steady-state solutions obtained from program BED3NEW FORTRAN.

```
C **********************************************************************
C   CALCULATE THE EIGENVALUES FOR THE STABILITY ANALYSIS OF           *
C   THE STAEDY STATE SOLUTION GENERATED BY BED3NEW                    *
C   ASSUMING A FIRST ORDER REACTION                                   *
C   WITH OPEN T   AND LINEAR EXPANSION OF THE EXPONENTIAL             *
C   AND NUMERICAL INTEGRATIONS                                        *
C **********************************************************************
C --- The steady state coefficients are read in having been calculated
C --- by BED3NEW
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NEQN=14)
      REAL*8 K, L
      REAL*8 Y(200)
      REAL*8 AA(NEQN,NEQN), RR(NEQN), RI(NEQN)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /COEFF/ B(-20:+1,-20:+1,20),
     #               C(-20:+1,-20:+1,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
      COMMON /TRPSUM/ SUM5
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /SUBS/ IR, IS, IG
      COMMON /TRC/ ITRC(-20:+1,-20:+1,20)
      INTEGER INTGER(NEQN)
      DATA IN /1/, NOUT /2/, ITERM /6/, ICOEF /8/, ICOUT /9/, INIT /16/
     #     , NEIG /7/
C --- Set parameters
        PHI=400.D0
        PI=3.14159265358979D0
        XLE=1.0452D0
C --- Read truncation parameter, wave number and task flag for uss/ss
      READ(IN,*)M, A, ITASK
C --- Read physical parameters
      READ(IN,*)DP, EPSI, EA, XKO, XKEQ, XMU
      READ(IN,*)TO, RHO, GRAV, HTCAP, DELTAH, CHRL
      READ(IN,*)IREAD
        PERM=DP*DP*EPSI**3/150.0D0/(1.00D0-EPSI)**2
        RA=GRAV*3.67D-3*TO*CHRL*RHO**2*PERM*HTCAP/XKEQ/XMU
        GAMMA=EA/8.314D0/TO
        BETA=DELTAH/0.029D0/TO/HTCAP/4.76D0
        THIELE=6.0D0*XKO*RHO*HTCAP*(1.0D0-EPSI)*CHRL*CHRL
     #         /(XKEQ*DP)*DEXP(-GAMMA)
        DE=6.0D0*XKO*RHO**2*CHRL**2.5*HTCAP**1.5*DEXP(-GAMMA)
     #     /XKEQ**1.5*(GRAV*TO*3.67D-3*EPSI**3/150.0D0/XMU)**0.5
C --- Calculate parameters for the positive exponential approximation
        RASTAR=RA/GAMMA
        RA=RASTAR
        FK=BETA*THIELE*GAMMA
        WRITE(NOUT,*)'FF', FK
C --- Calculate wave numbers
        L=DSQRT(A*A/4.0D0)
        K=DSQRT(A*A-L*L)
C --- Maximum and minimum values for the subscripts
        MAXIR=0
        MAXIS=0
        MAXIG=1
        MINIR=0
        MINIS=0
        MINIG=1
C --- ICOUNT counts the number of coefficients i.e. the number of
C --- equations that satisfy the truncation and parity
        READ(ICOUT,*)ICOUNT
        DO 11 I=1,ICOUNT
          READ(ICOUT,*)IR, IS, IG, B(IR, IS, IG)
          Y(I)=B(IR,IS,IG)
          ISUB1(I)=IR
          ISUB2(I)=IS
          ISUB3(I)=IG
          IF(IR.GT.MAXIR) MAXIR=IR
          IF(IS.GT.MAXIS) MAXIS=IS
          IF(IG.GT.MAXIG) MAXIG=IG
          IF(IR.LT.MINIR) MINIR=IR
          IF(IS.LT.MINIS) MINIS=IS
          IF(IG.LT.MINIG) MINIG=IG
          IF(I.LE.ICOUNT/2) THEN
            ITRC(IR,IS,IG)=I
          END IF
11      CONTINUE
        WRITE(INIT,*)ICOUNT
        DO 111 I=1,ICOUNT
          WRITE(INIT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
111     CONTINUE
        WRITE(ITERM,*)'NUMBER OF TERMS', ICOUNT
C ---
C --- Start loop to assemble matrix
C --- Assemble matrix for the energy equation
        IFLAG=1
        DO 15 J=1,ICOUNT/2
          IR=ISUB1(J)
          IS=ISUB2(J)
          IG=ISUB3(J)
          DO 10 I=1,ICOUNT
            CALL SUMB(Y, ICOUNT, I)
            CALL SUMTRIP(Y, ICOUNT, IFLAG, I)
            IF(I.EQ.J) THEN
              AA(J,I)=(-0.5D0*(
     #               (DFLOAT(ISUB1(I))*K)**2+(DFLOAT(ISUB2(I))*L)**2
     #               +PI**2*DFLOAT(ISUB3(I)**2))
     #               -2.0D0*SUM3+FK*(0.5D0+SUM1+SUM2+SUM4+
     #               0.5D0*SUM5))*2.0D0/PHI
            ELSE
              AA(J,I)=(-2.0D0*SUM3+FK*(0.5D0+SUM1+SUM2+SUM4+
     #               0.5D0*SUM5))*2.0D0/PHI
            END IF
          WRITE(NOUT,*)AA(J,I), J, I
10        CONTINUE
15      CONTINUE
C ---
C --- Assemble matrix for the concentration equation
        IFLAG=2
        DO 25 J=ICOUNT/2+1,ICOUNT
          IR=ISUB1(J)
          IS=ISUB2(J)
          IG=ISUB3(J)
          DO 50 I=1,ICOUNT
            CALL SUMB(Y, ICOUNT, I)
            CALL SUMTRIP(Y, ICOUNT, IFLAG, I)
      IF(I.EQ.J) THEN
        AA(J,I)=(   D0*XLE*(
     #          (DFLOAT(ISUB1(I)*K)**2+(DFLOAT(ISUB2(I))*L)**2
     #          +(  *2*DFLOAT(ISUB3(I)**2))
     #          -SUM3+FK/(BETA*GAMMA)
     #          *(0.5D0+SUM1+SUM2+SUM4+
     #          0.5D0*SUM5))/EPSI
      ELSE
        AA(J,I)=(SUM3+FK/(BETA*GAMMA)
     #          *(0.5D0+SUM1+SUM2+SUM4+
     #          0.5D0*SUM5))/EPSI
      END IF
      WRITE(NOUT,*)AA(J,I), J, I
      CONTINUE
      CONTINUE

C --- Use F02AFF to find the eigenvalues of AA
C --- Parameters for F02AFF
      N=ICOUNT
      IA=ICOUNT
      IFAIL=1
      CALL F02AFF(AA, IA, N, RR, RI, INTGER, IFAIL)
      IF(IFAIL.GT.0) THEN
        WRITE(NEIG,*)'IFAIL', IFAIL
        STOP
      END IF
      WRITE(NEIG,*)'EIGENVALUES'
      DO   I=1,ICOUNT
      WRITE(NEIG,*)RR(I), RI(I)
      CONTINUE
      STOP
      END

      SUBROUTINE PARCHK(IR, IS, IPRFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Check parity
      SUM=DFLOAT(IR+IS)
      TEST=DMOD(SUM,2.0D0)
      IF(TEST.NE.0.0D0) THEN
        IPRFLG=1
      ELSE
        IPRFLG=0
      END IF
      RETURN
      END

      SUBROUTINE TRNCHK(IR, IS, IG, M, ITFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA ITERM /3/
C --- Check truncation
      TEST=0.75D0*DFLOAT(IR**2)+DFLOAT(IS**2)+0.25D0*DFLOAT(IG*IG)
      IF(TEST.GT.(DFLOAT(M*M+1))) THEN
        ITFLG=1
      ELSE
        ITFLG=0
      END IF
      RETURN
      END
```

```
      SUBROUTINE SUMB(Y, ICOUNT, INDEX)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the multiple summation arising from non-linear term
      REAL*8 KAPPA, K, L
      REAL*8 Y(200)
      COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
      COMMON /COEFF/ B(-20:41,-20:41,20),
     #               C(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      COMMON /SUBS/ IR, IS, IG
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /TRC/ ITRC(-20:41,-20:41,20)
      EXTERNAL F1, F2, F3, F5, F6, F7, FAKZO, FAKZ1, FBKZO, FBKZ1,
     #         FCKZO, FCKZ1, FDKZO, FDKZ1
      DATA ITERM /3/
      SUM1=0.0D0
      SUM2=0.0D0
      SUM3=0.0D0
      SUM4=0.0D0
      E=DEXP(1.0D0)
C --- Parameters for DQDAG
C --- We split up the region of integration to avoid errors with
C --- periodic functions
      X1=0.0D0
      X2=0.7D0
      X3=1.0DD
      AERR=1.0D-5
      RERR=1.0D-5
C --- Quadrature rule for oscillatory functions
      IRULE=6
C ---
      DO 50 IF=1,MAXIG
        XIF=DFLOAT(IF)
        XIG=DFLOAT(IG)
        CALL DQDAG(F1, X1, X2, AERR, RERR, IRULE, A11, ER)
        CALL DQDAG(F1, X2, X3, AERR, RERR, IRULE, A12 ,ER)
        A1=A11+A12
        IF(IER.GT.0) WRITE(ITERM,*)'IER A1', IER
        CALL DQDAG(F5, X1, X2, AERR, RERR, IRULE, A51, ER)
        CALL DQDAG(F5, X2, X3, AERR, RERR, IRULE, A52, ER)
        A5=A51+A52
        IF(IER.GT.0) WRITE(ITERM,*)'IER A5', IER
C --- Energy equation
        IF(IFLAG.EQ.1) THEN
          N=ITRC(IR,IS,IF)
          IF((N+ICOUNT/2).EQ.INDEX) THEN
            SUM1=A1
          END IF
        END IF
C --- Concentration equation
        IF(IFLAG.EQ.2) THEN
          N=ITRC(IR,IS,IF)
          IF(N.EQ.INDEX) THEN
            SUM1=A5
          END IF
        END IF
50    CONTINUE
```

```
[illegible] and IT[illegible] are counters for the pq and tu coeffs
[illegible] P+[illegible], P+Q, T+U even, P,Q,,T,U truncated
[illegible] NT/2
  IP=[illegible]
  [illegible]
  [illegible] IP,[illegible]
  [illegible] ITUCNT 1,ICOUNT/2
    [illegible](ITUCNT)
    [illegible](ITUCNT)
    [illegible](ITUCNT)
        [illegible] NE.(R) [illegible] 10
        [illegible] NE.(S) [illegible] 10

[illegible] subscripts that are real
          [illegible]FLOAT(IR)
          [illegible]AT(IT)
          [illegible]AT(IU)

[illegible]

[illegible]
      IF [illegible] 1) THEN
        [illegible] X1, X2, AERR, RERR, IRULE, A21, ER)
        [illegible] X2, X3, AERR, RERR, IRULE, A22, ER)
        [illegible]
        [illegible] WRITE(ITERM,*)'IER A2', IER
        [illegible] THEN
            [illegible]
        END IF
        IF [illegible] INDEX) THEN
          [illegible] A2
        END IF
      END IF
[illegible] equation
      IF [illegible] 2) THEN
        [illegible] X1, X2, AERR, RERR, IRULE, A61, ER)
        [illegible] X2, X3, AERR, RERR, IRULE, A62, ER)
        [illegible]
        [illegible] WRITE(ITERM,*)'IER A6' IER
        IF [illegible] INDEX) THEN
            [illegible] ITUCNT+ICOUNT[illegible] A6
        [illegible]
        [illegible] EQ INDEX) THEN
          [illegible] A6
        END IF
      END IF

[illegible]

[illegible] equation

      IF(IFLAG[illegible] THEN
        IF(IP[illegible] INDEX) THEN
        KAPPA=SQRT((DFLOAT(IP)[illegible]2+(DFLOAT[illegible]**2)
        IF(((IP EQ 0) AND (IQ EQ 0)) OR ((IT[illegible] AND [illegible] EQ.0)))
        THEN
          CALL DQDAG (FAKZ0, X1, X2, AERR, [illegible] RULE, AKZ1, ER)
          CALL DQDAG (FAKZ0, X2, X3, AERR, RERR [illegible] RULE, AKZ2, ER)
```

```
          AKZ=AKZ1+AKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ', IER
          CALL DQDAG (FBKZ0, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
          CALL DQDAG (FBKZ0, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
          BKZ=BKZ1+BKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ', IER
        ELSE
          C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *           (PI**2*XIG**2+KAPPA**2)
          C2=RA/(PI**2*XIG**2+KAPPA**2)
          CALL DQDAG (FAKZ1, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
          CALL DQDAG (FAKZ1, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
          AKZ=AKZ1+AKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ', IER
          CALL DQDAG (FBKZ1, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
          CALL DQDAG (FBKZ1, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
          BKZ=BKZ1+BKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ', IER
        END IF
        SUM3=SUM3+Y(ITUCNT)*
     *       (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*AKZ
     *       +KAPPA*KAPPA*BKZ)
        END IF
      END IF
C---
C --- Concentration equation
C---
      IF(IFLAG.EQ.2) THEN
        KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
        IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *  THEN
          CALL DQDAG(FCKZ0, X1, X2, AERR, RERR, IRULE, CKZ1, ER)
          CALL DQDAG(FCKZ0, X2, X3, AERR, RERR, IRULE, CKZ2, ER)
          CKZ=CKZ1+CKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER CKZ', IER
          CALL DQDAG(FDKZ0, X1, X2, AERR, RERR, IRULE, DKZ1, ER)
          CALL DQDAG(FDKZ0, X2, X3, AERR, RERR, IRULE, 'DKZ2, ER)
          DKZ=DKZ1+DKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER DKZ', IER
        ELSE
C --- KAPPA.NE.0
          C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *           (PI**2*XIG**2+KAPPA**2)
          C2=RA/(PI**2*XIG**2+KAPPA**2)
          CALL DQDAG(FCKZ1, X1, X2, AERR, RERR, IRULE, CKZ1, ER)
          CALL DQDAG(FCKZ!, X2, X3, AERR, RERR, IRULE, CKZ2, ER)
          CKZ=CKZ1+CKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ', IER
          CALL DQDAG(FDKZ1, X1, X2, AERR, RERR, IRULE, DKZ1, ER)
          CALL DQDAG(FDKZ1, X2, X3, AERR, RERR, IRULE, DKZ2, ER)
          DKZ=DKZ1+DKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER DKZ', IER
        END IF
        IF(IPQCNT.EQ.INDEX) THEN
          SUM3=SUM3+Y(ITUCNT+ICOUNT/2)*
     *       (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*CKZ
     *       -KAPPA*KAPPA*DKZ)
        END IF
        IF((ITUCNT+ICOUNT/2).EQ.INDEX) THEN
```

```
          SUM    M3+Y(IPQCNT)*
                   (DFLOAT(IP*IT)*K*K+DFLOA( [illegible] IU)*L*L)-CK2
*                  KAPPA*KAPPA-DK2)
         END
       END IF
C --- CALCULATE [illegible]

C --- Energy equation
         IF(IFLAG.EQ.1) THEN
           CALL DQDAG(F3, X1, X2, AERR, RERR, IRULE, A31, ER)
           CALL DQDAG(F3, X2, X3, AERR, RERR, IRULE, A32, ER)
           A3=A31+A32
           IF(IER.GT.0) WRITE(ITERM,*)'IER A3', IER
           IF(IT[illegible]CNT.EQ.INDEX) THEN
             [illegible]M4=SUM4+Y(ITUCNT)*A3
           END IF
         END IF
C --- Concentration equation
         IF(IFLAG.EQ.2) THEN
           CALL DQDAG(F7, X1, X2, AERR, RERR, IRULE, A71, ER)
           CALL DQDAG(F7, X2, X3, AERR, RERR, IRULE, A72, ER)
           A7=A71+A72
           IF([illegible]GT.0) WRITE(ITERM,*)'IER A7', IER
           IF(IT[illegible]CNT.EQ.INDEX) THEN
             [illegible]M4=SUM4+Y(ITUCNT)*A7
           END IF
         END IF
     CONTINUE
     RETURN
     END

     SUBROUTINE [illegible]IF(Y, ICOUNT, IFLAG, INDEX)
     IMPLICIT REAL*8 (A-H,O-Z)
C    Evaluate the triple summation arising from non-linear term
     REAL*8 KAPPA, K, L
     REAL*8 Y(200)
     COMMON /COEFF/ [illegible](-20:41,-20:41,20),
     #                  [illegible](-20:41,-20:41,20)
     COMMON /WAVE/ K, L, M
     COMMON /PARAM/ BETA, GAMMA, PI, THILLE, RA, PHI, [illegible]HKL, FK, XLE
     COMMON /SUBS/ [illegible], IS, IG
     COMMON /REALS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
     COMMON /MAXS/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
     COMMON /TRUN/ ITRUN(-20:41,-20:41,20)
     COMMON /NFSUM/ SUM5
     COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
     EXTERNAL F4, F8
     DATA ITERM /6/
     SUM5=0.0D0
     [illegible]
C --- Parameters for DQDAG
C --- We split up the region of integration to avoid errors with
C --- periodic functions
     X1=0.0D0
     X2=0.[illegible]D0
     X3=1.0D0
     AERR=1.0D-5
     RERR=1.0D-5
C --- Quadrature rule for oscillatory functions
```

```
      IRULE=6
C ---
C --- IPQCNT and ITUCNT IMNCNT are counters for the pq, tu mn coeffs
C --- Check that P+T+M=R, Q+U+N=S, P+Q, T+U M+N even,
C --- P,Q, T,U M,N truncated
      DO 10 IPQCNT=1,ICOUNT/2
        IP=ISUB1(IPQCNT)
        IQ=ISUB2(IPQCNT)
        IH=ISUB3(IPQCNT)
        DO 10 ITUCNT=1,ICOUNT/2
          IT=ISUB1(ITUCNT)
          IU=ISUB2(ITUCNT)
          IIF=ISUB3(ITUCNT)
          DO 10 IMNCNT=1,ICOUNT/2
            IM=ISUB1(IMNCNT)
            I[illegible]SUB2(IMNCNT)
            I[illegible]SUB3(IMNCNT)
                IF((IP+IT+IM).NE.IR) GO TO 10
                IF((IQ+IU+IN).NE.IS) GO TO 10
C ---
C --- Assign dummy subscripts that are real
                XIG=DFLOAT(IG)
                XIF=DFLOAT(IIF)
                XIH=DFLOAT(IH)
                XIL=DFLOAT(IL)
C --- Calculate summation
C ---
C --- Energy equation
              IF(IFLAG.EQ.1) THEN
                CALL DQDAG(F4, X1, X2, AERR, RERR, IRULE, A41, ER)
                CALL DQDAG(F4, X2, X3, AERR, RERR, IRULE, A42, ER)
                A4=A41+A42
                IF(IER.GT.0) WRITE(ITERM,*)'IER A4', IER
                IF(IPQCNT.EQ.INDEX) THEN
                  SUM5=SUM5+Y(ITUCNT)*Y(IMNCNT+ICOUNT/2)*A4
                END IF
                IF(ITUCNT.EQ.INDEX) THEN
                  SUM5=SUM5+Y(IPQCNT)*Y(IMNCNT+ICOUNT/2)*A4
                END IF
                IF((IMNCNT+ICOUNT/2).EQ.INDEX) THEN
                  SUM5=SUM5+Y(IPQCNT)*Y(ITUCNT[illegible]
                END IF
              END IF
C --- Concentration equation
              IF(IFLAG.EQ.2) THEN
                CALL DQDAG(F8, X1, X2, AERR, RERR, IRULE, A81, ER)
                CALL DQDAG(F8, X2, X3, AERR, RERR, IRULE, A82, ER)
                A8=A81+A82
                IF(IER.GT.0) WRITE(ITERM,*)'IER A8', IER
                IF(IPQCNT.EQ.INDEX) THEN
                  SUM5=SUM5+Y(ITUCNT)*Y(IMNCNT+ICOUNT/2)*A8
                END IF
                IF(ITUCNT.EQ.INDEX) THEN
                  SUM5=SUM5+Y(IPQCNT)*Y(IMNCNT+ICOUNT/2)*A8
                END IF
                IF((IMNCNT+ICOUNT/2).EQ.INDEX) THEN
                  SUM5=SUM5+Y(IPQCNT)*Y(ITUCNT)*A8
                END IF
              END IF
10    CONTINUE
```

```
      RETURN
      END
C
C --- The following functions contain the integrands for DQDAG
C
      DOUBLE PRECISION FUNCTION F1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F1=DSIN(XIG*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION F2(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F2=DSIN(XIH*PI*Z)*DSIN(XIG*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION F3(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F3=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION F4(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F4=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)*DCOS((XIL-0.5D0)*
     #   PI*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION F5(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F5=DSIN(XIF*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION F6(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F6=DSIN(XIH*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F7(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F7=DSIN(XIH*PI*Z)*DSIN(XIF*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F8(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F8=DSIN(XIH*PI*Z)*DSIN(XIL*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-
     #   0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FAKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      FAKZ0=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      RA/PI/XIH*(1.0D0+DCOS(PI*XIH*Z))
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FAKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      FAKZ1=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*C1*KAPPA*(EXP(KAPPA*Z)+
     #      EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FBKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      FBKZ0=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FBKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      FBKZ1=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FCKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      FCKZ0=DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)*
     #      RA/PI/XIH*(1.0D0+DCOS(PI*XIH*Z))
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FCKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      FCKZ1=DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
     #                                   *C1*KAPPA*(EXP(KAPPA*Z)+
     #      EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FDKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      FDKZ0=PI*(XIF-0.5D0)*DSIN((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
     #      *RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FDKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL, FK, XLE
      FDKZ1=PI*(XIF-0.5D0)*DSIN((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
     #      *C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
      RETURN
      END
```

The following program written in Fortran 77 (IGN3 FORTRAN) is used to calculate directly ignition points in the infinite layer assuming a first order reactien.

```
C**************************************************************************
C        CALCULATE THE IGNITION POINT FOR THE UNBOUNDED LAYER            *
C        ASSUMING A FIRST ORDER REACTION                                 *
C        WITH OPEN TOP AND LINEAR EXPANSION OF THE EXPONENTIAL           *
C        AND NUMERICAL INTEGRATIONS
C**************************************************************************
 -- Uses C05NBF
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K, L
      REAL*8 WA(70000), F(200)
      REAL*8 Y(200)
      COMMON /SUB/ ISUB1(200), ISUP2(200), ISUB3(200)
      COMMON /COEFF/ B(-20:41,-20:41,20),
     #               C(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /TRC/ ITRC(-20:41,-20:41,20)
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL FUNC
      EXTERNAL TDERV
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
 -  Call to ensure that there is sufficient workspace for DIVPAG
 -- and DQDAG
      CALL IWKIN(10070)
 -  Model parameters
        SIGMA=900.0D0
        PI=3.14159265357930D0
        XLE=0.0353D0
 --    ad truncation parameter, wave number
       READ(IN,*)M, A
      ead physical parameters
      READ(IN,*) EPSI, EA, XKO, XKEQ, XMU
      READ(IN,*)TO, RHO, GRAV, HTCAP, DELTAH, CHRL
      READ(IN,*)IREAD
        PERM=DP*DP*EPSI**3/150.0D0/(1.00D0-EPSI)**2
        RA=GRAV*3.67D-3*TO*CHRL*RHO**2*PERM*HTCAP/XKEQ/XMU
        GAMMA=EA/8.314D0/TO
        BETA=DELTAH/0.?29D0/TO/HTCAP/4.76D0
        THIELE=6.0D0*XKO*RHO*HTCAP*(1.0D0-EPSI)*CHRL*CHRL
     *       /(XKEQ*DP)*DEXP(-GAMMA)
        DD=6.0D0*XKO*RHO**2*CHRL**2.5*HTCAP**1.5*DEXP(-GAMMA)
     *    /XKEQ**1.5*(GRAV*TO*3.67D-3*EPSI**3/150.0D0/XMU)**0.5
        Parameters for the positive exponential approximation
        RASTAR=RA/GAMMA
        RA=RASTAR
        FK=BETA*THIELE*GAMMA
      alculate wave numbers
        L=SQRT(A*A/4.0D0)
        K=SQRT(A*A-L*L)
 -    Maximum and minimum values for the subscripts
        MAXIR=0
        MAXIS=0
        MAXIG=1
        MINIR=0
        MINIS=0
        MINIG=1
      IF(IREAD.EQ.1) THEN
        READ(ICOEF,*)ICOUNT
        NEQN=ICOUNT+1
        DO 11 I=1,ICOUNT
          READ(ICOEF,*  R, IS, IG, B(IR, IS, IG)
          Y(I)=B(IR,  ,IG)
          ISUB1(I)=IR
          ISUB2(I)=IS
          ISUB3(I)=IG
          IF(IR.GT.MAXIR) MAXIR=IR
          IF(IS.OT.MAXIS) MAXIS=IS
          IF(IG.GT.MAXIG) MAXIG=IG
          IF(IR.LT.MINIR) MINIR=IR
          IF(IS.LT.MINIS) MINIS=IS
          IF(IG.LT.MINIG) MINIG=IG
          IF(I.LE.ICOUNT/2) THEN
            ITRC(IR,IS,IG)=I
          END IF
11      CONTINUE
        READ(ICOEF,*)FK
        Y(NEQN)=FK
      ELSE
C --- ICOUNT counts the number of coufficients i.e. the number ot
C --- equations that satisfy the truncation and parity
C --- Set up coefficients for the energy equation
      ICOUNT=0
      NCOUNT=0
      DO 10 IR=-10, 10
        DO 10 IS=-10, 10
          DO 10 IG=1,10
C --- Check parity and tr ncatiou
            CALL TRNCHK(IR, IS, IG, M, ITFLG)
             IF(ITFLG.NE.0) GO TO 10
            CALL PARCHK(IR, IS, IPRFLG)
             IF(IPRFLG.NE.0) GO TO 10
            ICOUNT=ICOUNT+1
C --- Initialise B
            IF(IR.EQ.0) THEN
              B(IR,IS,IG)=1.0D0
            ELSE
              B(IR,IS,IG)=-1.0D-3
            END IF
C --- Copy b into dummy variables Y is a vector
            Y(ICOUNT)=B(IR,IS,IG)
            ISUB1(ICOUNT)=IR
            ISUB2(ICOUNT)=IS
            ISUB3(ICOUNT)=IG
            ITRC(IR,IS,IG)=ICOUNT
            IF(IR.GT.MAXIR) MAXIR-IR
            IF(IS.OT.MAXIS) MAXIS=IS
            IF(IG.GT.MAXIG) MAXIG=IG
            IF(IR.LT.MINIR) MINIR=IR
            IF(IS.LT.MINIS) MINIS=IS
            IF(IG.LT.MINIG) MINIG=IG
10    CONTINCE
C --- Set up coefficients for the concentration equation
      DO 110 IR=-10, 10
        DO 110 IS=-10, 10
          DO 110 IG=1,10
C --- Check parity and truncation
```

```
      CALL TRNCHK(IR, IS, IO, M, ITFLG)
       IF(ITFLG.NE.0) GO TO 110
      CALL PARCHK(IR, IS, IPRFLG)
       IF(IPRFLG.NE.0) GO TO 110
      ICOUNT=ICOUNT+1
[illegible]
      IF(IR [illegible] THEN
        [illegible]
      ELSE
        [illegible]
      END IF
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
        WRITE(NOUT,*) [illegible] ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
        WRITE(ICOUT,*) ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
      CONTINUE
      WRITE(NOUT, [illegible]
      CALL [illegible]
      STOP
      END
C ---
      SUBROUTINE PARCHK(IR, IS, IPRFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Check parity
        SUM=DFLOAT(IR+IS)
        TEST=DMOD(SUM,2.0D0)
        IF(TEST.NE.0.0D0) THEN
          IPRFLG=1
        ELSE
          IPRFLG=0
        END IF
        RETURN
      END
C ---
      SUBROUTINE TRNCHK(IR, IS, IO, M, ITFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA ITERM /3/
C --- Check truncation
        TEST=0.75D0*DFLOAT(IR**2)+DFLOAT(IS**2)*0.25D0+DFLOAT(IO*IO)
        IF(TEST.GT.(DFLOAT(M*M+1))) THEN
          ITFLG=1
        ELSE
          ITFLG=0
        END IF
        RETURN
      END
C
      SUBROUTINE FUNC (NEQN, Y, F, PAR)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the equation for coefficients, for C05NBF
C --- Evaluates the equation for  det(Jacobian)=0
C --- Y(NEQN)=FK
      PARAMETER (NVAR=14)
      REAL*8 K, L, NU
      REAL*8 F(200), Y(200), AA(NVAR,NVAR), RR(NVAR), RI(NVAR)
      COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
      COMMON /SUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /TRPSUM/  SUM5
      COMMON /COEFF/ B(-20:41,-20:41,20),
     #                 C(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      COMMON /SUBS/ IR, IS, IO
      COMMON /WAVE/ K, L, M
      INTEGER INTGER(NVAR)
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
      ICOUNT=NEQN-1
      REWIND (UNIT=9)
      WRITE(ICOUT,*)'Y IN FUNC'
      DO 55 I=1,ICOUNT
        WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
55    CONTINUE
C --- The correct element of Y is assigned on the same basis
C --- as in the main program i.e. using the counter Ncount which is
C --- incremented for every admissible rs pair
C --- Equation for energy balance
      DO 20 NCOUNT=1,ICOUNT/2
        IR=ISUB1(NCOUNT)
```

```
         IS=ISUB2(NCOUNT)
         IG=ISUB3(NCOUNT)
            NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
C --- Set flag for equation
            IFLAG=1
            CALL SUMB(Y, ICOUNT)
            CALL SUMTRIP(Y, ICOUNT, IFLAG)
            IF((IR.EQ.0) AND (IS.EQ.0)) THEN
              TERM=(1.0D0-(-1.0D0)**IG)/DFLOAT(IG)/PI
                  +0.5D0*Y(NCOUNT)
                 +SUM1+SUM2+0.5D0*(SUM4+SUM5)
            ELSE
              TERM=0.5D0*Y(NCOUNT)+SUM1+SUM2+0.5D0*(SUM4+SUM5)
            END IF
            F(NCOUNT)=-SUM3-(PI*PI*DFLOAT(IG*IG)+NU*NU)*0.5D0*Y(NCOUNT)
                 +Y(NEQN)*TERM
            WRITE(ITERM,*)F(NCOUNT), NCOUNT
      CONTINUE
C --- Assemble Jacobian for the energy equation
      DO 15 J=1,ICOUNT,2
         IR=ISUB1(J)
         IS=ISUB2(J)
         IG=ISUB3(J)
         DO 20 I=1,ICOUNT
           CALL DYSUMB(Y, ICOUNT, I)
           CALL DYSUMTRP(Y, ICOUNT, IFLAG, I)
           IF(I.EQ.J) THEN
             AA(J,I)=-0.5D0*(
                 (DFLOAT(ISUB1(I))*K)**2+(DFLOAT(ISUB2(I))*L)**2
                 +PI**2*DFLOAT(ISUB3(I)**2))
                 -SUM3+Y(NEQN)*(0.5D0+SUM1+SUM2+0.5D0*(SUM4+
                 SUM5))
           ELSE
             AA(J,I)=-SUM3+Y(NEQN)*(0.5D0+SUM1+SUM2+0.5D0*(SUM4+
                 SUM5))
           END IF
         WRITE(NOUT,*)AA(J,I)
         CONTINUE
      CONTINUE
C ---
C --- Equation for the concentration balance
      DO 30 NCOUNT=ICOUNT/2+1, ICOUNT
         IR=ISUB1(NCOUNT)
         IS=ISUB2(NCOUNT)
         IG=ISUB3(NCOUNT)
            NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
C --- Set flag for equation
            IFLAG=2
            CALL SUMB(Y, ICOUNT)
            CALL SUMTRIP(Y, ICOUNT, IFLAG)
            IF((IR.EQ.0) AND (IS.EQ.0)) THEN
              TERM=DSIN((DFLOAT(IG)-0.5D0)*PI)/(PI*(DFLOAT(IG)-0.5D0))
                  +0.5D0*Y(NCOUNT)
                  +SUM1+SUM2+0.5D0*(SUM4+SUM5)
            ELSE
              TERM=0.5D0*Y(NCOUNT)+SUM1+SUM2+0.5D0*(SUM4+SUM5)
            END IF
            F(NCOUNT)=-SUM3-(PI*PI*(DFLOAT(IG)-0.5D0)**2+NU*NU)
                          *0.5D0*Y(NCOUNT)*XLE
     *              -Y(NEQN)/BETA/GAMMA*TERM
C           WRITE(ITERM,*)F(NCOUNT), NCOUNT
30    CONTINUE
C --- Assemble Jacobian for the concentration equation
      DO 25 J=ICOUNT/2+1,ICOUNT
         IR=ISUB1(J)
         IS=ISUB2(J)
         IG=ISUB3(J)
         DO 50 I=1,ICOUNT
           CALL DYSUMB(Y, ICOUNT, I)
           CALL DYSUMTRP(Y, ICOUNT, IFLAG, I)
           IF(I.EQ.J) THEN
              AA(J,I)=-0.5D0*XLE*(
     #              (DFLOAT(ISUB1(I))*K)**2+(DFLOAT(ISUB2(I))*L)**2
     #              +PI**2*DFLOAT(ISUB3(I)**2))
     #              -SUM3+Y(NEQN)*(0.5D0+SUM1+SUM2+0.5D0*(SUM4+
     #              SUM5))
           ELSE
             AA(J,I)=-SUM3+Y(NEQN)*(0.5D0+SUM4+SUM2+0.5D0*(SUM4+
     #              SUM5))
           END IF
C        WRITE(NOUT,*)AA(J,I)
50       CONTINUE
25    CONTINUE
C ---
C --- Call F02AFF to find the eigenvalues of AA
C --- Parameters for F02AFF
      N=ICOUNT
      IA=ICOUNT
      IFAIL=1
      CALL F02AFF(AA, IA, N, RR, RI, INTGER, IFAIL)
      IF(IFAIL.GT.0) THEN
        WRITE(NOUT,*)'IFAIL', IFAIL
        STOP
      END IF
      WRITE(NOUT,*)'EIGENVALUES'
      DO 60 I=1,ICOUNT
       WRITE(NOUT,*)RR(I), RI(I)
60    CONTINUE
      F(NEQN)=1.0D0
      DO 70 I=1,ICOUNT
        F(NEQN)=RR(I)*F(NEQN)
70    CONTINUE
      F(NEQN)=F(NEQN)*1.0D-4
      WRITE(ITERM,*)F(NEQN), Y(NEQN)
      RETURN
      END
C
      SUBROUTINE SUMB(Y, ICOUNT)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the multiple summation arising from non-linear term
      REAL*8 KAPPA, K, L
      REAL*8 Y(200)
      COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
      COMMON /COEFF/ B(-20:41,-20:41,20),
     #               C(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      COMMON /SUBS/ IR, IS, IG
```

```
[illegible] 10
[illegible] zeros except those that are real
[illegible]
      [illegible], AERR, RERR, IRULE, A21, ER)
      [illegible], AERR, RERR, IRULE, A22, ER)
      [illegible] 'IER A2', IER
      [illegible] A2
      [illegible], AERR, RERR, IRULE, A61, ER)
      [illegible], AERR, RERR, IRULE, A62, ER)
      [illegible] 'IER A6', IER
      [illegible] A6
[illegible]
      [illegible] AND [illegible]
      [illegible], AERR, RERR, IRULE, AK21, ER)
      [illegible], AERR, RERR, IRULE, AK22, ER)
      [illegible] 'IER AK2', IER
      [illegible], AERR, RERR, IRULE, BK21, ER)
      [illegible], AERR, RERR, IRULE, BK22, ER)
      [illegible] 'IER BK2', IER
      [illegible] KAPPA) [illegible]
      [illegible], AERR, RERR, IRULE, AK21, ER)
      [illegible], AERR, RERR, IRULE, AK22, ER)
      [illegible] 'IER AK2', IER
      [illegible], AERR, RERR, IRULE, BK21, ER)
      [illegible], AERR, RERR, IRULE, BK22, ER)
      [illegible] 'IER BK2', IER
      [illegible] AK2
      [illegible]
      COMMON /SUB/ ISUB, [illegible], ISUB[illegible]
      COMMON /REALS/ BS, [illegible], SIG, SIH, [illegible], KAPPA, SA
      COMMON /MAXS/ MAXIK, MAXIG, MAXI[illegible], MINIK, MINIG, MINI[illegible]
      COMMON /[illegible]/ [illegible]
      COMMON /WORK/ [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
```

```
      IF(IQ(IP).NE.IS) GO TO 10
C
C  reassign dummy subscripts that are real
      SIG=DFLOAT(IG)
      SIE=DFLOAT(IE)
      SIH=DFLOAT(IH)
C
C  calculate SUM2
C
C  energy equation
         IF(IFLAG.EQ.1) THEN
           CALL DQDAG(F2, X1, X2, AERR, RERR, IRULE, A21, ER)
           CALL DQDAG(F2, X2, X3, AERR, RERR, IRULE, A22, ER)
           A2=A21+A22
           IF (IER.GT.0) WRITE(ITERM,*)'IER A2', IER
           SUM2=SUM2+Y(IP,CNT)*Y(IQ,CNT)[illegible]A2
         END IF
C  concentration equation
         IF(IFLAG.EQ.2) THEN
           CALL DQDAG(F6, X1, X2, AERR, RERR, IRULE, A61, ER)
           CALL DQDAG(F6, X2, X3, AERR, RERR, IRULE, A62, ER)
           A6=A61+A62
           IF (IER.GT.0) WRITE(ITERM,*)'IER A6', IER
           SUM6=SUM6+[illegible]A6
         END IF
C
C  calculate [illegible]
C
C  energy equation
C
        IF(IFLAG.EQ.1) THEN
          KAPPA=[illegible]
          IF [illegible] AND [illegible]
     THEN
            CALL DQDAG(FAK21, X1, X2, AERR, RERR, IRULE, AK21, ER)
            CALL DQDAG(FAK21, X2, X3, AERR, RERR, IRULE, AK22, ER)
            AK2=AK21+AK22
            IF (IER.GT.0) WRITE(ITERM,*)'IER AK2', IER
            CALL DQDAG(FBK21, X1, X2, AERR, RERR, IRULE, BK21, ER)
            CALL DQDAG(FBK21, X2, X3, AERR, RERR, IRULE, BK22, ER)
            BK2=BK21+BK22
            IF (IER.GT.0) WRITE(ITERM,*)'IER BK2', IER
          ELSE
            [illegible] KAPPA=(DEXP(KAPPA)+DEXP(-KAPPA))
     [illegible]*KAPPA**2)
            [illegible]*KAPPA**2)
            CALL DQDAG(FAK21, X1, X2, AERR, RERR, IRULE, AK21, ER)
            CALL DQDAG(FAK21, X2, X3, AERR, RERR, IRULE, AK22, ER)
            AK2=AK21+AK22
            IF(IER.GT.0) WRITE(ITERM,*)'IER AK2', IER
            CALL DQDAG(FBK21, X1, X2, AERR, RERR, IRULE, BK21, ER)
            CALL DQDAG(FBK21, X2, X3, AERR, RERR, IRULE, BK22, ER)
            BK2=BK21+BK22
            IF(IER.GT.0) WRITE(ITERM,*)'IER BK2', IER
          END IF
          SUM3=SUM3+Y(IP,CNT)*Y(IQ,CNT)*
     &         (-(DFLOAT(IP-1))*K*K+DFLOAT(IQ-1)[illegible]AK2
     &         *KAPPA*KAPPA*BK2)
        END IF
```

```
C --- Concentration equation
         IF(IFLAG.EQ.2) THEN
           KAPPA=DSQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
           IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
           THEN
             CALL DQDAG(FCK20, X1, X2, AERR, RERR, IRULE, CK21, ER)
             CALL DQDAG(FCK20, X2, X3, AERR, RERR, IRULE, CK22, ER)
             CK2=CK21+CK22
             IF(IER.GT.0) WRITE(ITERM,*)'IER CK2', IER
             CALL DQDAG(FDK20, X1, X2, AERR, RERR, IRULE, DK21, ER)
             CALL DQDAG(FDK20, X2, X3, AERR, RERR, IRULE, DK22, ER)
             DK2=DK21+DK22
             IF(IER.GT.0) WRITE(ITERM,*)'IER DK2', IEP
           ELSE
C --- KAPPA.NE.0
             C1=RA*PI*XIG*KAPPA*(DEXP(KAPPA)+DEXP(-KAPPA))/
     #             (PI**2*XIG**2+KAPPA**2)
             C2=RA*(PI**2*XIG**2+KAPPA**2)
             CALL DQDAG(FCK21, X1, X2, AERR, RERR, IRULE, CK21, ER)
             CALL DQDAG(FCK21, X2, X3, AERR, RERR, IRULE, CK22, ER)
             CK2=CK21+CK22
             IF(IER.GT.0) WRITE(ITERM,*)'IER AK2', IER
             CALL DQDAG(FDK21, X1, X2, AERR, RERR, IRULE, DK21, ER)
             CALL DQDAG(FDK21, X2, X3, AERR, RERR, IRULE, DK22, ER)
             DK2=DK21+DK22
             IF(IER.GT.0) WRITE(ITERM,*)'IER DK2', IER
           END IF
           SUM3=SUM3+Y(ICOUNT)*Y(ICOUNT+ICOUNT/2)*
     #          ([illegible]ATE(IT)*K*K+DFLOAT(IQ*IU)*L*L)*CK2
     #          +KAPPA*KAPPA*DK2)
         END IF
C --- Calculate SUM4
C --- Energy equation
         IF(IFLAG.EQ.1) THEN
           CALL DQDAG(F3, X1, X2, AERR, RERR, IRULE, A31, ER)
           CALL DQDAG(F3, X2, X3, AERR, RERR, IRULE, A32, ER)
           A3=A31+A32
           IF(IER.GT.0) WRITE(ITERM,*)'IER A3', IER
           SUM4=SUM4+Y(IHCNT)*Y(ILCNT)*A3
         END IF
C --- Concentration equation
         IF(IFLAG.EQ.2) THEN
           CALL DQDAG(F7, X1, X2, AERR, RERR, IRULE, A71, ER)
           CALL DQDAG(F7, X2, X3, AERR, RERR, IRULE, A72, ER)
           A7=A71+A72
           IF(IER.GT.0) WRITE(ITERM,*)'IER A7', IER
           SUM4=SUM4+Y(IHCNT)*Y(ILCNT)*A7
         END IF
   50 CONTINUE
      RETURN
      END

      SUBROUTINE DYSUMB(Y, ICOUNT, IJAC)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the multiple summation arising from non-linear term
      REAL*8 KAPPA, K, L
      REAL*8 Y(200)
      COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
      COMMON /COEFF/ B(-20:41,-20:41,20),
     #                C(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      COMMON /SUBS/ IR, IS, IG
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /TRC/ ITRC(-20:41,-20:41,20)
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL F1, F2, F3, F5, F6, F7, FAKZO, FAKZ1, FBK20, FBK21,
     #         FCK20, FCK21, FDK20, FDK21
      DATA ITERM /3/
      RA=(BETA*DD*DSQRT(GAMMA)/Y(ICOUNT+1))**2
      SUM1=0.0D0
      SUM2=0.0D0
      SUM3=0.0D0
      SUM4=0.0D0
      E=DEXP(1.0D0)
C --- Parameters for DQDAG
C --- We split up the region of integration to avoid errors with
C --- periodic functions
      X1=0.0D0
      X2=0.7D0
      X3=1.0D0
      AERR=1.0D-5
      RERR=1.0D-5
C --- Quadrature rule for oscillatory functions
      IRULE=6
C ---
      DO 50 IF=1,MAXIG
        XIF=DFLOAT(IF)
        XIG=DFLOAT(IG)
        CALL DQDAG(F1, XI, X2, AERR, RERR, IRULE, A11, ER)
        CALL DQDAG(F1, X2, X3, AERR, RERR, IRULE, A12, ER)
        A1=A11+A12
        IF(IER.GT.0) WRITE(ITERM,*)'IER A1', IER
        CALL DQDAG(F5, XI, X2, AERR, RERR, IRULE, A51, ER)
        CALL DQDAG(F5, X2, X3, AERR, RERR, IRULE, A52, ER)
        A5=A51+A52
        IF(IER.GT.0) WRITE(ITERM,*)'IER A5', IER
C --- Energy equation
        IF(IFLAG.EQ.1) THEN
          N=ITRC(IR,IS,IF)
          IF(N.EQ.IJAC) THEN
            TERM=1.0D0
          ELSE
            TERM=0.0D0
          END IF
          SUM1=SUM1+A1*TERM
        END IF
C --- Concentration equation
        IF(IFLAG.EQ.2) THEN
          N=ITRC(IR,IS,IF)+ICOUNT/2
          IF(N.EQ.IJAC) THEN
            TERM=1.0D0
          ELSE
            TERM=0.0D0
```

```
C --- Concentration equation
          IF(IFLAG.EQ.2) THEN
            KAPPA=DSQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
            IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
            THEN
              CALL DQDAG(FCK20, X1, X2, AERR, RERR, IRULE, CK21, ER)
              CALL DQDAG(FCK20, X2, X3, AERR, RERR, IRULE, CK22, ER)
              CK2=CK21+CK22
              IF(IER.GT.0) WRITE(ITERM,*)'IER CK2', IER
              CALL DQDAG(FDK20, X1, X2, AERR, RERR, IRULE, DK21, ER)
              CALL DQDAG(FDK20, X2, X3, AERR, RERR, IRULE, DK22, ER)
              DK2=DK21+DK22
              IF(IER.GT.0) WRITE(ITERM,*)'IER DK2', IER
            ELSE
C --- KAPPA.NE.0
              C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     #              (PI**2*XIG**2+KAPPA**2)
              C2=RA/(PI**2*XIG**2+KAPPA**2)
              CALL DQDAG(FCK21, X1, X2, AERR, RERR, IRULE, CK21, ER)
              CALL DQDAG(FCK21, X2, X3, AERR, RERR, IRULE, CK22, ER)
              CK2=CK21-CK22
              IF(IER.GT.0) WRITE(ITERM,*)'IER AK2', IER
              CALL DQDAG(FDK21, X1, X2, AERR, RERR, IRULE, DK21, ER)
              CALL DQDAG(FDK21, X2, X3, AERR, RERR, IRULE, DK22, ER)
              DK2=DK21+DK22
              IF(IER.GT.0) WRITE(ITERM,*)'IER DK2', IER
            END IF
            SUM3=SUM3+Y(IPQCNT)*Y(ITUCNT+ICOUNT/2)*
     #          (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*CK2
     #          -KAPPA*KAPPA*DK2)
          END IF
C --- CALCULATE SUM4

C --- Energy equation
          IF(IFLAG.EQ.1) THEN
            CALL DQDAG(F3, X1, X2, AERR, RERR, IRULE, A31, ER)
            CALL DQDAG(F3, X2, X3, AERR, RERR, IRULE, A32, ER)
            A3=A31+A32
            IF(IER.GT.0) WRITE(ITERM,*)'IER A3', IER
            SUM4=SUM4+Y(IPQCNT)*Y(ITUCNT)*A3
          END IF
C --- Concentration equation
          IF(IFLAG.EQ.2) THEN
            CALL DQDAG(F7, X1, X2, AERR, RERR, IRULE, A71, ER)
            CALL DQDAG(F7, X2, X3, AERR, RERR, IRULE, A72, ER)
            A7=A71+A72
            IF(IER.GT.0) WRITE(ITERM,*)'IER A7', IER
            SUM4=SUM4+Y(IPQCNT)*Y(ITUCNT)*A7
          END IF
 50   CONTINUE
      RETURN
      END

      SUBROUTINE DYSUMB(Y, ICOUNT, IJAC)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the multiple summation arising from non-linear term
      REAL*8 KAPPA, K, L
      REAL*8 Y(200)
      COMMON /SUM/ SUM1, SUM2, SUM3, SUM4, IFLAG, NCOUNT
      COMMON /COEFF/ B(-20:41,-20:41,20),
     #               C(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      COMMON /SUBS/ IR, IS, IO
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIO, MINIR, MINIS, MINIO
      COMMON /TRC/ ITRC(-20:41,-20.41,20)
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL F1, F2, F3, F5, F6, F7, FAK20, FAK21, FBK20, FBK21,
     #         FCK20, FCK21, FDK20, FDK21
      DATA ITERM /3/
      RA=(BETA*DD*DSQRT(GAMMA)/Y(ICOUNT+1))**2
      SUM1=0.0D0
      SUM2=0.0D0
      SUM3=0.0D0
      SUM4=0.0D0
      E=DEXP(1.0D0)
C --- Parameters for DQDAG
C --- We split up the region of integration to avoid errors with
C --- periodic functions
      X1=0.0D0
      X2=0.7D0
      X3=1.0D0
      AERR=1.0D-5
      RERR=1.0D-5
C --- Quadrature rule for oscillatory functions
      IRULE=6
C ---
      DO 50 IF=1,MAXIO
        XIF=DFLOAT(IF)
        XIG=DFLOAT(IO)
        CALL DQDAG(F1, X1, X2, AERR, RERR, IRULE, A11, ER)
        CALL DQDAG(F1, X2, X3, AERR, RERR, IRULE, A12, ER)
        A1=A11+A12
        IF(IER.GT.0) WRITE(ITERM,*)'IER A1', IER
        CALL DQDAG(F5, X1, X2, AERR, RERR, IRULE, A51, ER)
        CALL DQDAG(F5, X2, X3, AERR, RERR, IRULE, A52, ER)
        A5=A51+A52
        IF(IER.GT.0) WRITE(ITERM,*)'IER A5', IER
C --- Energy equation
        IF(IFLAG.EQ.1) THEN
          N=ITRC(IR,IS,IF)
          IF(N.EQ.IJAC) THEN
            TERM=1.0D0
          ELSE
            TERM=0.0D0
          END IF
          SUM1=SUM1+A1*TERM
        END IF
C --- Concentration equation
        IF(IFLAG.EQ.2) THEN
          N=ITRC(IR,IS,IF)+ICOUNT/2
          IF(N.EQ.IJAC) THEN
            TERM=1.0D0
          ELSE
            TERM=0.0D0
      END IF
      SUM1=SUM1+A5*[illegible]
    END IF
 [illegible]NTINUE

C [illegible]NT and IT[illegible] [illegible] counters for the pq and tu [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible] ITUCNT=1,I[illegible]NT,2
    [illegible]
    [illegible]
    [illegible]
      [illegible] ([illegible]P+IT) NE [illegible] GO TO 10
      [illegible] ([illegible]Q+IU) NE [illegible] GO TO 10

[illegible] subscripts that are real
      [illegible]
      [illegible]
      [illegible]

[illegible]

[illegible]
        [illegible] THEN
          [illegible]
          [illegible]
          [illegible]
          [illegible] WRITE(ITERM,*)'IER A2', IER
          [illegible] THEN
            [illegible]
          [illegible] THEN
            [illegible]
          [illegible]
            [illegible]
          END IF
          [illegible]+TERM A[illegible]
        END IF
 [illegible] equation
        [illegible] THEN
          CALL [illegible]
          CALL [illegible] A62, ER)
          A6=A61+[illegible]
          IF(IER GT 0) WRITE(ITERM,*)'IER A6', IER
          IF([illegible] THEN
            TERM=Y([illegible])
          ELSE IF([illegible] THEN
            TERM=Y([illegible])
          ELSE
            TERM=0.0D0
          END IF
          SUM2=SUM2+TERM*A6
        END IF

[illegible] SUM2

[illegible] Energy equation
```

```
          IF(IFLAG EQ.1) THEN
            KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
            IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *      THEN
              CALL DQDAG (FAKZ0, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
              CALL DQDAG (FAKZ0, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
              AKZ=AKZ1+AKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ', IER
              CALL DQDAG (FBKZ0, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
              CALL DQDAG (FBKZ0, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
              BKZ=BKZ1+BKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ', IER
            ELSE
              C1=RA*PI*XI[illegible]APPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *              (PI*[illegible]IG**2+KAPPA**2)
              C2=RA/(PI*[illegible]IG**2+KAPPA**2)
              CALL DQDAG (FAKZ1, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
              CALL DQDAG (FAKZ1, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
              AKZ=AKZ1+AKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ', IER
              CALL DQDAG (FBKZ1, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
              CALL DQDAG (FBKZ1, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
              BKZ=BKZ1+BKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ', IER
            END IF
            IF((IPQCNT.EQ.IJAC).AND.(ITUCNT.NE.IJAC)) THEN
              TERM=Y(ITUCNT)
            ELSE IF((ITUCNT.EQ.IJAC).AND.(IPQCNT.NE.IJAC)) THEN
              TERM=Y(IPQCNT)
            ELSE IF((IPQCNT.EQ.IJAC).AND.(ITUCNT.EQ.IJAC)) THEN
              TERM=2.0D0*Y(IJAC)
            ELSE
              TERM=0.0D0
            END IF
            SUM3=SUM3+TERM*
     *           (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*AKZ
     *           +KAPPA*KAPPA*BKZ)
          END IF
C---
C --- Concentration equation
C---
          IF(IFLAG.EQ.2) THEN
            KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
            IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *      THEN
              CALL DQDAG(FCKZ0, X1, X2, AERR, RERR, IRULE, CKZ1, ER)
              CALL DQDAG(FCKZ0, X2, X3, AERR, RERR, IRULE, CKZ2, ER)
              CKZ=CKZ1+CKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER CKZ', IER
              CALL DQDAG(FDKZ0, X1, X2, AERR, RERR, IRULE, DKZ1, ER)
              CALL DQDAG(FDKZ0, X2, X3, AERR, RERR, IRULE, DKZ2, ER)
              DKZ=DKZ1+DKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER DKZ', IER
            ELSE
C --- KAPPA.NE.0
              C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *              (PI**2*XIG**2+KAPPA**2)
              C2=RA/(PI**2*XIG**2+KAPPA**2)
              CALL DQDAG(FCKZ1, X1, X2, AERR, RERR, IRULE, CKZ1, ER)
          END IF
          SUM1=SUM1+A5*TERM
        END IF
 10   CONTINUE
C
C --- IPQCNT and ITUCNT are counters for the pq and tu coeffs
      DO 20 IPQCNT=1,ICOUNT/2
        IP=ISUB1(IPQCNT)
        IQ=ISUB2(IPQCNT)
        IG=ISUB3(IPQCNT)
        DO 20 ITUCNT=1,ICOUNT/2
          IT=ISUB1(ITUCNT)
          IU=ISUB2(ITUCNT)
          IH=ISUB3(ITUCNT)
          IF((IP+IT).NE.IR) GO TO 10
          IF((IQ+IU).NE.IS) GO TO 10
C
C --- make dummy subscripts that are real
          XIG=DFLOAT(IG)
          XIF=DFLOAT(IIF)
          XIH=DFLOAT(IH)
C
C --- calculate SUM2
C
C --- Energy equation
          IF(IFLAG.EQ.1) THEN
            CALL DQDAG(F2, X1, X2, AERR, RERR, IRULE, A21, ER)
            CALL DQDAG(F2, X2, X3, AERR, RERR, IRULE, A22, ER)
            A2=A21+A22
            IF(IER.GT.0) WRITE(ITERM,*)'IER A2', IER
            IF(IPQCNT.EQ.IJAC) THEN
              TERM=Y(ITUCNT+ICOUNT/2)
            ELSE IF((ITUCNT+ICOUNT/2).EQ.IJAC) THEN
              TERM=Y(IPQCNT)
            ELSE
              TERM=0.0D0
            END IF
            SUM2=SUM2+TERM*A2
          END IF
C --- Concentration equation
          IF(IFLAG.EQ.2) THEN
            CALL DQDAG(F6, X1, X2, AERR, RERR, IRULE, A61, ER)
            CALL DQDAG(F6, X2, X3, AERR, RERR, IRULE, A62, ER)
            A6=A61+A62
            IF(IER.GT.0) WRITE(ITERM,*)'IER A6', IER
            IF(IPQCNT.EQ.IJAC) THEN
              TERM=Y(ITUCNT+ICOUNT/2)
            ELSE IF((ITUCNT+ICOUNT/2).EQ.IJAC) THEN
              TERM=Y(IPQCNT)
            ELSE
              TERM=0.0D0
            END IF
            SUM2=SUM2+TERM*A6
          END IF
C
C --- calculate SUM3
C
C --- Energy equation
C
          IF(IFLAG.EQ.1) THEN
            KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
            IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *      THEN
              CALL DQDAG (FAKZ0, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
              CALL DQDAG (FAKZ0, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
              AKZ=AKZ1+AKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ', IER
              CALL DQDAG (FBKZ0, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
              CALL DQDAG (FBKZ0, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
              BKZ=BKZ1+BKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ', IER
            ELSE
              C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *               (PI**2*XIG**2+KAPPA**2)
              C2=RA/(PI**2*XIG**2+KAPPA**2)
              CALL DQDAG (FAKZ1, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
              CALL DQDAG (FAKZ1, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
              AKZ=AKZ1+AKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ', IER
              CALL DQDAG (FBKZ1, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
              CALL DQDAG (FBKZ1, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
              BKZ=BKZ1+BKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ', IER
            END IF
            IF((IPQCNT.EQ.IJAC).AND.(ITUCNT.NE.IJAC)) THEN
              TERM=Y(ITUCNT)
            ELSE IF((ITUCNT.EQ.IJAC).AND.(IPQCNT.NE.IJAC)) THEN
              TERM=Y(IPQCNT)
            ELSE IF((IPQCNT.EQ.IJAC).AND.(ITUCNT.EQ.IJAC)) THEN
              TERM=2.0D0*Y(IJAC)
            ELSE
              TERM=0.0D0
            END IF
            SUM3=SUM3+TERM*
     *            ((DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*AKZ
     *            +KAPPA*KAPPA*BKZ)
          END IF
C---
C --- Concentration equation
C---
          IF(IFLAG.EQ.2) THEN
            KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
            IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *      THEN
              CALL DQDAG(FCKZ0, X1, X2, AERR, RERR, IRULE, CKZ1, ER)
              CALL DQDAG(FCKZ0, X2, X3, AERR, RERR, IRULE, CKZ2, ER)
              CKZ=CKZ1+CKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER CKZ', IER
              CALL DQDAG(FDKZ0, X1, X2, AERR, RERR, IRULE, DKZ1, ER)
              CALL DQDAG(FDKZ0, X2, X3, AERR, RERR, IRULE, DKZ2, ER)
              DKZ=DKZ1+DKZ2
              IF(IER.GT.0) WRITE(ITERM,*)'IER DKZ', IER
            ELSE
C --- KAPPA.NE.0
              C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *               (PI**2*XIG**2+KAPPA**2)
              C2=RA/(PI**2*XIG**2+KAPPA**2)
              CALL DQDAG(FCKZ1, X1, X2, AERR, RERR, IRULE, CKZ1, ER)
```

[illegible]

```
      END
C ---
      SUBROUTINE SUMTRIP(Y, ICOUNT, IFLAG)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the triple summation arising from non-linear term
      REAL*8 KAPPA, K, L
      REAL*8 Y(200)
      COMMON /COEFF/ B(-20:41,-20:41,20),
     #                C(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      COMMON /SUBS/ IR, IS, IO
      COMMON /REALSUBS/ XIF, XIO, XIH, XIL, C1, C2, KAPPA, RA
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIO, MINIR, MINIS, MINIO
      COMMON /TRC/ ITRC(-20:41,-20:41,20)
      COMMON /TRPSUM/  SUMS
      COMMON /SUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL F4, F8
      DATA ITERM /3/
      RA= BETA*DD*DSQRT(GAMMA)/Y(ICOUNT+1))**2
      SUMS=0.0D0
      E=DEXP(1.0D0)
C --- Parameters for DQDAG
C --- We split up the region of integration to avoid errors with
C --- periodic functions
      X1=0.0D0
      X2=0.5D0
      X3=1.0D0
      AERR=1.0D-5
      RERR=1.0D-5
C --- quadrature rule for oscillatory functions
      IRULE=6
C ---
C --- Check that P+T+M=R, Q+U+N=S, P+Q, T+U M+N even,
C --- P,Q, T,U M,N truncated
      DO 10 IPQCNT=1,ICOUNT/2
        IP=ISUB1(IPQCNT)
        IQ=ISUB2(IPQCNT)
        IH=ISUB3(IPQCNT)
        DO 10 ITUCNT=1,ICOUNT/2
          IT=ISUB1(ITUCNT)
          IU=ISUB2(ITUCNT)
          I.F=ISUB3(ITUCNT)
          DO 10 IMNCNT=1,ICOUNT/2
            IM=ISUB1(IMNCNT)
            IN=ISUB2(IMNCNT)
            IL=ISUB3(IMNCNT)
              IF((IP+IT+IM).NE.IR) GO TO 10
              IF((IQ+IU+IN).NE.IS) GO TO 10
C ---
C --- assign dummy subscripts that are real
              XIG=DFLOAT(IG)
              XIF=DFLOAT(IIF)
              XIH=DFLOAT(IH)
              XIL=DFLOAT(IL)
C --- Calculate summation
C ---
C --- Energy equation
```

```
      IF(IFLAG.EQ.1) THEN
        CALL DQDAG(F4, X1, X2, AERR, RERR, IRULE, A41, ER)
        CALL DQDAG(F4, X2, X3, AERR, RERR, IRULE, A42, ER)
        A4=A41+A42
        IF(IER.GT.0) WRITE(ITERM,*)'IER A4', IER
        SUM[illegible]+Y(IPQCNT)*Y(IT[illegible])*Y(IMNCNT+ICOUNT/2)*A4
      END IF
C --- [illegible]ation equation
      IF(IFLAG.EQ.2) THEN
        CALL DQDAG(F8, X1, X2, AERR, RERR, IRULE, A81, ER)
        CALL DQDAG(F8, X2, X3, AERR, RERR, IRULE, A82, ER)
        A8=A81+A82
        IF(IER.GT.0) WRITE(ITERM,*)'IER A8', IER
        SUM5=SUM5+Y(IPQCNT)*Y(ITUCNT+ICOUNT/2)*Y(IMNCNT)*A8
      END IF

      [illegible]
      [illegible]
C [illegible] the triple summation arising from non-linear term
C [illegible] K, L
      [illegible]
      [illegible] KAPPA, KA
      [illegible]
      [illegible] GAMMA [illegible]
      [illegible]
C [illegible] for DQDAG
C [illegible] the region of integration to avoid errors with
C [illegible] functions

C [illegible] for [illegible] auxiliary functions
      [illegible]

C [illegible] ITUCNT IMNCNT are counters for the pq, tu mn coeffs
C [illegible] P+T+M=R, Q+U+N=S, P+Q, T+U M+N even,
C [illegible] M,N truncated
      DO 10 IPQCNT=1,ICOUNT/2
        IP=ISUB1(IPQCNT)
        IQ=ISUB2(IPQCNT)
        IG=ISUB3(IPQCNT)
```

```
        DO 10 ITUCNT=1,ICOUNT/2
          IT=ISUB1(ITUCNT)
          IU=ISUB2(ITUCNT)
          IIF=ISUB3(ITUCNT)
          DO 10 IMNCNT=1,ICOUNT/2
            IM=ISUB1(IMNCNT)
            IN=ISUB2(IMNCNT)
            IL=ISUB3(IMNCNT)
              IF((IP+IT+IM).NE.IR) GO TO 10
              IF((IQ+IU+IN).NE.IS) GO TO 10
C ---
C --- assign dummy subscripts that are real
              XIG=DFLOAT(IG)
              XIF=DFLOAT(IIF)
              XIH=DFLOAT(IH)
              XIL=DFLOAT(IL)
C --- Calculate summation
C ---
C --- Energy equation
              IF(IFLAG.EQ.1) THEN
                CALL DQDAG(F4, X1, X2, AERR, RERR, IRULE, A41, ER)
                CALL DQDAG(F4, X2, X3, AERR, RERR, IRULE, A42, ER)
                A4=A41+A42
                IF(IER.GT.0) WRITE(ITERM,*)'IER A4', IER
                IF((IPQCNT.EQ.IJAC).AND.(ITUCNT.NE.IJAC).AND.
     #            ((IMNCNT+ICOUNT/2).NE.IJAC)) THEN
                  TERM=Y(ITUCNT)*Y(IMNCNT+ICOUNT/2)
                ELSE IF((IPQCNT.NE.IJAC).AND.(ITUCNT.EQ.IJAC).AND.
     #            ((IMNCNT+ICOUNT/2).NE.IJAC)) THEN
                  TERM=Y(IPQCNT)*Y(IMNCNT+ICOUNT/2)
                ELSE IF((IPQCNT.EQ.IJAC).AND.(ITUCNT.EQ.IJAC).AND.
     #            ((IMNCNT+ICOUNT/2).NE.IJAC)) THEN
                  TERM=2.0D0*Y(IJAC)*Y(IMNCNT+ICOUNT/2)
                ELSE IF((IPQCNT.NE.IJAC).AND.(ITUCNT.NE.IJAC).AND.
     #            ((IMNCNT+ICOUNT/2).EQ.IJAC)) THEN
                  TERM=Y(IPQCNT)*Y(ITUCNT)
                ELSE
                  TERM=0.0D0
                END IF
                SUM5=SUM5+TERM*A4
              END IF
C --- Con[illegible]ation equation
              IF(IFLAG.EQ.2) THEN
                CALL DQDAG(F8, X1, X2, AERR, RERR, IRULE, A81, ER)
                CALL DQDAG(F8, X2, X3, AERF, RERR, IRULE, A82, ER)
                A8=A81+A82
                IF(IER.GT.0) WRITE(ITERM,*)'IER A8', IER
                IF((IPQCNT.EQ.IJAC).AND.((ITUCNT+ICOLNT/2).NE.IJAC).AND.
     #            (IMNCNT.NE.IJAC)) THEN
                  TERM=Y(ITUCNT+ICOUNT/2)*Y(IMNCNT)
                ELSE IF((IPQCNT.NE.IJAC).AND.((ITUCNT+ICOUNT/2).EQ.IJAC)
     #              .AND.(IMNCNT.NE.IJAC)) THEN
                  TERM=Y(IPQCNT)*Y(IMNCNT)
                ELSE IF((IPQCNT.EQ.IJAC).AND.((ITUCNT+ICOUNT/2).NE.IJAC)
     #          .AND.(IMNCNT.EQ.IJAC)) THEN
                  TERM=2.0D0*Y(IJAC)*Y(ITUCNT+ICOUNT/2)
                ELSE IF((IPQCNT.NE.IJAC).AND.((ITUCNT+ICOUNT/2).NE.IJAC)
     #          .AND.(IMNCNT.EQ.IJAC)) THEN
                  TERM=Y(IPQCNT)*Y(ITUCNT+ICOUNT/2)
            ELSE
              TERM=0.0D0
            END IF
            SUMS=SUMS+TERM*AA
          END IF
    CONTINUE
    RETURN
    END
C ---
    SUBROUTINE FLOW(Y, ICOUNT)
    IMPLICIT REAL*8 (A-H,O-Z)
C -- Results for plotting
    REAL*8 K, L , KAPPA
    COMMON /WAVE/ K, L, M
    COMMON /COEFF/ B(-20:41,-20:41,20),
   #               C(-20:41,-20:41,20)
    REAL*8 Y(200)
    COMMON /SUB/ ISUB1(200), ISUB2(200), ISUB3(200)
    REAL*8 COEFF(-20:41,-20:41,0)
    COMMON /PARAMS/ BETA, GAMMA, FI, CHRL, DD, XLE
    DATA TERM/0/
    RA=[illegible]DSQRT[illegible]GAMMA[illegible]INT+1[illegible]2
C -- Energy equation
    DO [illegible],ICOUNT/2
        B(I[illegible]BI(I),ISUB2(I),[illegible](I))=Y(I)
        [illegible]
        [illegible]
        [illegible]
    [illegible]
C -- [illegible] equation
    [illegible]
    [illegible]
    [illegible]
    [illegible]
    [illegible]
    CONTINUE
C -- [illegible] the height ZZ
C
    [illegible]
    DO [illegible]
      THETA=0.0D0
      VELZ=0.0D0
      CONC=0.0D0
C -- Energy equation
      DO 20 NCOUNT=1,ICOUNT/2
        I=ISUB1(NCOUNT)
        J=ISUB2(NCOUNT)
        KK=ISUB3(NCOUNT)
          SUM=COEFF(I,J,KK)
   *          *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
   *          -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
          KAPPA=DSQRT(DFLOAT(I)**2*K**2+DFLOAT(J)**2*L**2)
          IF((I.EQ.0).AND.(J.EQ.0)) THEN
            F=RA*ZZ/PI/KK+RA/PI**2/KK**2*DSIN(KK*PI*ZZ)
          ELSE
            A=RA/(PI**2*KK**2+KAPPA**2)
            C1=A*PI*KK/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))
            F=C1*(DEXP(KAPPA*ZZ)-DEXP(-KAPPA*ZZ))+A*DSIN(KK*PI*ZZ)
          END IF
```

```
          THETA=THETA+SUM*DSIN(PI*DFLOAT(KK)*ZZ)
          VELZ=VELZ+SUM*KAPPA**2*F
20      CONTINUE
C ---
C --- Concentration equation
        DO 30 NCOUNT=1,ICOUNT/2
          I=ISUB1(NCOUNT)
          J=ISUB2(NCOUNT)
          KK=ISUB3(NCOUNT)
             SUM=C(I,J,KK)
     *          *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
     *          -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
             CONC=CONC+SUM*DCOS(PI*(DFLOAT(KK)-0.5D0)*ZZ)
30      CONTINUE
C ---
        THETA=THETA/GAMMA
        CONC=0.21D0*(CONC+1.0D0)
        WRITE(7,*)293.0*(THETA+1.0), CONC,
     #         VELZ*2.0E-4/CHRL, X*CHRL,ZZ*CHRL
15     CONTINUE
5     CONTINUE
      RETURN
      END
C ---
C --- The following functions contain the integrands for DQDAG
C ---
      DOUBLE PRECISION FUNCTION F1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F1=DSIN(XIG*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F2(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F2=DSIN(XIH*PI*Z)*DSIN(XIG*PI*Z)*DCOS((XIF-0.5D0)*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F3(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F3=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F4(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F4=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)*DCOS((XIL-0.5D0)*
#     PI*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION F[illegible](Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      F[illegible]D0
#     [illegible]DSIN(XIF[illegible]PI*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION F[illegible](Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      F[illegible]
#     [illegible]PI[illegible](XIG-0.5D0)*PI*Z)
      RETURN
      END

      DOUBLE PRECISION FUNCTION F[illegible](Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
         [illegible]
         [illegible]PI*Z)
      RETURN

      DOUBLE PRECISION FUNCTION F[illegible](Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
         [illegible]
         [illegible]
      RETURN
      END

      DOUBLE PRECISION FUNCTION F[illegible](Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      FA[illegible]XIF*PI*Z)[illegible]PI*Z)
#         RA/PI/XIH[illegible]XIH*Z))
      RETURN

      DOUBLE PRECISION FUNCTION FA[illegible]1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      FAK21=[illegible]XIF*PI*Z)[illegible]PI*Z)*C1*KAPPA*(EXP(KAPPA*Z)+
#         EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
      RETURN
```

```
      END
C ---
      DOUBLE PRECISION FUNCTION FBKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      FBKZ0=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
#         RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FBKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      FBKZ1=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
#         C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FCKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      FCKZ0=DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)*
#         RA/PI/XIH*(1.0D0+DCOS(PI*XIH*Z))
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FCKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      FCKZ1=DCOS((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
#                                    *C1*KAPPA*(EXP(KAPPA*Z)+
#         EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FDKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      FDKZ0=PI*(XIF-0.5D0)*DSIN((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
#         *RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FDKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, XIL, C1, C2, KAPPA, RA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, DD, XLE
      FDKZ1=PI*(XIF-0.5D0)*DSIN((XIF-0.5D0)*PI*Z)*DCOS((XIG-0.5D0)*PI*Z)
```

```
#       *C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
 RETURN
 END
```

APPENDIX F

LISTING OF THE PROGRAM FOR THE INTERIOR MODEL

In this appendix a listing is given of the computer programs used to obtain the results for the interior model of Chapter 3. The programs were written in FORTRAN 77, and run on an IBM 3083 or equivalent machine. The first program (BED2NEW FORTRAN) is used to calculate steady or unsteady-state solutions to the model.

```
C*******************************************************************************
C        CALCULATE THE FOURIER COEFFICIENTS FOR THE STABILITY                  *
C        ANALYSIS OF FLOW IN AN INFINITE COAL BED                              *
C        WITH OPEN TOP AND 2 TERM EXPANSION OF THE EXPONENTIAL                 *
C        WITH NUMERICAL INTEGRATIONS                                           *
C*******************************************************************************
C --- Can use either DIVPAG for the unsteady problem or C05NBF for
C --- the steady problem
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K, L
      REAL*8 F(200), WA(70000)
      REAL*8 BN(200)
      REAL*8 W(200,9), C(24)
      REAL*8 Y(200)
      REAL*8 PARA(50), AIVPAG(1,1)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL FUNC
      EXTERNAL TDERV
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
      DATA NTMAX / 17/
      DATA BN /200*0.0D0/
C --- Call to ensure that there is sufficient workspace for DIVPAG
C --- and DQDAG
      CALL IWKIN(10070)
C --- Model parameters
        SIGMA=900.0D0
        PI=3.1415926535793D0
C --- Read truncation parameter, wave number and task
      READ(IN,*)M, A,  ITASK
C --- Read physical parameters
      READ(IN,*)DP, EPSI, EA, XKO, XKEQ, XMU
      READ(IN,*)TO, RHO  GRAV, HTCAP, DELTAH, CHRL
      READ(IN,*)IREAD
C --- Calculate dimensionless groups
        PERM=DP*DP*EPSI**3/150.0D0/(1.00D0-EPSI)**2
        RA=GRAV*3.67D-3*TO*CHRL*RHO**2*PERM*HTCAP/XKEQ/XMU
        GAMMA=EA/8.314D0/TO
        BETA=DELTAH/0.029D0/TO/HTCAP/4.76D0
        THIELE=6.0D0*XKO*RHO*HTCAP*(1.0D0-EPSI)*CHRL*CHRL
     *       /(XKEQ*DP)*DEXP(-GAMMA)
        D=6.0D0*XKO*RHO**2*CHRL**2.5*HTCAP**1.5*DEXP(-GAMMA)
     *    /XKEQ**1.5*(GRAV*TO*3.67D-3*EPSI**3/150.0D0/XMU)**0.5
        PHI=1.0D0
C --- Calulate parameters for the positive exponential approximation
        RASTAR=RA/GAMMA
        RA=RASTAR
        FK=BETA*THIELE*GAMMA
        WRITE(NOUT,*)'FK', FK
C --- calculate wave numbers
        L=SQRT(A*A/4.0D0)
        K=SQRT(A*A-L*L)
C --- Maximum and minimum values for the subscripts
        MAXIR=0
```

```
C*********************************************************************
C        CALCULATE THE FOURIER COEFFICIENTS FOR THE STABILITY        *
C        ANALYSIS OF FLOW IN AN INFINITE COAL BED                    *
C        WITH OPEN TOP AND 2 TERM EXPANSION OF THE EXPONENTIAL       *
C        WITH NUMERICAL INTEGRATIONS                                 *
C*********************************************************************
C --- Can use either DIVPAG for the unsteady problem or C05NBF for
C --- the steady problem
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K, L
      REAL*8 F(200), WA(70000)
      REAL*8 BN(200)
      REAL*8 W(200,9), C(24)
      REAL*8 Y(200)
      REAL*8 PARA(50), AIVPAG(1,1)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL FUNC
      EXTERNAL TDERV
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
      DATA NTMAX / 17/
      DATA BN /200*0.0D0/
C --- Call to ensure that there is sufficient workspace for DIVPAG
C --- and DQDAG
      CALL IWKIN(10070)
C --- Model parameters
        SIGMA=900.0D0
        PI=3.1415926535793D0
C --- Read truncation parameter, wave number and task
      READ(IN,*)M, A,  ITASK
C --- Read physical parameters
      READ(IN,*)DP, EPSI, EA, XKO, XKEQ, XMU
      READ(IN,*)TO, RHO, GRAV, HTCAP, DELTAH, CHRL
      READ(IN,*)IREAD
C --- Calculate dimensionless groups
        PERM=DP*DP*EPSI**3/150.0D0/(1.00D0-EPSI)**2
        RA=GRAV*3.67D-3*TO*CHRL*RHO**2*PERM*HTCAP/XKEQ/XMU
        GAMMA=EA/8.314D0/TO
        BETA=DELTAH/0.029D0/TO/HTCAP/4.76D0
        THIELE=6.0D0*XKO*RHO*HTCAP*(1.0D0-EPSI)*CHRL*CHRL
     *      /(XKEQ*DP)*DEXP(-GAMMA)
        D=6.0D0*XKO*RHO**2*CHRL**2.5*HTCAP**1.5*DEXP(-GAMMA)
     *   /XKEQ**1.5*(GRAV*TO*3.67D-3*EPSI**3/150.0D0/XMU)**0.5
        PHI=1.0D0
C --- Calulate parameters for the positive exponential approximation
        RASTAR=RA/GAMMA
        RA=RASTAR
        FK=BETA*THIELE*GAMMA
        WRITE(NOUT,*)'FK', FK
C --- calculate wave numbers
        L=SQRT(A*A/4.0D0)
        K=SQRT(A*A-L*L)
C --- Maximum and minimum values for the subscripts
        MAXIR=0
        MAXIS=0
        MAXIG=1
        MINIR=0
        MINIS=0
        MINIG=1
      IF(IREAD.EQ.1) THEN
C --- Read coefficients
        READ(ICOEF,*)ICOUNT
        DO 11 I=1,ICOUNT
          READ(ICOEF,*)IR, IS, IG, B(IR, IS, IG)
          Y(I)=B(IR,IS,IG)
          ISUB1(I)=IR
          ISUB2(I)=IS
          ISUB3(I)=IG
          IF(IR.GT.MAXIR) MAXIR=IR
          IF(IS.GT.MAXIS) MAXIS=IS
          IF(IG.GT.MAXIG) MAXIG=IG
          IF(IR.LT.MINIR) MINIR=IR
          IF(IS.LT.MINIS) MINIS=IS
          IF(IG.LT.MINIG) MINIG=IG
11      CONTINUE
      ELSE
C --- ICOUNT counts the number of coefficients i.e. the number of
C --- equations that satisfy the truncation and parity
      ICOUNT=0
      NCOUNT=0
      DO 10 IR=-10, 10
        DO 10 IS=-10, 10
          DO 10 IG=1,10
C --- Check parity and truncation
            CALL TRNCHK(IR, IS, IG, M, ITFLG)
             IF(ITFLG.NE.0) GO TO 10
            CALL PARCHK(IR, IS, IPRFLG)
             IF(IPRFLG.NE.0) GO TO 10
            ICOUNT=ICOUNT+1
C --- Assign N:N=3R**2+S**2
            SUBN=(3.0*DFLOAT(IR**2)+DFLOAT(IS**2))/4.0
            DO 60 I=1,ICOUNT
              IF(SUBN.EQ.BN(I)) THEN
                GO TO 70
              ELSE
                GO TO 60
              END IF
60          CONTINUE
            NCOUNT=NCOUNT+1
            BN(NCOUNT)=SUBN
70          CONTINUE
C --- Initialise B
C           IF((IABS(IS).GE.3).OR.(IABS(IR).GE.2)) THEN
            IF(IR.NE.0) THEN
              B(IR,IS,IG)=0.0D0
            ELSE
              B(IR,IS,IG)=1.0D-2
            END IF
C --- Copy B into dummy variables for NAG/IMSL, Y is a vector
            Y(ICOUNT)=B(IR,IS,IG)
            ISUB1(ICOUNT)=IR
            ISUB2(ICOUNT)=IS
            ISUB3(ICOUNT)=IG
            IF(IR.GT.MAXIR) MAXIR=IR
            IF(IS.GT.MAXIS) MAXIS=IS
            IF(IG.GT.MAXIG) MAXIG=IG
            IF(IR.LT.MINIR) MINIR=IR
            IF(IS.LT.MINIS) MINIS=IS
            IF(IG.LT.MINIG) MINIG=IG
10    CONTINUE
      END IF
      WRITE(INIT,*)ICOUNT
      DO 111 I=1,ICOUNT
        WRITE(INIT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
111   CONTINUE
      WRITE(ITERM,*)'NUMBER OF TERMS', ICOUNT
C --- Ei.her integrate ODES or solve non-linear ss equations
      IF(ITASK.EQ.0) THEN
C ---   Integrate o.d.e.'S using DIVPAG
C ---   Parameters for DIVPAG
            READ (IN,*) TINIT, TFINAL, NTSTEP
            TSTEP=(TFINAL-TINIT)/DFLOAT(NTSTEP)
            TSTART=TINIT
            TDIF=TSTEP
            TOLR=1.0D-4
C             METH=1 -> ADAMS METHOD    MITER=0 -> FUNCTIONAL ITERN.
C             METH=2 -> Stiff method
C --- Set parameters for DIVPAG to defaults
            CALL SSET(50, 0.0D0, PARA, 1)
            INORM=3
            METH=2
            MITER=0
            IDO=1
            PARA(10)=INORM
            PARA(12)=METH
            IF(NTSTEP.EQ.1) THEN
              N1=2
              N2=2
            ELSE
              N1=2
              N2=NTSTEP
            END IF
            DO 20 ITSTEP=N1, N2
C             TEND=TEND+TDIF
C --- Steps closer at beginning
C             TEND=TINIT+(TFINAL-TINIT)*(1.0D0-DCOS(PI/2.0D0*
C    #             DFLOAT(ITSTEP-1)/DFLOAT(NTSTEP-1)))
C --- Steps closer at end
              TEND=TINIT+(TFINAL-TINIT)*DSIN(PI/2.0D0*
     #             DFLOAT(ITSTEP-1)/DFLOAT(NTSTEP-1))
              CALL DIVPAG(IDO, ICOUNT, TDERV, PDERV, AIVPAG, TSTART,
     #                    TEND, TOLR, PARA, Y)
              WRITE(4,*)'TEND, IDO = ',TEND, IDO
              REWIND (UNIT=2)
              REWIND (UNIT=9)
              REWIND (UNIT=7)
              WRITE(NOUT,*)'TAU', TEND, 'FK', FK
              WRITE(NOUT,*)'TEND', TEND*CHRL**2/2.0E-4/3600.0/24.0,'DAY'
              WRITE(NOUT,*)'RA', RA*GAMMA, 'THIELE', THIELE
              WRITE(NOUT,*)'D', D
              WRITE(NOUT,*)'WAVE NUMBERS', K, L
              WRITE(ICOUT,*)ICOUNT
            DO 30 I=1,ICOUNT
              WRITE(NOUT,*)'B(R,S,G)',ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
              WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
30          CONTINUE
            CALL PLOT(Y, ICOUNT)
            CALL TMAX(Y, ICOUNT, TEND)
C ---
C --- Loops to check all Bn with the same n
          DO 80 J=1,NCOUNT
            DO 40 I=1,ICOUNT
              SUBN=(3.*DFLOAT(ISUB1(I)**2)+DFLOAT(ISUB2(I)**2))/4.0
              IF(SUBN.EQ.BN(J)) WRITE(NOUT,*)'N', BN(J), SUBN, Y(I)
40          CONTINUE
80        CONTINUE
C ---
          CONTINUE
        END IF
        IF (TASK [illegible] 1) THEN
C ---     [illegible] using [illegible] NAG
C ---     Parameters [illegible] NAG
          LABEL(1:[illegible]) [illegible]
          LWA=[illegible]
          IFAIL=1
          CALL [illegible] ([illegible], ICOUNT, Y, F, LABEL, WA, LWA, IFAIL)
          IF (IFAIL [illegible]) THEN
            WRITE(NOUT,*)'IFAIL', IFAIL
          ELSE
            WRITE([illegible]) ICOUNT
            DO [illegible] ICOUNT
              ENORM=[illegible]
              WRITE(NOUT,*)'[illegible]',ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
              WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
            CONTINUE
            WRITE(NOUT,*)'[illegible]', [illegible](ENORM)
            CALL PLOT(Y, ICOUNT)
            CALL TMAX(Y, ICOUNT)
          END IF
        END IF
            WRITE(NOUT,*)'KA', KA GAMMA, 'THIELE', THIELE
            WRITE(NOUT,*)[illegible]
            WRITE(NOUT,*)'WAVE NUMBERS', K, L
            ENORM=0
            WRITE([illegible]) ICOUNT
              DO [illegible] ICOUNT
              WRITE(NOUT,*)'B(R,S,G)',ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
              ENORM=ENORM+Y(I)**2
              WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
            CONTINUE
            WRITE(NOUT,*)DSQRT(ENORM), 'L2 NORM'
            CALL PLOT(Y, ICOUNT)
            CALL TMAX(Y, ICOUNT)
C ---
C --- Loops to check all Bn with the same n
          DO 1[illegible] J=1,NCOUNT
            DO 14[illegible] I=1,ICOUNT
              SUBN=(3.*DFLOAT(ISUB1(I)**2)+DFLOAT(ISUB2(I)**2))/4.0
              IF(SUBN.EQ.BN(J)) WRITE(NOUT,*)'N', BN(J), SUBN, Y(I)
1[illegible]        CONTINUE
1[illegible]      CONTINUE
```

```
C ---
      END IF
      STOP
      END
C ---
      SUBROUTINE PARCHK(IR, IS, IPRFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Check parity
        SUM=DFLOAT(IR+IS)
        TEST=DMOD(SUM,2.0D0)
        IF(TEST.NE.0.0D0) THEN
          IPRFLG=1
        ELSE
          IPRFLG=0
        END IF
        RETURN
      END
C ---
      SUBROUTINE TRNCHK(IR, IS, IG, M, ITFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA ITERM /3/
C --- Check truncation
        TEST=0.75D0*DFLOAT(IR**2)+DFLOAT(IS**2)*0.25D0+DFLOAT(IG*IG)
        IF(TEST.GT.(DFLOAT(M*M+1))) THEN
          ITFLG=1
        ELSE
          ITFLG=0
        END IF
        RETURN
      END
C
      SUBROUTINE TDERV(ICOUNT, X, Y, YDERV)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the time derivative of the coefficients for DIVPAG
      REAL*8 K, L, NU
      REAL*8 Y(200), YDERV(200)
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK
      COMMON /SUBS/ IR, IS, IG
      COMMON /WAVE/ K, L, M
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
      DO 20 NCOUNT=1,ICOUNT
        IR=ISUB1(NCOUNT)
        IS=ISUB2(NCOUNT)
        IG=ISUB3(NCOUNT)
            NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
            CALL SUMB(BSUM, THETA, Y, ICOUNT)
            IF((IR.EQ.0).AND.(IS.EQ.0)) THEN
              TERM=(1.0D0-(-1.0D0)**IG)/DFLOAT(IG)/PI
     *             +0.5D0*Y(NCOUNT)
     *             +THETA
            ELSE
              TERM=0.5D0*Y(NCOUNT)+THETA
            END IF
            BDERV(IR,IS,IG)=(-BSUM-(PI*PI*DFLOAT(IG*IG)+NU*NU)
     *                      *0.5D0*Y(NCOUNT)
     *           +FK*TERM)/SIGMA*2.0D0
            YDERV(NCOUNT)=BDERV(IR,IS,IG)
```

```
20    CONTINUE
      RETURN
      END
C
      SUBROUTINE FUNC (ICOUNT, Y, F, IFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the equation for coefficients, for C05NBF
      REAL*8 K, L, NU
      REAL*8 Y(200), F(200)
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK
      COMMON /SUBS/ IR, IS, IG
      COMMON /WAVE/ K, L, M
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
C     REWIND (UNIT=9)
C     WRITE(ICOUT,*)'Y IN FUNC'
C     DO 15 I=1,ICOUNT
C       WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
C15     CONTINUE
      DO 20 NCOUNT=1,ICOUNT
        IR=ISUB1(NCOUNT)
        IS=ISUB2(NCOUNT)
        IG=ISUB3(NCOUNT)
            NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
            CALL SUMB(BSUM, THETA, Y, ICOUNT)
            IF((IR.EQ.0).AND.(IS.EQ.0)) THEN
              TERM=(1.0D0-(-1.0D0)**IG)/DFLOAT(IG)/PI
     *              +0.5D0*Y(NCOUNT)
     *             +THETA
            ELSE
              TERM=0.5D0*Y(NCOUNT)+THETA
            END IF
            F(NCOUNT)=-BSUM-(PI*PI*DFLOAT(IG*IG)+NU*NU)*0.5D0*Y(NCOUNT)
     *             +FK*TERM
C           WRITE(ITERM,*)F(NCOUNT), NCOUNT
20    CONTINUE
      RETURN
      END
C
      SUBROUTINE SUMB(BSUM, THETA, Y, ICOUNT)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the multiple summation arising from non-linear term
      REAL*8 Y(200)
      REAL*8 KAPPA, K, L
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK
      COMMON /SUBS/ IR, IS, IG
      COMMON /REALSB/ XIF, XIG, XIH, C1, C2, KAPPA
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL  F1, FAKZ0, FAKZ1, FBKZ0, FBKZ1
      DATA ITERM /3/
      THETA=0.0D0
      BSUM=0.0D0
C --- Parameters for  DQDAG
```

```
C --- We split up the region of integr[illegible]ion to avoid errors with
C --- integration of whole perio[illegible]
      X1=0.0D0
      X2=0.7D0
      X3=1.0D0
      AERR=1.0D-5
      RERR=1.0D-5
C --- Quadrature rule for oscillatory functions
      IRULE=6
C --- IPQCNT and ITUCNT are counters for the pq and tu coeffs
      DO 20 IPQCNT=1,ICOUNT
        IP=ISUB1(IPQCNT)
        IQ=ISUB2(IPQCNT)
        IH=ISUB3(IPQCNT)
        DO 20 ITUCNT=1,ICOUNT
          IT=ISUB1(ITUCNT)
          IU=ISUB2(ITUCNT)
          IIF=ISUB3(I[illegible]F)
          IF((IP+IT[illegible]) GO TO 20
          IF((IQ+I[illegible] GO TO 20
C ---
C --- assign d[illegible]hat are real
                           [illegible])
                           [illegible]F)
                  A[illegible]FL[illegible])
C
                  CALL DQDAG (F1, X1, X2, AERR, RERR, IRULE, A1, ER)
                  CALL DQDAG (F1, X2, X3, AERR, RERR, IRULE, A2, ER)
                  A=A1+A2
                  IF(IER.GT.0) WRITE(ITERM,*)'IER A', IER
              THETA=THETA+Y(IPQCNT)*Y(ITUCNT)*A
C ---
                  KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
              IF((((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *        THEN
                CALL DQDAG (FAKZ0, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
                CALL DQDAG (FAKZ0, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
                AKZ=AKZ1+AKZ2
                IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ0', IER
                CALL DQDAG (FBKZ0, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
                CALL DQDAG (FBKZ0, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
                BKZ=BKZ1+BKZ2
                IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ0', IER
              ELSE
                  C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *              (PI**2*XIG**2+KAPPA**2)
                  C2=RA/(PI**2*KK**2+KAPPA**2)
                CALL DQDAG (FAKZ1, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
                CALL DQDAG (FAKZ1, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
                AKZ=AKZ1+AKZ2
                IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ1', IER
                CALL DQDAG (FBKZ1, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
                CALL DQDAG (FBKZ1, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
                BKZ=BKZ1+BKZ2
                IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ1', IER
              END IF
              BSUM=BSUM+Y(IPQCNT)*Y(ITUCNT)*
     *            (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*AKZ
     *            +KAPPA*KAPPA*BKZ)
```

```
C---
20    CONTINUE
      RETURN
      END
C
      SUBROUTINE PDERV(N,X,Y,PD)
      IMPLICIT REAL*8 (A-H,O-Z)
      RETURN
      END
C
C
      SUBROUTINE PLOT(Y, ICOUNT)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Results for plotting
      REAL*8 K, L , KAPPA
      COMMON /WAVE/ K, L, M
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      REAL*8 Y(200)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      REAL*8 COEFF(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK
      DATA ITERM /3/
      DO 10 I=1,ICOUNT
        COEFF(1SUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
10    CONTINUE
C --- Choose the height ZZ
      Z=0.0
      DO 5 X=0.0,0.4001D0, 0.4D-1
        DO 15 ZZ=0.0D0,1.0001D0, 1.00D-1
          THETA=0.0D0
          VELZ=0.0D0
          DO 20 NCOUNT=I,ICOUNT
            I=ISUB1(NCOUNT)
            J=ISUB2(NCOUNT)
            KK=ISUB3(NCOUNT)
                SUM=COEFF(I,J,KK)
     *                *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
     *                -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
                KAPPA=SQRT(DFLOAT(I)**2*K**2+DFLOAT(J)**2*L**2)
                IF((I.EQ.0).AND.(J.EQ.0)) THEN
                  F=RA*ZZ/PI/KK+RA/PI**2/KK**2*DSIN(KK*PI*ZZ)
                ELSE
                  A=RA/(PI**2*KK**2+KAPPA**2)
                  C=A*PI*KK/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))
                  F=C*(DEXP(KAPPA*ZZ)-DEXP(-KAPPA*ZZ))+A*DSIN(KK*PI*ZZ)
                END IF
                THETA=THETA+SUM*DSIN(PI*DFLOAT(KK)*ZZ)
                VELZ=VELZ+SUM*KAPPA**2*F
20        CONTINUE
          THETA=THETA/GAMMA
          WRITE(7,*)293.0*(THETA+1.0), VELZ*2.0E-4/CHRL, X*CHRL,ZZ*CHRL
15      CONTINUE
5     CONTINUE
      RETURN
      END
C ---
```

```
      SUBROUTINE TMAX(Y, ICOUNT, TEND)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Calculate maximum temperature
      REAL*8 K, L , KAPPA
      COMMON /WAVE/ K, L, M
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      REAL*8 Y(200)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      REAL*8 COEFF(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
      DATA NTMAX /17/
      DO 10 I=I,ICOUNT
        COEFF(ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
10    CONTINUE
C --- Choose the height X and Y first check at hexagon centre
      X=0.0D0
      Z=0.0D0
      THTMAX=0.0D0
30      DO 15 ZZ=0.0D0,1.0001D0, 1.00D-1
          THETA=0.0D0
          VELZ=0.0D0
          DO 20 NCOUNT=1,ICOUNT
            I=ISUB1(NCOUNT)
            J=ISUB2(NCOUNT)
            KK=ISUB3(NCOUNT)
                SUM=COEFF(I,J,KK)
     *                *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
     *                -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
                KAPPA=SQRT(DFLOAT(I)**2*K**2+DFLOAT(J)**2*L**2)
                IF((I.EQ.0).AND.(J.EQ.0)) THEN
                  F=RA*ZZ/PI/KK+RA/PI**2/KK**2*DSIN(KK*PI*ZZ)
                ELSE
                  A=RA/(PI**2*KK**2+KAPPA**2)
                  C=A*PI*KK/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))
                  F=C*(DEXP(KAPPA*ZZ)-DEXP(-KAPPA*ZZ))+A*DSIN(KK*PI*ZZ)
                END IF
                THETA=THETA+SUM*DSIN(PI*DFLOAT(KK)*ZZ)
                VELZ=VELZ+SUM*KAPPA**2*F
20        CONTINUE
          THETA=THETA/GAMMA
          IF(THETA.GT.THTMAX) THEN
            THTMAX=THETA
            HEIGHT=ZZ
            YCOORD=Z
          END IF
15      CONTINUE
C --- Now check at vertex
        IF(Z.EQ.0.0D0) THFN
          Z=PI/K*2.0D0/DSQRT(3.0D0)
          GO TO 30
        END IF
        WRITE(17,1)293.0*(THTMAX+1.0), THTMAX, YCOORD*CHRL, HEIGHT*CHRL
     #                 , TEND, TEND*CHRL**2/2.0E-4/24.0/3600.0
1     FORMAT(6F12.6)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION F1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.1415926535793D0
      F1=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FAKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FAKZ0=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      RA/PI/XIH*(1.0D0+DCOS(PI*XIH*Z))
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FAKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FAKZ1=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*C1*KAPPA*(EXP(KAPPA*Z)+
     #      EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FBKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FBKZ0=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FBKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSB/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FBKZ1=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
      RETURN
      END
```

The following program (IGNNEW FORTRAN) is used to calculate directly ignition points in the layer.

```
*********************  ******************************************* ****
      CALCULATE THE FRANK-KAMENETSKII NUMBER FOR IGNITION          *
      IN AN INFINITE COAL BED                                      *
      WITH OPEN TOP AND 2 TERM EXPANSION OF THE EXPONENTIAL        *
      AND NUMERICAL INTEGRATIONS
  ******************************************************************
    es C05NBF for the solution of the non-linear problem
    culates the determinant of the Jacobian from the product
    f the eigenvalues obtained from F02AFF
   IMPLICIT REAL*8 (A-H,O-Z)
   REAL*8 K, L
   REAL*8 BN(200)
   REAL*8 F(200), WA(70000)
   REAL*8 Y(200)
    COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
    COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
    COMMON /WAVE/ K, L, M
    COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
    COMMON /TRC/ ITRC(-20:41,-20:41,20)
    COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
   EXTERNAL FUNC
   EXTERNAL TDERV
   DATA NIN /1/, NOUT /[illegible]/, ITERM /4/, ICOEF /8/, ICOUT /9/, INIT /16/
    Set parameters
        PI=3.14159265358979
      TRUNCATION PARAMETER, WAVE NUMBER
     READ(NIN,*) M, A
    Physical parameters
     READ(NIN,*) DP, EPSI, EA, XKO, XKEQ, XMU
     READ(NIN,*) TO, RHO, GRAV, HTCAP, DELTAH, CHRL
     READ(NIN,*) IREAD
     PERM=DP*DP*EPSI**3/150.0D0/(1.0D0-EPSI)**2
     RA=GRAV*3.67D-3*TO*CHRL*RHO**2*PERM*HTCAP/XKEQ/XMU
     GAMMA=EA/8.314D0/TO
    BETA=DELTAH*0.02[illegible]*TO/HTCAP/4.76D0
    D=[illegible]D0*XKO*RHO**2*CHRL**2.5*HTCAP**1.5*DEXP(-GAMMA)
       XKEQ**1.5*(GRAV*TO*3.67D-3*EPSI**3/150.0D0/XMU))**0.5
     RASTAR=RA/GAMMA
     RA=RASTAR
    Calculate wave numbers
     L=DSQRT(A*A/4.0D0)
     K=DSQRT(A*A-L*L)
   Maximum and minimum values for the subscripts
    MAXIR=0
    MAXIS=0
    MAXIG=1
    MINIR=0
    MINIS=0
    MINIG=1
   IF(IREAD.EQ.1) THEN
    READ(ICOEF,*)ICOUNT
    NEQN=ICOUNT+1
    DO 11 I=1,ICOUNT
      READ(ICOEF,*)IR, IS, IG, B(IR, IS, IG)
      Y(I)=B(IR,IS,IG)
      ISUB1(I)=IR
      ISUB2(I)=IS
      ISUB3(I)=IG
      IF(IR.GT.MAXIR) MAXIR=IR
        IF(IS.GT.MAXIS) MAXIS=IS
        IF(IG.GT.MAXIG) MAXIG=IG
        IF(IR.LT.MINIR) MINIR=IR
        IF(IS.LT.MINIS) MINIS=IS
        IF(IG.LT.MINIG) MINIG=IG
 11     CONTINUE
        READ(ICOEF,*)FK
        Y(NEQN)=FK
      ELSE
C --- ICOUNT counts the number of coefficients i.e. the number of
C --- equations that satisfy the truncation and parity
      ICOUNT=0
      DO 10 IR=-10, 10
        DO 10 IS=-10, 10
          DO 10 IG=1,10
C --- Check parity and truncation
            CALL TRNCHK(IR, IS, IG, M, ITFLG)
             IF(ITFLG.NE.0) GO TO 10
            CALL PARCHK(IR, IS, IPRFLG)
             IF(IPRFLG.NE.0) GO TO 10
              ICOUNT=ICOUNT+1
C --- Initialise B
            IF(IR.NE.0) THEN
              B(IR,IS,IG)=0.0D0
            ELSE
              B(IR,IS,IG)=1.0D-1
            END IF
C --- Copy B into dummy variables Y is a vector
            Y(ICOUNT)=B(IR,IS,IG)
            ISUB1(ICOUNT)=IR
            ISUB2(ICOUNT)=IS
            ISUB3(ICOUNT)=IG
            IF(IR.GT.MAXIR) MAXIR=IR
            IF(IS.GT.MAXIS) MAXIS=IS
            IF(IG.GT.MAXIG) MAXIG=IG
            IF(IR.LT.MINIR) MINIR=IR
            IF(IS.LT.MIN'S) MINIS=IS
            IF(IG.LT.MINIG) MINIG=IG
 10     CONTINUE
        NEQN=ICOUNT+1
C --- Initial estimate for FK
        Y(NEQN)=5.0D0
        END IF
        WRITE(INIT,*)ICOUNT
        DO 111 I=1,ICOUNT
          WRITE(INIT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
 111    CONTINUE
        WRITE(INIT,*)Y(NEQN)
        WRITE(ITERM,*)'NUMBER OF TERMS', ICOUNT
C ---     Solve eqs using C05NBF
C ---     Parameters for C05NBF
          ERRREL=DSQRT(X02AAF(0.0D0))
          LWA=70000
          IFAIL=1
          CALL FUNC(NEQN, Y, F, IFLAG)
C         CALL C05NBF (FUNC, NEQN, Y, F, ERRREL, WA, LWA, IFAIL)
          IF (IFAIL.GT.0) THEN
            WRITE(NOUT,*)'IFAIL', IFAIL
            FNORM=0.0D0
```

```
C *******************************************************************
C     CALCULATE THE FRANK-KAMENETSKII NUMBER FOR IGNITION           *
C     IN AN INFINITE COAL BED                                       *
C     WITH OPEN TOP AND 2 TERM EXPANSION OF THE EXPONENTIAL         *
C     AND NUMERICAL INTEGRATIONS
C *******************************************************************
C Uses C05NBF for the solution of the non-linear problem
C Calculates the determinant of the Jacobian from the product
C of the eigenvalues obtained from F02AFF
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K, L
      REAL*8 BN(200)
      REAL*8 F(200), WA(70000)
      REAL*8 Y(200)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
      COMMON /TRC/ ITRC(-20:41,-20:41,20)
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      EXTERNAL FUNC
      EXTERNAL TDERV
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /10/
C Set parameters
        PI=3.14159265358979D0
C Truncation parameter, wave number
      READ(IN,*)M, A
C Physical parameters
      READ(IN,*)DP, EPSI, EA, XKO, XKEQ, XMU
      READ(IN,*)TO, RHO, GRAV, HTCAP, DELTAH, CHRL
      READ(IN,*)IREAD
      PERM=DP*DP*EPSI**3/150.0D0/(1.0D0-EPSI)**2
      RA=GRAV*3.67D-3*TO*CHRL*RHO**2*PERM*HTCAP/XKEQ/XMU
      GAMMA=EA/8.314D0/TO
      BETA=DELTAH/0.029D0/TO/HTCAP/4.76D0
      D=[illegible]D0*XKO*RHO**2*CHRL**2.5*HTCAP**1.5*DEXP(-GAMMA)
     *  /XKEQ**1.5*(GRAV*TO*3.67D-3*EPSI**3/150.0D0/XMU)**0.5
      RASTAR=RA/GAMMA
      RA=RASTAR
C Calculate wave numbers
      L=DSQRT(A*A/4.0D0)
      K=DSQRT(A*A-L*L)
C Maximum and minimum values for the subscripts
      MAXIR=0
      MAXIS=0
      MAXIG=1
      MINIR=0
      MINIS=0
      MINIG=1
      IF(IREAD.EQ.1) THEN
      READ(ICOEF,*)ICOUNT
      NEQN=ICOUNT+1
      DO 11 I=1,ICOUNT
        READ(ICOEF,*)IR, IS, IG, B(IR, IS, IG)
        Y(I)=B(IR,IS,IG)
        ISUB1(I)=IR
        ISUB2(I)=IS
        ISUB3(I)=IG
        IF(IR.GT.MAXIR) MAXIR=IR
        IF(IS.GT.MAXIS) MAXIS=IS
        IF(IG.GT.MAXIG) MAXIG=IG
        IF(IR.LT.MINIR) MINIR=IR
        IF(IS.LT.MINIS) MINIS=IS
        IF(IG.LT.MINIG) MINIG=IG
11      CONTINUE
        READ(ICOEF,*)FK
        Y(NEQN)=FK
      ELSE
C --- ICOUNT counts the number of coefficients i.e. the number of
C --- equations that satisfy the truncation and parity
      ICOUNT=0
      DO 10 IR=-10, 10
        DO 10 IS=-10, 10
          DO 10 IG=1,10
C --- Check parity and truncation
            CALL TRNCHK(IR, IS, IG, M, ITFLG)
              IF(ITFLG.NE.0) GO TO 10
            CALL PARCHK(IR, IS, IPRFLG)
              IF(IPRFLG.NE.0) GO TO 10
              ICOUNT=ICOUNT+1
C --- Initialise B
            IF(IR.NE.0) THEN
              B(IR,IS,IG)=0.0D0
            ELSE
              B(IR,IS,IG)=1.0D-1
            END IF
C --- Copy B into dummy variables Y is a vector
            Y(ICOUNT)=B(IR,IS,IG)
            ISUB1(ICOUNT)=IR
            ISUB2(ICOUNT)=IS
            ISUB3(ICOUNT)=IG
            IF(IR.GT.MAXIR) MAXIR=IR
            IF(IS.GT.MAXIS) MAXIS=IS
            IF(IG.GT.MAXIG) MAXIG=IG
            IF(IR.LT.MINIR) MINIR=IR
            IF(IS.LT.MINIS) MINIS=IS
            IF(IG.LT.MINIG) MINIG=IG
10    CONTINUE
      NEQN=ICOUNT+1
C --- Initial estimate for FK
      Y(NEQN)=5.0D0
      END IF
      WRITE(INIT,*)ICOUNT
      DO 111 I=1,ICOUNT
        WRITE(INIT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
111   CONTINUE
      WRITE(INIT,*)Y(NEQN)
      WRITE(ITERM,*)'NUMBER OF TERMS', ICOUNT
C ---   Solve eqs using C05NBF
C ---   Parameters for C05NBF
        ERRREL=DSQRT(X02AAF(0.0D0))
        LWA=70000
        IFAIL=1
        CALL FUNC(NEQN, Y, F, IFLAG)
C       CALL C05NBF (FUNC, NEQN, Y, F, ERRREL, WA, LWA, IFAIL)
        IF (IFAIL.GT.0) THEN
          WRITE(NOUT,*)'IFAIL', IFAIL
          FNORM=0.0D0
      WRITE(ICOUT,*)ICOUNT
      DO 430 I=1,ICOUNT
        FNORM=FNORM+F(I)**2
        WRITE(NOUT,*)'B(R,S,G)' ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
        WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
      CONTINUE
      WRITE(NOUT,*)'L2 NORM', DSQRT(FNORM)
      CALL PLOT(Y, ICOUNT)
      STOP
    END IF
      FK=Y(NEQN)
      WRITE(NOUT,*)'FK*', FK, ,'FK', FK/DSQRT(RA), 'RA*', RA
      WRITE(NOUT,*)'D', D
      WRITE(NOUT,*)'WAVE NUMBERS', K, L
      FNORM=0.0D0
      WRITE(ICOUT,*)ICOUNT
      DO [illegible] I=1,ICOUNT
        FNORM=FNORM+F(I)**2
        WRITE(NOUT,*)'B(R,S,G)',ISUB1(I),ISUB2(I),ISUB3(I),Y(I)
        WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
        CONTINUE
      WRITE(NOUT,*)'L2 NORM', DSQRT(FNORM)
       CALL PLOT(Y, ICOUNT)


      SUBROUTINE PARCHK(IR, IS, IPFLAG)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Parity
        SUM=DFLOAT(IR+IS)
       TEST=DMOD(SUM,2.0D0)
       IF(TEST.NE.0.0D0) THEN
         IPFLAG=1
       ELSE
         IPFLAG=0
       END IF
       RETURN
      END

      SUBROUTINE TRNCHK(IR, IS, IG, M, ITFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA ITERM /3/
C --- Check truncation
        TEST=0.75D0*DFLOAT(IR**2)+DFLOAT(IS**2)*0.25D0+DFLOAT(IG*IG)
        IF((TEST.GT.(DFLOAT(M*M+1))) THEN
          ITFLG=1
        ELSE
          ITFLG=0
        END IF
        RETURN
      END

      SUBROUTINE FUNC (NEQN, Y, F, IFLAG)
      IMPLICIT REAL*8 (A-H,O-Z)
C - - Evaluates the equations for coefficients, for C05NBF
    - Evaluates the equation for  det(Jacobian)=0
    - Y(NEQN)=FK
      PARAMETER (NVAR=3, LDEVEC=NVAR)
      REAL*8 K, L, NU
      REAL*8 Y(200), F(200), AA(NVAR,NVAR), RR(NVAR), RI(NVAR)
      REAL*8 ACOPY(NVAR,NVAR)
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
      COMMON /SUBS/ IR, IS, IG
      COMMON /WAVE/ K, L, M
      COMMON /TRC/ ITRC(-20:41,-20:41,20)
      COMMON /I[illegible]/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /W[illegible]P/ RWKSP
      COMPLEX*8 EVAL(NVAR), EVEC(LDEVEC, NVAR)
      INTEGER INTGER(NVAR)
      REAL*8 RWKSP(10070)
      DATA IN /1/, NOUT /2/, ITERM/4/, ICOEF /8/, ICOUT /9/, INIT /16/
      ICOUNT=NEQN-1
      REWIND (UNIT=9)
      WRITE(ICOUT,*)'Y IN FUNC'
      DO 55 I=1,ICOUNT
        WRITE(ICOUT,*)ISUB1(I), ISUB2(I), ISUB3(I), Y(I)
55    CONTINUE
C --- The correct element of Y is assigned on the same basis
C --- as in the main program i.e. using the counter Ncount which is
C --- incremented for every admissible rs pair
      DO 20 NCOUNT=1,ICOUNT
        IR=ISUB1(NCOUNT)
        IS=ISUB2(NCOUNT)
        IG=ISUB3(NCOUNT)
            NU=SQRT((DFLOAT(IR)*K)**2+(DFLOAT(IS)*L)**2)
            CALL SUMB(BSUM, THETA, Y, NEQN)
            IF((IR.EQ.0).AND.(IS.EQ.0)) THEN
              TERM=(1.0D0-(-1.0D0)**IG)/DFLOAT(IG)/PI
     .              +0.5D0*Y(NCOUNT)
     *             +THETA
            ELSE
              TERM=0.5D0*Y(NCOUNT)+THETA
            END IF
            F(NCOUNT)=-BSUM-(PI*PI*DFLOAT(IG*IG)+NU*NU)*0.5D0*Y(NCOUNT)
     *             +Y(NEQN)/DSQRT(RA)*TERM
C           WRITE(ITERM,*)F(NCOUNT)
20    CONTINUE
C --- Start loop to assemble Jacobian matrix
      DO 15 J=1,ICOUNT
        IR=ISUB1(J)
        IS=ISUB2(J)
        IG=ISUB3(J)
        DO 10 I=1,ICOUNT
          CALL SUM(SUM1, SUM2, Y, I, NEQN)
          IF(I.EQ.J) THEN
            AA(J,I)=-0.5D0*(
     #              (DFLOAT(ISUB1(I))*K)**2+(DFLOAT(ISUB2(I))*L)**2
     #              +PI**2*DFLOAT(ISUB3(I)**2))
     #              -SUM1+Y(NEQN)/DSQRT(RA)*0.5D0*(1.0D0+SUM2)
          ELSE
            AA(J,I)=-SUM1+0.5D0*Y(NEQN)/DSQRT(RA)*SUM2
          END IF
          WRITE(NOUT,*)AA(J,I)
          ACOPY(J,I)=AA(J,1)
10      CONTINUE
```

```
15    CONTINUE
C --- Call FO2AFF to find the eigenvalues of AA
C --- Parameters for FO2AFF
      N=ICOUNT
      IA=ICOUNT
      IFAIL=1
      CALL FO2AFF(AA, IA, N, RR, RI, INTGER, IFAIL)
      CALL DEVCRG(N, ACOPY, IA, EVAL, EVEC, IDEVEC)
      CALL WRCRN ('EIGENVALUES', 1,N, EVAL, 1, 0)
      IF(IFAIL.GT.0) THEN
        WRITE(NOUT,*)'IFAIL FO2', IFAIL
        STOP
      END IF
      WRITE(NOUT,*)'EIGENVALUES'
      DO 50 I=1,ICOUNT
       WRITE(NOUT,*)RR(I), RI(I)
       WRITE(NOUT,*)EVAL(I)
      CONTINUE
      F(NEQN)=1.0D0
      DO 70 I=1,ICOUNT
        F(NEQN)=RR(I)*F(NEQN)
        F(NEQN)=EVAL(I)*F(NEQN)
      CONTINUE
      RETURN
      END


      SUBROUTINE SUMB(BSUM, THETA, Y, NEQN)
       IMPLICIT REAL*8 (A-H,O-Z)
      evaluate the multiple summation arising from non-linear term
      DIMENSION Y(200)
      REAL*8 KAPPA, K, L
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
      COMMON /SUBS/ IR, IS, IG
      COMMON /REALSUBS/ XIF, XIO, XIH, C1, C2, KAPPA
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /WORKSP/  RWKSP
      REAL*8 RWKSP(10070)
      EXTERNAL  F1, FAKZ0, FAKZ1, FBKZ0, FBKZ1
      DATA ITERM /3/
      THETA=0.0D0
      BSUM=0.0D0
  --- Parameters for DQDAG
  --- We split up the region of integration to avoid errors with
  --- integration of whole periods
      X1=0.0D0
      X2=0.7D0
      X3=1.0D0
      AERR=1.0D-6
      RERR=1.0D-6
  --- Quadrature rule for oscillatory functions
      IRULE=6
      ICOUNT=NEQN-1
 ---- IPQCNT and ITUCNT are counters for the pq and tu coeffs
      DO 10 IPQCNT=1,ICOUNT
        IP=ISUB1(IPQCNT)
        IQ=ISUB2(IPQCNT)
        IH=ISUB3(IPQCNT)
        DO 10 ITUCNT=1,ICOUNT
          IT=ISUB1(ITUCNT)
          IU=ISUB2(ITUCNT)
          IIF=ISUB3(ITUCNT)
            IF((IP+IT).NE.IR) GO TO 10
            IF((IQ+IU).NE.IS) GO TO 10
C ---
C --- assign dummy subscripts that are real
                  XIG=DFLOAT(IG)
                  XIF=DFLOAT(IIF)
                  XIH=DFLOAT(IH)
C
                  CALL DQDAG(F1, X1, X2, AERR, RERR, IRULE, A1, ER)
                  CALL DQDAG(F1, X2, X3, AERR, RERR, IRULE, A2, ER)
                  A=A1+A2
                  IF(IER.GT.0) WRITE(ITERM,*)'IER A', IER
              THETA=THETA+Y(IPQCNT)*Y(ITUCNT)*A
C ---
                  KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
              IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *        THEN
                CALL DQDAG(FAKZ0, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
                CALL DQDAG(FAKZ0, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
                AKZ=AKZ1+AKZ2
                IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ0', IER
                CALL DQDAG(FBKZ0, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
                CALL DQDAG(FBKZ0, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
                BKZ=BKZ1+BKZ2
                IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ0', IER
              ELSE
                 C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *              (PI**2*XIG**2+KAPPA**2)
                 C2=RA/(PI**2*KK**2+KAPPA**2)
                CALL DQDAG(FAKZ1, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
                CALL DQDAG(FAKZ1, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
                AKZ=AKZ1+AKZ2
                IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ1', IER
                CALL DODAG(FBKZ1, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
                CALL DQDAG(FBKZ1, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
                BKZ=BKZ1+BKZ2
                IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ1', IER
              END IF
              BSUM=BSUM+Y(IPQCNT)*Y(ITUCNT)*
     *             (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*AKZ
     *             +KAPPA*KAPPA*BKZ)
C---
10    CONTINUE
      RETURN
      END
C
      SUBROUTINE SUM(SUM1, SUM2, Y, IJAC, NEQN)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Evaluate the multiple summation arising from non-linear term
      REAL*8 Y(200)
      REAL*8 KAPPA, K, L
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      COMMON /WAVE/ K, L, M
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
      COMMON /SUBS/ IR, IS, IG
      COMMON /TRC/ ITR(-20:41,-20:41,20)
      COMMON /REALSUB/ XIF, XIG, XIH, C1, C2, KAPPA
      COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      COMMON /WORKSP/ RWKSP
      REAL*8 RWKSP([illegible])
      EXTERNAL F1, FAKZ0, FAKZ1, FBKZ0, FBKZ1
      DATA ITERM /3/
      [illegible]
      [illegible]
C Parameters for DQDAG
C We split up the region of integration to avoid errors with
C integration of whole periods
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
C Quadrature rule for oscillatory functions
      IRULE=6
      [illegible]
C [illegible] and IT[illegible]NT are [illegible] for the pq and tu coeffs
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
C [illegible] dummy subscripts that are real
          XIG=DFLOAT(IG)
          XIF=DFLOAT(IF)
          XIH=DFLOAT(IH)

          CALL DQDAG(F1, X1, X2, AERR, RERR, IRULE, A1, ER)
          CALL DQDAG(F1, X2, X3, AERR, RERR, IRULE, A2, ER)
          A=A1+A2
          IF(IER.GT.0) WRITE(ITERM,*)'IER A', IER
        IF((IPQCNT.EQ.IJAC).AND.(ITUCNT.NE.IJAC)) THEN
          TERM=Y(ITUCNT)
        ELSE IF((ITUCNT.EQ.IJAC).AND.(IPQCNT.NE.IJAC)) THEN
          TERM=Y(IPQCNT)
        ELSE IF((IPQCNT.EQ.ICOUNT).AND.(ITUCNT.EQ.ICOUNT)) THEN
          TERM=2.0D0*Y(ICOUNT)
        ELSE
          TERM=0.0D0
        END IF
        SUM2=SUM2+TERM*A

          KAPPA=SQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
        IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
     *  THEN
          CALL DQDAG(FAKZ0, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
          CALL DQDAG(FAKZ0, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
          AKZ=AKZ1+AKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ0', IER
          CALL DQDAG(FBKZ0, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
          CALL DQDAG(FBKZ0, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
          BKZ=BKZ1+BKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ0', IER
        ELSE
            C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *          (PI**2*XIG**2+KAPPA**2)
            C2=RA/(PI**2*KK**2+KAPPA**2)
          CALL DQDAG(FAKZ1, X1, X2, AERR, RERR, IRULE, AKZ1, ER)
          CALL DQDAO(FAKZ1, X2, X3, AERR, RERR, IRULE, AKZ2, ER)
          AKZ=AKZ1+AKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ1', IER
          CALL DQDAG(FBKZ1, X1, X2, AERR, RERR, IRULE, BKZ1, ER)
          CALL DQDAG(FBKZ1, X2, X3, AERR, RERR, IRULE, BKZ2, ER)
          BKZ=BKZ1+BKZ2
          IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ1', IER
        END IF
        IF((IPQCNT.EQ.IJAC).AND.(ITUCNT.NE.IJAC)) THEN
          TERM=Y(ITUCNT)
        ELSE IF((ITUCNT.EQ.IJAC).AND.(IPQCNT.NE.IJAC)) THEN
          TERM=Y(IPQCNT)
        ELSE IF((IPQCNT.LQ.ICOUNT).AND.(ITUCNT.EQ.ICOUNT)) THEN
          TERM=2.0D0*Y(ICOUNT)
        ELSE
          TERM=0.0D0
        END IF
        BSUM=BSUM+TERM*
     *       (-(DFLOAT(IP*IT)*K*K+DFLOAT(IQ*IU)*L*L)*AKZ
     *       +KAPPA*KAPPA*BKZ)
C---
10    CONTINUE
      RETURN
      END
C
      SUBROUTINE PLOT(Y, ICOUNT)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Results for plotting
      REAL*8 K, L , KAPPA
      COMMON /WAVE/ K, L, M
      COMMON /COEFF/ BDERV(-20:41,-20:41,20), B(-20:41,-20:41,20)
      DIMENSION Y(200)
      COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
      REAL*8 COEFF(-20:41,-20:41,20)
      COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
      DATA ITERM /3/
      DO 10 I=1,ICOUNT
        COEFF(ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
C       COEFF(-ISUB1(I),-ISUB2(I),ISUB3(I))=Y(I)
10    CONTINUE
C --- Choose the height ZZ
      ZZ=0.5
      DO 5 X=0.0,0.4001D0, 0.4D-1
        DO 15 Z=0.0D0,0.40001D0, 0.40D-1
          THETA=0.0D0
      VELZ=0.0D0
      DO 20 NCOUNT=1,ICOUNT
         I=ISUB1(NCOUNT)
         J=ISUB2(NCOUNT)
         KK=ISUB3(NCOUNT)
            SUM=COEFF(I,J,KK)
     *            *(DCOS(DFLOAT(I)*K*X)*DCOS(DFLOAT(J)*L*Z)
     *            -DSIN(DFLOAT(I)*K*X)*DSIN(DFLOAT(J)*L*Z))
            KAPPA=SQRT(DFLOAT(I)**2*K**2+DFLOAT(J)**2*L**2)
            IF((I.EQ.0).AND.(J.EQ.0)) THEN
              F=RA*ZZ/PI/KK+RA/PI**2/KK**2*DSIN(KK*PI*ZZ)
            ELSE
              A=RA/(PI**2*KK**2+KAPPA**2)
              C=A*PI*KK/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))
              F=C*(DEXP(KAPPA*ZZ)-DEXP(-KAPPA*ZZ))+A*DSIN(KK*PI*ZZ)
            END IF
            THETA=THETA+SUM*DSIN(PI*DFLOAT(KK)*ZZ)
            VELZ=VELZ+SUM*KAPPA**2*F
      CONTINUE
      THETA=THETA/GAMMA
      WRITE(7,*)293.0*(THETA+1.0), VELZ*2.0E-4/CHRL, X*CHRL, Z*CHRL
    CONTINUE
   CONTINUE
   RETURN
   END
C ---
   DOUBLE PRECISION FUNCTION F1(Z)
   IMPLICIT REAL*8 (A-H,O-Z)
   COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
   REAL*8 KAPPA
   PI=3.14159265358979D0
   F1=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)
   RETURN
   END

   DOUBLE PRECISION FUNCTION FAKZ0(Z)
   IMPLICIT REAL*8 (A-H,O-Z)
   COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
   REAL*8 KAPPA
   COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
   FAKZ0=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*
  #      RA/PI/XIH*(1.0D0+DCOS(PI*XIH*Z))
   RETURN
   END

   DOUBLE PRECISION FUNCTION FAKZ1(Z)
   IMPLICIT REAL*8 (A-H,O-Z)
   COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
   REAL*8 KAPPA
   COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
   FAKZ1=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*C1*KAPPA*(EXP(KAPPA*Z)+
  #      EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
   RETURN

C ---
   DOUBLE PRECISION FUNCTION FBKZ0(Z)
   IMPLICIT REAL*8 (A-H,O-Z)
   COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
   REAL*8 KAPPA
   COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
   FBKZ0=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
  #      RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
   RETURN
   END
C ---
   DOUBLE PRECISION FUNCTION FBKZ1(Z)
   IMPLICIT REAL*8 (A-H,O-Z)
   COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
   REAL*8 KAPPA
   COMMON /PARAMS/ BETA, GAMMA, PI, CHRL, D, RA
   FBKZ1=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
  #      C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
   RETURN
   END
```

The following program (STABNEW FORTRAN) is used to calculate the eigenvalues for the stability analysis of the of the steady-state solutions supplied by BED2NEW FORTRAN.

```
C **************************************************************
C     CALCULATE THE [illegible]ENVALUES FOR THE PERTURBED STEADY FLOW
C     IN AN INFINIT[illegible]AL BED
C     WITH QUADRAT[illegible] EXPANSION OF THE TEMPERATURE EXPONENTIAL
C     AND NUMERICA[illegible] NTEGRATIONS
C **************************************************************
C The steady state coefficients are read, having been calculated
C by program BED2
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K, L
      REAL*8 B(-20:+1,-20:+1,20)
      REAL*8 AA(20,20)
      REAL*8 WK(10000)
      REAL*8 RW(52), RZ([illegible])
      COMPLEX*8 W(26), Z(26,26)
      EQUIVALENCE (W(1),RW(1)), (Z(1,1), RZ(1))
        COMMON /INDEX/ IR, IS, IG
        COMMON /SUBS/ ISUB1(200), ISUB2(200), ISUB3(200)
        COMMON /COEFF/ Y(200)
        COMMON /TRACE/ ITRC(-20:+1,-20:+1,20)
        COMMON /WAVE/ K, L, M
        COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, KA, PHI, THEL
        COMMON /MAXMIN/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
       [illegible]
          NEIG [illegible]
C [illegible]
  [illegible]
  [illegible] wave number [illegible]
    READ(IN,*)M, A, [illegible]
C [illegible] parameters
    READ(IN,*) [illegible]
    READ(IN,*) [illegible], BETAH, THEL
    READ(IN,*) [illegible]
      [illegible]
      [illegible] THEL [illegible]
      [illegible]
      BETA [illegible] HICAP [illegible]
      THIELE [illegible] HICAP [illegible] THEL
              [illegible] DEXP(-GAMMA)
      [illegible]
C Calculate parameters for the positive exponential approximation
      [illegible]KA GAMMA
      [illegible]
      FK BETA THIELE GAMMA
C Calculate wave numbers
      [illegible] A*A+[illegible]
      [illegible] A*A+L[illegible]
C [illegible] counts the number of [illegible], i.e. the number of
C [illegible] that satisfy the [illegible] and parity
      READ [illegible] ICOUNT
C Read the Fourier coefficients and the subscripts and assign trace
C Maximum and minimum values for the subscripts
      MAXIR=0
      MAXIS=0
      MAXIG=1
      MINIR=0
      MINIS=0
      MINIG=1
      DO 11 I=1,ICOUNT
         READ(ICOUT,*)IR, IS, IG, B(IR, IS, IG)
         Y(I)=B(IR,IS,IG)
         ISUB1(I)=IR
         ISUB2(I)=IS
         ISUB3(I)=IG
         IF(IR.GT.MAXIR) MAXIR=IR
         IF(IS.GT.MAXIS) MAXIS=IS
         IF(IG.GT MAXIG) MAXIG=IG
         IF(IR.LT.MINIR) MINIR=IR
         IF(IS.LT.MINIS) MINIS=IS
         IF(IG.LT.MINIG) MINIG=IG
         ITRC(IR,IS,IG)=I
11    CONTINUE
C --- Start loop to assemble matrix
      DO 15 J=1,ICOUNT
         IR=ISUB1(J)
         IS=ISUB2(J)
         IG=ISUB3(J)
         DO 10 I=1,ICOUNT
           CALL SUMB(BSUM, THETA, I, ICOUNT)
           IF(I.EQ J) THEN
             AA(J,I[illegible]2.0D0/PHI*(-0.5D0*((ISUB1(I)*K)**2
     *                 +(ISUB2(I)*L)**2
     *                 +FK*(0.5D0+THET[illegible]+2.0D0*BSUM)
           ELSE
             AA(J,I)=2.0D0/PHI*(FK*THETA+2.0D0*BSUM)
           END IF
           WRITE(NOUT,*)AA(J,I), J, I
10       CONTINUE
15    CONTINUE
C --- Call EIGRF to find the eigenvalues of AA
C --- Parameters for EIGRF
      N=ICOUNT
      IA=ICOUNT
      IJOB=2
      IZ=ICOUNT
      CALL EIGRF(AA, N, IA, IJOB, RW, RZ, IZ, WK, IER)
      IF(IER.GT.0) THEN
        WRITE(ITERM,*)'IER', IER
        WRITE(NOUT,*)'IER', IER
        STOP
      END IF
      WRITE(NEIG,*)'PERFORMANCE INDEX'
      WRITE(NEIG,*)WK(1)
      WRITE(NEIG,*)'EIGENVALUES'
      DO 50 I=1,2*ICOUNT
       WRITE(NEIG,*)W(I)
50    CONTINUE
       WRITE(NEIG,*)'*****************'
      WRITE(NEIG,*)RW
      DO 60 I=1,2*ICOUNT,2
       IF(RW(I).GT.0.0D0) TH
         WRITE(3,*)'*****************************************************'
         WRITE(3,*)'*********************UNSTABLE************************'
         WRITE(3,*)'*****************************************************'
         STOP
       END IF
60    CONTINUE
      STOP
```

```
END

SUBROUTINE SUMB([illegible] M, THETA, INDEX, ICOUNT)
IMPLICIT REAL*8 (A-H,O-Z)
Evaluate the multiple summation arising from non-linear term
REAL*8 KAPPA, K, L
COMMON /COEFF/ Y(200)
COMMON /WAVE/ K, L, M
COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, PHI, CHRL
COMMON /SUBS/ IR, IS, IG
COMMON /TRC/ ITRC(-20:41,-20:41,20)
COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
COMMON /MAXSUB/ MAXIR, MAXIS, MAXIG, MINIR, MINIS, MINIG
COMMON /ISUB/ ISUB1(200), ISUB2(200), ISUB3(200)
EXTERNAL F1, FAKZ0, FAKZ1, FBKZ0, FBKZ1
DATA ITERM /3/
[illegible]=0.D0
THETA=0.0D0
Parameters for DCADRE
We split up the region of integration to avoid errors with
integration of whole periods
X1=[illegible]
X2=[illegible]
X3=[illegible]
AERR=[illegible]D-6
RERR=1.D-6
IPQCNT and ITUCNT are counters for the pq and tu coeffs
      DO 20 IPQCNT=1,ICOUNT
     IP=ISUB1(IPQCNT)
    IQ=ISUB2(IPQCNT)
   [illegible]=ISUB3(IPQCNT)
      DO 20 ITUCNT=1,ICOUNT
     IT=ISUB1(ITUCNT)
     IU=ISUB2(ITUCNT)
     IIF=ISUB3(ITUCNT)
        IF((IP+IT).NE.IR) GO TO 20
        IF((IQ+IU).NE.IS) GO TO 20

Assign array subscripts that are real
            XIG=FLOAT(IG)
            XIF=FLOAT(IIF)
            XIH=FLOAT(IH)

            A1=DCADRE(F1, X1, X2, AERR, RERR, ERROR, IER)
            A2=DCADRE(F1, X2, X3, AERR, RERR, ERROR, IER)
            A=A1+A2
            IF(IER.GT.0) WRITE(ITERM,*)'IER A', IER
            IF(IPQCNT.EQ.INDEX) THEN
              THETA=Y(ITUCNT)*A+THETA
            END IF

            KAPPA=DSQRT((DFLOAT(IP)*K)**2+(DFLOAT(IQ)*L)**2)
         IF(((IP.EQ.0).AND.(IQ.EQ.0)).OR.((IT.EQ.0).AND.(IU.EQ.0)))
         THEN
           AKZ1=DCADRE(FAKZ0, X1, X2, AERR, RERR, ERROR, IER)
           AKZ2=DCADRE(FAKZ0, X2, X3, AERR, RERR, ERROR, IER)
           AKZ=AKZ1+AKZ2
           IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ0', IER
           BKZ1=DCADRE(FBKZ0, X1, X2, AERR, RERR, ERROR, IER)
           BKZ2=DCADRE(FBKZ0, X2, X3, AERR, RERR, ERROR, IER)
           BKZ=BKZ1+BKZ2
           IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ0', IER
         ELSE
           C1=RA*PI*XIG/KAPPA/(DEXP(KAPPA)+DEXP(-KAPPA))/
     *        (PI**2*XIG**2+KAPPA**2)
           C2=RA/(PI**2*KK**2+KAPPA**2)
           AKZ1=DCADRE(FAKZ1, X1, X2, AERR, RERR, ERROR, IER)
           AKZ2=DCADRE(FAKZ1, X2, X3, AERR, RERR, ERROR, IER)
           AKZ=AKZ1+AKZ2
           IF(IER.GT.0) WRITE(ITERM,*)'IER AKZ1', IER
           BKZ1=DCADRE(FBKZ1, X1, X2, AERR, RERR, ERROR, IER)
           BKZ2=DCADRE(FBKZ1, X2, X3, AERR, RERR, ERROR, IER)
           BKZ=BKZ1+BKZ2
           IF(IER.GT.0) WRITE(ITERM,*)'IER BKZ1', IER
         END IF
C ---
           IF(IPQCNT.EQ.INDEX) THEN
             BSUM=BSUM+Y(ITUCNT)*
     *             (-(FLOAT(IP*IT)*K*K+FLOAT(IQ*IU)*L*L)*AKZ
     *             +KAPPA*KAPPA*BKZ)
           END IF
C---
20    CONTINUE
      RETURN
      END
C
      SUBROUTINE PARCHK(IR, IS, IPRFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
C --- Check parity
        SUM=FLOAT(IR+IS)
        TEST=DMOD(SUM,2.0D0)
        IF(TEST.NE.0.0) THEN
          IPRFLG=1
        ELSE
          IPRFLG=0
        END IF
        RETURN
      END
C ---
      SUBROUTINE TRNCHK(IR, IS, IG, M, ITFLG)
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA ITERM /3/
C --- Check truncation
        TEST=0.75D0*FLOAT(IR**2)+FLOAT(IS**2)*0.25D0+FLOAT(IG*IG)
        IF(TEST.GT.(FLOAT(M*M+1))) THEN
          ITFLG=1
        ELSE
          ITFLG=0
        END IF
        RETURN
      END
      DOUBLE PRECISION FUNCTION F1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF  XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      PI=3.14159265357930D0
      F1=DSIN(XIG*PI*Z)*DSIN(XIF*PI*Z)*DSIN(XIH*PI*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FAKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FAKZ0=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      RA/PI/XIH*(1.0D0+DCOS(PI*XIH*Z))
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FAKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FAKZ1=DSIN(XIF*PI*Z)*DSIN(XIG*PI*Z)*C1*KAPPA*(EXP(KAPPA*Z)+
     #      EXP(-KAPPA*Z))+PI*XIH*C2*DCOS(PI*XIH*Z)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FBKZ0(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FBKZ0=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      RA/PI/XIH*(Z+DSIN(PI*XIH*Z)/PI/XIH)
      RETURN
      END
C ---
      DOUBLE PRECISION FUNCTION FBKZ1(Z)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /REALSUBS/ XIF, XIG, XIH, C1, C2, KAPPA
      REAL*8 KAPPA
      COMMON /PARAMS/ BETA, GAMMA, PI, THIELE, RA, SIGMA, CHRL, FK, XLE
      FBKZ1=PI*XIF*DCOS(XIF*PI*Z)*DSIN(XIG*PI*Z)*
     #      C1*(EXP(KAPPA*Z)-EXP(-KAPPA*Z))+C2*DSIN(PI*XIH*Z)
      RETURN
      END
```

APPENDIX G

PUBLICATIONS ARISING FROM THE WORK DESCRIBED IN THIS THESIS

SPONTANEOUS COMBUSTION OF COAL STOCKPILES - AN UNUSUAL CHEMICAL REACTION ENGINEERING PROBLEM

Kevin Brooks, Steven Bradshaw and David Glasser
Department of Chemical Engineering
University of the Witwatersrand
WITS 2050
South Africa

(Published in Chemical Engineering Science, Vol. 43, No. 8, pp. 2139-2145, 1988)

ABSTRACT

Spontaneous combustion of stockpiled coal occurs because of reaction between atmospheric gases and coal, and causes pollution, as well as the potential loss of all or part of the stockpile. The stockpile may be viewed as a chemical reactor, and chemical engineering methods may be used to analyse and understand this problem.

The analysis is complicated by the fact that coal stockpiles in general possess little or no symmetry. In addition, it has been shown that the flow in coal stockpiles is mainly due to natural convection, which makes the problem rather different to those traditionally analysed in the chemical reaction engineering literature. In order that a realistic model of this phenomenon be formulated, it is necessary to derive mass, momentum and energy balance equations.

The various models that have been used to describe this phenomenon are discussed, starting with simple steady-state one-dimensional models, and culminating in a three-dimensional formulation. The simple one-dimensional model gives valuable insights into the behaviour of stockpiled coal. The solution of the more realistic model, using finite element techniques, is described. The models are compared in the light of the assumptions made in their derivation. Finally the practical implications of the work are discussed.

Keywords: Spontaneous combustion, mathematical model, coal storage, natural convection

SELF-IGNITION AND CONVECTION PATTERNS IN AN INFINITE COAL LAYER

Steven Bradshaw, David Glasser and Kevin Brooks
Department of Chemical Engineering
University of the Witwatersrand
Wits 2050
South Africa
Telephone 011-7162413

(Submitted to Chemical Engineering Communications, April 1989)

ABSTRACT

Spontaneous combustion can occur in a coal stockpile if the heat generated by oxidation cannot be dissipated at near ambient temperature. Determination of conditions for which combustion occurs is of great importance in designing coal stockpiles. An approximate analysis is used to obtain natural convection patterns in a laterally-unbounded layer of coal. Down-hexagons and two-dimensional rolls appear to be the stable flow planforms. A continuation procedure gives a simple criterion for the point of ignition in the layer in terms of easily measurable parameters. This criterion can be used to determine ignition points in the interior of a large coal stockpile and complements earlier work in which a similar criterion was developed for the edge of a stockpile.

Keywords:spontaneous combustion, ignition, natural convection, coal

SELF-IGNITION AND CONVECTION PATTERNS IN AN INFINITE COAL LAYER

Steven Bradshaw, David Glasser and Kevin Brooks
Department of Chemical Engineering
University of the Witwatersrand
Wits 2050
South Africa
Telephone 011-7162413

(Submitted to Chemical Engineering Communications, April 1989)

ABSTRACT

Spontaneous combustion can occur in a coal stockpile if the heat generated by oxidation cannot be dissipated at near ambient temperature. Determination of conditions for which combustion occurs is of great importance in designing coal stockpiles. An approximate analysis is used to obtain natural convection patterns in a laterally-unbounded layer of coal. Down-hexagons and two-dimensional rolls appear to be the stable flow planforms. A continuation procedure gives a simple criterion for the point of ignition in the layer in terms of easily measurable parameters. This criterion can be used to determine ignition points in the interior of a large coal stockpile and complements earlier work in which a similar criterion was developed for the edge of a stockpile.

Keywords:spontaneous combustion, ignition, natural convection, coal

SPONTANEOUS COMBUSTION IN BEDS OF COAL

David Glasser and Steven Bradshaw

Department of Chemical Engineering
University of the Witwatersrand
Wits 2050
South Africa
Telephone 011-7162413

To appear in Handbook of Heat and Mass Transfer, vol. 4, Combustion Science and Technology, Gulf Publishing, Matawan, New Jersey.

APPENDIX H

DECOMPOSITION OF A SOLENOIDAL FIELD

The material in this appendix is based on Joseph,1976.

A solenoidal field is one that satisfies:

$$\text{div}\ \underset{\sim}{u} = 0 \tag{H.1}$$

Such a field can be represented as the sum of two fields:

$$\underset{\sim}{u} = \underset{\sim}{u}_1 + \underset{\sim}{u}\ _2 \tag{H.2}$$

Where:

$$\text{div}\ \underset{\sim}{u}_1 = \underset{\sim}{r} \cdot \text{ourl}\ \underset{\sim}{u}_1 = 0 \tag{H.3}$$

$$\text{div}\ \underset{\sim}{u}_2 = \underset{\sim}{r} \cdot \underset{\sim}{u}_2 = 0 \tag{H.4}$$

$$\underset{\sim}{u}_1 = \underset{\sim}{\delta}\ \chi = \text{grad} \left(r \frac{\partial\ \chi}{\partial\ r} + \chi \right) - \underset{\sim}{r}\ \nabla^2\ \chi = \text{curl}^2\ (\ \underset{\sim}{r}\ \chi) \tag{H.5}$$

$$\underset{\sim}{u}_2 = \underset{\sim}{r} \times \text{grad}\ \Psi = -\ \text{curl}\ (\underset{\sim}{r}\ \Psi) \tag{H.6}$$

$\underset{\sim}{u}_1$ has no vertical vorticity and $\underset{\sim}{u}_2$ has no vertical velocity. χ is the poloidal potential and Ψ the toroidal potential.

Now consider the Darcy-Oberbeck Boussinesq momentum equation Eq. (3.2):

$$\nabla\ \Pi + \underset{\sim}{u} - \text{Ra}\ \theta\ \hat{z} = 0 \tag{H.7}$$

By taking the curl of Eq. (H.7) pressure Π is eliminated, and the z-component is:

$$\frac{\partial\ u_y}{\partial\ x} - \frac{\partial\ u_x}{\partial\ y} = 0 \tag{H.8}$$

The vorticity is defined:

$$\underset{\sim}{\Omega} = \text{curl}\ \underset{\sim}{u} \tag{H.9}$$

and it can be seen that:

$$\Omega_z = \frac{\partial\ u_y}{\partial\ x} - \frac{\partial\ u_x}{\partial\ y} = 0 \tag{H.10}$$

Thus we have shown that $\underset{\sim}{u}$ is purely poloidal.

REFERENCES

Acharya, S., Patankar, S.V., "Use of an Adaptive grid procedure for parabolic flow problems", Int. J.Heat Mass Transfer, 28 (6), 1057, (1985)

Akin, J.E., "Application and Implementation of Finite Element Methods", Academic Press, London, (1982)

Anderson, P., "Description of 'Fepde' - a Finite Element Code for the Solution of Systems of Steady-state, Two-Dimensional Non-linear Partial Differential Equations", Report, Department of Chemical Engineering, University of the Witwatersrand, (1986)

Anderson, P., "Anisotropic Transfer of Thermal Energy in Porous Media", Report, Department of Chemical Engineering, University of the Witwatersrand, (1987)

Anderson, P., Bradshaw, S.M., "PDFELM A Finite Element Code for the Solution of a General System of Steady or Unsteady-state Nonlinear Partial Differential Equations", Report, Department of Chemical Engineering, University of the Witwatersrand, (1987)

Bejan, A., "Lateral Intrusion of Natural Convection Into a Horizontal Porous Structure", J. Heat Transfer, 103, 237, (1981)

Bejan, A., Khair, K.R., "Heat and Mass Transfer by Natural Convection in a Porous Medium", Int. J. Heat Mass Transfer, 28 (5), 909, (1985)

Beukema, K.J., Bruin, S., Schenk J. "Three-dimensional Natural Convection in a Confined Porous Medium With Internal Heat Generation", Int. J. Heat Mass Transfer, 26(3) 451, (1983)

Bird R.B., Stewart, W.E., Lightfoot, E.N., "Transport Phenomena", Wiley, (1960)

Bowes, P.C., "Self-Heating:Evaluacing and Controlling the Hazards", Elsevier, New York, (1984)

Brooks, K.S., "A Simplifed Model For Spontaneous Combustion in Coal Stockpiles", PhD Thesis, University of the Witwatersrand, (1986) Brooks, K., Glasser, D., "A Simplified Model of Spontaneous Combustion in Coal Stockpiles", Fuel, 65, 1035, (1986)

Brooks, K., Balakotiah, V., Luss, D., "Effect of Natural Convection on Spontaneous Combustion of Coal Stockpiles", AIChE J., 34(3), 353, (1988a)

Brooks, K., [illegible], N., Glasser, D., "Evaluating the Risk of Spontaneous Combustion in Coal Stockpiles", Fuel, 67, 651, (1988b)

Brooks, K., Bradshaw, S., Glasser, D., "Spontaneous Combustion of Coal Stockpiles-An Unusual Chemical Reaction Engineering Problem", Chem. Eng. Sci., 43(8), 2139, (1988c)

Busse, F.H., Frick, H., "Square Pattern Convection in Fluids With Strongly Temperature-dependent Viscosity", J. Fluid Mech., 150, 451, (1985)

Carey, G.F., Oden, J.T., "Finite Elements:Fluid Mechanics", Prentice-Hall, Englewood Cliffs, (1986)

Carey, G.F., Plover, T., "Variable Upwinding and Adaptive Mesh Refinement in Convection-Diffusion", Int. J. Num. Meth. Engng., 19, 341, (1983)

Chan, Y.T., Banerjee, S., "Analysis of Transient Three-dimensional Natural Convection in porous Media", J. Heat Transfer, 103, 242, (1981)

Chandrasekhar, S., "Hydrodynamic and Hydromagnetic Stability", Clarendon Press, Oxford, (1961)

De Villiers, N.D., "Continuation Methods for the Numerical Solution of Nonlinear Problems in Engineering Science", PhD Thesis, University of the Witwatersrand, (1984)

Donea, J., "A Taylor-Galerkin Method for Convective Transport Problems", Int. J. Num. Meth. Engng, 20, 101, (1984)

Frank-Kamenetskii, D.A., "Diffusion and Heat Transfer in Chemical Kinetics", Plenum Press, New York, (1969) Gartling, D.K., & Becker, E.B., "Finite Element Analysis of Viscous Incompressible Fluid Flow. Part 1", Comp. Meths. Appl. Mech. Engng., 8, 51, (1976)

Gatica, J.E., Viljoen, H., Hlavacek, V., "Stability Analysis of Chemical Reaction and Free Convection in Porous Media", Int. Comm. Heat Mass Transfer, 14, 391, (1987a)

Gatica, J.E., Viljoen, H., Hlavacek, V., "Thermal Instability of Non-linearly Stratified Fluids", Int. Comm. Heat Mass Transfer, 14, 673, (1987b)

Gatica, J.E., Viljoen, H., Hlavacek, V., "Influence of Secondary Flows on the Stability of Chemically Reacting Systems", AIChE J., 34(2), 209 (1988)

Gijbels, M.R., Bruining, J., "Spontaneous Ignition Time of Underground Coal", Erdol und Kohle Erdgas, 35, 370 (1982)

Glasser, D., Bradshaw, S.M., "Spontaneous Combustion in Beds of Coal", to be published in Handbook of Heat and Mass Transfer, vol 4., Combustion Science and Technology, Gulf, New Jersey, (1989)

Gresho, P.M., Lee, R.L., "Don't Suppress The Wiggles - They're Telling You Something", Computers and Fluids, 9, 223, (1981)

Guney, M., "Oxidation and Spontaneous Heating of Coal", J. Pure Appl. Sci., 5, 1, (1972)

Handa, T., Morita, M., Sugawa, O., Ishii, T., Hayashi, K., "Computer Simulation of the Spontaneous Combustion of Coal Storage", Fire Science and Technology, 3 (1), 13, (1983)

Havstad, M.A., Burns, P.J., "Convective Heat Transfer in Vertical Cylindrical Annuli Filled With a Porous Medium", Int. J. Heat Mass Transfer, 25, 1755, (1982)

Heinrich, J.C., Zienkiewicz, O.C., "Quadratic Finite Element Schemes For Two-dimensional Convective Transport Problems", Int. J. Num. Meth. Engng, 11, 1831, (1977) Hickox, C.E., Gartling, D.K., "A Numerical Study of Natural Convection in a Vertical Porous Annular Layer", Int. J. Heat Mass Transfer, 28, 720, (1985)

Itay, M., "The Low Temperature Oxidation of Coal: Its Kinetics and Implications For Spontaneous Combustion", PhD Thesis, University of the Wiwatersrand, (1984)

Itay, M., Hill, C.R., Glasser, D., "A Study of the Low Temperature Oxidation of Coal", Fuel Processing Technology, 21, 81, (1989)

Jones, D.R., "The Dynamic Stability of Confined, Exothermically Reacting Fluids", Int. J. Heat Mass Transfer, 16, 157, (1973)

Joseph, D.D., "Stability of Fluid Motions I&II", Springer-Verlag, Berlin, (1976)

Kantorovich, L.V., Krylov, V.I., "Approximate Methods of Higher Analysis", Noordhof, Groningen, (1964)

Kimura, S., Schubert, G., Strauss, J.M., "Instabilities of Steady, Periodic and Quasi-Periodic Modes of Convection in Porous Media", J. Heat Transfer, 109, 350, (1987)

Kordylewski, W., Krajewski, Z., "Convection Effects on Thermal Ignition in Porous Media", Chem. Eng. Sci., 39(3), 610, (1984)

Kubicek, M., Marek, M., "Computational Methods in Bifurcation Theory and Dissipative Structures", Springer-Verlag, New York, (1983)

Lennie, T.B., McKenzie, D.P., Moore, D.R., Weiss, N.O., "The Breakdown of Steady Convection", J. Fluid Mech., 188, 47, (1988)

Lillington, J.N., Shepherd, I.M., "Central Difference Approximations To the Heat Transport Equation", Int. J. Num. Meth. Engng. 12, 1697, (1978)

Lin, Y.S., Akins, R.G., "A Suggested Characteristic Dimension For Natural Convection in Enclosures", Chem. Eng. Commun., 49, 119, (1986) Markatos, N.C., Pericleous, K.A., "Laminar and Turbulent Natural Convection in an Enclosed Cavity", Int. J. Heat Mass Transfer, 127, 755, (1984)

McDonald, H., "Combustion Modelling in Two and Three Dimensions - Some Numerical Considerations", Prog. Energy. Combust. Sci., 5, 97, (1979)

Merzhanov, A.G., Shtessel, E.A., "Free Convection and Thermal Explosion in Reacting Systems", Acta Astronautica, 18, 191, (1973)

Morita, M., Nishimoto, T., Kodama, J., Oshawa, T., Hayashi, K., "Spontaneous Combustion of Coal IV Computer Simulation", Fire Science and Technology, 6, 7, (1986)

NAG Finite Element Library, Greenough, C., Robinson, K., eds., Science and Engineering Research Council, Rutherford Appleton Laboratory, Oxfordshire, (1982)

Norton, P., "A Model for the Self-heating Reaction of Coal and Char", Fuel, 58, 456, (1979)

Norrie, D., De Vries, G., "The Finite Element Method", Academic Press, New York, (1973)

Poulikakos, D., Bejan, A., "Penetrative Convection in Porous Medium Bounded by a Horizontal Wall With Hot and Cold Spots", Int. J. Heat Mass Transfer, 27 (10), 1749, (1984)

Prasad, V., Kulacki, F.A., "Natural Convection in a Vertical Porous Annulus", Int. J. Heat Mass Transfer, 27, 207, (1984)

Roberts, P.H., Convection in Horizontal Layers with Internal Heat Generation. Theory", J. Fluid. Mech., 30(1), 33, (1967)

Runchal, A.K., "Convergence and Accuracy of Three Finite Difference Schemes For a Two-dimensional Conduction and Convection Problem", Int. J. Num. Meth. Engng, 4, 541, (1972)

Saatdjian, E., Caltagirone, J.P., "Natural Convection in a Porous Layer Under the Influence of an Exothermic Decompositon Reaction", J. Heat Transfer, 102, 654, (1980) Schmal, D., "A Model for Spontaneous Heating of Stored Coal", PhD Thesis, University of Delft, (1987)

Schmal, D., Duyzer, D.H., van Heuven, J.W., "A Model for the Spontaneous Heating of Coal", Fuel, 64, 963, (1985)

Schmidt, L.D., Elder, J.L., "Atmospheric Oxidation of Coal at Moderate Temperatures; Rates of Oxidation Reaction for Representative Coking Coals", Ind. Eng. Chem., 32, 249, (1940)

Schreiber, R., Keller, H.B., "Spurious Solution in Driven Cavity Calculation", J. Comp. Phys., 49, 165, (1983)

Schulenberg, T. Muller, V., "Natural Convection in Saturated Porous Layers With Internal Heat Sources", Int. J. Heat Mass Transfer, 27 (5), 677, (1984)

Shiralkar, G.S, Tien, C.L., "A Numerical Study of Laminar Natural Convection in Shallow Cavities", J. of Heat Transfer, 103, 228, (1981)

Smith, M.A., PhD Thesis, "Oxidation of Various Coals at Low Temperature", University of the Witwatersrand, to be submitted

Smith, R.M., "Finite Element Solutions of the Energy Equation at High Peclet Number", Computers and Fluids, 8, 335, (1980)

Sondreal, E.A., Ellman, R.C., "Laboratory Determination of Factors AffectingStorage North Dakota Lignite", Bureau of Mines Report, R.I., 7887, (1974)

Spalding, D.B., "A Novel Finite Difference Formulation For Differential Expressions Involving Both First and Second Derivatives", Int. J. Num. Meth. Engng, 4, 551-559, (1972)

Steinberg, V., Brand, H., "Convective Instabilities of Binary Mixtures With Fast Chemical Reaction in a Porous Medium", J. Chem Phys., 78(5), 2655, (1983)

Tveitereid. M., "Thermal Convection in a Horizontal Porous Layer with Internal Heat Sources", Int. J. Heat Mass Transfer, 20, 1049, (1977) Tveitereid, M., Palm, E., "Convection Due to Internal Sources", J. Fluid. Mech., 76(3), 481, (1976)

Van Doornum, G.A.W., "The Spontaneous Heating of Coal", J. Inst. Fuel., 27, 482, (1954)

Van Krevelen, D.W., "Coal", Elsevier, Amsterdam, (1981)

Vasseur, P., Hung Nguyen, T., Robillard, L., Tong Thi, V.K., "Natural Convection Between Horizontal Concentric Cylinders Filled With a Porous Layer With Internal Heat Generation", Int. J. Heat Mass Transfer, 27 (3), 337, (1984)

Viljoen, H., Hlavacek, V., "Chemically Driven Convection in a Porous Medium", AIChE J., 33, 1344, (1987)

Viljoen, H.J., Gatica, J.E., Hlavacek, V., "Induction Times for Thermal Explosion and Natural Convection in Porous Media", Chem. Eng. Sci., 43(11), 2951, (1988)

Viljoen, H.J., Gatica, J.E., Hlavacek, V., "Convective Regimes in Reactive Fluid Media due to the Interaction with Catalytic surfaces", Physics of Fluids (in press)

White, D.B., "The Planforms and Onset of Convection with a Temperature-dependent Viscosity", J. Fluid Mech., 191, 247, (1988)

Young, B.D., Two-dimensional Conduction and Convection in a Porous Medium: Application to Spontaneous Combustion of Coal", MSc Thesis, University of the Witwatersrand, (1985)

Young, B.D., Williams, D.F., Bryson, A.W., "Two-Dimensional Natural Convection and Conduction in a Packed Bed Containing a Hot-Spot and Its Relevance to the Transport of Air in a Coal-Dump", Int. J. Heat Mass Transfer, 29, 331, (1986)