SSCL-N-881

A Numerical Investigation of Spatial Approximations for Strongly Convective Flows

W. Schiesser, A. Yücel, R. Carcagno, and J. Demko

Superconducting Super Collider Laboratory*
2550 Beckleymeade Ave.
Dallas, TX 75237

November 1994

*Operated by the Universities Research Association, Inc., for the U.S. Department of Energy under Contract No. DE-AC35-89ER40486.

Index

Abstract 1

(1) The Advection Equation 3

(2) Numerical Approximation of the Advection Equation 6

(3) Programming Two Point Upwind Approximations 8

(4) An Evaluation of Approximations for the Advection Equation 15

(5) The Effect of Smoother Test Functions 18

(6) Numerical Solutions to the Advection Equation by Finite Elements 29

(7) The Quadratic Upwind/Petrov-Galerkin Finite Element Formulation 51

(8) Finite Volume Approximation of the Advection Equation 54

(9) The Nonhomogeneous Advection Equation 54

(10) Simultaneous PDEs 67

(11) Nonlinear Advection - Burgers' Equation 97

(12) Conclusion 108

References 110

Appendix 1: Solution of the Homogeneous Advection Equation 112

Appendix 2: Listing of Subroutine DSS012 115

Appendix 3: *Numerical Diffusion in the Two Point Upwind Approximation of the Advection Equation* 117

Appendix 4: Main Program SRKF45 121

Appendix 5: Listing of Subroutine DSS014 125

Appendix 6: Listing of Subroutine DSS018 126

Appendix 7: Listing of Subroutine DSS019 127

Appendix 8: Listing of Subroutine DSS020 129

Appendix 9: Listing of Subroutine DSS021 131

Appendix 10: Listing of Subroutine LEO1 133

Appendix 11: Listing of Subroutine LEO2 135

Appendix 12: Listing of Subroutine DSS002 137

Appendix 13: Listing of Subroutine DSS004 134

Appendix 14: Formulas for A_0, A_1, A_2 in Equations (56) and (57) 150

Appendix 15: Subroutines INITAL, DERV and PRINT, and Subordinate Routines for the QU/PG Solution of Advection Equation (7) 153

Appendix 16: Analytical Solution of the Nonhomogeneous Advection Equation 169

Appendix 17: Analytical Solution of the Simultaneous PDEs 175

Index for Figures 1 to 10 181

Index for Figures 11 to 20 187

Index for Figures 21 to 30 193

Index for Figures 31 to 40 199

Index for Figures 41 to 50 205

Index for Figures 51 to 55 211

Index for Figures 56 to 60 215

Figure 61 219

Figure 62 220

Figures 63 and 64 221

Figure 65 and 66 222

Figure 67 223

Abstract

The simulation of physical systems with strongly convective flow is a particularly challenging numerical problem because such systems tend to propagate steep spatial fronts (e.g., in temperature), and even discontinuities (shocks) which are difficult to represent numerically. Much effort has been devoted to this problem by numerical analysts, scientists and engineers, and this report is another attempt to arrive at a computationally useful procedure. Within the context of the SSC, the flow of supercritical helium (He) through the magnets is a process that must be understood to assure adequate cooling of the magnets according to an acceptable time schedule. Thus, the propagation of sharp temperature fronts along the magnets is a phenomena which must be understood through modeling and simulation.

In this study, we investigate approximations of the basic advection group

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial t} \tag{1}$$

which we write in subscript notation (to facilitate the discussion and analysis)

$$u_t + vu_x \tag{2}$$

v is a linear, constant velocity, although the numerical procedures to be developed and evaluated can be applied to nonlinear variants of (1) and (2), including two and three dimensions.

Ten approximations, which are essentially finite differences, are evaluated through a series of five test problems of varying difficulty, all of which have an exact solution which can be used to assess the quality of the numerical solutions. We conclude as a result of these 50 tests (10 approximations x five test problems = 50 cases) that the five point biased upwind approximation with five point approximations at the boundaries is the best of the 10 approximations. We also include listings of our code so that other investigators can know exactly how we did the calculations, and thereby evaluate our work. Finally, a set of subroutines is provided which can be easily called for use in applications.

In addition to finite differences, finite volumes are mentioned briefly, and then the five test problems are executed with two different finite elements approximations, giving an additional 10 tests

(two appoximations x five test problems = 10 cases). The study concludes with two tests based on the nonhomogeneous advection equation, four tests based on a problem involving two PDEs and one test based on the nonlinear Burgers' equation, a total of seven additional tests. Thus, the complete test set has 67 cases. In all of these cases, analytical solutions were used to evaluate the numerical solutions.

(1) The Advection Equation

We start by analyzing the convective system illustrated in Figure 1 in terms of an energy balance written for an incremental length Δx

rate of accumulation of energy =
rate of heat flowing in - rate of heat flowing out

$$A\Delta\rho C_p\frac{\partial T}{\partial t} = Av\rho C_p T_x - Av\rho C_p T_{x+\Delta x} \tag{3}$$

Division by the coefficient of the t derivative, followed by $\Delta x \rightarrow 0$, gives

$$\frac{\partial T}{\partial t} = -v\frac{\partial T}{\partial x} \tag{4}$$

We have made several assumptions in deriving equation (4), e.g., no heat transfer to the fluid, constant fluid (temperature independent) properties, constant velocity, all of which can be relaxed, i.e., in applying the numerical approximations to be discussed to more general and realistic problem systems.

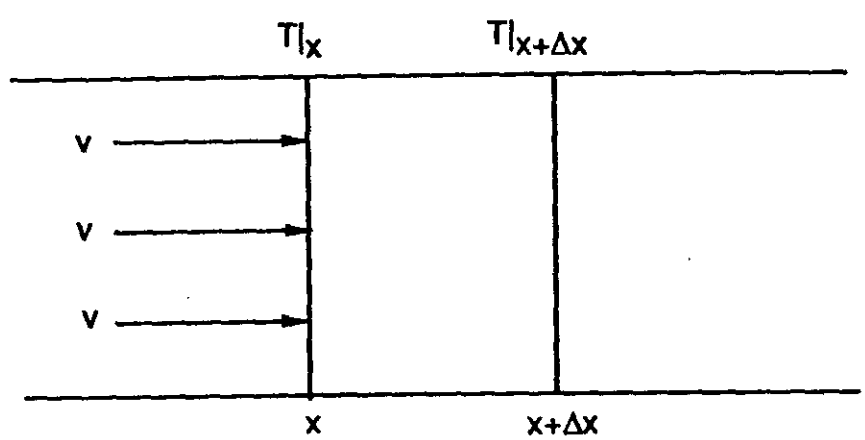


Figure 1.1: One-dimensional Convective System

Equation (4) is termed the *advection equation* since it basically models flow. The group of derivatives $\frac{\partial T}{\partial t} + v\frac{\partial T}{\partial x}$ is termed the *advection group*, and essentially what we are considering are ways to

approximate this advection group. Somewhat ironically, although equation (4) is one of the simplest partial differential equations (PDEs) we can conceive, it also is *one of the most difficult to integrate numerically* for reasons which will soon be apparent. Mathematically, it is termed a *first order hyperbolic* PDE.

Before proceeding with the numerical analysis of equation (4), we introduce some notation that will simplfy the discussion, and the programming of numerical approximations for the advection group. If, instead of using the full derivative notation, we use a subscript to denote a partial derivative, equation (4) can be written simply as

$$T_t + vT_x = 0 \tag{5}$$

An independent variable appearing as a subscript denotes a partial derivative with respect to that independent variable. Thus, $\frac{\partial T}{\partial t}$ is denoted as T_t, and $\frac{\partial T}{\partial x}$ is denoted as T_x. Other subscripts (not with respect to the independent variables of a PDE) are interpreted in the usual way (as just subscripts). Finally, in order to follow the usual convention of the numerical analysis literature for PDEs, we denote the dependent variable as "u" (not to be confused with internal energy), so equations (4) and (5) become

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} \tag{6}$$

$$u_t + vu_x = 0 \tag{7}$$

Since equation (7) (or (6)) is first order in t and x, it requires one initial condition and one boundary condition

$$u(x,0) = g(x),\ u(0,t) = f(t), \tag{8)(9}$$

We now consider some approximations for the basic advection or convective problem, equations (7) to (9). First, we again state that equation (7) is deceptively simple; it is *one of the most difficult PDEs to integrate numerically.* To illustrate this point, we first consider the analytical solution to equations (7), (8) and (9) for the special case $g(x) = 0$ and $f(t) = 0$ for $t < 0$

$$u(x,t) = f(t - x/v)h(t - x/v) \tag{10}$$

h(t) is the Heaviside unit step function

$$h(t) = \begin{cases} 0, & t < 0 \\ 1, & t > 0 \end{cases} \tag{11}$$

Equation (10) is derived in Appendix 1.

This solution can easily be verified. If we define $\lambda = t - x/v$, then from equation (10), $u(x,t) = f(\lambda)h(\lambda)$, i.e., u is a function of only λ. The terms in equation (7) become

$$u_t = (du/d\lambda)(\partial\lambda/\partial t) = du/d\lambda$$

$$vu_x = v(du/d\lambda)(\partial\lambda/\partial x) = v(du/d\lambda)(-1/v)$$

and, upon addition of these two equations, we see that equation (7) is satisfied. Also, equation (10) satisfies the initial and boundary conditions, equations (8) and (9) (again, for the special case $g(x) = 0$ and $f(t) = 0$ for $t < 0$).

Now, consider the additional special case $f(t) = 1$, $t > 0$, so that at $x = 0$, the entering temperature undergoes a unit step change at $t = 0$

$$f(t) = \begin{cases} 0, & t < 0 \\ 1, & t > 0 \end{cases} \tag{12}$$

i.e., f(t) defined by equation (12) is the Heaviside unit step function, h(t).

In physical terms, the temperature is initially zero since $g(x) = 0$ (this does not imply absolute zero, but rather, that the datum upon which the temperature is defined is zero, e.g., zero degrees centigrade). Then, at $t = 0$, the entering temperature (at $x = 0$) jumps from zero to one according to equation (12),

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

It then follows from equation (10)

$$u(x,t) = h(t - x/v) \tag{13}$$

i.e., *the unit step in temperature at $x = 0$ propagates left to right at velocity v*; eventually, at $t = L/v$, the unit step will reach the other end of the system (L is the overall length). At any position x when $t = x/v$, an observer would see a unit step function pass by. In other words, equation (7) *propagates a finite discontinuity for this special case*, which is a major source of numerical difficulties.

Why does this cause a problem in computing a numerical solution? Consider the slope of the solution at any point x when $t = x/v$; the *slope u_x in equation (7) is infinite!* (which follows from equation (13)). Clearly, any numerical procedure based on well behaved functions will fail under this condition. In a sense, we are asking the impossible of any numerical approximation we might consider for calculating spatial derivatives such as u_x in equation (7). With this rather sobering conclusion in mind, we can now consider what we might do to approximate equation (7) (and, again, recall it is probably the simplest possible description of convection).

(2) Numerical Approximation of the Advection Equation

We now consider how equation (7) might be approximated for numerical solution. One approach, within the context of the *numerical method of lines* [Schiesser (1)] is to replace the spatial derivative, u_x, in equation (7) with a discrete approximation such as a *first order backward finite difference*

$$u_x(x,t) \doteq \frac{u(x,t) - u(x-\Delta x,t)}{\Delta x} + O(\Delta x) \tag{14}$$

where $O(\Delta x)$ is interpreted as "of order Δx" and means that the principal (truncation) error of equation (14) in calculating u_x is proportional to Δx. Substitution of equation (14) in equation (7) gives

$$\frac{du(x,t)}{dt} = -v\left\{\frac{u(x,t) - u(x-\Delta x,t)}{\Delta x}\right\} \tag{15}$$

Note that equation (15) is now an ODE since x has essentially been removed as an independent variable, and only t remains as an independent variable.

We can now use equation (15) at each of the N points in a spatial grid in x, i.e., at $i = 1, 2, \ldots, N$, with grid spacing $\Delta x = L/(N - 1)$, so that $x = (i - 1)\Delta x$ and, of course, $L = (N - 1)\Delta x$. Thus, if $u(x,t) = u_i$, $u(x-\Delta x,t) = u_{i-1}$, equation (7) can be written as

$$\frac{du_i}{dt} = -v\left(\frac{u_i - u_{i-1}}{\Delta x}\right) \tag{16}$$

At $i = 1$, we cannot use equation (6.27) since it would involve $u_{i-1} = u_0$ which is a *fictitious point* outside the left boundary of the grid. Rather, for $i = 1$, we use a *first order forward finite difference approximation*

$$u_x(x,t) \doteq \frac{u(x+\Delta,t) - u(x,t)}{\Delta x} + O(\Delta x) \tag{17}$$

which when substituted in equation (7) gives

$$\frac{du_1}{dt} = -v\left(\frac{u_2 - u_1}{\Delta x}\right) \tag{18}$$

Note that from boundary condition (9), $u_1 = f(t)$ in equation (18). Thus, as an alternative to equation (18), we can directly apply boundary condition (9) as $u_1 = f(t)$, with $du_1/dt = 0$ or $du_1/dt = df(t)/dt$; either of these last two ODEs (for du_1/dt) will lead to the same numerical solution.

Thus, we now have a system of N ODEs for the N dependent variables $u_1, u_2, \ldots, u_N$. All that remains is to specify the initial conditions for these N ODEs, which come from equation (8)

$$u_i(0) = g((i - 1)\Delta x, \; i = 1, 2, \ldots, N \tag{19}$$

For flow in the negative x direction (right to left), modeled by

$$u_t - vu_x = 0 \tag{20}$$

equation (7) becomes

$$\frac{du_i}{dt} = v\left(\frac{u_{i+1} - u_i}{\Delta x}\right), \; i = 1, 2, \ldots, N - 1 \tag{21}$$

$$\frac{du_N}{dt} = v\left(\frac{u_{N-1} - u_N}{\Delta x}\right) \tag{22}$$

Equation (8) again provides the initial conditions for equations (21) and (22). However, in order to program flow in both the positive and negative x directions, we must have some logic in the coding to select either equations (16) and (18), or equations (21) and (22). These equations are generally termed *two point upwind approximations* since they involve two points along the spatial grid, e.g., u_{i-1} and u_i in equation (16), and one of these is the upwind (or upstream) point, e.g., u_{i-1} in equation (16).

In summary, through the use of first order forward and backward finite difference approximations, we have replaced the advection equation, equation (7) for flow in the positive x direction, or equation (20) for flow in the negative x direction, with a set of approximating ODEs that can be integrated by a library routine for ODEs, e.g., RKF45 or LSODES. This is the essence of the

method of lines.

(3) Programming Two Point Upwind Approximations

Equations (16) and (18), and (21) and (22), can now be put in a general-purpose library routine for the calculation of u_x in equations (7) and (20), respectively (or in any advection-like equation as will be discussed subsequently). Here is the coding for stepping along the spatial grid, which appears in subroutine DSS012 listed in Appendix 2

```
      DX=(XU-XL)/FLOAT(N-1)
      IF(V.LT.0.)GO TO 10
C...
C...      (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
               UX(1)=(U(2)-U(1))/DX
               DO 1 I=2,N
               UX(I)=(U(I)-U(I-1))/DX
1              CONTINUE
               RETURN
C...
C...      (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
10             NM1=N-1
               DO 2 I=1,NM1
               UX(I)=(U(I+1)-U(I))/DX
2              CONTINUE
               UX(N)=(U(N)-U(N-1))/DX
               RETURN
```

Program 1: Coding for Two Point Upwind Approximations
(taken from Subroutine DSS012 in Appendix 2)

The following points should be noted about this code:

(1) The left and right values of x, XL and XR, are available as inputs through the arguments of subroutine DSS012, along with the total number of grid points in the spatial grid, N. Then, the grid spacing, DX (= Δx) can be calculated.

(2) The velocity in equations (7) and (20) also is available through an argument of DSS012, and can be used to select either equation (16) (programmed first) or equation (21) (programmed at statement 10).

(3) DO loop1 and 2 are straightforward implementations of equations (16) and (18), or (21) and (22), respectively (array U is also available through an argument of subroutine DSS012).

Thus, this coding returns the first derivative UX (u_X in equation (7) or equation (16)) to the calling program (through an argument of DSS012). Subroutine DSS012 can then be called in the derivative subroutine for ODEs to implement equations (16) and (18), or (21) and (22) (depending on the sign of v). This is illustrated by the following coding for:

(1) Subroutine INITAL which defines the initial conditions for equations (16) and (18), or equations (21) and (22). i.e., equation (19).

(2) Subroutine DERV where the derivatives in equations (16) and (18), or equations (21) and (22) are programmed after DSS012 is called (for the first run, NORUN = 1). Note also the implementation of boundary condition (9) at the beginning of subroutine DERV, i.e., u(0,t) = f(t) = h(t) (U(1) = 1 in Fortran), followed by $du_1/dt = 0$ (UT(1) = 0 in Fortran). Also, the coding of equation (7) in DO loop 1 bears a close resemblance to equation (7), which is a significant advantage of the method of lines.

(3) Subroutine PRINT which prints the numerical and analytical solutions, and plots these solutions.

```
      SUBROUTINE INITAL
C...
C...  PURE CONVECTION THROUGH A TUBE WITH A UNIFORM VELOCITY PROFILE CAN
C...  BE DEPICTED AS
C...
C...
C...          ..............................................
C...                         V......+.   .
C...                         V......+.   .
C...                         V......+.DX. U(X,T)
C...                         V......+.   .
C...                         V......+.   .
C...          ..............................................
C...        X=0                                            X=XL
C...
C...  IF A MASS OR ENERGY BALANCE IS WRITTEN FOR A DIFFERENTIAL SECTION
C...  OF THE TUBE, THE WELL-KNOWN ADVECTION EQUATION RESULTS
```

```
C...
C...      U + V*U  = 0                                                    (1)
C...       T     X
C...
C...   SINCE EQUATION (1) IS FIRST-ORDER IN THE SPATIAL VARIABLE, X, AND
C...   THE TEMPORAL VARIABLE, T, IT REQUIRES ONE BOUNDARY CONDITION AND
C...   ONE INITIAL CONDITION
C...
C...      U(0,T) = F(T), U(X,0) = G(X)                                  (2)(3)
C...
C...   FOR WHICH THE EXACT SOLUTION TO EQUATIONS (1) TO (3) IS
C...
C...      U(X,T) = 0, X GT V*T
C...                                                                      (4)
C...      U(X,T) = F(T - X/V), X LT V*T
C...
C...   THE SPATIAL DERIVATIVE IN EQUATION (1), U  IS REPLACED BY A FINITE
C...                                            X
C...   DIFFERENCE APPROXIMATION OVER A GRID OF N (= 21) POINTS IN X.  A
C...   SYSTEM OF N INITIAL-VALUE ORDINARY DIFFERENTIAL EQUATIONS IN T
C...   RESULTS, ONE DIFFERENTIAL EQUATION FOR EACH GRID POINT.  THIS
C...   SYSTEM OF ODES IS THEN INTEGRATED SIMULTANEOUSLY TO OBTAIN THE
C...   NUMERICAL SOLUTION TO EQUATIONS (1) TO (3).
C...
       COMMON/T/      T,  NSTOP,  NORUN
     1       /Y/  U(21)
     2       /F/ UT(21)
     3      /SD/ UX(21)
     4       /C/      V,     XL,      N,     DX,    SSE,     IP
C...
C...   SET THE VELOCITY, LENGTH OF THE SYSTEM, NUMBER OF SPATIAL GRID
C...   POINTS, LENGTH OF THE GRID INTERVAL, SUM OF SQUARES OF ERRORS OF
C...   THE NUMERICAL SOLUTION, COUNTER FOR THE PLOTTED SOLUTION USED IN
C...   SUBROUTINE PRINT
       V=1.
       XL=1.
       N=21
       DX=XL/FLOAT(N-1)
       SSE=0.
       IP=0
C...
C...   INITIAL CONDITION
       DO 1 I=1,N
       U(I)=0.
1      CONTINUE
C...
C...   INITIALIZE THE TEMPORAL DERIVATIVES IN COMMON/F/
       CALL DERV
       RETURN
```

```
      END

      SUBROUTINE DERV
      COMMON/T/       T,   NSTOP,   NORUN
     1       /Y/  U(21)
     2       /F/ UT(21)
     3      /SD/ UX(21)
     4       /C/       V,      XL,       N,      DX,     SSE,      IP
C...
C...  UNIT STEP FUNCTION
      U(1)=1.
      UT(1)=0.
C...
C...  COMPUTE THE SPATIAL DERIVATIVE IN EQUATION (1) BY ONE OF A
C...  SERIES OF SPATIAL DISCRETIZATIONS
C...
C...  TWO POINT UPWIND
      IF(NORUN.EQ.1)CALL DSS012(0.,XL,N,U,UX,V)
C...
C...  PDE
      DO 1 I=2,N
      UT(I)=-V*UX(I)
1     CONTINUE
      RETURN
      END

      SUBROUTINE PRINT(NI,NO)
      COMMON/T/       T,   NSTOP,   NORUN
     1       /Y/  U(21)
     2       /F/ UT(21)
     3      /SD/ UX(21)
     4       /C/       V,      XL,       N,      DX,     SSE,      IP
C...
C...  DIMENSION THE ARRAYS FOR PLOTTING THE SOLUTION
      DIMENSION TPE(201),TPN(201),UPE(201),UPN(201)
C...
C...  PRINT A HEADING FOR NUMERICAL BSOLUTION
      IF(IP.EQ.0)WRITE(NO,1)
1     FORMAT(9X,1HT,6X,6HU(0,T),6X,6HU(1,T),7X,5HEXACT,11X,1HE)
C...
C...  STORE THE EXACT SOLUTION IN ARRAY UPE FOR PLOTTING
      X=1.0
      TXV=T-X/V
      DO 3 I=1,NP
C...
C...  UNIT STEP FUNCTION
      IF(TXV.LE.0.)EXACT=0.
      IF(TXV.GT.0.)EXACT=1.0
3     CONTINUE
```

```
C...
C...  COMPUTE THE FIGURE OF MERIT, SSE
      E=U(N)-EXACT
      SSE=SSE+E**2
C...
C...  PRINT THE NUMERICAL AND EXACT SOLUTIONS, ERROR
      IF((IP/10*10).EQ.IP)
     1 WRITE(NO,20)T,U(1),U(N),EXACT,E
20    FORMAT(F10.2,4F12.3)
C...
C...  STORE THE NUMERICAL SOLUTION FOR SUBSEQUENT PLOTTING VIA TOP
C...  DRAWER
      IP=IP+1
      UPN(IP)=U(N)
C...
C...  STORE THE EXACT SOLUTION IN ARRAY UPE FOR PLOTTING
      UPE(IP)=EXACT
C...
C...  STORE THE INDEPENDENT VARIABLE FOR SUBSEQUENT PLOTTING VIA TOP
C...  DRAWER
      NP=201
      TPE(IP)=2.0*FLOAT(IP-1)/FLOAT(NP-1)
      TPN(IP)=TPE(IP)
C...
C...  TEST FOR THE END OF A RUN
      IF(IP.LT.NP)RETURN
C...
C...  OPEN FILE FOR TOP DRAWER PLOTTING
         OPEN(4,FILE='T.TOP',STATUS='NEW')
C...
C...  WRITE TOP DRAW FILE FOR PLOTTING
         WRITE(4,17)
17       FORMAT(' SET LIMITS X FROM 0 to 2 Y FROM -0.5 TO 1.5',/,
     1           ' SET FONT DUPLEX')
         WRITE(4,13)
13       FORMAT(' SET WINDOW X 2 TO 6 Y 2 TO 8')
      WRITE(4,14)(TPN(I),UPN(I),I=1,NP)
14    FORMAT(2F10.3)
      WRITE(4,16)
16    FORMAT(' JOIN 1')
      WRITE(4,14)(TPE(I),UPE(I),I=1,NP)
      WRITE(4,16)
      WRITE(4,18)NORUN,SSE
18    FORMAT(
     1' Title 3.5 9.0 "Fig. ',I2,'"'
     2,/,' TITLE LEFT "u(1,t)" '
     3,/,' TITLE BOTTOM "t"'
     4,/,'Title 3.0, 0.75 "SSE = ',F6.3,'"')
      WRITE(4,15)
```

```
15    FORMAT(' NEW FRAME')
C...
C...  LEGEND OF RUNS AT THE END
      IF(NORUN.EQ.10)THEN
      WRITE(4,19)
19    FORMAT(
     1' Title 3.25 9.0 "Heaviside Unit Step Funtion"',/,
     1' Title 3.25 8.5 "Fig.  1: Two Point Upwind"',/,
     1' Title 3.25 3.5 "All solutions are for a 21-point grid"')
      END IF
      RETURN
      END
```

Program 2: Subroutines INITAL, DERV and PRINT for Equations (7) to (9)
with $g(x) = 0$, $f(t) = h(t)$ (Unit Step Function)

The numerical and graphical output from subroutine PRINT is listed in Table 1 and Figur 1 (the figures appear in one place at the end of this report so that can be more easily compared)

```
RUN NO. -    1  METHOD OF LINES SOLUTION OF THE ADVECTION EQUATION - DSS012

INITIAL T -  0.000E+00

  FINAL T -  0.200E+01

  PRINT T -  0.100E-01

NUMBER OF DIFFERENTIAL EQUATIONS -  21

MAXIMUM INTEGRATION ERROR -  0.100E-03
```

T	U(0,T)	U(1,T)	EXACT	E
0.00	1.000	0.000	0.000	0.000
0.10	1.000	0.000	0.000	0.000
0.20	1.000	0.000	0.000	0.000
0.30	1.000	0.000	0.000	0.000
0.40	1.000	0.000	0.000	0.000
0.50	1.000	0.003	0.000	0.003
0.60	1.000	0.021	0.000	0.021
0.70	1.000	0.077	0.000	0.077
0.80	1.000	0.188	0.000	0.188
0.90	1.000	0.349	0.000	0.349
1.00	1.000	0.530	0.000	0.530
1.10	1.000	0.694	1.000	-0.306
1.20	1.000	0.820	1.000	-0.180
1.30	1.000	0.903	1.000	-0.097

1.40	1.000	0.952	1.000	-0.048
1.50	1.000	0.978	1.000	-0.022
1.60	1.000	0.991	1.000	-0.009
1.70	1.000	0.996	1.000	-0.004
1.80	1.000	0.999	1.000	-0.001
1.90	1.000	0.999	1.000	-0.001
2.00	1.000	1.000	1.000	0.000

Table 1: Numerical Output from Program 2

The following points should be noted about the tabular and graphical outoput in Table 1 and Figure 1.

(1) The numerical solution only approximates the exact solution. In particular, the numerical solution is "smoothed" which is generally termed *numerical diffusion.* This is a feature of the two point, upwind approximations and results from an additional diffusion term introduced into the advection equation (equation (7)) by the upwind approximations, as demonstrated in Appendix 3. Recall again that this is essentially an impossible problem since u_x in equation (7) is infinite at the point of discontinuity in the solution $u(x,t) = h(t - x/v)$, so that the numerical solution can only be an approximation in some fashion.

(2) A sum of squares of errors, SSE (the sum of squares of the differences between the exact and numerical solutions), is computed in subroutine PRINT. SSE is then printed at the end of the run in the graphical output (of Figure 1); in this case, SSE = 5.263. SSE can then be used as a figure of merit in judging the performance of various approximations of the advection equation to be considered subsequently.

In order to provide the complete Fortran program SRKF45 used to compute the results in Table 1, we have listed the main program in Appendix 4 which calls library integrator RKF45 [Forsythe, et al (1)]. Also, SRKF45 reads three lines of data to control the integration (via formats 1000, 1001 and 1002) that are listed below

```
METHOD OF LINES SOLUTION OF THE ADVECTION EQUATION - DSS012
0.        2.0        0.01
   21                        0.0001
END OF RUNS
```

Table 2: Data for Program 2 Read by Main Program SRKF45

The data are printed in a summary at the beginning of each run (see Table 1) and are explained below:

(1) Line 1 - a documentation title which is merely read, then printed at the beginning of the run read with the format 20A4.

(2) Line 2 - the initial, final and print interval values of time (T0, TF and TP in main program SRKF45) read with the format 3E10.0. In the case of Table 2, the solution is stored for plotting (via calls to subroutine PRINT) every 0.01 time units, so that there are a total of 2/0.01 + 1 = 201 points stored, including the initial condition at t = 0 (this large number was selected to give smooth plots such as in Figure 1). Note however, that the solution is printed every 0.1 time units via FORMAT 20 through the logic of the counter IP so that a reasonable number of solution points are printed (21 rather than 201).

(3) Line 3 - the number of ODEs and the error tolerance, in this case 21 and 0.0001, read with the format I5, 20X,E10.0.

Finally, the characters END OF RUNS in columns 1 to 11 are read to terminate the run.

In summary, although the two point upwind approximation is generally stable computationally in the sense that the solutions are bounded (in this case $0 \leq u(x,t) \leq 1$), it suffers from numerical diffusion which may be an unacceptable distortion of the solution. We therefore now consider a series of approximations, some of which offer better performance.

(4) An Evaluation of Approximations for the Advection Equation

We now consider how some alternative approximations for u_x in equation (7). We can start with the idea that if two point upwind approximations produce significant distortion in the numerical solution (Figure 1), then additional points might improve the solution. We therefore consider *three point upwind approximations* in which two (rather than one) points upwind of the point where the derivative u_x is to be computed are used. The finite difference equations can be derived by a Taylor series analysis, or by a simple algorithm developed by Fornberg [Schiesser (1.2)]; the resulting three point upwind approximations are second order correct, i.e., $O(\Delta x^2)$, which would lead one to believe that the solutions should be better than for the two point upwind approximations of equations (16) and

(18), or (21) and (22) which are $O(\Delta x)$.

Subroutine DSS014 with three point upwind approximations is listed in Appendix 5. The code to call DSS014 in subroutine DERV (for NORUN = 2) is

```
C...
C...  COMPUTE THE SPATIAL DERIVATIVE IN EQUATION (1) BY ONE OF A
C...  SERIES OF SPATIAL DISCRETIZATIONS
C...
C...  TWO POINT UPWIND
      IF(NORUN.EQ.1)CALL DSS012(0.,XL,N,U,UX,V)
C...
C...  THREE POINT UPWIND
      IF(NORUN.EQ.2)CALL DSS014(0.,XL,N,U,UX,V)
```

In order for the code to execute a second run and thereby call DSS014, an additional set of data must be added, i.e.,

```
METHOD OF LINES SOLUTION OF THE ADVECTION EQUATION - DSS012
0.        2.0       0.01
   21                        0.0001
METHOD OF LINES SOLUTION OF THE ADVECTION EQUATION - DSS014
0.        2.0       0.01
   21                        0.0001
END OF RUNS
```

Thus, the first set of data is read (NORUN = 1) and a run executed with DSS012. Then the second set of data is read (NORUN = 2) and a second run is executed with DSS014. Additional sets of data can be added and main program SRKF45 will continue to execute runs until the END OF RUNS data line is read.

Because of the many finite difference approximations, and associated subroutines and outputs to be considered, we will now discuss the performance of the various approximations only through the graphical output. For example, the output from the three point upwind approximations in subroutine DSS014 is plotted in Figure 2. We note the second major form of numerical distortion, *numerical oscillation*, to the left of the discontinuity. Thus, we conclude that merely adding more points to an upwind scheme does not improve the performance of the method of lines solution (although SSE = 3.494 which is somewhat lower than for the two point upwind approximation for which SSE = 5.263).

We therefore consider the performance of *four point biased upwind* approximations which use two points upstream and one point downstream of the point where u_x is calculated. Again, these

approximations can be derived by a Taylor series analysis or by Fornberg's algorithm, and they are $O(\Delta x^3)$. The final results are coded in subroutine DSS018 listed in Appendix 6. We see in Figure 3 that the numerical solution is substantially improved over Figures 1 and 2, with SSE = 1.711. We therefore conclude that biasing does help (i.e., using downstream and well as upstream points, but more upstream than downstream points, which is the reason for the term "biased"). Note also that four point approximations are used at the boundaries in subroutine DSS018 (at x = XL and XR), but they cannot be four point biased upwind approximations if we are to avoid going outside the spatial grid running from x = XL to XR.

We can therefore consider a variant of DSS018 in which we use two point approximations at the points on and near the boundaries, then switch to four point biased upwind approximations at the interior points. This combination is programmed in subroutine DSS019 in Appendix 7. The corresponding graphical output is given in Figure 4, which appears to be essentially equivalent to Figure 3, but with a slightly higher SSE (2.157 vs 1.711 for Figure 3). Thus, we conclude that switching to lower order approximations at the boundaries does not offer any advantage, and this conclusion is confirmed with subsequent examples.

Since biasing appears to improve the performance of the spatial differentiators used to calculate u_x in equation (7), we consider *five point biased upwind approximations* originally proposed by Carver and Hinds (3), which use three points upwind and one point downwind, and are $O(\Delta x^4)$. These approximations are programmed in subroutine DSS020 in Appendix 8. The graphical output is given in Figure 5, which has the lowest SSE so far (1.392). Also, two point approximations at the boundaries rather than five point approximations are programmed in subroutine DSS021 in Appendix 9. The output is in Figure 6, which has a higher SSE (= 2.036) than Figure 5, so again we conclude that using lower order approximations at the boundaries offers no advantage.

Next, we consider a four point biased upwind approximation proposed by Leonard (4), as coded in subroutine LEO1 in Appendix 10. The solution is plotted in Figure 7 for which SSE = 1.881. This example illustrates that the SSE figure of merit tends to emphasize discrepancies in vertical distances due to numerical diffusion more than due to numerical oscillation. Also, two point approximations at the boundaries rather than four point approximations are used in LEO2 in Appendix 11; the solution is plotted in Figure 8 with SEE = 2.133. As might be expected, the two point approximations tend to reduce the oscillation (recall the oscillation-free solution of Figure 1) at the expense of increasing the numerical diffusion.

Finally, since biasing has produced better numerical solutions (relative to pure upwinding), we might conclude that further biasing, say to centered approximations, should improve the solution. In subroutine DSS002 listed in Appendix 12 we have three point approximations programmed which are $O(\Delta x^2)$. The numerical solution is given in Figure 9, with SSE = 3.654. The obvious problem here is excessive numerical oscillation. We might consider using more points in a centered approximation. This is done in subroutine DSS004 in Appendix 13 with graphical output in Figure 10, for which SSE = 2.456. Clearly the numerical oscillation is worse, and we can therefore generally conclude that *centered approximations should not used in the method of lines solution of the advection equation if the forcing function has a discontinuity.*

This last statement also suggests that the performance of these various finite difference approximations may be strongly dependent on the characteristics of the forcing function, f(t) in equation (9). In other words, as we move away from the unit step function, f(t) = h(t), (which has a finite discontinuity and a first derivative $dh(t)/dt = \delta(t)$, the unit impulse or Dirac function, that really cannot be represented by any of the finite difference approximations considered previously), and consider smoother functions, the performance of the various approximations should improve. This is indeed the case, as we now demonstrate. We conclude this section by noting that f(t) = h(t) is not realistic in physical problems anyway. Some degree of smoothness will occur in all realistic applications.

(5) <u>The Effect of Smoother Test Functions</u>

We consider as a test function f(t) = r(t) where r(t) is a truncated ramp function defined as

$$\begin{aligned} r(t) &= 0,\ t < 0 \\ r(t) &= s,\ 0 < t < 1/s \\ r(t) &= 1,\ t > 1/s \end{aligned} \tag{23}$$

where we take s = 5. r(t) has a steep, but finite, slope. Therefore, it is continuous, but its first derivative is discontinuous, so it is a stringent test problem.

The exact solution to equation (7) with f(t) = r(t) in boundary condition (9) is

$$u(x,t) = r(t - x/v)h(t - x/v) \tag{24}$$

which follows immediately from equation (10) with f(t) = r(t) and which can be used to evaluate the 10 finite difference approximations considered previously. Before proceeding to the final results, we list subroutine: (1) DERV with equation (23) programmed as a boundary condition (forcing function) for the 10 finite difference approximation (NORUN = 1 to 10) and (2) subroutine PRINT with equation (24) programmed as the exact solution and the extension of the plotting for all 10 finite difference approximations (v. FORMAT 19):

```
      SUBROUTINE DERV
      COMMON/T/       T,   NSTOP,   NORUN
     1       /Y/  U(21)
     2       /F/ UT(21)
     3      /SD/ UX(21)
     4       /C/       V,      XL,       N,      DX,     SSE,      IP
C...
C...  TRUNCATED RAMP FUNCTION
      S=5.0
      IF(T.LT.0.)U(1)=0.
      IF(T.GT.(1./S))U(1)=1.
      IF((T.GE.0.).AND.(T.LE.(1./S)))U(1)=S*T
      UT(1)=0.
C...
C...  COMPUTE THE SPATIAL DERIVATIVE IN EQUATION (1) BY ONE OF A
C...  SERIES OF SPATIAL DISCRETIZATIONS
C...
C...  TWO POINT UPWIND
      IF(NORUN.EQ.1)CALL DSS012(0.,XL,N,U,UX,V)
C...
C...  THREE POINT UPWIND
      IF(NORUN.EQ.2)CALL DSS014(0.,XL,N,U,UX,V)
C...
C...  FOUR POINT BIASED UPWIND WITH FOUR POINT APPROXIMATIONS
C...  AT THE BOUNDARIES
      IF(NORUN.EQ.3)CALL DSS018(0.,XL,N,U,UX,V)
C...
C...  FOUR POINT BIASED UPWIND WITH TWO POINT APPROXIMATIONS
C...  AT THE BOUNDARIES
      IF(NORUN.EQ.4)CALL DSS019(0.,XL,N,U,UX,V)
C...
C...  FIVE POINT BIASED UPWIND WITH FIVE POINT APPROXIMATIONS
C...  AT THE BOUNDARIES
      IF(NORUN.EQ.5)CALL DSS020(0.,XL,N,U,UX,V)
C...
C...  FIVE POINT BIASED UPWIND WITH TWO POINT APPROXIMATIONS
C...  AT THE BOUNDARIES
      IF(NORUN.EQ.6)CALL DSS021(0.,XL,N,U,UX,V)
C...
```

```
C...  LEONARD WITH FOUR POINT APPROXIMATIONS AT THE BOUNDARIES
      IF(NORUN.EQ.7)CALL   LEO1(0.,XL,N,U,UX,V)
C...
C...  LEONARD WITH TWO POINT APPROXIMATIONS AT THE BOUNDARIES
      IF(NORUN.EQ.8)CALL   LEO2(0.,XL,N,U,UX,V)
C...
C...  THREE POINT CENTERED
      IF(NORUN.EQ.9)CALL   DSS002(0.,XL,N,U,UX)
C...
C...  FIVE POINT CENTERED
      IF(NORUN.EQ.10)CALL  DSS004(0.,XL,N,U,UX)
C...
C...  PDE
      DO 1 I=2,N
      UT(I)=-V*UX(I)
1     CONTINUE
      RETURN
      END

      SUBROUTINE PRINT(NI,NO)
      COMMON/T/      T,  NSTOP,  NORUN
     1       /Y/  U(21)
     2       /F/ UT(21)
     3      /SD/ UX(21)
     4       /C/      V,     XL,      N,     DX,    SSE,     IP
C...
C...  DIMENSION THE ARRAYS FOR PLOTTING THE SOLUTION
      DIMENSION TPE(201),TPN(201),UPE(201),UPN(201)
C...
C...  PRINT A HEADING FOR NUMERICAL BSOLUTION
      IF(IP.EQ.0)WRITE(NO,1)
1     FORMAT(9X,1HT,6X,6HU(0,T),6X,6HU(1,T),7X,5HEXACT,11X,1HE)
C...
C...  STORE THE EXACT SOLUTION IN ARRAY UPE FOR PLOTTING
      X=1.0
      TXV=T-X/V
      S=5.0
      DO 3 I=1,NP
C...
C...  TRUNCATED RAMP FUNCTION
      IF(TXV.LT.0.)EXACT=0.
      IF(TXV.GT.(1./S))EXACT=1.
      IF((TXV.GE.0.).AND.(TXV.LE.(1./S)))EXACT=S*TXV
3     CONTINUE
C...
C...  COMPUTE THE FIGURE OF MERIT, SSE
      E=U(N)-EXACT
      SSE=SSE+E**2
C...
```

```
C...  PRINT THE NUMERICAL AND EXACT SOLUTIONS, ERROR
      IF((IP/10*10).EQ.IP)
     1 WRITE(NO,20)T,U(1),U(N),EXACT,E
20    FORMAT(F10.2,4F12.3)
C...
C...  STORE THE NUMERICAL SOLUTION FOR SUBSEQUENT PLOTTING VIA TOP
C...  DRAWER
      IP=IP+1
      UPN(IP)=U(N)
C...
C...  STORE THE EXACT SOLUTION IN ARRAY UPE FOR PLOTTING
      UPE(IP)=EXACT
C...
C...  STORE THE INDEPENDENT VARIABLE FOR SUBSEQUENT PLOTTING VIA TOP
C...  DRAWER
      NP=201
      TPE(IP)=2.0*FLOAT(IP-1)/FLOAT(NP-1)
      TPN(IP)=TPE(IP)
C...
C...  TEST FOR THE END OF A RUN
      IF(IP.LT.NP)RETURN
C...
C...  OPEN FILE FOR TOP DRAWER PLOTTING
         OPEN(4,FILE='T.TOP',STATUS='NEW')
C...
C...  WRITE TOP DRAW FILE FOR PLOTTING
         WRITE(4,17)
17       FORMAT(' SET LIMITS X FROM 0 TO 2 Y FROM -0.5 TO 1.5',/,
     1          ' SET FONT DUPLEX')
         WRITE(4,13)
13       FORMAT(' SET WINDOW X 2 TO 6 Y 2 TO 8')
      WRITE(4,14)(TPN(I),UPN(I),I=1,NP)
14    FORMAT(2F10.3)
      WRITE(4,16)
16    FORMAT(' JOIN 1')
      WRITE(4,14)(TPE(I),UPE(I),I=1,NP)
      WRITE(4,16)
      WRITE(4,18)NORUN+10,SSE
18    FORMAT(
     1' Title 3.5 9.0 "Fig. ',I2,'"'
     2,/,' TITLE LEFT "u(1,t)" '
     3,/,' TITLE BOTTOM "t"'
     4,/,'Title 3.0, 0.75 "SSE = ',F6.3,'"')
      WRITE(4,15)
15    FORMAT(' NEW FRAME')
C...
C...  LEGEND OF RUNS AT THE END
      IF(NORUN.EQ.10)THEN
      WRITE(4,19)
```

```
19    FORMAT(
     1' Title 3.25 9.0 "Truncated Ramp Funtion"',/,
     1' Title 3.25 8.5 "Fig. 11: Two Point Upwind"',/,
     1' Title 3.25 8.0 "Fig. 12: Three Point Upwind"',/,
     1' Title 3.25 7.5 "Fig. 13: Four Point Biased Upwind, 4PB"',/,
     1' Title 3.25 7.0 "Fig. 14: Four Point Biased Upwind, 2PB"',/,
     1' Title 3.25 6.5 "Fig. 15: Five Point Biased Upwind, 5PB"',/,
     1' Title 3.25 6.0 "Fig. 16: Five Point Biased Upwind, 2PB"',/,
     1' Title 3.25 5.5 "Fig. 17: Leonard, 4PB"',/,
     1' Title 3.25 5.0 "Fig. 18: Leonard, 2PB"',/,
     1' Title 3.25 4.5 "Fig. 19: Three Point Centered"',/,
     1' Title 3.25 4.0 "Fig. 20: Five Point Centered"',/,
     1' Title 3.25 3.5 "All solutions are for a 21-point grid"')
      END IF
      RETURN
      END
```

Program 3: Subroutines DERV and PRINT for the Truncated Ramp

The preceding subroutine DERV also requires 10 sets of data for NORUN = 1 to NORUN = 10. Since these are essentially replications of the data in Table 2, they are not presented here. The preceding FORMAT 19 serves as an index for the figures produced by the 10 finite difference approximations, Figures 11 to 20. The SSE values are tabulated below:

11	2.636
12	0.892
13	0.121
14	0.326
15	0.037
16	0.242
17	0.192
18	0.323
19	1.069
20	0.141

Table 3: Summary of the SSE Values for the Ramp Function

As in the case of the unit step function, the five point biased upwind approximations in subroutine

```
1' Title 3.25 8.5 "Fig. 21: Two Point Upwind"',/,
1' Title 3.25 8.0 "Fig. 22: Three Point Upwind"',/,
1' Title 3.25 7.5 "Fig. 23: Four Point Biased Upwind, 4PB"',/,
1' Title 3.25 7.0 "Fig. 24: Four Point Biased Upwind, 2PB"',/,
1' Title 3.25 6.5 "Fig. 25: Five Point Biased Upwind, 5PB"',/,
1' Title 3.25 6.0 "Fig. 26: Five Point Biased Upwind, 2PB"',/,
1' Title 3.25 5.5 "Fig. 27: Leonard, 4PB"',/,
1' Title 3.25 5.0 "Fig. 28: Leonard, 2PB"',/,
1' Title 3.25 4.5 "Fig. 29: Three Point Centered"',/,
1' Title 3.25 4.0 "Fig. 30: Five Point Centered"',/,
1' Title 3.25 3.5 "All solutions are for a 21-point grid"')
```

Program 4: Programming for the Cosine Step

The SSE values for the graphical solutions in Figures 21 to 30 are

21	2.155
22	0.564
23	0.063
24	0.201
25	0.024
26	0.148
27	0.101
28	0.196
29	0.637
30	0.058

Table 4: Summary of the SSE Values for the Cosine Step Function

As before, the five point biased upwind approximations in subroutine DSS020 have the smallest value of SSE, and the numerical solution closely approximates the exact solution (Figure 25).

Next, we consider a cosine step function which has continuous left and right derivatives

$$\begin{aligned} sc(t) &= 0,\ \omega t < 0 \\ sc(t) &= [1 - \cos(\omega t)]/2,\ 0 < \omega t < \pi \\ sc(t) &= 0,\ \omega t > \pi \end{aligned} \tag{27}$$

DSS020 have the lowest value of SSE (Figure 15), and give a good approximation to the exact solution.

Next we consider a cosine step function with a continuous left derivative and a discontinuous right derivative which can be used to investigate the local effect of continuity in the derivative

$$\begin{aligned} c(t) &= 0,\ \omega t < 0 \\ c(t) &= 1 - \cos(\omega t),\ 0 < \omega t < \pi/2 \\ c(t) &= 1,\ \omega t > \pi/2 \end{aligned} \tag{25}$$

The exact solution for this boundary condition function again follows immediately from equation (10) since $f(t) = c(t)$

$$u(x,t) = c(t - x/v)h(t - x/v) \tag{26}$$

The programming of equation (25) as the boundary condition in subroutine DERV is

```
C...
C...  COSINE STEP FUNCTION
      W=5.
      PI=4.*ATAN(1.)
      IF(T.LT.0.)U(1)=0.
      IF((W*T).GT.(PI/2.))U(1)=1.
      IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI/2.)))U(1)=1.-COS(W*T)
      UT(1)=0.
```

and the programming of the exact solution, equation (26) is subroutine PRINT, plus FORMAT 19, which serves as an index for the graphical solutions is

```
      X=1.0
      TXV=T-X/V
      W=5.0
      PI=4.*ATAN(1.)
      DO 3 I=1,NP
C...
C...  COSINE STEP FUNCTION
      IF(TXV.LT.0.)EXACT=0.
      IF((W*TXV).GT.(PI/2.))EXACT=1.
      IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI/2.)))EXACT=1.-COS(W*TXV)
3     CONTINUE


      WRITE(4,19)
19    FORMAT(
     1' Title 3.25 9.0 "Cosine Step Funtion"',/,
```

The exact solution for this boundary condition function again follows immediately from equation (10) since f(t) = sc(t)

$$u(x,t) = sc(t - x/v)h(t - x/v) \tag{28}$$

The programming of equation (27) as the boundary condition in subroutine DERV is

```
C...
C...  SMOOTH COSINE STEP FUNCTION
      W=5.
      PI=4.*ATAN(1.)
      IF(T.LT.0.)U(1)=0.
      IF((W*T).GT.(PI))U(1)=1.
      IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI)))U(1)=(1.-COS(W*T))/2.0
      UT(1)=0.
```

and the programming of the exact solution, equation (28) in subroutine PRINT, plus FORMAT 19, which serves as an index for the graphical solutions is

```
      X=1.0
      TXV=T-X/V
      W=5.0
      PI=4.*ATAN(1.)
      DO 3 I=1,NP
C...
C...  SMOOTH COSINE STEP FUNCTION
      IF(TXV.LT.0.)EXACT=0.
      IF((W*TXV).GT.(PI))EXACT=1.
      IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI)))
     +   EXACT=(1.-COS(W*TXV))/2.
3     CONTINUE


      WRITE(4,19)
19    FORMAT(
     1' Title 3.25 9.0 "Smooth Cosine Step Funtion"',/,
     1' Title 3.25 8.5 "Fig. 31: Two Point Upwind"',/,
     1' Title 3.25 8.0 "Fig. 32: Three Point Upwind"',/,
     1' Title 3.25 7.5 "Fig. 33: Four Point Biased Upwind, 4PB"',/,
     1' Title 3.25 7.0 "Fig. 34: Four Point Biased Upwind, 2PB"',/,
     1' Title 3.25 6.5 "Fig. 35: Five Point Biased Upwind, 5PB"',/,
     1' Title 3.25 6.0 "Fig. 36: Five Point Biased Upwind, 2PB"',/,
     1' Title 3.25 5.5 "Fig. 37: Leonard, 4PB"',/,
     1' Title 3.25 5.0 "Fig. 38: Leonard, 2PB"',/,
     1' Title 3.25 4.5 "Fig. 39: Three Point Centered"',/,
```

```
1' Title 3.25 4.0 "Fig. 40: Five Point Centered"',/,
1' Title 3.25 3.5 "All solutions are for a 21-point grid"')
```

Program 5: Programming for the Smooth Cosine Step

The SSE values for the graphical solutions in Figures 31 to 40 are

31	0.99303
32	0.07744
33	0.00224
34	0.03109
35	0.00049
36	0.02384
37	0.00487
38	0.02851
39	0.04865
40	0.00118

Table 5: Summary of the SSE Values for the Smooth Cosine Step Function

As before, the five point biased upwind approximations in subroutine DSS020 have the smallest value of SSE, and the numerical solution is essentially indistinguishable from the exact solution (Figure 35). Also, as expected, the various finite differences approximate the exact solution more closely as the forcing or boundary condition function becomes smoother. This is an important result since for many applications, the forcing function has the necessary smoothness for the various approximations, with the possible exception of centered differences, to produce accurate numerical solutions. Also, it may be possible in some cases to choose the boundary condition function to be smooth enough to ensure an accurate solution; for example, if only the steady state solution (spatial profile) is required, then the boundary condition function $u(0,t) = f(t)$ can change slowly, with the only requirement that $f(t)$ approach the correct steady state value.

We conclude this section with one more forcing function which again provides a stringent test of the various approximations

$$
\begin{aligned}
p(t) &= 0,\ \omega t < 0 \\
p(t) &= 1 - \cos(\omega t),\ 0 < \omega t < \pi/2 \\
p(t) &= 1 + \cos(\omega t),\ \pi/2 < \omega t < \pi \\
p(t) &= 0,\ \omega t > \pi
\end{aligned}
\tag{29}
$$

The exact solution for this cosine pulse from equation (10) is

$$
u(x,t) = p(t - x/v)h(t - x/v) \tag{30}
$$

The programming of equation (29) as the boundary condition in subroutine DERV is

```
C...
C...   COSINE PULSE FUNCTION
       W=5.
       PI=4.*ATAN(1.)
       IF(T.LT.0.)U(1)=0.
       IF((W*T).GT.(PI))U(1)=0.
       IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI/2.)))U(1)=1.-COS(W*T)
       IF(((W*T).GT.(PI/2.)).AND.((W*T).LE.(PI)))  U(1)=1.+COS(W*T)
       UT(1)=0.
```

and the programming of the exact solution, equation (30) in subroutine PRINT, plus FORMAT 19, which serves as an index for the graphical solutions is

```
       X=1.0
       TXV=T-X/V
       W=5.0
       PI=4.*ATAN(1.)
       DO 3 I=1,NP
C...
C...   COSINE PULSE FUNCTION
       IF(TXV.LT.0.)EXACT=0.
       IF((W*TXV).GT.(PI))EXACT=0.
       IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI/2.)))EXACT=1.-COS(W*TXV)
       IF(((W*TXV).GT.(PI/2.)).AND.((W*TXV).LE.(PI)))  EXACT=1.+COS(W*TXV)
3      CONTINUE

       WRITE(4,19)
19     FORMAT(
      1' Title 3.25 9.0 "Cosine Pulse Function"',/,
      1' Title 3.25 8.5 "Fig. 41: Two Point Upwind"',/,
      1' Title 3.25 8.0 "Fig. 42: Three Point Upwind"',/,
```

```
1' Title 3.25 7.5 "Fig. 43: Four Point Biased Upwind, 4PB"',/,
1' Title 3.25 7.0 "Fig. 44: Four Point Biased Upwind, 2PB"',/,
1' Title 3.25 6.5 "Fig. 45: Five Point Biased Upwind, 5PB"',/,
1' Title 3.25 6.0 "Fig. 46: Five Point Biased Upwind, 2PB"',/,
1' Title 3.25 5.5 "Fig. 47: Leonard, 4PB"',/,
1' Title 3.25 5.0 "Fig. 48: Leonard, 2PB"',/,
1' Title 3.25 4.5 "Fig. 49: Three Point Centered"',/,
1' Title 3.25 4.0 "Fig. 50: Five Point Centered"',/,
1' Title 3.25 3.5 "All solutions are for a 21-point grid"')
```

Program 6: Programming for the Cosine Pulse

The SSE values for the graphical solutions in Figures 41 to 50 are

41	4.645
42	1.947
43	0.244
44	0.680
45	0.094
46	0.498
47	0.384
48	0.671
49	2.335
50	0.235

Table 6: Summary of the SSE Values for the Cosine Pulse Function

As before, the five point biased upwind approximations in subroutine DSS020 have the smallest value of SSE by a substantial margin (Figure 45). Also, we see from these results that using two point approximations at the boundaries rather than approximations of the same order as at the interior points has an obvious negative effect on the accuracy of the numerical solution, e.g., compare the SSE values for Figures 45 and 46.

In conclusion, based on the results from the five test functions, we recommend the use of the five point biased upwind finite differences in subroutine DSS020 for problems which have strongly convective characteristics. Of course, another way to improve the accuracy of numerical solutions is to

increase the number of grid points. Experience has indicated that 21 points is minimal for forcing or

boundary conditions functions which change rapidly with time. Thus, the rule-of-thumb that only a small number of grid points or cells is required to obtain accurate solutions is generally not valid; even with very smooth or slowly varying boundary condition functions, the spatial discretization errors could be large if only a small number of grid points is used, and these errors would appear even at steady state.

(6) Numerical Solutions to the Advection Equation by Finite Elements

So far, we have considered approximate numerical solutions computed by finite differences. We now consider two other general approaches to numerical solutions: *finite elements* and in the next section, *finite volumes*. The problem to be considered is again equations (7) to (9). The general approach in finite elements is to assume a trial solution consisting (at least in part) of selected approximating functions, then attempt to complete the solution so as to minimize its error. In the present case of equations (7) to (9), we assume a *separated solution* (a solution with the x and t dependency in separate functions) of the form

$$u(x,t) \approx \sum_{i=1}^{N} c_i(t)\phi_i(x) \tag{31}$$

Note that this solution is separated since the $c_i(t)$ depend only on t and the $\phi_i(x)$, which are termed the *basis functions*, depend only on x. Equation (31) is then based on the assumption that an adequate numerical solution can be computed as a sum or superposition of N of the products of these two functions.

Next, we choose the *basis or shape functions*, $\phi_i(x)$, somewhat arbitrarily (or perhaps more accurately, based on knowledge of the problem and experience with the finite element method). In the present case, we make the simplest choice, *linear* or *triangular finite elements*, also termed *hat*, *roof* or *chapeau functions*, as depicted in Figure 6.1. Note that $\phi_i(x)$ is centered at $x = x_i$, and consists of two linear segments which intersect the x axis at $x_i \pm \Delta x$ ($= x_{i+1}$ and x_{i-1}, respectively); also $\phi_i(x_i) = 1$. These linear finite elements make the analysis required to complete the trial solution, equation (31), relatively easy since they are easily differentiated and integrated.

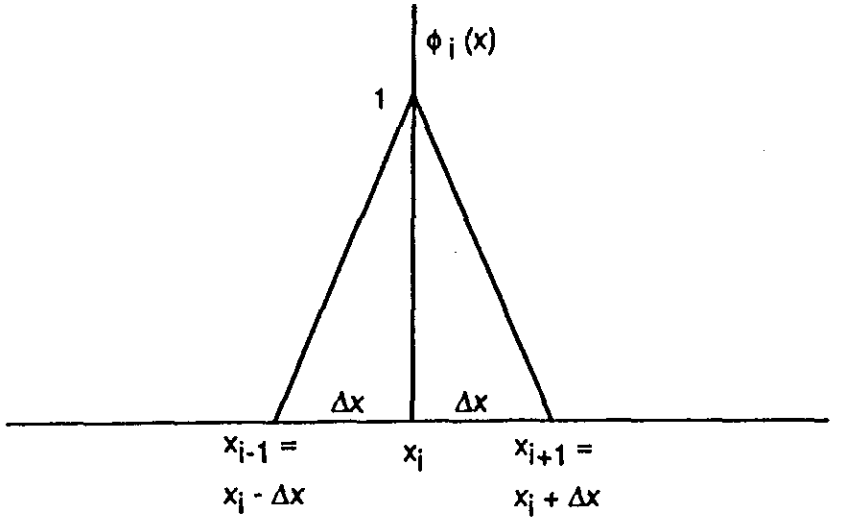


Figure 6.1: A Linear Finite Element

Also, they have the property of *local support* since, for any given x_i, only three terms in the sum of equation (31) need be considered which is apparent from a consideration of Figure 6.1 (only the functions ϕ_{i-1}, ϕ_i and ϕ_{i+1} are nonzero in the interval $x_i \pm \Delta x$).

We now substitute equation (31) in equation (7) (where we see a major advantage of the separated form of equation (31), that is, the partial derivatives in equation (7) are easily computed)

$$u_t(x,t) \approx \sum_{i=1}^{N} c_i'(t)\phi_i(x) \tag{32}$$

$$u_x(x,t) \approx \sum_{i=1}^{N} c_i(t)\phi_i'(x) \tag{33}$$

where ($'$) denotes differentiation (and, again, there is no confusion about this differentiation since $c_i(t)$ and $\phi_i(x)$ are functions of only one variable). Substitution of equations (32) and (33) in equation (7) gives

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x) \approx -v\sum_{i=1}^{N} c_i(t)\phi_i'(x) \tag{34}$$

Equation (31) will probably, at best, be only an approximate solution, and therefore equation (34) will not be truly satisfied. Thus we arrange it as

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x) + v\sum_{i=1}^{N} c_i(t)\phi_i'(x) = R(x,t) \tag{35}$$

where R(x,t) is a *residual*, which ideally is zero for all x and t (if we could find the exact solution to equation (7)). Since equation (31) will only be approximate, we attempt to minimize R(x,t) in some fashion. Generally, this is done by forming the integral

$$\int_{x_1}^{x_N} w(x)R(x,t)dx = 0 \tag{36}$$

where x_1 and x_N are the left and right boundary values of x, respectively, and w(x) is a *weighting function* selected by the analyst. Thus the use of equation (36) is called the *method of weighted residuals* (as suggested by the integrand in equation (36)). Basically, we are *making the residual, $R(x,t)$, orthogonal to the weighting function, $w(t)$.*

The solution resulting from the use of equation (36) is said to be of the *weak form* since it does not satisfy equation (7) for all x and t. Several well known choices for w(x) can be considered. We consider here just one, $w(x) = \delta(x - x_i)$ (the *unit impulse* or *Dirac delta function*), for which equation (36) becomes

$$\int_{x_1}^{x_N} \delta(x - x_i)R(x,t)dx = R(x_i,t) = 0 \tag{37}$$

where we have made use of the *sifting property* of the delta function,

$$\int \delta(x)f(x) = f(0) \tag{38}$$

Equation (37) indicates that we merely *set the residual to zero at the x_i*, which is the *collocation method*; the x_i i = 1, 2, . . ., N, are called the *collocation points.* It is also a *strong form* of the solution at these collocation points since *the residuals are zero at these points*, or in other words, equation (7) is *satisfied exactly at the collocation points.*

We apply the collocation method, equation (37), to equation (35) at x_j, j = 1, 2, . . ., N (the N collocation points), by multiplying equation (35) by $\delta(x - x_j)$ and integrating from x_1 to x_N

$$\int_{x_1}^{x_N} \delta(x - x_j)\sum_{i=1}^{N} c_i'(t)\phi_i(x)dx + v\int_{x_1}^{x_N} \delta(x - x_j)\sum_{i=1}^{N} c_i(t)\phi_i'(x)dx = \int_{x_1}^{x_N} \delta(x - x_j)R(x,t)dx = 0$$

or

$$\sum_{i=1}^{N} c_i'(t)\int_{x_1}^{x_N} \delta(x - x_j)\phi_i(x)dx + v\sum_{i=1}^{N} c_i(t)\int_{x_1}^{x_N} \delta(x - x_j)\phi_i'(x)dx = \int_{x_1}^{x_N} \delta(x - x_j)R(x,t)dx = 0 \qquad (39)$$

where we have interchanged the order of integration and summation (which can be done for finite sums). Then, making use of equation (37),

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x_j) + v\sum_{i=1}^{N} c_i(t)\phi_i'(x_j) = 0 \qquad (40)$$

From the properties of $\phi_i(x)$ in Figure 6.1, i.e.,

$$\phi_i(x_j) \begin{cases} = 1, \ i = j \\ = 0, \ i \neq j \end{cases}$$

we have

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x_j) = c_i'(t) \qquad (41)$$

i.e., there is only one nonzero term in this sum corresponding to $i = j$.

However, we encounter some difficulties with the sum

$$\sum_{i=1}^{N} c_i(t)\phi_i'(x_j) \qquad (42)$$

since $\phi_i'(x_j)$ is discontinuous at $i = j - 1$, j and $j + 1$ (v. Figure 6.1); in other words, it's not clear what values we should use for $\phi_i'(x_j)$ in equation (42) (this can perhaps be better understood by selecting specific values of i and j as examples and sketching $\phi_i'(x_j)$, say $j = 4$, $i = 3$, 4 and 5 or $i = 4$, $j = 3$, 4 and 5). We therefore conclude that *collocation cannot be applied to the solution of equation* (7) *with the linear finite elements of Figure* 6.1. Higher order finite elements, e.g., quadratics or cubics, with continuous first derivatives would circumvent this problem; B-splines [Strang, (5)] are widely used basis functions which meet this condition of continuity in various derivatives.

However, we will continue with the linear finite elements of Figure 6.1, and use a different weighting function, w(x), in equation (36). Specifically, we use the *basis functions as the weighting functions*, which is the *Galerkin method*

$$\int_{x_1}^{x_N} \phi_j(x)R(x,t)dx = 0 \qquad (43)$$

In other words, we make the *residual, $R(x,t)$, orthogonal to the basis functions.* Multiplying equation (35) by $c_j(x)$ and integrating from x_1 to x_N, we have

$$\int_{x_1}^{x_N} \phi_j(x)\sum_{i=1}^{N} c_i'(t)\phi_i(x)dx + v\int_{x_1}^{x_N} \phi_j(x)\sum_{i=1}^{N} c_i(t)\phi_i'(x)dx = \int_{x_1}^{x_N} \phi_j(x)R(x,t)dx = 0$$

or

$$\sum_{i=1}^{N} c_i'(t)\int_{x_1}^{x_N} \phi_j(x)\phi_i(x)dx + v\sum_{i=1}^{N} c_i(t)\int_{x_1}^{x_N} \phi_j(x)\phi_i'(x)dx = \int_{x_1}^{x_N} \phi_j(x)R(x,t)dx = 0 \qquad (44)$$

where we have interchanged the order of integration and summation.

We now must consider three cases for each of the two LHS integrals in equation (44). For the first integral

(1) $\displaystyle\int_{x_1}^{x_N} \phi_j(x)\phi_i(x)dx$

(1.1) $i = j$

$$\int_{x_1}^{x_N} \phi_i(x)\phi_i(x)dx = \int_{x_{i-1}}^{x_i} \left\{\frac{x - x_{i-1}}{x_i - x_{i-1}}\right\}^2 dx + \int_{x_i}^{x_{i+1}} \left\{\frac{x_{i+1} - x}{x_{i+1} - x_i}\right\}^2 dx$$

$$= \frac{1}{3\Delta x^2}(x - x_{i-1})^3 \Big|_{x_{i-1}}^{x_i} - \frac{1}{3\Delta x^2}(x_{i+1} - x)^3 \Big|_{x_i}^{x_{i+1}} = (2/3)\Delta x$$

(1.2) $i = j+1$ (or $j = i-1$)

$$\int_{x_1}^{x_N} \phi_i(x)\phi_{i+1}(x)dx = \int_{x_i}^{x_{i+1}} \left\{\frac{x_{i+1} - x}{x_{i+1} - x_i}\right\}\left\{\frac{x - x_i}{x_{i+1} - x_i}\right\}dx = -\int_{x_i}^{x_{i+1}} \left\{\frac{x - x_{i+1}}{x_{i+1} - x_i}\right\}\left\{\frac{x - x_i}{x_{i+1} - x_i}\right\}dx$$

Using integration by parts, we have

$$-\int_{x_i}^{x_{i+1}} \left\{\frac{x - x_{i+1}}{x_{i+1} - x_i}\right\}\left\{\frac{x - x_i}{x_{i+1} - x_i}\right\}dx = \frac{-1}{\Delta x^2}\left\{(x - x_{i+1})(1/2)(x - x_i)^2 \Big|_{x_i}^{x_{i+1}} - (1/2)\int_{x_i}^{x_{i+1}} (x - x_i)^2 dx\right\}$$

$$= \frac{1}{\Delta x^2}\left\{(1/6)(x - x_i)^3 \Big|_{x_i}^{x_{i+1}}\right\} = (1/6)\Delta x$$

so

$$\int_{x_1}^{x_N} \phi_i(x)\phi_{i+1}(x)dx = (1/6)\Delta x$$

(1.3) $i = j\text{-}1$ (or $j = i+1$)

$$\int_{x_1}^{x_N} \phi_i(x)\phi_{i-1}(x)dx = \int_{x_{i-1}}^{x_i} \left\{\frac{x - x_{i-1}}{x_i - x_{i-1}}\right\}\left\{\frac{x_i - x}{x_i - x_{i-1}}\right\}dx = -\int_{x_{i-1}}^{x_i} \left\{\frac{x - x_{i-1}}{x_i - x_{i-1}}\right\}\left\{\frac{x - x_i}{x_i - x_{i-1}}\right\}dx$$

$$= (1/6)\Delta x$$

which follows from the preceding case with $i+1 \rightarrow i$.

(2) $\displaystyle\int_{x_1}^{x_N} \phi_j(x)\phi_i'(x)dx$

(2.1) $i = j$

$$\int_{x_1}^{x_N} \phi_i(x)\phi_i'(x)dx = \int_{x_{i-1}}^{x_i} \left\{\frac{x - x_{i-1}}{x_i - x_{i-1}}\right\}\left(\frac{1}{\Delta x}\right)dx + \int_{x_i}^{x_{i+1}} \left\{\frac{x_{i+1} - x}{x_{i+1} - x_i}\right\}\left(\frac{-1}{\Delta x}\right)dx$$

$$= \left(\frac{1}{2\Delta x^2}\right)(x - x_{i-1})^2 \Big|_{x_{i-1}}^{x_i} + \left(\frac{1}{2\Delta x^2}\right)(x_{i+1} - x)^2 \Big|_{x_i}^{x_{i+1}} = \frac{1}{2} - \frac{1}{2} = 0$$

(2.2) $i = j+1$ (or $j = i-1$)

$$\int_{x_1}^{x_N} \phi_i(x)\phi_{i+1}'(x)dx = \int_{x_i}^{x_{i+1}} \left\{\frac{x_{i+1} - x}{x_{i+1} - x_i}\right\}\left(\frac{1}{\Delta x}\right)dx = \left(\frac{-1}{2\Delta x^2}\right)(x_{i+1} - x)^2 \Big|_{x_i}^{x_{i+1}} = \frac{1}{2}$$

(2.3) $i = j-1$ (or $j = i+1$)

$$\int_{x_1}^{x_N} \phi_i(x)\phi_{i-1}'(x)dx = \int_{x_{i-1}}^{x_i} \left\{\frac{x - x_{i-1}}{x_i - x_{i-1}}\right\}\left(\frac{-1}{\Delta x}\right)dx = \left(\frac{-1}{2\Delta x^2}\right)(x - x_{i-1})^2 \Big|_{x_{i-1}}^{x_i} = -\frac{1}{2}$$

Thus, equation (44) becomes

$$(\Delta x/6)c_{i+1}' + (4\Delta x/6)c_i' + (\Delta x/6)c_{i-1}' = -v\left\{(1/2)c_{i+1} - (1/2)c_{i-1}\right\}$$

or

$$(1/6)c_{i-1}' + (4/6)c_i' + (1/6)c_{i+1}' = -v\left\{\frac{c_{i+1} - c_{i-1}}{2\Delta x}\right\} \tag{45}$$

Equation (45) is the Galerkin approximation to the advection equation (7) based on the linear finite element of Figure 6.1. It has the following properties:

(1) The LHS is a linear combination of three time derivatives, c_{i+1}', c_i', c_{i-1}' ($= dc_{i+1}/dt$, dc_i/dt, dc_{i-1}/dt) so that equation (45) is *linearly implicit*; in other words, it is not possible to explicitly compute each time derivative as in preceding approximations, and therefore library routines such as RKF45, which are for *explicit ODEs* cannot be used directly. We have essentially two choices: (a) apply a linear equation solver to equation (45) to obtain c_{i+1}', c_i'

and c_{i-1}' explicitly which can then be sent to a solver like RKF45, or (b) use an ODE integrator designed specifically for implicit ODEs, such as LSODI or DASSL. We will use the former in the subsequent coding so that we can retain the use of RKF45.

(2) The RHS of equation (45) is just the usual two point, centered approximation of the derivative in u_x in equation (7).

In summary, *weighted residual methods such as Galerkin's method generally lead to implicit ODEs* such as equation (45).

In order to complete the Galerkin formulation of equation (7), we must also include the initial and boundary conditions, equations (8) and (9). For the initial condition, we take $g(x) = 0$ as in the preceding examples, in which case $c_i(t) = 0$, $i = 1, 2, \ldots, N$ in equation (31) so that $u(x,0) = 0$. For the boundary condition, we have from equation (31)

$$u(0,t) = f(t) = c_1(t)\phi_1(0) + c_2(t)\phi_2(0) + \ldots + c_N(t)\phi_N(0)$$

But, we see from Figure 6.1 that $\phi_1(0) = 1$ and $\phi_i(0) = 0$, $i = 2, 3, \ldots, N$ so that

$$c_1(t) = f(t) \tag{46}$$

However, this result is rather specific to the linear finite element in Figure 6.1. We shall therefore also consider the integrals at $x = x_1$ which lead to this result to demonstrate how we can use finite elements other than the linear finite element in Figure 6. These integrals result from applying equation (44) with $\phi_j(x) = \phi_1(x)$, and neglecting any integrations for $x < x_1$.

$$\sum_{i=1}^{N} c_i'(t)\int_{x_1}^{x_N} \phi_1(x)\phi_i(x)dx + v\sum_{i=1}^{N} c_i(t)\int_{x_1}^{x_N} \phi_1(x)\phi_i'(x)dx = \int_{x_1}^{x_N} \phi_1(x)R(x,t)dx = 0 \tag{47}$$

Because of the properties of the linear finite element in Figure 6.1, equation (47) becomes

$$c_1'(t)\int_{x_1}^{x_2} \phi_1(x)\phi_1(x)dx + c_2'(t)\int_{x_1}^{x_2} \phi_1(x)\phi_2(x)dx$$

$$+ vc_1(t)\int_{x_1}^{x_2} \phi_1(x)\phi_1'(x)dx + vc_2(t)\int_{x_1}^{x_2} \phi_1(x)\phi_2'(x)dx = 0 \tag{48}$$

Thus, we have four integrals to evaluate in equation (48).

Also, we are primarily interested in $u(x_N,t)$ for which we require an equation. If in equation (44) we take $\phi_j(x) = \phi_N(x)$ and neglect any integrations for $x > x_N$, we again arrive at four integrals resulting from the application of Galerkin's method

$$\sum_{i=1}^{N} c_i'(t)\int_{x_1}^{x_N} \phi_N(x)\phi_i(x)dx + v\sum_{i=1}^{N} c_i(t)\int_{x_1}^{x_N} \phi_N(x)\phi_i'(x)dx = \int_{x_1}^{x_N} \phi_N(x)R(x,t)dx = 0 \tag{49}$$

which, because of the properties of the linear finite element in Figure 6.1, becomes

$$c_{N-1}'(t)\int_{x_{N-1}}^{x_N} \phi_N(x)\phi_{N-1}(x)dx + c_N'(t)\int_{x_{N-1}}^{x_N} \phi_N(x)\phi_N(x)dx$$

$$+ vc_{N-1}(t)\int_{x_{N-1}}^{x_N} \phi_N(x)\phi_{N-1}'(x)dx + vc_N(t)\int_{x_{N-1}}^{x_N} \phi_N(x)\phi_N'(x)dx = 0 \tag{50}$$

Thus, we again have four integrals to evaluate.

(3) $\displaystyle\int_{x_{N-1}}^{x_N} \phi_j(x)\phi_i(x)dx$

(3.1) $i = j = 1$

$$\int_{x_1}^{x_2} \phi_1(x)\phi_1(x)dx = \int_{x_1}^{x_2} \left\{\frac{x_2 - x}{x_2 - x_1}\right\}^2 dx = \frac{1}{3\Delta x^2}(x_2 - x)^3 \Big|_{x_1}^{x_2} = (1/3)\Delta x$$

which is 1/2 of the result from integral (1.1) as expected.

(3.2) $i = 2, j = 1$

$$\int_{x_1}^{x_2} \phi_1(x)\phi_2(x)dx = \int_{x_1}^{x_2} \left\{\frac{x_2 - x}{x_2 - x_1}\right\}\left\{\frac{x - x_1}{x_2 - x_1}\right\}dx = -\int_{x_1}^{x_2} \left\{\frac{x - x_2}{x_2 - x_1}\right\}\left\{\frac{x - x_1}{x_2 - x_1}\right\}dx$$

Using integration by parts, we have

$$-\int_{x_1}^{x_2} \left\{\frac{x - x_2}{x_2 - x_1}\right\}\left\{\frac{x - x_1}{x_2 - x_1}\right\}dx = \frac{-1}{\Delta x^2}\left\{(x - x_2)(1/2)(x - x_1)^2 \Big|_{x_1}^{x_2} - (1/2)\int_{x_1}^{x_2} (x - x_1)^2 dx\right\}$$

$$= \frac{1}{\Delta x^2}\left\{(1/6)(x - x_1)^3 \Big|_{x_1}^{x_2}\right\} = (1/6)\Delta x$$

which agrees with integral (1.2).

(3.3) $i = j = N$

$$\int_{x_{N-1}}^{x_N} \phi_N(x)\phi_N(x)dx = \int_{x_{N-1}}^{x_N} \left\{\frac{x - x_{N-1}}{x_N - x_{N-1}}\right\}^2 dx = \frac{1}{3\Delta x^2}(x - x_{N-1})^3 \Big|_{x_{N-1}}^{x_N} = (1/3)\Delta x$$

which is 1/2 of the result from integral (1.1) as expected.

(3.4) $i = N\text{-}1, j = N$

$$\int_{x_{N-1}}^{x_N} \phi_N(x)\phi_{N-1}(x)dx = \int_{x_{N-1}}^{x_N} \left\{\frac{x - x_{N-1}}{x_N - x_{N-1}}\right\}\left\{\frac{x_N - x}{x_N - x_{N-1}}\right\}dx = -\int_{x_{N-1}}^{x_N} \left\{\frac{x - x_{N-1}}{x_N - x_{N-1}}\right\}\left\{\frac{x - x_N}{x_N - x_{N-1}}\right\}dx$$

Using integration by parts, we have

$$-\int_{x_{N-1}}^{x_N} \left\{\frac{x - x_{N-1}}{x_N - x_{N-1}}\right\}\left\{\frac{x - x_N}{x_N - x_{N-1}}\right\}dx = \frac{-1}{\Delta x^2}\left\{(x - x_{N-1})(1/2)(x - x_N)^2 \Big|_{x_{N-1}}^{x_N} - (1/2)\int_{x_{N-1}}^{x_N} (x - x_N)^2 dx\right\}$$

$$= \frac{1}{\Delta x^2}\left\{(1/6)(x - x_N)^3 \Big|_{x_{N-1}}^{x_N}\right\} = (1/6)\Delta x$$

which agrees with integral (1.3).

(4) $\displaystyle\int_{x_{N-1}}^{x_N} \phi_j(x)\phi_i{}'(x)dx$

(4.1) $i = j = 1$

$$\int_{x_1}^{x_2} \phi_1(x)\phi_1'(x)dx = \int_{x_1}^{x_2} \left\{\frac{x_2 - x}{x_2 - x_1}\right\}\left(\frac{-1}{\Delta x}\right)dx = \left(\frac{1}{2\Delta x^2}\right)(x_2 - x)^2 \Big|_{x_1}^{x_2} = -\frac{1}{2}$$

which agrees with the first part of integral (2.1) as expected.

(4.2) $i = 2, j = 1$

$$\int_{x_1}^{x_2} \phi_1(x)\phi_2'(x)dx = \int_{x_1}^{x_2} \left\{\frac{x_2 - x}{x_2 - x_1}\right\}\left(\frac{1}{\Delta x}\right)dx = \left(\frac{-1}{2\Delta x^2}\right)(x_2 - x)^2 \Big|_{x_1}^{x_2} = \frac{1}{2}$$

which agrees with integral (2.2).

(4.3) $i = j = N$

$$\int_{x_{N-1}}^{x_N} \phi_N(x)\phi_N'(x)dx = \int_{x_{N-1}}^{x_N} \left\{\frac{x - x_{N-1}}{x_N - x_{N-1}}\right\}\left(\frac{1}{\Delta x}\right)dx = \left(\frac{1}{2\Delta x^2}\right)(x - x_{N-1})^2 \Big|_{x_{N-1}}^{x_N} = \frac{1}{2}$$

which agrees with the first part of integral (2.1).

(4.4) $i = N\text{-}1, j = N$

$$\int_{x_{N-1}}^{x_N} \phi_N(x)\phi_{N-1}'(x)dx = \int_{x_{N-1}}^{x_N} \left\{\frac{x - x_{N-1}}{x_N - x_{N-1}}\right\}\left(\frac{-1}{\Delta x}\right)dx = \left(\frac{-1}{2\Delta x^2}\right)(x - x_{N-1})^2 \Big|_{x_{N-1}}^{x_N} = -\frac{1}{2}$$

which agrees with integral (2.3).

Thus, for $x = x_1$, we have in place of equation (45)

$$(2/6)c_1' + (1/6)c_2' = -v\left\{\frac{c_2 - c_1}{2\Delta x}\right\}$$

or

$$(4/6)c_1' + (2/6)c_2' = -v\left\{\frac{c_2 - c_1}{\Delta x}\right\} \tag{51}$$

Also, for $x = x_N$, we have in place of equation (45)

$$(1/6)c_{N-1}' + (2/6)c_N' = -v\left\{\frac{c_N - c_{N-1}}{2\Delta x}\right\}$$

or

$$(2/6)c_{N-1}' + (4/6)c_N' = -v\left\{\frac{c_N - c_{N-1}}{\Delta x}\right\} \tag{52}$$

We can now summarize the entire system of ODEs from equations (45), (51) and (52)

$$
\begin{aligned}
(4/6)c_1' + (2/6)c_2' &= -v\left\{\frac{c_2 - c_1}{\Delta x}\right\} \\
(1/6)c_1' + (4/6)c_2' + (1/6)c_3' &= -v\left\{\frac{c_3 - c_1}{2\Delta x}\right\} \\
(1/6)c_2' + (4/6)c_3' + (1/6)c_4' &= -v\left\{\frac{c_4 - c_2}{2\Delta x}\right\} \\
&\vdots \\
(1/6)c_{N-3}' + (4/6)c_{N-2}' + (1/6)c_{N-1}' &= -v\left\{\frac{c_{N-1} - c_{N-3}}{2\Delta x}\right\} \\
(1/6)c_{N-2}' + (4/6)c_{N-1}' + (1/6)c_N' &= -v\left\{\frac{c_N - c_{N-2}}{2\Delta x}\right\} \\
(2/6)c_{N-1}' + (4/6)c_N' &= -v\left\{\frac{c_N - c_{N-1}}{\Delta x}\right\}
\end{aligned}
\tag{53}
$$

We have equation from (46), $c_1 = f(t)$, and it follows that $c_1' = f'(t)$. However, since c_1' is then integrated by the ODE solver to produce c_1, we could equivalently set $c_1 = f(t)$ and use $c_1' = 0$ as the first equation in equations (53), i.e., use the boundary condition $c_1 = f(t)$ to set c_1 rather than use the ODE integrator to compute c_1. However, this approach is rather specific to the linear finite element in Figure 6.1, and therefore we will use the more general approach of equations (53) in the subsequent programming.

We note that the LHS weighting coefficients of equations (53) form a tridiagonal matrix. Therefore, we can invert the LHS using a tridiagonal solver to obtain c_1', c_2', . . . , c_{N-1}', c_N' explictly, then send this vector of derivatives to an explicit ODE solver, e.g., RKF45. The solution of particular interest is $u(x_N,t)$, which, from equation (31), is

$$u(x_N,t) \approx C_N(t) \tag{54}$$

and $C_N(t)$ is computed by simultaneous solution of equations (50).

The programming of equations (53) in subroutine DERV is listed in Program 7 along with the data files in Table 7 for the five test functions (NORUN = 1 to 5). Program 7 also includes a listing of subroutine INITAL which has been extended to set the tridiagonal coefficient matrix of equations (53). Subroutine PRINT has also been modified slightly to give the correct titles to the plots

```
      SUBROUTINE INITAL
C...
C...  PURE CONVECTION THROUGH A TUBE WITH A UNIFORM VELOCITY PROFILE CAN
C...  BE DEPICTED AS
C...
C...
C...          ..............................................
C...                         V......+.   .
C...                         V......+.   .
C...                         V......+.DX. U(X,T)
C...                         V......+.   .
C...                         V......+.   .
C...          ..............................................
C...        X=0                                            X=XL
C...
C...  IF A MASS OR ENERGY BALANCE IS WRITTEN FOR A DIFFERENTIAL SECTION
C...  OF THE TUBE, THE WELL-KNOWN ADVECTION EQUATION RESULTS
C...
C...     U  + V*U  = 0                                                  (1)
C...      T      X
C...
C...  SINCE EQUATION (1) IS FIRST-ORDER IN THE SPATIAL VARIABLE, X, AND
C...  THE TEMPORAL VARIABLE, T, IT REQUIRES ONE BOUNDARY CONDITION AND
C...  ONE INITIAL CONDITION
C...
C...     U(0,T) = F(T), U(X,0) = G(X)                                 (2)(3)
C...
C...  FOR WHICH THE EXACT SOLUTION TO EQUATIONS (1) TO (3) IS
C...
C...     U(X,T) = 0, X GT V*T
C...                                                                    (4)
C...     U(X,T) = F(T - X/V), X LT V*T
C...
C...  THE SPATIAL DERIVATIVE IN EQUATION (1), U  IS REPLACED BY A FINITE
C...                                          X
C...  ELEMENT APPROXIMATION OVER A GRID OF N (= 21) POINTS IN X.  A
C...  SYSTEM OF N INITIAL-VALUE ORDINARY DIFFERENTIAL EQUATIONS IN T
C...  RESULTS, ONE DIFFERENTIAL EQUATION FOR EACH GRID POINT.  THIS
C...  SYSTEM OF ODES IS THEN INTEGRATED SIMULTANEOUSLY TO OBTAIN THE
C...  NUMERICAL SOLUTION TO EQUATIONS (1) TO (3).
C...
      COMMON/T/      T,  NSTOP,  NORUN
     1       /Y/  U(21)
     2       /F/ UT(21)
     3      /SD/ UX(21)
     4       /C/      V,    SSE,     FT
     5      /FE/     X1,     XN,     DX,
     6  AL(21), BM(21), CU(21),BRHS(21)
```

```
C...  UPPER DIAGONAL
      DO 25 I=1,N
         IF(I.EQ.1)THEN
C...
C...     FIRST ODE NOT USED
C           CU(1)=0.0E0
C...
C...     FIRST ODE USED
            CU(1)=2.0E0/6.0E0
         ELSE
     +   IF(I.EQ.N)THEN
            CU(N)=0.0E0
         ELSE
            CU(I)=1.0E0/6.0E0
         END IF
25    CONTINUE
C...
C...  INITIAL CONDITION (5)
      DO 1 I=1,N
      U(I)=0.
1     CONTINUE
C...
C...  INITIALIZE THE TEMPORAL DERIVATIVES IN COMMON/F/
      CALL DERV
C...
C...  PRINT THE COEFFICIENT MATRIX FOR VERIFICATION
      IF(NORUN.EQ.1)THEN
      DO 27 I=1,N
         WRITE(NO,28)I,AL(I),BM(I),CU(I)
28       FORMAT(I5,3F12.4)
27    CONTINUE
      END IF
      RETURN
      END

      SUBROUTINE DERV
      COMMON/T/       T,   NSTOP,   NORUN
     1       /Y/  U(21)
     2       /F/ UT(21)
     3      /SD/ UX(21)
     4       /C/       V,     SSE,      FT
     5      /FE/      X1,      XN,      DX,
     6  AL(21), BM(21), CU(21),BRHS(21)
     7       /I/      IP,       N
C...
C...  UNIT STEP FUNCTION
      IF(NORUN.EQ.1)THEN
         FT=1.0E0
C...
```

```
     7       /I/      IP,       N
C...
C...  COMMON AREA FOR I/O UNIT NUMBERS
      COMMON/IO/     NI,      NO
C...
C...  SET THE VELOCITY, LENGTH OF THE SYSTEM, NUMBER OF SPATIAL GRID
C...  POINTS, LENGTH OF THE GRID INTERVAL, SUM OF SQUARES OF ERRORS OF
C...  THE NUMERICAL SOLUTION, COUNTER FOR THE PLOTTED SOLUTION USED IN
C...  SUBROUTINE PRINT
      V=1.
      X1=0.
      XN=1.
      N=21
      DX=(XN-X1)/FLOAT(N-1)
      SSE=0.
      IP=0
C...
C...  SET UP THE COEFFICIENT MATRIX IN BAND STORAGE MODE. THIS IS
C...  DONE ONLY ONCE SINCE THE COEFFICIENT MATRIX IS CONSTANT
C...
C...  LOWER DIAGONAL
      DO 23 I=1,N
         IF(I.EQ.1)THEN
            AL(1)=0.0E0
         ELSE
     +   IF(I.EQ.N)THEN
            AL(N)=2.0E0/6.0E0
         ELSE
            AL(I)=1.0E0/6.0E0
         END IF
23    CONTINUE
C...
C...  MAIN DIAGONAL
      DO 24 I=1,N
         IF(I.EQ.1)THEN
C...
C...     FIRST ODE NOT USED
C           BM(1)=1.0E0
C...
C...     FIRST ODE USED
            BM(1)=4.0E0/6.0E0
         ELSE
     +   IF(I.EQ.N)THEN
            BM(N)=4.0E0/6.0E0
         ELSE
            BM(I)=4.0E0/6.0E0
         END IF
24    CONTINUE
C...
```

```
C...   TRUNCATED RAMP FUNCTION
       ELSE IF(NORUN.EQ.2)THEN
          S=5.0
          IF(T.LT.0.)FT=0.
          IF(T.GT.(1./S))FT=1.
          IF((T.GE.0.).AND.(T.LE.(1./S)))FT=S*T
C...
C...   COSINE STEP FUNCTION
       ELSE IF(NORUN.EQ.3)THEN
          W=5.
          PI=4.*ATAN(1.)
          IF(T.LT.0.)FT=0.
          IF((W*T).GT.(PI/2.))FT=1.
          IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI/2.)))FT=1.-COS(W*T)
C...
C...   SMOOTH COSINE STEP FUNCTION
       ELSE IF(NORUN.EQ.4)THEN
          W=5.
          PI=4.*ATAN(1.)
          IF(T.LT.0.)FT=0.
          IF((W*T).GT.(PI))FT=1.
          IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI)))
     +       FT=(1.-COS(W*T))/2.0
C...
C...   COSINE PULSE FUNCTION
       ELSE IF(NORUN.EQ.5)THEN
          W=5.
          PI=4.*ATAN(1.)
          IF(T.LT.0.)FT=0.
          IF((W*T).GT.(PI))FT=0.
          IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI/2.)))FT=1.-COS(W*T)
          IF(((W*T).GT.(PI/2.)).AND.((W*T).LE.(PI))) FT=1.+COS(W*T)
       END IF
C...
C...   APPLY BOUNDARY CONDITION (4) AS A CONSTRAINT
       U(1)=FT
C...
C...   RIGHT HAND SIDE VECTOR
       DO 1 I=1,N
          IF(I.EQ.1)THEN
C...
C...      FIRST ODE NOT USED
C            BRHS(1)=0.0E0
C...
C...      FIRST ODE USED
            BRHS(1)=-(V/(1.0E0*DX))*(U(2)-U(1))
          ELSE
     +    IF(I.EQ.N)THEN
            BRHS(N)=-(V/(1.0E0*DX))*(U(N)-U(N-1))
```

```
         ELSE
            BRHS(I)=-(V/(2.0E0*DX))*(U(I+1)-U(I-1))
         END IF
1     CONTINUE
C...
C...  SOLVE THE LINEAR ALGEBRAIC EQUATIONS BY SUBROUTINE TRIDAG, WHICH
C...  RETURNS THE DERIVATIVE VECTOR UT IN COMMON/F/
      CALL TRIDAG(AL,BM,CU,BRHS,UT,N)
      RETURN
      END

      SUBROUTINE PRINT(NI,NO)
      COMMON/T/       T,   NSTOP,   NORUN
     1       /Y/  U(21)
     2       /F/ UT(21)
     3      /SD/ UX(21)
     4       /C/       V,     SSE,      FT
     5      /FE/      X1,      XN,      DX,
     6  AL(21), BM(21), CU(21),BRHS(21)
     7       /I/      IP,       N
C...
C...  DIMENSION THE ARRAYS FOR PLOTTING THE SOLUTION
      DIMENSION TPE(201),TPN(201),UPE(201),UPN(201)
C...
C...  PRINT A HEADING FOR NUMERICAL BSOLUTION
      IF(IP.EQ.0)WRITE(NO,1)
1     FORMAT(9X,1HT,6X,6HU(0,T),6X,6HU(1,T),7X,5HEXACT,11X,1HE)
C...
C...  STORE THE EXACT SOLUTION IN ARRAY UPE FOR PLOTTING
      X=1.0
      TXV=T-X/V
      DO 3 I=1,NP
C...
C...  UNIT STEP FUNCTION
      IF(NORUN.EQ.1)THEN
         IF(TXV.LE.0.)EXACT=0.
         IF(TXV.GT.0.)EXACT=1.0
C...
C...  TRUNCATED RAMP FUNCTION
      ELSE IF(NORUN.EQ.2)THEN
         S=5.0
         IF(TXV.LT.0.)EXACT=0.
         IF(TXV.GT.(1./S))EXACT=1.
         IF((TXV.GE.0.).AND.(TXV.LE.(1./S)))EXACT=S*TXV
C...
C...  COSINE STEP FUNCTION
      ELSE IF(NORUN.EQ.3)THEN
         W=5.0
         PI=4.*ATAN(1.)
```

```
          IF(TXV.LT.0.)EXACT=0.
          IF((W*TXV).GT.(PI/2.))EXACT=1.
          IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI/2.)))
     +       EXACT=1.-COS(W*TXV)
C...
C...   SMOOTH COSINE STEP FUNCTION
       ELSE IF(NORUN.EQ.4)THEN
          W=5.0
          PI=4.*ATAN(1.)
          IF(TXV.LT.0.)EXACT=0.
          IF((W*TXV).GT.(PI))EXACT=1.
          IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI)))
     +       EXACT=(1.-COS(W*TXV))/2.
C...
C...   COSINE PULSE FUNCTION
       ELSE IF(NORUN.EQ.5)THEN
             W=5.0
             PI=4.*ATAN(1.)
             IF(TXV.LT.0.)EXACT=0.
             IF((W*TXV).GT.(PI))EXACT=0.
             IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI/2.)))
     +          EXACT=1.-COS(W*TXV)
             IF(((W*TXV).GT.(PI/2.)).AND.((W*TXV).LE.(PI)))
     +          EXACT=1.+COS(W*TXV)
       END IF
3      CONTINUE
C...
C...   COMPUTE THE FIGURE OF MERIT, SSE
       E=U(N)-EXACT
       SSE=SSE+E**2
C...
C...   PRINT THE NUMERICAL AND EXACT SOLUTIONS, ERROR
       IF((IP/10*10).EQ.IP)
      1 WRITE(NO,20)T,FT,UN,EXACT,E
20     FORMAT(F10.2,4F12.3)
C...
C...   STORE THE NUMERICAL SOLUTION FOR SUBSEQUENT PLOTTING VIA TOP
C...   DRAWER
       IP=IP+1
       UPN(IP)=U(N)
C...
C...   STORE THE EXACT SOLUTION IN ARRAY UPE FOR PLOTTING
       UPE(IP)=EXACT
C...
C...   STORE THE INDEPENDENT VARIABLE FOR SUBSEQUENT PLOTTING VIA TOP
C...   DRAWER
       NP=201
       TPE(IP)=2.0*FLOAT(IP-1)/FLOAT(NP-1)
       TPN(IP)=TPE(IP)
```

```
C...
C...  TEST FOR THE END OF A RUN
      IF(IP.LT.NP)RETURN
C...
C...  OPEN FILE FOR TOP DRAWER PLOTTING
         OPEN(4,FILE='T.TOP',STATUS='NEW')
C...
C...  WRITE TOP DRAW FILE FOR PLOTTING
         WRITE(4,17)
17       FORMAT(' SET LIMITS X FROM 0 TO 2 Y FROM -0.5 TO 1.5',/,
     1          ' SET FONT DUPLEX')
         WRITE(4,13)
13       FORMAT(' SET WINDOW X 2 TO 6 Y 2 TO 8')
      WRITE(4,14)(TPE(I),UPE(I),I=1,NP)
14    FORMAT(2F10.4)
      WRITE(4,16)
16    FORMAT(' JOIN 1')
      WRITE(4,14)(TPN(I),UPN(I),I=1,NP)
      WRITE(4,16)
      WRITE(4,18)NORUN+50,SSE
18    FORMAT(
     1' Title 3.5 9.0 "Fig. ',I2,'"'
     2,/,' TITLE LEFT "u(1,t)" '
     3,/,' TITLE BOTTOM "t"'
     4,/,'Title 3.0, 0.75 "SSE = ',F6.3,'"')
      WRITE(4,15)
15    FORMAT(' NEW FRAME')
C...
C...  LEGEND OF RUNS AT THE END
      IF(NORUN.EQ.5)THEN
      WRITE(4,19)
19    FORMAT(
     1' Title 3.25 9.0 "Galerkin with Linear FInite Elements"',/,
     1' Title 3.25 8.5 "Fig. 51: Heaviside Unit Step Function"',/,
     1' Title 3.25 8.0 "Fig. 52: Truncated Ramp"',/,
     1' Title 3.25 7.5 "Fig. 53: Cosine Step"',/,
     1' Title 3.25 7.0 "Fig. 54: Smooth Cosine Step"',/,
     1' Title 3.25 6.5 "Fig. 55: Cosine Pulse"',/,
     1' Title 3.25 6.0 "All solutions are for a 21-point grid"')
      END IF
      RETURN
      END
```

Program 7: Subroutines DERV and PRINT for the Galerkin

Solution with Linear Finite Elements - Five Test Functions

The following points should be noted about the programming in subroutines DERV and

PRINT:

(1) Tridiagonal equations (53) are programmed in two steps:

(1.1) In INITAL, the subdiagonal is stored in array AL, the main diagonal in array BM, and the superdiagonal in array CU. This is done only once in INITAL since these diagonals are constant.

(1.2) In DERV, after the boundary condition function, f(t), is computed (for NORUN = 1 to 5), the RHS vector of equations (53) is stored in array BRHS.

(2) After the tridiagonal matrix and RHS vector are defined, subroutine TRIDAG [Press, et al, (6)] is called (at the end of DERV) to compute c_1' to c_N' (array UT). Then, in the usual way, UT is sent through COMMON/F/ to RKF45, which returns array U (c_1 to c_N) in COMMON/Y/.

(3) From equation (54), we have $u(x_N,t) = c_N(t)$, which is printed and plotted in PRINT.

The following data in Table 7 are for five runs corresponding to the five test functions:

```
GALERKIN NUMOL SOLUTION OF ADVECTION EQN - UNIT STEP
0.        2.0        0.01
   21                        0.0001
GALERKIN NUMOL SOLUTION OF ADVECTION EQN - TRUNCATED RAMP
0.        2.0        0.01
   21                        0.0001
GALERKIN NUMOL SOLUTION OF ADVECTION EQN - COSINE STEP
0.        2.0        0.01
   21                        0.0001
GALERKIN NUMOL SOLUTION OF ADVECTION EQN - SMOOTH COSINE STEP
0.        2.0        0.01
   21                        0.0001
GALERKIN NUMOL SOLUTION OF ADVECTION EQN - COSINE PULSE
0.        2.0        0.01
   21                        0.0001
END OF RUNS
```

Table 7: Data for Program 7

Main program SRKF45 in Appendix 4 was again used to read the data in Table 7 and call RKF45 (and thus INITAL, DERV and PRINT). The graphical output produced by subroutine PRINT is in Figures 51 to 55. The SSE values for these runs are tabulated in Table 8

51	1.271
52	0.037
53	0.027
54	0.002
55	0.099

Table 8: Summary of the SSE Values for the Galerkin on Linear Finite Element Solutions

Again, we see numerical oscillation, which is probably to be expected because of the centered approximation of u_x in the RHS of equations (53). This centered approximation results from the symmetric form of the linear finite element in Figure 6.1. Therefore, we might expect better results from an upwind finite element, and in fact, we shall explore this possibility next. Generally, however, the linear finite element of Figure 6.1 produced surprisingly good solutions; the SSE values are close to those for the five point, biased upwind approximations in subroutine DSS020, e.g.,

Approximation	f(t)	Figure	SSE
Five point biased upwind	Unit step	5	1.392
Five point biased upwind	Cosine pulse	45	0.094
Linear finite element	Unit step	51	1.271
Linear finite element	Cosine pulse	55	0.099

(7) The Quadratic Upwind/Petrov-Galerkin Finite Element Formulation

We now consider the finite element formulation of the preceding section, with the linear finite element of Figure 6.1 replaced by the Quadratic Upwind/Petrov-Galerkin element (QU/PG) of Figure 7.1. This QU/PG element, which has a general form discussed by Brooks and Hughes (7), clearly weighs the upwind direction (e.g., $x < 0$ for flow in the positive x direction) more heavily

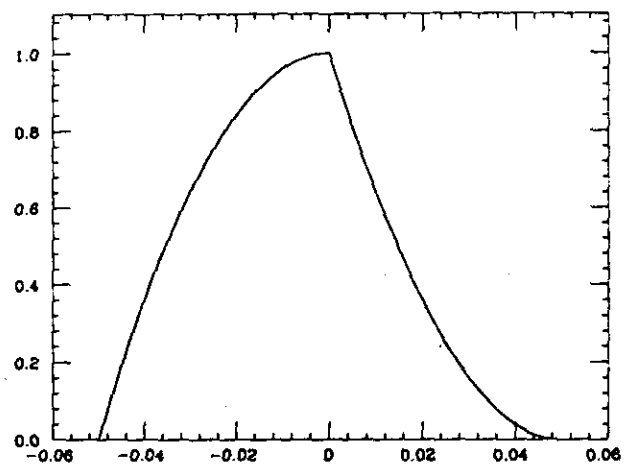


Fig. 7.1: The QU/PG Element

The QU/PG element of Figure 7.1 consists of two sections for $x < 0$ and $x > 0$, each composed of a quadratic of the form

$$\phi_i(x) = a_0 + a_1 x + a_2 x^2 \tag{55}$$

For $x < 0$, the coefficients a_0, a_1 and a_2 are computed so the quadratic passes through the three points

$$\phi_i(0) = 1$$

$$\phi_i(-\Delta x/2) = \phi_L \tag{56}$$

$$\phi_i(-\Delta x) = 0$$

where Δx is the half-width of the element, and ϕ_L is selected by the analyst. Again, as in the case of

the linear finite element of Figure 6.1, we take $\phi_i(x) = 0$, $x < -\Delta x$.

Similarly, for $x > 0$, we choose the three coefficients so the quadratic of equation (55) passes through the three points

$$\phi_i(0) = 1$$

$$\phi_i(\Delta x/2) = \phi_R \tag{57}$$

$$\phi_i(\Delta x) = 0$$

where $\phi_i(x) = 0$, $x > \Delta x$ and ϕ_R is selected by the analyst. $\phi_i(x)$ has the following noteworthy properties:

(1) Continuity at $x = 0$, with the value $\phi_i(0) = 1$.

(2) Any two basis functions overlap at most by the distance Δx.

(3) As a consequence of (1) and (2), the sums in equation (44) are relatively easy to compute, and the solution $u(x_N,t)$ consists of a single term in the series, i.e., $u(x_N,t) = c_N(t)$ (again, $\phi_N(x_N) = 1$ and $\phi_i(x_N) = 0$, $i < N$).

The formulas for the three coefficients a_0, a_1 and a_2 for each of the three sets of points defined by equations (56) and (57) are derived in Appendix 14. In the case of Figure 7.1, $\phi_L = 0.75$, $\phi_R = 0.25$.

As in the case of the linear finite elements of Figure 6.1, we now must evaluate integrals (1.1) to (4.4) to use the element of Figure 7.1 (a total of 14 integrals, although some redundancy is reflected in this figure since several of the integrals follow as special cases of others, but we choose to list and evaluate them in detail to clarify the discussion). The myriad of details for an analytical integration (the integrands involve $\phi_i(x)$ from equation (55) and its first derivative defined on the two sections of the element, $x < 0$ and $x > 0$) precludes an analysis of reasonable length. We therefore use numerical integration via a library quadrature routine, QUANC8 [Forsythe, et al (1)].

The code for the QU/PG element of Figure 7.1 is listed in Appendix 15. Either of two elements can be selected: (a) The linear finite element of Figure 6.1, with the intergals computed

numerically (ITYPE = 1 set in INITAL), and (b) The QU/PG element of Figure 7.1 (ITYPE = 2) with the integrals computed numerically. The code was tested in two ways:

(1) The solution for the linear finite element with the integrals computed numerically (ITYPE = 1) was compared with the solution obtained with the integrals done analytically as considered previously (Program 7).

(2) The solution for the QU/PG element (ITYPE = 2) for the case $\phi_L = \phi_R = 0.5$ was compared with the linear finite element solution obtained with the integrals done analytically as considered previously (Program 7) ($\phi_L = \phi_R = 0.5$ corresponds to the linear finite element).

In both cases, the three solutions were essentially identical. Thus all of the code in the program of Appendix 15 was tested against Program 7 for the case $\phi_L = \phi_R = 0.5$. In other words, only the small portion of the code in functions PHIM, PHIP, PHIDM and PHIDP for $\phi_L \neq 0.5$ and/or $\phi_R \neq 0.5$ was not tested. We mention this since the results for the QU/PG element in general were surprisingly poor; this point is substantiated with a subsequent discussion of the solution for $\phi_L = 0.75$, $\phi_R = 0.25$ (the element in Figure 7.1). In any case, we report exactly how the calculations were done through the listing of the program in Appendix 15.

The QU/PG solutions for $\phi_L = 0.75$, $\phi_R = 0.25$ and the five test functions are given in Figures 56 to 60. These solutions are poor in two respects: (a) numerical oscillation exceeds that of the approximations considered previously, and (b) perhaps more importantly, the solutions have a phase error (time displacement)(see, in particular, Figure 59). We again mention that these solutions were produced by the program in Appendix 15, so the reader can check the details of our calculations for errors. We therefore conclude, that at least for our implementation, the QU/PG element is not effective.

Also, more generally, the finite element approach has the following disadvantages:

(1) Integrals must be evaluated for all of the terms in a PDE problem system. While some of the integrals may already be available (e.g., tabulated), a new PDE problem may require additional integrals, which can be complicated depending on the form of the individual terms in the PDEs.

(2) Boundary conditions may be difficult to implement relative to the finite difference method.

(3) The time dependent ODEs resulting from the application of a weighted residual method are implicitly coupled, and therefore require some additional mathematical programming, e.g., the solution of tridiagonal equations, or an ODE solver designed specifically for implicit ODEs.

However, the finite element method offers significant advantages, particularly in higher dimensions, e.g., for irregular geometries, although recent advances in finite difference methods have offset the advantages of finite element methods to some extent.

(8) Finite Volume Approximation of the Advection Equation

So far, we have considered finite difference and finite element approximations of advection equation (7). We now consider briefly the third principal method of approximating PDEs, the *finite volume* or *control volume* method. Finite volume methods have the principal advantage of conservation, e.g., of mass or energy. However, this property does not necessarily guarantee an accurate solution (conservation might be considered necessary, but not sufficient).

For example, if we consider the finite volume approximation of the advection equation proposed by Patankar [(8), pp 83-85] we observe that it is just the two point upwind approximation which we have already noted suffers from excessive numerical diffusion (recall the solutions from subroutine DSS012 listed in Appendix 2). Other approximations are possible [Lick (9)] which will give better results than the two point upwind approximations, but we do not consider this further.

(9) The Nonhomogeneous Advection Equation

All of the preceding tests of various approximations are based on equations (7) to (9) for five different test functions (different f(t) in boundary condition (9)). We now consider the following extension of equations (7) to (9):

$$u_t + vu_x = c(x,t) \tag{58}$$

$$u(x,0) = f(0),\ u(0,t) = f(t) \tag{59)(60}$$

Note in particular the nonhomogeneous term, c(x,t), in equation (58). The analytical solution to

equations (58) to (60), for the special case $f(t) = 0$, $c(x,t) = g(x)$ is derived in Appendix 16 (with $d(x)$ used in place of $g(x)$). In particular, we shall use the following property of the solution expressed as equation (a.69)

$$\lim_{t \to \infty} u(x,t) = \lim_{s \to 0} s\bar{u}(x,s) = (1/v)\int_0^x g(\lambda)d\lambda \tag{61}$$

i.e., the steady state solution is just the integral of $g(x)$.

Equation (61) is the basis for a test problem first proposed by Leonard (4), and subsequently used by Brooks and Hughes (7). In particular, we consider the nonhomogeneous function

$$g(x) = (-1.5/6)x + 1,\ 0 \le x \le 6$$

$$g(x) = (0.5/2)(x - 8),\ 6 < x \le 8 \tag{62}$$

$$g(x) = 0,\ 8 < x \le 15$$

This $g(x)$ therefore consists of three linear segments.

Program 8 for equations (58) to (62) is similar to the preceding programs, but double precision versions of main program SRKF45 and integrator RKF45 wre used (the double precision coding was used to minimize any errors in the steady state solution when compared with equation (61))

```
      SUBROUTINE INITAL
C...
C...      U   + V*U   = G(X)                                        (1)
C...       T        X
C...
C...      U(X,0) = F(0) = 0, U(0,T) = F(T) = 0                      (2)(3)
C...
C...  G(X) IS REPRESENTED GRAPHICALLY BELOW
C
C                .
C                .(0,1)
C                . .
C          G(X).    .
C                .      .
C                .        .          (8,0)        (15,0)
C             ....................................
C                X=0     X     .     .
```

```
C                                    .(6,-0.5)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (N=16)
      COMMON/T/       T,  NSTOP,  NORUN
     1       /Y/   U(N)
     2       /F/  UT(N)
     3       /S/  UX(N),  XG(N),      XS,  GI(N)
C...
C...  LENGTH
      XS=15.0D0
C...
C...  SPATIAL GRID
      DO 2 I=1,N
      XG(I)=XS*DFLOAT(I-1)/DFLOAT(N-1)
2     CONTINUE
C...
C...  INITIAL CONDITION (T = 0)
      DO 1 I=1,N
      U(I)=F(T)
1     CONTINUE
      RETURN
      END

      SUBROUTINE DERV
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (N=16)
      COMMON/T/       T,  NSTOP,  NORUN
     1       /Y/   U(N)
     2       /F/  UT(N)
     3       /S/  UX(N),  XG(N),      XS,  GI(N)
C...
C...  BOUNDARY CONDITION AT X = 0
      U(1)=F(T)
      UT(1)=0.0D0
C...
C...  DERIVATIVE UX
      V=1.0D0
      IF(NORUN.EQ.1)CALL DSS012(0.0D0,XS,N,U,UX,V)
      IF(NORUN.EQ.2)CALL DSS020(0.0D0,XS,N,U,UX,V)
C...
C...  PDE
      DO 1 I=2,N
         UT(I)=-V*UX(I)+G(I)
1     CONTINUE
      RETURN
      END

      SUBROUTINE PRINT(NI,NO)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (N=16)
      COMMON/T/       T,  NSTOP,  NORUN
     1       /Y/   U(N)
     2       /F/  UT(N)
     3       /S/  UX(N),  XG(N),      XS,  GI(N)
C...
C...  PRINT THE NUMERICAL SOLUTION
      IF(T.LT.49.5D0)THEN
         WRITE(NO,1)
1        FORMAT(//,9X,'T',9X,'X',6X,'U(X,T)',5X,'UT(X,T)',
     +   5X,'UX(X,T)')
         WRITE(NO,2)T,XG(1),U(1),UT(1),UX(1)
2        FORMAT(F10.1,F10.2,3F12.4)
         WRITE(NO,3) (XG(I),U(I),UT(I),UX(I),I=2,N)
3        FORMAT(  10X,F10.2,3F12.4)
C...
C...  AT STEADY STATE, ALSO PRINT THE EXACT SOLUTION
      ELSE
         CALL EXACT
         WRITE(NO,4)
4        FORMAT(//,9X,'T',9X,'X',6X,'U(X,T)',5X,'UT(X,T)',
     +   5X,'UX(X,T)',5X,'EXACT')
         WRITE(NO,5)T,XG(1),U(1),UT(1),UX(1),GI(1)
5        FORMAT(F10.1,F10.2,4F12.4)
         WRITE(NO,6) (XG(I),U(I),UT(I),UX(I),GI(I),I=2,N)
6        FORMAT(  10X,F10.2,4F12.4)
C...
C...  CREATE AN OUTPUT FILE FOR TOP DRAWER PLOTTING
      IF(NORUN.EQ.1)THEN
         OPEN(4,FILE='T.TOP',STATUS='NEW')
C...
C...  WRITE TOP DRAWER FILE FOR PLOTTING NUMERICAL AND EXACT STEADY
C...  STATE SOLUTIONS VS X
C...
C...  EXACT SOLUTION
         WRITE(4,7)
7        FORMAT(' SET LIMITS X FROM 0 TO 15 Y FROM 0 TO 2.5',/,
     1           ' SET FONT DUPLEX')
         WRITE(4,10)(XG(I),GI(I),I=1,N)
         WRITE(4,13)
13       FORMAT(' JOIN')
      END IF
C...
C...  NUMERICAL SOLUTIONS
         IF(NORUN.EQ.1)WRITE(4,8)
8        FORMAT(' SET SYMBOL 1')
         IF(NORUN.EQ.2)WRITE(4,9)
9        FORMAT(' SET SYMBOL 2')
```

```
         WRITE(4,10)(XG(I),U(I),I=1,N)
10       FORMAT(2F10.3)
         WRITE(4,11)
11       FORMAT(' PLOT')
         IF(NORUN.EQ.2)THEN
            WRITE(4,12)
12          FORMAT(
     1      ' Title 5.25 9.5 "Fig. 61: u(x,t) vs x at steady state"'
     2      ,/,' TITLE LEFT " u(x,t)"'
     3      ,/,' TITLE BOTTOM "x"'
     4      ,/,' TITLE 4.5 0.75 '
     5      '"Solid - exact, 1 - 2PUW, 2 - 5PBUW"')
         END IF
      END IF
      RETURN
      END

      DOUBLE PRECISION FUNCTION G(I)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (N=16)
      COMMON/T/      T,  NSTOP,  NORUN
     1       /Y/   U(N)
     2       /F/  UT(N)
     3       /S/  UX(N),  XG(N),     XS,  GI(N)
C...
C...  NONHOMOGENEOUS TERM IN THE ADVECTION EQUATION
      IF((XG(I).GE.0.0D0).AND.(XG(I).LE.6.0D0))THEN
         XL= 0.0D0
         XR= 6.0D0
         GL= 1.0D0
         GR=-0.5D0
      ELSE
     +IF((XG(I).GT.6.0D0).AND.(XG(I).LE.8.0D0))THEN
         XL= 6.0D0
         XR= 8.0D0
         GL=-0.5D0
         GR= 0.0D0
      ELSE
     +IF((XG(I).GT.8.0D0).AND.(XG(I).LE.15.0D0))THEN
         XL= 8.0D0
         XR=15.0D0
         GL= 0.0D0
         GR= 0.0D0
      END IF
C...
C...  SLOPE AND INTERCEPT
         S=(GR-GL)/(XR-XL)
         B=(GL*XR-GR*XL)/(XR-XL)
C...
```

```
C...   FUNCTION G(X)
          X=XS*DFLOAT(I-1)/DFLOAT(N-1)
          G=S*X+B
       RETURN
       END

       DOUBLE PRECISION FUNCTION F(T)
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       F=0.0D0
       RETURN
       END

       SUBROUTINE EXACT
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       PARAMETER (N=16)
       COMMON/T/       T,   NSTOP,   NORUN
      1       /Y/   U(N)
      2       /F/  UT(N)
      3       /S/  UX(N),   XG(N),      XS,   GI(N)
C...
C...   INITIALIZE THE INTEGRATION
          AREA=0.0D0
          DX=XS/DFLOAT(N-1)
C...
C...   CUMULATIVE AREA BY THE TRAPEZOIDAL RULE
       DO 1 I=2,N
          GL=G(I-1)
          GR=G(I  )
          AREA=AREA+0.5D0*(GL+GR)*DX
          GI(I)=AREA
1      CONTINUE
       RETURN
       END
```

Program 8: Subroutines DERV, INITAL and PRINT, and Subordinate Routines for the Solution of Equations (58) to (62)

The data for Program 8 are in Table 9

```
NUMOL SOLUTION OF NONHOMOGENSOUS ADVECTION EQUATION - DSS012
0.        50.0       10.0
   16                       0.0001
NUMOL SOLUTION OF NONHOMOGENEOUS ADVECTION EQUATION - DSS020
0.        50.0       10.0
   16                       0.0001
END OF RUNS
```

Table 9: Data for Program 8

The following points should be noted about Program 8:

(1) Initial condition (59) is set in INITAL via function F(T) (for f(t) = 0).

(2) The spatial derivative, u_x, in equation (58) is calculated by the two point upwind approximations in subroutine DSS012 (NORUN = 1), and the five point biased upwind approximations in subroutine DSS020 (NORUN = 2). Thus, Table 9 has two sets of data.

(3) g(x) in equation (62) is calculated by function G(I) called at the end of subroutine DERV in DO loop 1 for equation (58).

(4) Subroutine PRINT calls subroutine EXACT to compute the steady state solution according to equation (61). The integral in equation (61) is done by the trapezoidal rule, which is exact for linear functions, e.g., equation (62). Subroutine PRINT: (a) prints the transient (time dependent) numerical solution and (b) plots the steady state numerical and exact solutions.

A portion of the numerical solution from subroutine PRINT is listed in Table 10

```
 RUN NO. -    1  NUMOL SOLUTION OF NONHOMOGENEOUS ADVECTION EQUATION
                 - DSS012

INITIAL T -  0.000E+00

  FINAL T -  0.500E+02

  PRINT T -  0.100E+02

NUMBER OF DIFFERENTIAL EQUATIONS -  16

MAXIMUM INTEGRATION ERROR -  0.100E-03
```

T	X	U(X,T)	UT(X,T)	UX(X,T)
0.0	0.00	0.0000	0.0000	0.0000
	1.00	0.0000	0.7500	0.0000
	2.00	0.0000	0.5000	0.0000
	3.00	0.0000	0.2500	0.0000
	4.00	0.0000	0.0000	0.0000

5.00	0.0000	-0.2500	0.0000
6.00	0.0000	-0.5000	0.0000
7.00	0.0000	-0.2500	0.0000
8.00	0.0000	0.0000	0.0000
9.00	0.0000	0.0000	0.0000
10.00	0.0000	0.0000	0.0000
11.00	0.0000	0.0000	0.0000
12.00	0.0000	0.0000	0.0000
13.00	0.0000	0.0000	0.0000
14.00	0.0000	0.0000	0.0000
15.00	0.0000	0.0000	0.0000

T	X	U(X,T)	UT(X,T)	UX(X,T)
10.0	0.00	0.0000	0.0000	0.7500
	1.00	0.7500	0.0000	0.7500
	2.00	1.2496	0.0004	0.4996
	3.00	1.4977	0.0019	0.2481
	4.00	1.4907	0.0069	-0.0069
	5.00	1.2222	0.0185	-0.2685
		.	.	
		.	.	
		.	.	

T	X	U(X,T)	UT(X,T)	UX(X,T)
40.0	0.00	0.0000	0.0000	0.7500
	1.00	0.7500	0.0000	0.7500
	2.00	1.2500	0.0000	0.5000
	3.00	1.5000	0.0000	0.2500
	4.00	1.5000	0.0000	0.0000
	5.00	1.2500	0.0000	-0.2500
	6.00	0.7500	0.0000	-0.5000
	7.00	0.5000	0.0000	-0.2500
	8.00	0.5000	0.0000	0.0000
	9.00	0.5000	0.0000	0.0000
	10.00	0.5000	0.0000	0.0000
	11.00	0.5000	0.0000	0.0000
	12.00	0.5000	0.0000	0.0000
	13.00	0.5000	0.0000	0.0000
	14.00	0.5000	0.0000	0.0000
	15.00	0.5000	0.0000	0.0000

T	X	U(X,T)	UT(X,T)	UX(X,T)	EXACT
50.0	0.00	0.0000	0.0000	0.7500	0.0000
	1.00	0.7500	0.0000	0.7500	0.8750
	2.00	1.2500	0.0000	0.5000	1.5000
	3.00	1.5000	0.0000	0.2500	1.8750
	4.00	1.5000	0.0000	0.0000	2.0000

5.00	1.2500	0.0000	-0.2500	1.8750
6.00	0.7500	0.0000	-0.5000	1.5000
7.00	0.5000	0.0000	-0.2500	1.1250
8.00	0.5000	0.0000	0.0000	1.0000
9.00	0.5000	0.0000	0.0000	1.0000
10.00	0.5000	0.0000	0.0000	1.0000
11.00	0.5000	0.0000	0.0000	1.0000
12.00	0.5000	0.0000	0.0000	1.0000
13.00	0.5000	0.0000	0.0000	1.0000
14.00	0.5000	0.0000	0.0000	1.0000
15.00	0.5000	0.0000	0.0000	1.0000

RUN NO. - 2 NUMOL SOLUTION OF NONHOMOGENEOUS ADVECTION EQUATION - DSS020

INITIAL T - 0.000E+00

FINAL T - 0.500E+02

PRINT T - 0.100E+02

NUMBER OF DIFFERENTIAL EQUATIONS - 16

MAXIMUM INTEGRATION ERROR - 0.100E-03

T	X	U(X,T)	UT(X,T)	UX(X,T)
0.0	0.00	0.0000	0.0000	0.0000
	1.00	0.0000	0.7500	0.0000
	2.00	0.0000	0.5000	0.0000
	3.00	0.0000	0.2500	0.0000
	4.00	0.0000	0.0000	0.0000
	5.00	0.0000	-0.2500	0.0000
	6.00	0.0000	-0.5000	0.0000
	7.00	0.0000	-0.2500	0.0000
	8.00	0.0000	0.0000	0.0000
	9.00	0.0000	0.0000	0.0000
	10.00	0.0000	0.0000	0.0000
	11.00	0.0000	0.0000	0.0000
	12.00	0.0000	0.0000	0.0000
	13.00	0.0000	0.0000	0.0000
	14.00	0.0000	0.0000	0.0000
	15.00	0.0000	0.0000	0.0000

T	X	U(X,T)	UT(X,T)	UX(X,T)
10.0	0.00	0.0000	0.0000	0.9165
	1.00	0.8417	-0.0106	0.7606
	2.00	1.5052	-0.0568	0.5568
	3.00	1.9332	-0.0365	0.2865

	4.00	2.0498	0.0436	-0.0436
	5.00	1.8627	0.0881	-0.3381
		.	.	
		.	.	
		.	.	

T	X	U(X,T)	UT(X,T)	UX(X,T)
40.0	0.00	0.0000	0.0000	1.0071
	1.00	0.8777	0.0000	0.7500
	2.00	1.5023	0.0001	0.4999
	3.00	1.8775	0.0001	0.2499
	4.00	1.9999	-0.0001	0.0001
	5.00	1.8868	-0.0002	-0.2498
	6.00	1.4564	-0.0001	-0.4999
	7.00	1.0927	0.0002	-0.2502
	8.00	0.9885	0.0003	-0.0003
	9.00	0.9758	0.0000	0.0000
	10.00	0.9787	-0.0004	0.0004
	11.00	0.9810	-0.0003	0.0003
	12.00	0.9816	0.0002	-0.0002
	13.00	0.9812	0.0006	-0.0006
	14.00	0.9807	0.0004	-0.0004
	15.00	0.9807	-0.0005	0.0005

T	X	U(X,T)	UT(X,T)	UX(X,T)	EXACT
50.0	0.00	0.0000	0.0000	1.0069	0.0000
	1.00	0.8776	0.0000	0.7500	0.8750
	2.00	1.5023	0.0000	0.5000	1.5000
	3.00	1.8776	0.0000	0.2500	1.8750
	4.00	2.0000	0.0000	0.0000	2.0000
	5.00	1.8868	0.0000	-0.2500	1.8750
	6.00	1.4562	0.0001	-0.5001	1.5000
	7.00	1.0925	0.0000	-0.2500	1.1250
	8.00	0.9886	0.0000	0.0000	1.0000
	9.00	0.9761	-0.0001	0.0001	1.0000
	10.00	0.9789	0.0000	0.0000	1.0000
	11.00	0.9808	0.0001	-0.0001	1.0000
	12.00	0.9811	0.0001	-0.0001	1.0000
	13.00	0.9811	0.0000	0.0000	1.0000
	14.00	0.9812	-0.0002	0.0002	1.0000
	15.00	0.9814	-0.0002	0.0002	1.0000

Table 10: Abbreviated Output from Program 8

The following points can be noted about the output in Table 10:

(1) The solution starts out according to initial condition (59). Also, since this is a uniform (zero) solution and therefore $u_x(x,0) = 0$, the only nonzero term in equation (58) initially is $g(x)$, which appears as u_t in the output, i.e., equation at $t = 0$ is $u_t = g(x)$ (keeping in mind that $u_t(0,t) = 0$ is the boundary condition in subroutine DERV).

(2) The solution reaches steady state by $t = 40$; observe that the derivative u_t is small at $t = 40$, and the solution does not change from $t = 40$ to $t = 50$.

(3) The numerical solution for the two point upwind of u_x in equation (58) reaches the incorrect value of 0.5 at $x = 15$ (rather than 1). This error was previously reported by Leonard (4) and Brooks and Hughes (7).

(4) The solution for the five point biased upwind approximation of u_x reaches a steady state value of 0.9814 at $x = 15$ with an error of 0.0186.

The graphical output from subroutine PRINT is given in Figure 61, which is essentially identical to the plots reported by Leonard (4) and Brooks and Hughes (7).

The solution from Program 8 is for a 16-point grid in x. The output from Program 8 for a 31-point grid is listed below and in Figure 62

```
 RUN NO. -    1  NUMOL SOLUTION OF NONHOMOGENSOUS ADVECTION EQUATION
                 - DSS012

INITIAL T -  0.000E+00

  FINAL T -  0.500E+02

  PRINT T -  0.100E+02

NUMBER OF DIFFERENTIAL EQUATIONS -  31

MAXIMUM INTEGRATION ERROR -  0.100E-03

        T          X        U(X,T)       UT(X,T)       UX(X,T)
      0.0       0.00       0.0000        0.0000        0.0000
                0.50       0.0000        0.8750        0.0000
                1.00       0.0000        0.7500        0.0000
                1.50       0.0000        0.6250        0.0000
```

2.00	0.0000	0.5000	0.0000	
2.50	0.0000	0.3750	0.0000	
3.00	0.0000	0.2500	0.0000	
3.50	0.0000	0.1250	0.0000	
4.00	0.0000	0.0000	0.0000	
4.50	0.0000	-0.1250	0.0000	
5.00	0.0000	-0.2500	0.0000	
5.50	0.0000	-0.3750	0.0000	
6.00	0.0000	-0.5000	0.0000	
6.50	0.0000	-0.3750	0.0000	
7.00	0.0000	-0.2500	0.0000	
7.50	0.0000	-0.1250	0.0000	
8.00	0.0000	0.0000	0.0000	
8.50	0.0000	0.0000	0.0000	
9.00	0.0000	0.0000	0.0000	
9.50	0.0000	0.0000	0.0000	
10.00	0.0000	0.0000	0.0000	
	.	.		
	.	.		
	.	.		
T	X	U(X,T)	UT(X,T)	UX(X,T)	EXACT
50.0	0.00	0.0000	0.0000	0.8750	0.0000
	0.50	0.4375	0.0000	0.8750	0.4688
	1.00	0.8125	0.0000	0.7500	0.8750
	1.50	1.1250	0.0000	0.6250	1.2188
	2.00	1.3750	0.0000	0.5000	1.5000
	2.50	1.5625	0.0000	0.3750	1.7188
	3.00	1.6875	0.0000	0.2500	1.8750
	3.50	1.7500	0.0000	0.1250	1.9688
	4.00	1.7500	0.0000	0.0000	2.0000
	4.50	1.6875	0.0000	-0.1250	1.9688
	5.00	1.5625	0.0000	-0.2500	1.8750
	5.50	1.3750	0.0000	-0.3750	1.7188
	6.00	1.1250	0.0000	-0.5000	1.5000
	6.50	0.9375	0.0000	-0.3750	1.2813
	7.00	0.8125	0.0000	-0.2500	1.1250
	7.50	0.7500	0.0000	-0.1250	1.0313
	8.00	0.7500	0.0000	0.0000	1.0000
	8.50	0.7500	0.0000	0.0000	1.0000
	9.00	0.7500	0.0000	0.0000	1.0000
	9.50	0.7500	0.0000	0.0000	1.0000
	10.00	0.7500	0.0000	0.0000	1.0000
	10.50	0.7500	0.0000	0.0000	1.0000
	11.00	0.7500	0.0000	0.0000	1.0000
	11.50	0.7500	0.0000	0.0000	1.0000
	12.00	0.7500	0.0000	0.0000	1.0000
	12.50	0.7500	0.0000	0.0000	1.0000
	13.00	0.7500	0.0000	0.0000	1.0000
	13.50	0.7500	0.0000	0.0000	1.0000

14.00	0.7500	0.0000	0.0000	1.0000
14.50	0.7500	0.0000	0.0000	1.0000
15.00	0.7500	-0.0001	0.0001	1.0000

RUN NO. - 2 NUMOL SOLUTION OF NONHOMOGENEOUS ADVECTION EQUATION - DSS020

INITIAL T - 0.000E+00

FINAL T - 0.500E+02

PRINT T - 0.100E+02

NUMBER OF DIFFERENTIAL EQUATIONS - 31

MAXIMUM INTEGRATION ERROR - 0.100E-03

T	X	U(X,T)	UT(X,T)	UX(X,T)
0.0	0.00	0.0000	0.0000	0.0000
	0.50	0.0000	0.8750	0.0000
	1.00	0.0000	0.7500	0.0000
	1.50	0.0000	0.6250	0.0000
	2.00	0.0000	0.5000	0.0000
	2.50	0.0000	0.3750	0.0000
	3.00	0.0000	0.2500	0.0000
	3.50	0.0000	0.1250	0.0000
	4.00	0.0000	0.0000	0.0000
	4.50	0.0000	-0.1250	0.0000
	5.00	0.0000	-0.2500	0.0000
	5.50	0.0000	-0.3750	0.0000
	6.00	0.0000	-0.5000	0.0000
	6.50	0.0000	-0.3750	0.0000
	7.00	0.0000	-0.2500	0.0000
	7.50	0.0000	-0.1250	0.0000
	8.00	0.0000	0.0000	0.0000
	8.50	0.0000	0.0000	0.0000
	9.00	0.0000	0.0000	0.0000
	9.50	0.0000	0.0000	0.0000
	10.00	0.0000	0.0000	0.0000
		.	.	
		.	.	
		.	.	

T	X	U(X,T)	UT(X,T)	UX(X,T)	EXACT
50.0	0.00	0.0000	0.0000	1.0000	0.0000
	0.50	0.4688	0.0000	0.8750	0.4688
	1.00	0.8750	0.0000	0.7500	0.8750
	1.50	1.2188	0.0000	0.6250	1.2188
	2.00	1.5000	0.0000	0.5000	1.5000

2.50	1.7187	0.0000	0.3750	1.7188
3.00	1.8750	0.0000	0.2500	1.8750
3.50	1.9688	0.0000	0.1250	1.9688
4.00	2.0000	0.0000	0.0000	2.0000
4.50	1.9689	0.0000	-0.1250	1.9688
5.00	1.8745	0.0000	-0.2500	1.8750
5.50	1.7212	0.0000	-0.3750	1.7188
6.00	1.4883	0.0000	-0.5000	1.5000
6.50	1.2738	0.0000	-0.3750	1.2813
7.00	1.1161	0.0000	-0.2500	1.1250
7.50	1.0198	0.0000	-0.1250	1.0313
8.00	0.9954	0.0000	0.0000	1.0000
8.50	0.9933	-0.0001	0.0001	1.0000
9.00	0.9942	0.0000	0.0000	1.0000
9.50	0.9947	0.0000	0.0000	1.0000
10.00	0.9948	0.0001	-0.0001	1.0000
10.50	0.9948	0.0000	0.0000	1.0000
11.00	0.9948	-0.0001	0.0001	1.0000
11.50	0.9948	-0.0001	0.0001	1.0000
12.00	0.9948	0.0000	0.0000	1.0000
12.50	0.9948	0.0002	-0.0002	1.0000
13.00	0.9947	0.0001	-0.0001	1.0000
13.50	0.9947	-0.0001	0.0001	1.0000
14.00	0.9948	-0.0002	0.0002	1.0000
14.50	0.9949	-0.0001	0.0001	1.0000
15.00	0.9949	0.0003	-0.0003	1.0000

Table 11: Abbreviated Output from Program 8 for a 31-point Grid

Note that for the 31-point grid, the steady state solution for the two point upwind approximation is $u(15,\infty) = 0.75$ and for the five point biased upwind approximation, $u(15,\infty) = 0.9949$. These results are also clear from Figure 62.

Generally, we conclude that the five point biased upwind approximations give a numerical solution of good acuracy for the nonhomogeneous advection equation (58). Of course, the other approaches considered previously could be applied to this problem, i.e., subroutines LEO1 and LEO2 and the finite elements of Figure 6.1 and 7.1, but we do not consider these alternatives in order to keep the discussion of this problem to reasonable length.

(10) Simultaneous PDEs

All of the example we have considered so far have been for one PDE. We now consider an

example with two simultaneous PDEs for which an analytical solution can be derived by the Laplace transform [Bateman (10), pp 123-128]. The PDEs model a fluid-solid heat transfer system

$$v\left(\frac{\partial V}{\partial t} + c\frac{\partial V}{\partial x}\right) = k(U - V) \tag{63}$$

$$u\frac{\partial U}{\partial t} = k(V - U) \tag{64}$$

where

V	temperature of the fluid
U	temperature of the solid exchanging heat with the fluid
x	position along the system
t	time
v	ρC_p of the fluid
u	ρC_p of the solid
c	fluid velocity
k	heat transfer coefficient

Equations (63) and (64) require two initial conditions and one boundary condition, which will be designated in general format as

$$V(x,0),\ V(0,t) \text{ and } U(x,0) \qquad (65)(66)(67)$$

The analytical solution of equations (63) to (67) is developed in Appendix 17.

We now consider two types of boundary conditions (specifications of V(0,x)):

(1) A boundary condition which is consistent with the initial condition, and thereby avoids a discontinuity. An example is

$$V(0,t) = 1 - e^{-(k/u)t} \tag{68}$$

for which the analytical solution is (see Appendix 18)

$$V(x,t) = e^{-(k/(vc))x} (k/u)\int_0^t h(\lambda - x/c)e^{-(k/u)(\lambda - x/c)} I_o\left\{2k \sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\}d\lambda \tag{69}$$

(2) A boundary condition which is not consistent with the initial condition, and therefore introduces a discontinuity. Examples are

$$V(0,t) = e^{-(k/u)t} \tag{70}$$

$$V(0,t) = h(t) = \begin{cases} 0, & t < 0 \\ 1, & t > 0 \end{cases} \tag{71}$$

fow which the analytical solutions are

$$V(x,t) = e^{-(k/(vc))x} h(\lambda - x/c)e^{-(k/u)(\lambda - x/c)} I_o\left\{2k \sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\} \tag{72}$$

and

$$\begin{aligned} V(x,t) = {} & e^{-(k/(vc))x} (k/u)\int_0^t h(\lambda - x/c)e^{-(k/u)(\lambda - x/c)} I_o\left\{2k \sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\}d\lambda \\ & + \left\{ e^{-(k/(vc))x} h(\lambda - x/c)e^{-(k/u)(\lambda - x/c)} I_o\left\{2k \sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\}\right\} \end{aligned} \tag{73}$$

respectively.

These three examples (boundary conditions (68), (70 and (71)) provide a spectrum of test problems since, by varying k/u, the rate of change of V(0,t) can be varied for both the continuous and discontinuous cases. These examples are now used to test a computer code that implements a numerical solution of equations (63) and (64) with the same format as the preceding codes for single PDEs. Thus, the numerical and analytical solutions can be compared. The code is listed first as Program 9, and is followed by explanatory comments.

```
      SUBROUTINE INITAL
      COMMON/T/          T,       NSTOP,       NORUN
      DOUBLE PRECISION   T
C...
C...  V(O,T) = 1 - EXP(-(K/U)*T)
      IF(NORUN.EQ.1)CALL INIT1
```

```
C...
C...  V(0,T) = EXP(-(K/U)*T)
      IF(NORUN.EQ.2)CALL INIT2
C...
C...  V(0,T) = 1 (UNIT STEP)
      IF(NORUN.EQ.3)CALL INIT3
C...
C...  V(0,T) = T (UNIT RAMP)
      IF(NORUN.EQ.4)CALL INIT4
      RETURN
      END

      SUBROUTINE DERV
      COMMON/T/          T,      NSTOP,      NORUN
      DOUBLE PRECISION  T
C...
C...  V(0,T) = 1 - EXP(-(K/U)*T)
      IF(NORUN.EQ.1)CALL DERV1
C...
C...  V(0,T) = EXP(-(K/U)*T)
      IF(NORUN.EQ.2)CALL DERV2
C...
C...  V(0,T) = 1 (UNIT STEP)
      IF(NORUN.EQ.3)CALL DERV3
C...
C...  V(0,T) = T (UNIT RAMP)
      IF(NORUN.EQ.4)CALL DERV4
      RETURN
      END

      SUBROUTINE PRINT(NI,NO)
      COMMON/T/          T,      NSTOP,      NORUN
      DOUBLE PRECISION  T
C...
C...  V(0,T) = 1 - EXP(-(K/U)*T)
      IF(NORUN.EQ.1)CALL PRINT1(NI,NO)
C...
C...  V(0,T) = EXP(-(K/U)*T)
      IF(NORUN.EQ.2)CALL PRINT2(NI,NO)
C...
C...  V(0,T) = 1 (UNIT STEP)
      IF(NORUN.EQ.3)CALL PRINT3(NI,NO)
C...
C...  V(0,T) = T (UNIT RAMP)
      IF(NORUN.EQ.4)CALL PRINT4(NI,NO)
      RETURN
      END

      SUBROUTINE INIT1
```

```
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +              BESSIO
      PARAMETER (NX=21)
      COMMON/T/          T,     NSTOP,      NORUN
     +      /Y/     U1(NX),    U2(NX),        VAI
     +      /F/    U1T(NX),   U2T(NX),         VA
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
     +                EXP1,    ARG
C...
C...  SET THE PROBLEM PARAMETERS
         K=1.0D+0
         V=1.773D+00
         C=2.031D+00
         X=1.0D+0
         U=1.0D+01
C...
C...  PRECOMPUTE SOME QUANTITIES USED IN THE ANALYTICAL SOLUTION
      EXP1=DEXP(-K/(V*C)*X)
C...
C...  INITIAL CONDITIONS
      DO 1 I=1,NX
         U1(I)=0.D+0
         U2(I)=0.D+0
1     CONTINUE
      VAI=0.D+0
C...
C...  INITIAL DERIVATIVES
      CALL DERV
      RETURN
      END

      SUBROUTINE INIT2
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +              BESSIO
      PARAMETER (NX=21)
      COMMON/T/          T,     NSTOP,      NORUN
     +      /Y/     U1(NX),    U2(NX),        VAI
     +      /F/    U1T(NX),   U2T(NX),         VA
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
```

```
     +                EXP1,    ARG
C...
C...  SET THE PROBLEM PARAMETERS
         K=1.0D+0
         V=1.773D+00
         C=2.031D+00
         X=1.0D+0
         U=1.0D+01
C...
C...  PRECOMPUTE SOME QUANTITIES USED IN THE ANALYTICAL SOLUTION
      EXP1=DEXP(-K/(V*C)*X)
C...
C...  INITIAL CONDITIONS
      DO 1 I=1,NX
         U1(I)=0.D+0
         U2(I)=0.D+0
1     CONTINUE
      VAI=0.D+0
C...
C...  INITIAL DERIVATIVES
      CALL DERV
      RETURN
      END

      SUBROUTINE INIT3
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +              BESSIO
      PARAMETER (NX=21)
      COMMON/T/          T,     NSTOP,      NORUN
     +      /Y/     U1(NX),    U2(NX),        VAI
     +      /F/    U1T(NX),   U2T(NX),         VA
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
     +                EXP1,    ARG
C...
C...  SET THE PROBLEM PARAMETERS
         K=1.0D+0
         V=1.773D+00
         C=2.031D+00
         X=1.0D+0
         U=1.0D+01
C...
C...  PRECOMPUTE SOME QUANTITIES USED IN THE ANALYTICAL SOLUTION
      EXP1=DEXP(-K/(V*C)*X)
C...
```

```
C...   INITIAL CONDITIONS
       DO 1 I=1,NX
          U1(I)=0.D+0
          U2(I)=0.D+0
1      CONTINUE
       VAI=0.D+0
C...
C...   INITIAL DERIVATIVES
       CALL DERV
       RETURN
       END

       SUBROUTINE INIT4
C...
C...   DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
       DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +               BESSIO
       PARAMETER (NX=21,NT=3)
       COMMON/T/          T,     NSTOP,     NORUN
     +      /Y/     U1(NX),    U2(NX),    VAI(NT)
     +      /F/    U1T(NX),   U2T(NX),     VA(NT)
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
     +                EXP1,    ARG
C...
C...   SET THE PROBLEM PARAMETERS
          K=1.0D+0
          V=1.773D+00
          C=2.031D+00
          X=1.0D+0
          U=1.0D+01
C...
C...   PRECOMPUTE SOME QUANTITIES USED IN THE ANALYTICAL SOLUTION
       EXP1=DEXP(-K/(V*C)*X)
C...
C...   INITIAL CONDITIONS
       DO 1 I=1,NX
          U1(I)=0.D+0
          U2(I)=0.D+0
1      CONTINUE
       DO 2 I=1,NT
          VAI(I)=0.D+0
2      CONTINUE
C...
C...   INITIAL DERIVATIVES
       CALL DERV
       RETURN
```

```
      END

      SUBROUTINE DERV1
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +              BESSIO
      PARAMETER (NX=21)
      COMMON/T/          T,     NSTOP,      NORUN
     +      /Y/     U1(NX),    U2(NX),        VAI
     +      /F/    U1T(NX),   U2T(NX),         VA
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
     +                EXP1,    ARG
C...
C...  BOUNDARY CONDITION
       U1(1)=1.D+0-DEXP((-K/U)*T)
      U1T(1)=0.D+0
C...
C...  U1X
C...
C...     FIVE POINT BIASED UPWIND
         CALL DSS020(0.D+0,X,NX,U1,U1X,1.D+0)
C...
C...  PDES
      DO 1 I=1,NX
         IF(I.NE.1)THEN
            U1T(I)=(K/V)*(U2(I)-U1(I))-C*U1X(I)
         END IF
            U2T(I)=(K/U)*(U1(I)-U2(I))
1     CONTINUE
C...
C...  TEST FOR THE ELAPSED TIME
C...  T - X/C LT 0
      IF((T-X/C).LT.0.D+0)THEN
         VA=0.D+0
C...
C...  T - X/C GE 0
      ELSE
         ARG=2.0D+0*K*DSQRT((X/(U*V*C))*(T-X/C))
         VA=EXP1*DEXP(-(K/U)*(T-X/C))*BESSIO(ARG)
      END IF
      RETURN
      END

      SUBROUTINE DERV2
C...
```

```
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +              BESSIO
      PARAMETER (NX=21)
      COMMON/T/          T,     NSTOP,      NORUN
     +      /Y/     U1(NX),    U2(NX),        VAI
     +      /F/    U1T(NX),   U2T(NX),         VA
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
     +                EXP1,    ARG
C...
C...  BOUNDARY CONDITION
       U1(1)=DEXP((-K/U)*T)
      U1T(1)=0.D+0
C...
C...  U1X
C...
C...     FIVE POINT BIASED UPWIND
         CALL DSS020(0.D+0,X,NX,U1,U1X,1.D+0)
C...
C...  PDES
      DO 1 I=1,NX
         IF(I.NE.1)THEN
            U1T(I)=(K/V)*(U2(I)-U1(I))-C*U1X(I)
         END IF
            U2T(I)=(K/U)*(U1(I)-U2(I))
1     CONTINUE
C...
C...  TEST FOR THE ELAPSED TIME
C...  T - X/C LT 0
      IF((T-X/C).LT.0.D+0)THEN
         VA=0.D+0
C...
C...  T - X/C GE 0
      ELSE
         ARG=2.0D+0*K*DSQRT((X/(U*V*C))*(T-X/C))
         VA=EXP1*DEXP(-(K/U)*(T-X/C))*BESSIO(ARG)
      END IF
      RETURN
      END

      SUBROUTINE DERV3
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
```

```
     +                BESSI0
      PARAMETER (NX=21)
      COMMON/T/          T,      NSTOP,      NORUN
     +       /Y/     U1(NX),     U2(NX),        VAI
     +       /F/    U1T(NX),    U2T(NX),         VA
     +       /S/    U1X(NX)
     +       /C/          K,       V,      C,      X,      U,
     +                 EXP1,    ARG
C...
C...  BOUNDARY CONDITION
       U1(1)=1.D+0
      U1T(1)=0.D+0
C...
C...  U1X
C...
C...     FIVE POINT BIASED UPWIND
         CALL DSS020(0.D+0,X,NX,U1,U1X,1.D+0)
C...
C...  PDES
      DO 1 I=1,NX
         IF(I.NE.1)THEN
            U1T(I)=(K/V)*(U2(I)-U1(I))-C*U1X(I)
         END IF
            U2T(I)=(K/U)*(U1(I)-U2(I))
1     CONTINUE
C...
C...  TEST FOR THE ELAPSED TIME
C...  T - X/C LT 0
      IF((T-X/C).LT.0.D+0)THEN
         VA=0.D+0
C...
C...  T - X/C GE 0
      ELSE
         ARG=2.0D+0*K*DSQRT((X/(U*V*C))*(T-X/C))
         VA=EXP1*DEXP(-(K/U)*(T-X/C))*BESSI0(ARG)
      END IF
      RETURN
      END

      SUBROUTINE DERV4
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,       V,      C,      X,      U,
     +                 EXP1,    ARG,     VA,    VAI,      T,
     +                   U1,     U2,    U1T,    U2T,    U1X,
     +                BESSI0
      PARAMETER (NX=21,NT=3)
      COMMON/T/          T,      NSTOP,      NORUN
     +       /Y/     U1(NX),     U2(NX),    VAI(NT)
```

```
     +        /F/    U1T(NX),    U2T(NX),     VA(NT)
     +        /S/    U1X(NX)
     +        /C/          K,       V,       C,       X,       U,
     +                  EXP1,    ARG
C...
C...  BOUNDARY CONDITION
      IF(T.LE.1.D+0)U1(1)=T
      IF(T.GT.1.D+0)U1(1)=1.D+0
      U1T(1)=0.D+0
C...
C...  U1X
C...
C...     FIVE POINT BIASED UPWIND
         CALL DSS020(0.D+0,X,NX,U1,U1X,1.D+0)
C...
C...  PDES
      DO 1 I=1,NX
         IF(I.NE.1)THEN
            U1T(I)=(K/V)*(U2(I)-U1(I))-C*U1X(I)
         END IF
            U2T(I)=(K/U)*(U1(I)-U2(I))
1     CONTINUE
C...
C...  TEST FOR THE ELAPSED TIME FOR RESPONSE TO STEP AT T = 0
C...  T - X/C LT 0
      IF((T-X/C).LT.0.D+0)THEN
         VA(1)=0.D+0
C...
C...  T - X/C GE 0
      ELSE
         ARG=2.0D+0*K*DSQRT((X/(U*V*C))*(T-X/C))
         VA(1)=EXP1*DEXP(-(K/U)*(T-X/C))*BESSI0(ARG)
      END IF
C...
C...  TEST FOR THE ELAPSED TIME FOR RESPONSE TO STEP AT T = 1
C...  T - X/C - 1 LT 0
      IF((T-X/C-1.D+0).LT.0.D+0)THEN
         VA(2)=0.D+0
C...
C...  T - X/C - 1 GE 0
      ELSE
         ARG=2.0D+0*K*DSQRT((X/(U*V*C))*(T-X/C-1.D+0))
         VA(2)=-EXP1*DEXP(-(K/U)*(T-X/C-1.D+0))*BESSI0(ARG)
      END IF
C...
C...  SUPERIMPOSE TWO STEP RESPONSES
      VA(3)=VA(1)+(K/U)*VAI(1)+VA(2)+(K/U)*VAI(2)
      RETURN
      END
```

```
      SUBROUTINE PRINT1(NI,NO)
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C.      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +              BESSIO
      PARAMETER (NX=21)
      COMMON/T/          T,      NSTOP,      NORUN
     +      /Y/     U1(NX),     U2(NX),        VAI
     +      /F/    U1T(NX),    U2T(NX),        VA
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
     +                EXP1,    ARG
C...
C...  ARRAYS TO STORE THE SOLUTION FOR SUBSEQUENT PLOTTING
      DOUBLE PRECISION   TP(201),    VP(2,201)
C...
C...  INITIALIZE A COUNTER FOR THE PRINTING AND PLOTTING
      DATA IP/0/
      IP=IP+1
      IF(IP.EQ.1)THEN
C...
C...  MAP THE ODE JACOBIAN MATRIX AND COMPUTE THE TEMPORAL EIGENVALUES
C...  AT THE BEGINNING OF THE SOLUTION
      CALL MAP
C...
C...  PRINT A HEADING
         WRITE(NO,1)
1        FORMAT(//,14X,'T',9X,'V(X,T)',4X,'V(X,T) ANAL')
      END IF
C...
C...  PRINT THE SOLUTION EVERY 20TH CALL TO SUBROUTINE PRINT
      IF(((IP-1)/20*20).EQ.(IP-1))THEN
         WRITE(NO,2)T,U1(NX),(K/U)*VAI
2        FORMAT(F15.2,2F15.4)
C...
C...  WRITE NORUN, IP AND T ON THE SCREEN TO MONITOR THE PROGRESS OF
C...  THE CALCULATIONS
      WRITE(*,*)' NORUN = ',NORUN,' IP = ',IP,' T = ',T
      END IF
C...
C...  STORE THE SOLUTION FOR PLOTTING
      TP(IP)=T
      VP(1,IP)=U1(NX)
      VP(2,IP)=(K/U)*VAI
      IF(IP.LT.201)RETURN
C...
```

```
C...  MAP THE ODE JACOBIAN MATRIX AND COMPUTE THE TEMPORAL EIGENVALUES
C...  AT THE END OF THE SOLUTION
      CALL MAP
C...
C...  OPEN A FILE FOR THE PLOTTING
         OPEN(1,FILE='T.TOP',STATUS='NEW')
C...
C...  SCALE THE AXES OF THE PLOT
         WRITE(1,6)
6        FORMAT(' SET LIMITS X FROM 0 TO 10 Y FROM 0 TO 1',/,
     1           ' SET FONT DUPLEX')
         WRITE(1,7)
7        FORMAT(' SET SIZE 6 BY 6')
C...
C...  WRITE THE ANALYTICAL SOLUTION FOR PLOTTING
         WRITE(1,5)(TP(I),VP(2,I),I=1,IP)
5        FORMAT(2F10.4)
         WRITE(1,4)
4        FORMAT(' JOIN 1')
C...
C...  WRITE THE NUMERICAL SOLUTION FOR PLOTTING
         WRITE(1,5)(TP(I),VP(1,I),I=1,IP)
         WRITE(1,4)
C...
C...  LABEL THE PLOT
         WRITE(1,8)
8        FORMAT(
     1'  TITLE 1.75 5.8 "Fig. 63: Two First Order PDEs"'
     2,/,' TITLE LEFT "                V(1,t)"'
     3,/,' TITLE BOTTOM "t"'
     4,/,' TITLE 1.25 0.25 '
     5   '"V(1,t) vs t, V(0,t) = 1 - exp(-(k/u)t)"')
         WRITE(1,9)
9        FORMAT(' NEW FRAME')
C...
C...  RESET THE INTEGER COUNTER FOR THE NEXT RUN
      IP=0
      RETURN
      END

      SUBROUTINE PRINT2(NI,NO)
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +              BESSIO
      PARAMETER (NX=21)
      COMMON/T/          T,      NSTOP,      NORUN
```

```
     +       /Y/    U1(NX),    U2(NX),        VAI
     +       /F/   U1T(NX),   U2T(NX),         VA
     +       /S/   U1X(NX)
     +       /C/          K,      V,      C,      X,      U,
     +                 EXP1,   ARG
C...
C...  ARRAYS TO STORE THE SOLUTION FOR SUBSEQUENT PLOTTING
      DOUBLE PRECISION   TP(201),    VP(2,201)
C...
C...  INITIALIZE A COUNTER FOR THE PRINTING AND PLOTTING
      DATA IP/0/
      IP=IP+1
      IF(IP.EQ.1)THEN
C...
C...  MAP THE ODE JACOBIAN MATRIX AND COMPUTE THE TEMPORAL EIGENVALUES
C...  AT THE BEGINNING OF THE SOLUTION
      CALL MAP
C...
C...  PRINT A HEADING
         WRITE(NO,1)
1        FORMAT(//,14X,'T',9X,'V(X,T)',4X,'V(X,T) ANAL')
      END IF
C...
C...  PRINT THE SOLUTION EVERY 20TH CALL TO SUBROUTINE PRINT
      IF(((IP-1)/20*20).EQ.(IP-1))THEN
         WRITE(NO,2)T,U1(NX),VA
2        FORMAT(F15.2,2F15.4)
C...
C...  WRITE NORUN, IP AND T ON THE SCREEN TO MONITOR THE PROGRESS OF
C...  THE CALCULATIONS
      WRITE(*,*)' NORUN = ',NORUN,' IP = ',IP,' T = ',T
      END IF
C...
C...  STORE THE SOLUTION FOR PLOTTING
      TP(IP)=T
      VP(1,IP)=U1(NX)
      VP(2,IP)=VA
      IF(IP.LT.201)RETURN
C...
C...  MAP THE ODE JACOBIAN MATRIX AND COMPUTE THE TEMPORAL EIGENVALUES
C...  AT THE END OF THE SOLUTION
      CALL MAP
C...
C...  OPEN A FILE FOR THE PLOTTING
         OPEN(1,FILE='T.TOP',STATUS='NEW')
C...
C...  SCALE THE AXES OF THE PLOT
         WRITE(1,6)
6        FORMAT(' SET LIMITS X FROM 0 TO 10 Y FROM -0.1 TO 1.0',/,
```

```
     1            ' SET FONT DUPLEX')
         WRITE(1,7)
7        FORMAT(' SET SIZE 6 BY 6')
C...
C...  WRITE THE ANALYTICAL SOLUTION FOR PLOTTING
         WRITE(1,5)(TP(I),VP(2,I),I=1,IP)
5        FORMAT(2F10.4)
         WRITE(1,4)
4        FORMAT(' JOIN 1')
C...
C...  WRITE THE NUMERICAL SOLUTION FOR PLOTTING
         WRITE(1,5)(TP(I),VP(1,I),I=1,IP)
         WRITE(1,4)
C...
C...  LABEL THE PLOT
         WRITE(1,8)
8        FORMAT(
     1'  TITLE 1.75 5.8 "Fig. 64: Two First Order PDEs"'
     2,/,' TITLE LEFT "                V(1,t)"'
     3,/,' TITLE BOTTOM "t"'
     4,/,' TITLE 1.5 0.25 '
     5   '"V(1,t) vs t, V(0,t) = exp(-(k/u)t)"')
         WRITE(1,9)
9        FORMAT(' NEW FRAME')
C...
C...  RESET THE INTEGER COUNTER FOR THE NEXT RUN
      IP=0
      RETURN
      END

      SUBROUTINE PRINT3(NI,NO)
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,   ARG,    VA,    VAI,      T,
     +                  U1,    U2,   U1T,    U2T,    U1X,
     +               BESSIO
      PARAMETER (NX=21)
      COMMON/T/          T,     NSTOP,      NORUN
     +      /Y/     U1(NX),    U2(NX),        VAI
     +      /F/    U1T(NX),   U2T(NX),         VA
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
     +                EXP1,   ARG
C...
C...  ARRAYS TO STORE THE SOLUTION FOR SUBSEQUENT PLOTTING
      DOUBLE PRECISION   TP(201),    VP(2,201)
C...
C...  INITIALIZE A COUNTER FOR THE PRINTING AND PLOTTING
```

```
      DATA IP/0/
      IP=IP+1
      IF(IP.EQ.1)THEN
C...
C...  MAP THE ODE JACOBIAN MATRIX AND COMPUTE THE TEMPORAL EIGENVALUES
C...  AT THE BEGINNING OF THE SOLUTION
      CALL MAP
C...
C...  PRINT A HEADING
         WRITE(NO,1)
1        FORMAT(//,14X,'T',9X,'V(X,T)',4X,'V(X,T) ANAL')
      END IF
C...
C...  PRINT THE SOLUTION EVERY 20TH CALL TO SUBROUTINE PRINT
      IF(((IP-1)/20*20).EQ.(IP-1))THEN
         WRITE(NO,2)T,U1(NX),VA+(K/U)*VAI
2        FORMAT(F15.2,2F15.4)
C...
C...  WRITE NORUN, IP AND T ON THE SCREEN TO MONITOR THE PROGRESS OF
C...  THE CALCULATIONS
      WRITE(*,*)' NORUN = ',NORUN,' IP = ',IP,' T = ',T
      END IF
C...
C...  STORE THE SOLUTION FOR PLOTTING
      TP(IP)=T
      VP(1,IP)=U1(NX)
      VP(2,IP)=VA+(K/U)*VAI
      IF(IP.LT.201)RETURN
C...
C...  MAP THE ODE JACOBIAN MATRIX AND COMPUTE THE TEMPORAL EIGENVALUES
C...  AT THE END OF THE SOLUTION
      CALL MAP
C...
C...  OPEN A FILE FOR THE PLOTTING
         OPEN(1,FILE='T.TOP',STATUS='NEW')
C...
C...  SCALE THE AXES OF THE PLOT
         WRITE(1,6)
6        FORMAT(' SET LIMITS X FROM 0 TO 10 Y FROM -0.1 TO 1.1',/,
     1          ' SET FONT DUPLEX')
         WRITE(1,7)
7        FORMAT(' SET SIZE 6 BY 6')
C...
C...  WRITE THE ANALYTICAL SOLUTION FOR PLOTTING
         WRITE(1,5)(TP(I),VP(2,I),I=1,IP)
5        FORMAT(2F10.4)
         WRITE(1,4)
4        FORMAT(' JOIN 1')
C...
```

```
C...  WRITE THE NUMERICAL SOLUTION FOR PLOTTING
         WRITE(1,5)(TP(I),VP(1,I),I=1,IP)
         WRITE(1,4)
C...
C...  LABEL THE PLOT
         WRITE(1,8)
8        FORMAT(
     1'  TITLE 1.75 5.8 "Fig. 65: Two First Order PDEs"'
     2,/,' TITLE LEFT "                V(1,t)"'
     3,/,' TITLE BOTTOM "t"'
     4,/,' TITLE 1.75 0.25 '
     5   '"V(1,t) vs t, V(0,t) = unit step"')
         WRITE(1,9)
9        FORMAT(' NEW FRAME')
C...
C...  RESET THE INTEGER COUNTER FOR THE NEXT RUN
      IP=0
      RETURN
      END

      SUBROUTINE PRINT4(NI,NO)
C...
C...  DECLARE SELECTED VARIABLES AS DOUBLE PRECISION
      DOUBLE PRECISION  K,      V,      C,      X,      U,
     +                EXP1,    ARG,     VA,    VAI,      T,
     +                  U1,     U2,    U1T,    U2T,    U1X,
     +              BESSIO
      PARAMETER (NX=21,NT=3)
      COMMON/T/          T,     NSTOP,     NORUN
     +      /Y/     U1(NX),    U2(NX),   VAI(NT)
     +      /F/    U1T(NX),   U2T(NX),    VA(NT)
     +      /S/    U1X(NX)
     +      /C/          K,      V,      C,      X,      U,
     +                EXP1,    ARG
C...
C...  ARRAYS TO STORE THE SOLUTION FOR SUBSEQUENT PLOTTING
      DOUBLE PRECISION   TP(201),    VP(2,201)
C...
C...  INITIALIZE A COUNTER FOR THE PRINTING AND PLOTTING
      DATA IP/0/
      IP=IP+1
      IF(IP.EQ.1)THEN
C...
C...  MAP THE ODE JACOBIAN MATRIX AND COMPUTE THE TEMPORAL EIGENVALUES
C...  AT THE BEGINNING OF THE SOLUTION
      CALL MAP
C...
C...  PRINT A HEADING
         WRITE(NO,1)
```

```
1          FORMAT(//,14X,'T',9X,'V(X,T)',4X,'V(X,T) ANAL')
        END IF
C...
C...    PRINT THE SOLUTION EVERY 20TH CALL TO SUBROUTINE PRINT
        IF(((IP-1)/20*20).EQ.(IP-1))THEN
           WRITE(NO,2)T,U1(NX),VAI(3)
2          FORMAT(F15.2,2F15.4)
C...
C...    WRITE NORUN, IP AND T ON THE SCREEN TO MONITOR THE PROGRESS OF
C...    THE CALCULATIONS
        WRITE(*,*)' NORUN = ',NORUN,' IP = ',IP,' T = ',T
        END IF
C...
C...    STORE THE SOLUTION FOR PLOTTING
        TP(IP)=T
        VP(1,IP)=U1(NX)
        VP(2,IP)=VAI(3)
        IF(IP.LT.201)RETURN
C...
C...    MAP THE ODE JACOBIAN MATRIX AND COMPUTE THE TEMPORAL EIGENVALUES
C...    AT THE END OF THE SOLUTION
        CALL MAP
C...
C...    OPEN A FILE FOR THE PLOTTING
           OPEN(1,FILE='T.TOP',STATUS='NEW')
C...
C...    SCALE THE AXES OF THE PLOT
           WRITE(1,6)
6          FORMAT(' SET LIMITS X FROM 0 TO 10 Y FROM -0.1 TO 1.1',/,
     1            ' SET FONT DUPLEX')
           WRITE(1,7)
7          FORMAT(' SET SIZE 6 BY 6')
C...
C...    WRITE THE ANALYTICAL SOLUTION FOR PLOTTING
           WRITE(1,5)(TP(I),VP(2,I),I=1,IP)
5          FORMAT(2F10.4)
           WRITE(1,4)
4          FORMAT(' JOIN 1')
C...
C...    WRITE THE NUMERICAL SOLUTION FOR PLOTTING
           WRITE(1,5)(TP(I),VP(1,I),I=1,IP)
           WRITE(1,4)
C...
C...    LABEL THE PLOT
           WRITE(1,8)
8          FORMAT(
     1'    TITLE 1.75 5.8 "Fig. 66: Two First Order PDEs"'
     2,/,' TITLE LEFT "                V(1,t)"'
     3,/,' TITLE BOTTOM "t"'
```

```
     4,/,' TITLE 1.75 0.25 '
     5   '"V(1,t) vs t, V(0,t) = unit ramp"')
         WRITE(1,9)
9        FORMAT(' NEW FRAME')
C...
C...  RESET THE INTEGER COUNTER FOR THE NEXT RUN
      IP=0
      RETURN
      END

      SUBROUTINE MAP
      PARAMETER (N=42)
      COMMON/T/       T
     1       /Y/   Y(N)
     2       /F/   F(N)
C...
C...  SUBROUTINE MAP
C...
C...      (1)  CALLS SUBROUTINE JMAP TO MAP THE JACOBIAN MATRIX OF
C...           NTH-ORDER ODE SYSTEM.
C...
C...      (2)  CALLS A SERIES OF EISPACK ROUTINES VIA SUBROUTINE EIGEN
C...           TO COMPUTE THE TEMPORAL EIGENVALUES OF THE ODE SYSTEM,
C...           AND OPTIONALLY, THE ASSOCIATED EIGENVECTORS.
C...
C...  DECLARE SELECTED VARIABLES DOUBLE PRECISION
      DOUBLE PRECISION T, Y, F, YOLD(N), FOLD(N)
C...
C...  DEFINE SINGLE PRECISION ARRAYS REQUIRED BY SUBROUTINE JMAP (A)
C...  AND SUBROUTINE EIGEN (WR, WI, Z, RW, IW)
      REAL A(N,N), WR(N), WI(N), Z(N,N), RW(N)
      INTEGER      IW(N)
C...
C...  MAP THE JACOBIAN MATRIX OF THE ODE SYSTEM DEFINED IN SUBROUTINE
C...  DERV, AND COMPUTE ITS TEMPORAL EIGENVALUES.  NOTE THAT ARRAY
C...  A CONTAINING THE NUMERICAL JACOBIAN MATRIX ON OUTPUT FROM JMAP
C...  IS SINGLE PRECISION, I.E., IT IS NOT DECLARED IN THE PRECEDING
C...  DOUBLE PRECISION STATEMENT.  FURTHER EXPLANATION OF THIS POINT
C...  IS GIVEN IN THE COMMENTS BEFORE THE FOLLOWING CALL TO SUBROUTINE
C...  EIGEN
      CALL JMAP(N,A,Y,YOLD,F,FOLD)
C...
C...  SUBROUTINE EIGEN (PART OF DSS/2) CALLS A SERIES OF EISPACK ROU-
C...  TINES TO COMPUTE THE TEMPORAL EIGENVALUES, AND OPTIONALLY, THE
C...  EIGENVECTORS OF THE NTH-ORDER ODE SYSTEM JACOBIAN MATRIX.  NOTE
C...  THAT ALL OF THE ARRAYS GOING INTO EIGEN (A, WR, WI, Z, RW, IW)
C...  ARE SINGLE PRECISION.  THIS WAS DONE TO ALLOW THE USE OF THE
C...  SINGLE PRECISION EISPACK ROUTINES CALLED BY EIGEN
      CALL EIGEN(N,A,WR,WI,Z,RW,IW)
```

```
RETURN
END
```

Program 9: Subroutines DERV, INITAL and PRINT, and Subordinate Routines for the Solution of Equations (63) and (64)

We now consider each section of Program 9.

(1) Subroutine INITAL calls one of four initialization subroutines, INIT1 to INIT4, for four different test problems, depending on the run counter, NORUN. Subroutines INIT1, INIT2 and INIT3 are for boundary conditions (68), (70) and (71) respectively; INIT4 is for a unit ramp boundary condition, $V(0,t) = r(t)$, *where* $r(t)$ *is defined as*

$$r(t) = \begin{cases} t, & 0 \leq t \leq 1 \\ 1, & t > 1 \end{cases} \tag{74}$$

The exact solution for this case is obtained by integration and superposition of the solution for $V(0,t) = h(t)$, as explained subsequently. Note that four runs are executed (NORUN varies from 1 to 4) for these four boundary cobditions.

(2) Similarly, subroutine DERV calls one of four derivative subroutines, DERV1 to DERV4, corresponding to the four test problems described for INIT1 to INIT4, respectively.

(3) Finally, subroutine PRINT calls one of four derivative subroutines, PRINT1 to PRINT4, corresponding to the four test problems implemented in INIT1 to INIT4 and DERV1 to DERV4.

Now we consider the subroutines for each of the four test problems, starting with INIT1, DERV1 and PRINT1, which are for boundary condition (68). The following points should be noted about subroutine INIT1:

(1) The coding is in double precision format. This was done primarily to provide compatibility with function BESSI0 from Press et al [(14), pp 176-178] which is in double precision; this function computes the modified Bessel function I_0 in the analytical solutions, equations (69), (72) and (73)

(2) The model parameters are set in a series of statements, e.g., K=1.0D+0, and the exponential in the analytical solutions, equations (69), (72) and (73), which does not involve t, and therefore is constant throughout each solution is computed, i.e., EXP1=DEXP(-K/(V*C)*X).

(3) Initial conditions (65) and (67) are implemented in DO loop 1 (which are homogeneous). Also, the integral in equation (69) (VAI) is intialized

$$VAI = e^{-(k/(vc))x} (k/u) \int_0^t h(\lambda - x/c) e^{-(k/u)(\lambda - x/c)} I_0\left\{2k \sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\} d\lambda \qquad (69)$$

Note that VAI = 0 at t = 0. Also, VAI appears as the 43rd element in COMMON/Y/ since it is evaluated as an equivalent ODE, as explained subsequently.

In summary, INT1 initializes the program and sets the initial conditions for the first problem with boundary condition (68) (NORUN = 1). Then the ODE integrator calls DERV1 to step along the solution. DERV1 has the following features:

(1) Boundary condition (68) is first programmed as

```
C...
C...   BOUNDARY CONDITION
        U1(1)=1.D+0-DEXP((-K/U)*T)
       U1T(1)=0.D+0
```

Note that the temporal derivative U1T(1) is zeroed since U1(1) is defined by the boundary condition rather than by an ODE at grid point 1.

(2) The spatial derivative in equation (63), $\frac{\partial V}{\partial x}$, is computed by a call to subroutine DSS020 with the coding

```
C...
C...        FIVE POINT BIASED UPWIND
            CALL DSS020(0.D+0,X,NX,U1,U1X,1.D+0)
```

(3) The derivatives $\frac{\partial V}{\partial t}$ (U1T) and $\frac{\partial U}{\partial t}$ (U2T) in equations (63) and (64) are programmed in DO

loop 1 as

```
C...
C...   PDES
       DO 1 I=1,NX
          IF(I.NE.1)THEN
             U1T(I)=(K/V)*(U2(I)-U1(I))-C*U1X(I)
          END IF
             U2T(I)=(K/U)*(U1(I)-U2(I))
1      CONTINUE
```

Note that U1T(1) is not computed since it was zeroed for boundary condition (68). Also, the close resemblance of the PDEs, equations (63) and (64), and the coding in DO loop 1 is apparent. This is one of the major advantages of this approach to PDEs.

(4) The integrand in equation (69)

$$VA = e^{-(k/(vc))x}\,(k/u)h(\lambda - x/c)e^{-(k/u)(\lambda - x/c)}\,I_0\left\{2k\sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\} \tag{75}$$

is programmed after DO loop 1 as

```
C...
C...   TEST FOR THE ELAPSED TIME
C...   T - X/C LT 0
       IF((T-X/C).LT.0.D+0)THEN
          VA=0.D+0
C...
C...   T - X/C GE 0
       ELSE
          ARG=2.0D+0*K*DSQRT((X/(U*V*C))*(T-X/C))
          VA=(K/U)*EXP1*DEXP(-(K/U)*(T-X/C))*BESSI0(ARG)
       END IF
```

With this procedure, we are numerically integrating the ODE

$$dVAI/dt = VA \tag{76}$$

to obtain the integral in equation (69). This has the important advantage that additional programming outside the framework of the ODE integrator is not required, and also, we can take advantage of the features of the ODE integrator to insure an accurate value of the integral, i.e.,

automatic step size adjustment to achieve a user prescribed accuracy. Note also that VA appears as the 43rd element in COMMON/F/ as required so that it is integrated to VAI (the 43rd element in COMMON/Y/).

Subroutine PRINT1 provides the numerical and graphical output for the first problem, and has the following features

(1) At the beginning of the solution (IP = 1), a call to subroutine MAP maps the Jacobian matrix of the first 42 elements in COMMON/Y/ and /F/ and computes their temporal eigenvalues, which are then printed by MAP (as indicated in the following discussion of the output). Then a heading for the numerical and analytical solutions is printed via FORMAT 1.

(2) The numerical solution for V(1,t) (U1(NX)) and analytical solution for V(1,t) (VAI) are printed vs t for every 10th call to subroutine PRINT (PRINT is called 201 times so that enough points are generated to produce quality plots of the solutions).

(3) The numerical and analytical solutions are stored in array VP, and t is stored in array TP, for subsequent plotting. The plotting is done at the end of the run, when IP = 201, by the coding at the end of subroutine PRINT. The plots produced by this coding are subsequently discussed.

The data for Program 9 are listed in Table 12 for the four runs (four test problems)

```
BATEMAN TEST PROBLEM - DSS020 - V(0,T) = 1 - EXP(-(K/U)*T)
0.          10.0        0.05
   43 1000    15     1  ABS 0.0001
BATEMAN TEST PROBLEM - DSS020 - V(0,T) = EXP(-(K/U)*T)
0.          10.0        0.05
   42 1000    15     1  ABS 0.0001
BATEMAN TEST PROBLEM - DSS020 - V(0,T) = 1
0.          10.0        0.05
   43 1000    15     1  ABS 0.0001
BATEMAN TEST PROBLEM - DSS020 - unit ramp
0.          10.0        0.05
   45 1000    15     1  ABS 0.0001
END OF RUNS
```

Table 12: Data for Program 9

The following points pertain to the data:

(1) Time ranges over the interval $0 \leq t \leq 10$, with calls to subroutine PRINT1 at intervals of 0.05. Thus, there are 10/0.05 + 1 = 201 calls to PRINT1 (including the initial condition at t = 0).

(2) 43 ODEs are specified, which agrees with the number of elements in COMMON/Y/ and /F/.

(3) For this problem, subroutine RKF45 called by main program SRKF45 gave erratic results, e.g., the integration did not proceed to the correct output (print) times. Therefore the ODE integration was done with DSS/2 integrator 15 [Schiesser (15)], which is an implementation of the Rungke Kutta Fehlberg formulas [Forsythe, et al (1)], the same integration formulas as in subroutine RKF45. The additional data for DSS/2 are: (a) 1000 - the ratio of the print interval (0.05) to the minimum integration interval, so that the minimum allowable integration interval is 0.05/1000, (b) 15 - integrator 15, and (c) 1 - error messages are printed if the integration error reaches the minimum of 0.05/1000.

(4) An absolute error of 0.0001 in the solution of the 43 ODEs is specified.

Subroutine PRINT1 produces a substantial amount of numerical and graphical output, which will now be discussed in several parts.

(1) A data summary is first printed for each run. Thus, for NORUN = 1, the data summary is

```
RUN NO.   1 - BATEMAN TEST PROBLEM - DSS020
            -  V(0,T) = 1 - EXP(-(K/U)*T)

INITIAL VALUE OF TIME =  0.0000D+00

FINAL VALUE OF TIME =  0.1000D+02

PRINT INTERVAL OF TIME =  0.5000D-01

NUMBER OF FIRST-ORDER DIFFERENTIAL EQUATIONS =  43

PRINT INTERVAL/MINIMUM INTEGRATION INTERVAL =  1000
```

```
INTEGRATION ALGORITHM =  15
      1 - RUNGE KUTTA EULER
      2 - RUNGE KUTTA NIESSE
      3 - RUNGE KUTTA MERSON
      4 - RUNGE KUTTA TANAKA - 4
      5 - RUNGE KUTTA TANAKA - 5
      6 - RUNGE KUTTA CHAI
      7 - RUNGE KUTTA ENGLAND
      8 - RUNGE KUTTA WES - 4/1
      9 - RUNGE KUTTA WES - 4/2
     10 - RUNGE KUTTA WES - 4/3
     11 - RUNGE KUTTA WES - 4/4
     12 - RUNGE KUTTA WES - 4/5
     13 - RUNGE KUTTA WES - 5/1
     14 - RUNGE KUTTA WES - 5/2
     15 - RUNGE KUTTA FEHLBERG - RKF45

PRINT OPTION = 1
     NO INTEGRATION ERROR DIAGNOSTICS - 0
     SUMMARY OF INTEGRATION ERRORS    - 1

TYPE OF INTEGRATION ERROR = ABS

MAXIMUM INTEGRATION ERROR =  0.100D-03
```

The data summary should be self explanatory.

(2) The ODE Jacobian map produced by the first call to subroutine MAP appears next in the output

```
DEPENDENT VARIABLE COLUMN INDEX J (FOR YJ) IS PRINTED HORIZONTALLY

DERIVATIVE ROW INDEX I (FOR DYI/DT = FI(Y1,Y2,...,YJ,...,YN) IS PRINTED
VERTICALLY

JACOBIAN MATRIX ELEMENT IN THE MAP WITH INDICES I,J IS FOR PFI/PYJ
WHERE P DENOTES A PARTIAL DERIVATIVE


                 111111111122222222223333333333444
        123456789012345678901234567890123456789012
    1
    2    6665                    4
    3    6465                     4
    4    6666                      4
    5    56666                      4
    6     56666                      4
    7      56666                      4
```

```
 8        56666                    4
 9         56666                    4
10          56666                    4
11           56666                    4
12            56666                    4
13             56666                    4
14              56666                    4
15               56666                    4
16                56666                    4
17                 56666                    4
18                  56666                    4
19                   56666                    4
20                    56666                    4
21                    66776                    4
22                          3
23   3                       3
24    3                       3
25     3                       3
26      3                       3
27       3                       3
28        3                       3
29         3                       3
30          3                       3
31           3                       3
32            3                       3
33             3                       3
34              3                       3
35               3                       3
36                3                       3
37                 3                       3
38                  3                       3
39                   3                       3
40                    3                       3
41                     3                       3
42                      3                       3
```

This map requires some explanation:

(2.1) An integer index for the ODE dependent variables is listed across the top of the map, i.e., $1 \leq j \leq 42$. Similarly, an integer index for the ODE derivatives is listed down the left side of the map, i.e., $1 \leq i \leq 42$. Thus, a nonblank element in ith row and jth column indicates that the RHS of ODE i is a function of dependent variable j. $1 \leq j \leq 21$ is for U in equations (63) and (64) approximated over the 21 point grid (array U1(21) in COMMON/Y/), and $22 \leq j \leq 42$ is for V in equations (63) and (64) (array U2(21) in COMMON/Y/). As a specific example, the

second row of the map

2 6665 4

indicates $dy_2/dt = f_2(y_2,y_3,y_4,y_5,y_{23})$ or, in terms of the Fortran variables in subroutine DERV1, U1T(2) = f(U1(2),U1(3),U1(4),U1(5),U2(2)). The latter is just the ODE approximation for equation (63) at grid point 2; U1(2), U1(3), U1(4) and U1(5) result from the five point biased upwind approximations in subroutine DSS020 (which has only four points at grid point 2 since U1(1) is set by boundary condition (68) at x = 0, and not by an ODE), and U2(2) results from the coupling term involving V in equation (63). Of course, similar interpretations can be given to all of the other rows in the map. For example, the first row has only blank characters, indicating U1T(1) does not depend on any of the dependent variables, as expected, since boundary condition (68) in subroutine DERV1 includes the programming U1T(1) = 0.

(2.2) The map elucidates the structure of the ODEs. The five diagonals with, for example, 56666 result from the five point biased upwind approximations in subroutine DSS020 (used to calculate $\frac{\partial V}{\partial x}$ in equation (63)). Also, the outlying diagonal with 3 in rows 1 to 21 is due to the cross coupling between equations (63) and (64) due to the heat transfer term k(U - V) in equation (63). Thus, rows 1 to 21 reflect the ODE approximation of equation (63), including boundary condition (68).

(2.3) Similarly, rows 22 to 42 relect the ODE approximation of equation (64). Note that the main diagonal is not pentadiagonal since equation (64) does not have a derivative in x. Rather, the one main diagonal results from U in the heat transfer term k(V - U) in equation (64). The outlying diagonal with 4 results from V in this heat transfer term.

Finally, we should note that the numbers printed in the map give an indication of the magnitude of the elements of the Jacobian matrix. Thus a "6" indicates an element which can be up to two orders of magnitude (factor of 10^2) larger than an element designated with a "4". Additional information about how these numbers in the map are generated is available by studying subroutine MAP and the subroutines it calls (e.g., JMAP and EIGEN), which are all thoroughly commented. We can conclude from this map that generally it provides a "picture" of the ODE structure that is unique for the problem and thereby helps us to understand the ODE equation structure.

(3) In addition to producing the preceding map. subroutine MAP generates a numerical approximation to the Jacobian matrix through finite difference approximation of the elements of the Jacobian matrix (in subroutine JMAP called by MAP). This numerical Jacobian is then passed to another subroutine (EIGEN called by MAP) which computes the eigenvalues of the numerical Jacobian, and prints them. These eigenvalues are tabulated below for the first run ($NORUN = 1$)

I	REAL	IMAG
1	-9.309	47.090
2	-9.309	-47.090
3	-31.279	49.103
4	-31.279	-49.103
5	-31.912	46.532
6	-31.912	-46.532
7	-32.885	42.393
8	-32.885	-42.393
9	-34.045	36.904
10	-34.045	-36.904
11	-35.113	30.321
12	-35.113	-30.321
13	-35.512	22.861
14	-35.512	-22.861
15	-38.129	4.087
16	-38.129	-4.087
17	-36.802	12.158
18	-36.802	-12.158
19	-33.777	17.307
20	-33.777	-17.307
21	-0.098	0.000
22	-0.098	0.000
23	-0.099	0.001
24	-0.099	-0.001
25	-0.100	0.001
26	-0.100	-0.001
27	-0.099	0.000
28	-0.099	0.000
29	-0.099	0.001
30	-0.099	-0.001
31	-0.099	0.001
32	-0.099	-0.001
33	-0.099	0.001
34	-0.099	-0.001
35	-0.099	0.001
36	-0.099	-0.001
37	-0.099	0.001
38	-0.099	-0.001
39	-0.099	0.001
40	-0.099	-0.001

41	0.000	0.000
42	-0.100	0.000

The following points should be noted about these eigenvalues:

(3.1) The real parts are all negative, indicating the numerical solution of equations (63) and (64) is stable. Also, some of the imaginary parts are nonzero, indicating that some oscillation in the numerical solution might occur (although, as we shall see in examining the plotted numerical solution, any such oscillation is small, which is possible if the eigenvectors for the eigenvalues with nonzero imaginary parts are small).

(3.2) Eigenvalue 41 has both zero real and imaginary parts, at least to the three figures used in the printing (via a Fortran format used in subroutine EIGEN). This zero eigenvalue results from boundary condition (68), or in other words, from the first row of the preceding map which has all blank characters. More generally, each row of a Jacobian map with nothing but blanks (or all zeros in the row) will produce a zero eigenvalue.

(3.3) The smallest nonzero real part in the list of eigenvalues defines the problem time scale. In this case, this smallest value is -0.0910, which defines a time scale of approximately 100 since $e^{-0.910(100)}$ will be essentially negligible in comparison to 1. However, the numerical solution was calculated only over the interval $0 \leq t \leq 10$ as noted previously (see the preceding sets of data in Table 12) in order to elucidate the initial part of the solution, i.e., the calculation could have been continued to $t = 100$, but the plotted solutions would indicate mostly the steady state solution rather than the transient which is of primary interest in comparing the numerical and analytical solutions.

(3.4) The eigenvalue with the largest real part (in an absolute sense) defines the ODE system stiffness. In this case, this real part is -38.129, and when ratioed to the smallest value noted in (3.3), i.e., -38.128/-0.0910 = 419, indicates that the system of 42 ODEs is moderately stiff; we are able to use a nonstiff (explicit) time integrator, in this case the Runge Kutta Fehlberg method, and still have reasonable computer run times. More generally, the stiffness ratio would have to be about at least 1000 to consider the problem stiff, and therefore require a stiff (implicit) integrator. Another way to state the onset of stiffness is the condition

$$(|\text{largest eigenvalue}|)(\text{problem time scale}) >> 1$$

In the present case, this product is (38.128)(100) = 3812.8, which again indicates moderate stiffness (a product of 10^6, for example, would indicate a stiff problem).

(4) The numerical and analytical solutions are next printed by subroutine PRINT1

T	V(X,T)	V(X.T) ANAL
0.00	0.0000	0.0000
1.00	0.0378	0.0378
2.00	0.1082	0.1082
3.00	0.1737	0.1737
4.00	0.2346	0.2346
5.00	0.2911	0.2912
6.00	0.3436	0.3437
7.00	0.3924	0.3925
8.00	0.4377	0.4377
9.00	0.4797	0.4798
10.00	0.5187	0.5187

The agreement between the numerical and exact solutions is generally to about 3+ figures. The differences between the numerical and analytical solutions can be better understood by considering Figure 63 to 66 produced by subroutine PRINT1 from the four runs (the four sets of data in Table 12 corresponding to NORUN = 1 to 4).

(1) Figure 63: Boundary condition (68) is smooth in the sense of no inconsistency between it and the initial conditions (equations (65) and (67) with V(x,0) = U(x.0) = 0). Therefore, the differences between the numerical and analytical solution are imperceptible. As an incidental point, note that the solution has not reached steady state condition at t = 10, as noted previously, but if we continued the calculation beyond t = 10, the initial transient to t = 10 would be less obvious; this point is clearer in Figures 64 and 65.

(2) Figure 64: Boundary condition (70) introduces a step change at x = 0 and t = 0, so that the numerical solution oscillates slightly for short t. Generally, however, the agreement between the numerical and analytical solutions is acceptable, indicating again that the five point biased upwind approximations perform satisfactorily for this problem. Note also that if the calculation

were continued to t = 100, the initial response to the discontinuity would be difficult to discern.

(3) Figure 65: Boundary condition (71) again introduces a discontinuity at x = 0 and t = 0, so the conclusions for Figure 64 also apply here.

(4) Figure 66: Boundary condition (71), a unit step change, is modified to a ramp with a finite slope. As in preceding problems with a single PDE, when the ramp is used, the numerical and analytical solutions agree closely. We should also indicate how the analytical solution for the ramp function is computed. The unit ramp can be considered as the time integration of the sum of two step functions according to the equation

$$V(0,t) = \int_0^t \left\{ h(\lambda) - h(\lambda - 1) \right\} d\lambda \tag{77}$$

The integrand of equation (77) is computed in two parts in DERV4, as VA(1) and VA(2). These terms are integrated to VAI(1) and VAI(2). The analytical solution is then the integral of the sum of these four terms, i.e., V(0,t) = VAI(3), for which the integrand is programmed as

VA(3)=VA(1)+VAI(1)+VA(2)+VAI(2)

Therefore, three integrals, VAI(1), VAI(2) and VAI(3), are computed, and the data specfies 42 + 3 = 45 ODEs for NORUN = 4. The unit ramp itself, r(t), is programmed in DERV4 as

```
C...
C...  BOUNDARY CONDITION
      IF(T.LE.1.D+0)U1(1)=T
      IF(T.GT.1.D+0)U1(1)=1.D+0
      U1T(1)=0.D+0
```

In conclusion, we observe that the approximation of PDEs by systems of ODEs produces solutions of acceptable accuracy, and the coding of the PDEs is straightforward.

(11) Nonlinear Advection - Burgers' Equation

We conclude this discussion of numerical methods for strongly convective flows by considering

a famous test problem, Burgers' equation

$$u_t = -uu_x + \mu u_{xx} \tag{78}$$

where μ is a "viscosity" that can be used to adjust the relative magnitudes of the nonlinear convective term, $-uu_x$, and the linear diffusion term, μu_{xx}. Thus, for small μ, equation (78) is strongly hyperbolic, while for large μ, it is parabolic. The nonlinear advection term, $-uu_x$, produces some interesting properties in u(x,t) which did not occur with the linear advection equation (7). If we consider u to be a velocity in the term $-uu_x$, in analogy with the velocity v in the term $-vu_x$ of equation (7), we see for large u, the velocity is large (in $-uu_x$). Thus, if u is large for small x, and small for large x, the velocity at the left end of the system is higher than at the right end, and the solution will tend toward a steep front for flow from left to right, a so-called "front sharpening" feature of the solution; we will observe this front sharpening in a solution to be discussed next. If, however, u is small for small u, and large for large u, the velocity at the left end of the system is smaller than at the right end, and the solution will tend to smooth out with t.

Equation (78) requires one initial condition and two boundary conditions, which we take as

$$u(0,x) = \phi(0,x) \tag{79}$$

$$u(t,0) = \phi(t,0),\ u(1,t) = \phi(1,t) \tag{80)(81}$$

where

$$\phi(t,x) = \frac{0.1e^{-a} + 0.5e^{-b} + e^{-c}}{e^{-a} + e^{-b} + e^{-c}} \tag{82}$$

$$a = (0.05/\mu)(x - 0.5 + 4.95t)$$

$$b = (0.25/\mu)(x - 0.5 + 0.75t)$$

$$c = (0.5/\mu)(x - 0.375)$$

Additionally, $\phi(t.x)$ is the solution to equation (78) [Madsen and Sincovec (16)].

We can now list the reasons why Burgers' equation is considered a good test problem

(1) It is nonlinear, but has a known exact solution.

(2) It propagates steep fronts, depending on the choice of μ; in other words, its character can be changed from predominantly parabolic to predominantly hyperbolic.

(3) The solution has the front sharpening characteristic mentioned previously due to the nonlinear term $-uu_x$, and it therefore provides a stringent test of any numeriocal method of solution .

(4) Burgers' equation can easily be extended to two and three dimensions (with known exact solutions).

(5) The advection group $u_t + uu_x$ plays an important role in many applications, e.g., it appears in the Euler equations and the Navier Stokes equations.

We now consider the programming of a numerical solution of equations (78) in the format developed for the previous example, i.e., through a set of approximating ODEs. Subroutines INITAL, DERV and PRINT, and function PHI, which implements equation (82), is listed below

```
      SUBROUTINE INITAL
C...
C...  BURGERS' EQUATION IN ONE DIMENSION
C...
C...  SET NUMBER OF SPATIAL GRID POINTS
      PARAMETER(NX=101)
      COMMON/T/          T,       NFIN,      NORUN
     1      /Y/      U(NX)
     2      /F/     UT(NX)
     3     /SD/     UX(NX),    UXX(NX)
     4      /C/         XL,         XU,         DX,         VIS
C...
C...  PROBLEM PARAMETERS
      XL=0.
      XU=1.
      DX=(XU-XL)/FLOAT(NX-1)
      VIS=0.003
C...
C...  INITIAL CONDITION
      DO 1 I=1,NX
      X=DX*FLOAT(I-1)
```

```
      U(I)=PHI(0.,X)
1     CONTINUE
C...
C...  INITIALIZE THE DERIVATIVES, UX, UXX AND UT FOR PRINTING IN SUB-
C...  ROUTINE PRINT
      CALL DERV
      RETURN
      END

      REAL FUNCTION PHI(T,X)
C...
C...  FUNCTION PHI(T,X) COMPUTES THE EXACT SOLUTION TO BURGERS
C...  EQUATION.  IT IS USED IN THREE PLACES
C...
C...     (1)  TO PROVIDE THE EXACT SOLUTION IN SUBROUTINE PRINT FOR
C...          COMPARISON WITH THE NUMERICAL SOLUTION.
C...
C...     (2)  TO PROVIDE THE INITIAL CONDITION IN SUBROUTINE INITAL.
C...
C...     (3)  TO PROVIDE THE BOUNDARY CONDITIONS IN SUBROUTINE DERV.
C...
C...  ARGUMENT LIST
C...
C...      T   INITIAL-VALUE INDEPENDENT VARIABLE IN BURGERS EQUATION
C...          (INPUT)
C...
C...      X   BOUNDARY-VALUE INDEPENDENT VARIABLE IN BURGERS EQUATION
C...          (INPUT)
C...
C...  THE VALUE OF THE FUNCTION (PHI) IS THE EXACT SOLUTION TO BURGERS
C...  EQUATION AT X AND T.
C...
C...  TYPE REAL VARIABLES AS SINGLE PRECISION
C...
      COMMON/C/        XL,        XU,        DX,        VIS
      A=(0.05E+00/VIS)*(X-0.5E+00+4.95E+00*T)
      B=(0.25E+00/VIS)*(X-0.5E+00+0.75E+00*T)
      C=( 0.5E+00/VIS)*(X-0.375E+00)
      EA=EXP(-A)
      EB=EXP(-B)
      EC=EXP(-C)
C...
C...  THE FOLLOWING IF WAS ADDED FOR SHORT WORD LENGTH COMPUTERS, E.G.,
C...  32-BIT COMPUTERS SUCH AS THE VAX, WHICH CANNOT ACCOMMODATE WIDE
C...  VARIATIONS IN THE EXP FUNCTION.  THE INTENTION IS TO EFFECTIVELY
C...  AVOID A DIVISION BY ZERO.  IT IS BASED ON THE OBSERVATION THAT
C...  WHEN THE DIVISION BY ZERO OCCURS, EC MLT EB MLT EA MLT (WHERE
C...  MLT DENOTES "MUCH LESS THAN") SO THE EQUATION FOR PHI BECOMES
C...
```

```
C...      PHI = 0.1*EA/EA = 0.1
C...
      IF(ABS(EA+EB+EC).LT.1.0E-35)THEN
         PHI=0.1E+00
      ELSE
         PHI=(0.1E+00*EA+0.5E+00*EB+EC)/(EA+EB+EC)
      END IF
      RETURN
      END

      SUBROUTINE DERV
      PARAMETER(NX=101)
      COMMON/T/          T,        NFIN,      NORUN
     1       /Y/      U(NX)
     2       /F/     UT(NX)
     3      /SD/     UX(NX),    UXX(NX)
     4       /C/         XL,          XU,          DX,         VIS
C...
C...  BOUNDARY CONDITIONS
      U( 1)=PHI(T,0.)
      U(NX)=PHI(T,1.)
      UT( 1)=0.
      UT(NX)=0.
C...
C...  UXX BY FIVE POINT CENTERED APPROXIMATIONS
      NL=1
      NU=1
      CALL DSS044(XL,XU,NX,U,UX,UXX,NL,NU)
C...
C...  UX BY FIVE POINT BIASED UPWIND APPROXIMATIONS
      CALL DSS020(XL,XU,NX,U,UX,1.)
C...
C...  PDE
      NM1=NX-1
      DO 1 I=2,NM1
      UT(I)=VIS*UXX(I)-U(I)*UX(I)
1     CONTINUE
      RETURN
      END

      SUBROUTINE PRINT(NI,NO)
C...
C...  NP IS THE NUMBER OF SOLUTION CURVES TO BE PLOTTED
      PARAMETER(NX=101,NP=5)
      COMMON/T/          T,        NFIN,      NORUN
     1       /Y/      U(NX)
     2       /F/     UT(NX)
     3      /SD/     UX(NX),    UXX(NX)
     4       /C/         XL,          XU,          DX,         VIS
```

```
C...
C...  DIMENSION THE ARRAYS FOR STORING THE ANALYTICAL AND NUMERICAL
C...  SOLUTIONS
      DIMENSION UNP(NP,NX), UAP(NP,NX), XP(NX)
      DATA IP/0/
C...
C...  BECAUSE OF THE VOLUME OF PRINTED OUTPUT, THE NUMERICAL AND EXACT
C...  SOLUTIONS ARE PRINTED ONLY AT THE BEGINNING AND END OF THE RUN,
C...  I.E., T = 0, 0.8
      IP=IP+1
      IF((IP.EQ.1).OR.(IP.EQ.NP))THEN
C...
C...  PRINT A HEADING FOR THE NUMERICAL SOLUTION
      WRITE(NO,3)T
3     FORMAT(1H ,//,5H T = ,F5.2,/,
     1 9X,1HX,5X,5HU NUM,4X,6HU ANAL,5X,5HERROR,3X,7HVIS*UXX,5X,5H-U*UX,
     2 8X,2HUT)
C...
C...  COMPUTE THE EXACT SOLUTION, ERROR IN THE NUMERICAL SOLUTION AND
C...  THE INDIVIDUAL TERMS IN THE PDE
      DO 1 I=1,NX
      X=DX*FLOAT(I-1)
      UANAL=PHI(T,X)
      ERROR=U(I)-UANAL
      DIFF=VIS*UXX(I)
      CONV=-U(I)*UX(I)
C...
C...  PRINT THE VARIOUS ELEMENTS OF THE NUMERICAL SOLUTION, INCLUDING
C...  UT COMPUTED BY DERV AND AVAILABLE THROUGH COMMON/F/
      WRITE(NO,2)X,U(I),UANAL,ERROR,DIFF,CONV,UT(I)
2     FORMAT(F10.3,6F10.5)
1     CONTINUE
      END IF
C...
C...  STORE THE NUMERICAL AND ANALYTICAL SOLUTIONS FOR PLOTTING
      DO 4 I=1,NX
         XP(I)=DX*FLOAT(I-1)
         UNP(IP,I)=U(I)
         UAP(IP,I)=PHI(T,XP(I))
4     CONTINUE
C...
C...  CREATE AN OUTPUT FILE FOR TOP DRAWER PLOTTING
      IF((NORUN.EQ.1).AND.(IP.EQ.NP))THEN
         OPEN(4,FILE='T.TOP',STATUS='NEW')
         WRITE(4,5)
5        FORMAT(' SET LIMITS X FROM 0 TO 1 Y FROM 0 TO 1.25',/,
     1           ' SET FONT DUPLEX')
      END IF
C...
```

```
C...   WRITE TOP DRAWER FILE FOR PLOTTING NUMERICAL AND ANALYTICAL
C...   SOLUTIONS
       IF(IP.EQ.NP)THEN
C...
C...   ANALYTICAL SOLUTION
       DO 11 IP=1,NP
          WRITE(4,6)(XP(I),UAP(IP,I),I=1,NX)
6         FORMAT(2F10.4)
          WRITE(4,7)
7         FORMAT(' JOIN')
11     CONTINUE
C...
C...   NUMERICAL SOLUTION
       DO 12 IP=1,NP
          WRITE(4,8)IP
8         FORMAT(' SET SYMBOL ',I1)
          WRITE(4,6)(XP(I),UNP(IP,I),I=1,NX)
          WRITE(4,9)
9         FORMAT(' PLOT')
12     CONTINUE
C...
C...   TITLES
          WRITE(4,10)
10        FORMAT(
     +    ' Title 4.0 9.5 " Fig. 67: u(x,t) vs x,'
     +    ' t = 0, 0.2, 0.4, 0.6, 0.8"'
     +    ,/,' TITLE LEFT " u(x,t)"'
     +    ,/,' TITLE BOTTOM "x"'
     +    ,/,' TITLE 4.5 0.75 '
     +    '"Solid - exact; 1, 2, 3, 4, 5 - numerical "')
       IP=0
       END IF
       RETURN
       END
```

Program 10: Subroutines INITAL, DERV, PRINT and Function PHI

for Equations (78) to (82)

We can note the following points about Program 10:

(1) In INITAL we see the solution is computed on a 101 point grid in x (NX=101), and μ = 0.003 (VIS = 0.003). Initial condition (79) is set through the use of function PHI(T,X) in DO loop 1 (with T = 0). PHI follows INITAL in the listing of Program 10; the coding is apparent from equation (82).

(2) In DERV, boundary conditions (80) and (81) are first defined using function PHI(T,X) (with X = 0 and 1, respectively). Then the derivative u_{XX} in equation (78) is computed by a call to subroutine DSS044 which accepts u (array U(NX)) as an input and differentiates it to u_{XX} (array UXX(NX)). If Dirichlet boundary conditions are specified, as they are in this case, i.e., equations (80) and (81), NL = NU = 1. Subroutines DSS044 also can accept u_X at the boundaries as an input (if boundary conditions in u_X are specified, i.e., Neumann boundary conditions, for which NL = NU =2). The details of the differentiation formulas in DSS044 are discussed elsewhere [Schiesser (2)].

(3) The derivative u_X in equation (78) is then computed from u in the usual way by subroutine DSS020 for positive velocity (since the solution, u, is positive).

(4) Equation (78) is then implemented in DO loop 1 over the grid points i = 2 to NX-1. Note that the derivatives UT(1) and UT(NX) were previously set as part of boundary conditions (80) and (81). Also, the nonlinear term $-uu_X$ is easily programmed.

(5) Subroutine PRINT prints the numerical and analytical solutions, as well as all of the terms in equation (78), at the initial and final values of t (t = 0 and t = 0.8). The two solutions are then stored in arrays for subsequent plotting which produces Figure 67.

The data read by the DSS/2 main program are

```
MADSEN, ET AL, LAPIDUS, ET AL, NUM METH DIFF SYSTEMS, PP. 236-237
0.        0.8        0.2
  101 9999   15     1  REL 0.001
END OF RUNS
```

Table 13: Data for Program 10

Again, the ODE integration is done by the Rungke Kutta Fehlberg method in integrator 15.

Abbreviated numerical output from Program 10 is listed in Table 14

```
RUN NO.  1 - MADSEN, ET AL, LAPIDUS, ET AL, NUM METH
             DIFF SYSTEMS, PP. 236-237
```

```
INITIAL VALUE OF TIME =  0.0000E+00

FINAL VALUE OF TIME =  0.8000E+00

PRINT INTERVAL OF TIME =  0.2000E+00

NUMBER OF FIRST-ORDER DIFFERENTIAL EQUATIONS = 101

PRINT INTERVAL/MINIMUM INTEGRATION INTERVAL =  9999

INTEGRATION ALGORITHM =  15
      1 - RUNGE KUTTA EULER
      2 - RUNGE KUTTA NIESSE
      3 - RUNGE KUTTA MERSON
      4 - RUNGE KUTTA TANAKA - 4
      5 - RUNGE KUTTA TANAKA - 5
      6 - RUNGE KUTTA CHAI
      7 - RUNGE KUTTA ENGLAND
      8 - RUNGE KUTTA WES - 4/1
      9 - RUNGE KUTTA WES - 4/2
     10 - RUNGE KUTTA WES - 4/3
     11 - RUNGE KUTTA WES - 4/4
     12 - RUNGE KUTTA WES - 4/5
     13 - RUNGE KUTTA WES - 5/1
     14 - RUNGE KUTTA WES - 5/2
     15 - RUNGE KUTTA FEHLBERG - RKF45

PRINT OPTION = 1
     NO INTEGRATION ERROR DIAGNOSTICS - 0
     SUMMARY OF INTEGRATION ERRORS    - 1

TYPE OF INTEGRATION ERROR = REL

MAXIMUM INTEGRATION ERROR =  0.100E-02
```

T = 0.00

X	U NUM	U ANAL	ERROR	VIS*UXX	-U*UX	UT
0.000	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.010	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
	.				.	
	.				.	
	.				.	
0.100	1.00000	1.00000	0.00000	-0.00004	0.00015	0.00011
0.110	1.00000	1.00000	0.00000	-0.00009	0.00037	0.00028
0.120	0.99999	0.99999	0.00000	-0.00020	0.00085	0.00065
0.130	0.99998	0.99998	0.00000	-0.00047	0.00192	0.00145
0.140	0.99995	0.99995	0.00000	-0.00109	0.00441	0.00332
0.150	0.99988	0.99988	0.00000	-0.00248	0.01015	0.00767

0.160	0.99972	0.99972	0.00000	-0.00572	0.02329	0.01756
0.170	0.99936	0.99936	0.00000	-0.01312	0.05354	0.04042
0.180	0.99854	0.99854	0.00000	-0.03001	0.12264	0.09263
0.190	0.99665	0.99665	0.00000	-0.06812	0.27925	0.21114
0.200	0.99237	0.99237	0.00000	-0.15176	0.62762	0.47586
0.210	0.98278	0.98278	0.00000	-0.32386	1.36955	1.04570
0.220	0.96207	0.96207	0.00000	-0.62384	2.80639	2.18255
0.230	0.92057	0.92057	0.00000	-0.95035	5.07642	4.12607
0.240	0.84853	0.84853	0.00000	-0.85094	7.39095	6.54001
0.250	0.75000	0.75000	0.00000	-0.00001	7.86123	7.86123
0.260	0.65147	0.65147	0.00000	0.85097	5.84359	6.69455
0.270	0.57943	0.57943	0.00000	0.95031	3.22002	4.17033
0.280	0.53793	0.53793	0.00000	0.62385	1.52434	2.14818
0.290	0.51722	0.51722	0.00000	0.32386	0.70024	1.02410
0.300	0.50763	0.50763	0.00000	0.15176	0.31937	0.47113
0.310	0.50335	0.50335	0.00000	0.06810	0.14331	0.21140
0.320	0.50146	0.50146	0.00000	0.02998	0.06339	0.09337
0.330	0.50063	0.50063	0.00000	0.01306	0.02789	0.04095
0.340	0.50027	0.50027	0.00000	0.00559	0.01241	0.01800
0.350	0.50010	0.50010	0.00000	0.00225	0.00588	0.00812
0.360	0.50002	0.50002	0.00000	0.00061	0.00347	0.00408
0.370	0.49995	0.49995	0.00000	-0.00044	0.00331	0.00286
0.380	0.49988	0.49988	0.00000	-0.00158	0.00493	0.00335
0.390	0.49974	0.49974	0.00000	-0.00338	0.00893	0.00556
0.400	0.49949	0.49949	0.00000	-0.00670	0.01709	0.01038
0.410	0.49901	0.49901	0.00000	-0.01305	0.03305	0.02000
0.420	0.49808	0.49808	0.00000	-0.02521	0.06387	0.03866
0.430	0.49627	0.49627	0.00000	-0.04825	0.12275	0.07449
0.440	0.49281	0.49281	0.00000	-0.09082	0.23304	0.14222
0.450	0.48622	0.48622	0.00000	-0.16541	0.43225	0.26684
0.460	0.47401	0.47401	0.00000	-0.28270	0.76772	0.48502
0.470	0.45232	0.45232	0.00000	-0.42746	1.26212	0.83466
0.480	0.41656	0.41656	0.00000	-0.51192	1.82387	1.31195
0.490	0.36430	0.36430	0.00000	-0.38047	2.17092	1.79044
0.500	0.30000	0.30000	0.00000	0.00000	2.00833	2.00834
0.510	0.23570	0.23570	0.00000	0.38047	1.42093	1.80140
0.520	0.18344	0.18344	0.00000	0.51192	0.80926	1.32118
0.530	0.14768	0.14768	0.00000	0.42747	0.40984	0.83731
0.540	0.12599	0.12599	0.00000	0.28270	0.20167	0.48437
0.550	0.11378	0.11378	0.00000	0.16542	0.10029	0.26570
0.560	0.10719	0.10719	0.00000	0.09082	0.05059	0.14141
0.570	0.10373	0.10373	0.00000	0.04826	0.02575	0.07401
0.580	0.10192	0.10192	0.00000	0.02522	0.01316	0.03838
0.590	0.10099	0.10099	0.00000	0.01306	0.00674	0.01981
0.600	0.10051	0.10051	0.00000	0.00674	0.00346	0.01020
0.610	0.10026	0.10026	0.00000	0.00347	0.00178	0.00524
0.620	0.10013	0.10013	0.00000	0.00178	0.00091	0.00269
0.630	0.10007	0.10007	0.00000	0.00092	0.00047	0.00138
0.640	0.10004	0.10004	0.00000	0.00047	0.00024	0.00071

0.650	0.10002	0.10002	0.00000	0.00024	0.00012	0.00036
0.660	0.10001	0.10001	0.00000	0.00012	0.00006	0.00019
0.670	0.10000	0.10000	0.00000	0.00006	0.00003	0.00010
0.680	0.10000	0.10000	0.00000	0.00003	0.00002	0.00005
0.690	0.10000	0.10000	0.00000	0.00002	0.00001	0.00002
0.700	0.10000	0.10000	0.00000	0.00001	0.00000	0.00001
	.				.	
	.				.	
	.				.	
0.990	0.10000	0.10000	0.00000	0.00000	0.00000	0.00000
1.000	0.10000	0.10000	0.00000	-0.00001	0.00000	0.00000

T = 0.80

X	U NUM	U ANAL	ERROR	VIS*UXX	-U*UX	UT
0.000	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.010	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.020	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.030	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.040	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.050	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.060	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.070	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.080	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.090	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
0.100	1.00000	1.00000	0.00000	0.00000	0.00000	0.00000
	.				.	
	.				.	
	.				.	
0.710	1.00000	0.99999	0.00000	-0.00003	0.00010	0.00007
0.720	1.00000	0.99999	0.00001	0.00016	0.00001	0.00016
0.730	1.00000	0.99996	0.00004	0.00068	-0.00096	-0.00029
0.740	1.00002	0.99985	0.00016	0.00005	-0.00021	-0.00016
0.750	1.00000	0.99946	0.00054	-0.02111	0.03819	0.01707
0.760	0.99921	0.99784	0.00137	-0.18150	0.35503	0.17336
0.770	0.99269	0.99100	0.00170	-0.95186	2.01338	1.05549
0.780	0.95842	0.96230	-0.00389	-3.17697	7.76351	4.48265
0.790	0.83463	0.85501	-0.02037	-4.69094	17.83987	12.37055
0.800	0.56480	0.58668	-0.02188	0.89814	17.97342	17.16959
0.810	0.28422	0.28932	-0.00510	4.92708	5.77132	10.01829
0.820	0.14962	0.15158	-0.00196	2.46054	0.65958	3.07189
0.830	0.11169	0.11261	-0.00092	0.58251	0.09650	0.67684
0.840	0.10314	0.10313	0.00001	0.11039	0.04788	0.15800
0.850	0.10099	0.10085	0.00015	0.02858	0.01739	0.04595
0.860	0.10034	0.10027	0.00007	0.00953	0.00465	0.01418
0.870	0.10012	0.10010	0.00002	0.00311	0.00128	0.00440
0.880	0.10005	0.10004	0.00001	0.00101	0.00044	0.00145
0.890	0.10002	0.10002	0.00000	0.00037	0.00018	0.00055
0.900	0.10001	0.10001	0.00000	0.00015	0.00008	0.00023

	.				.	
	.				.	
	.				.	
0.980	0.10000	0.10000	0.00000	0.00000	0.00000	0.00000
0.990	0.10000	0.10000	0.00000	0.00000	0.00000	0.00000
1.000	0.10000	0.10000	0.00000	-0.00001	0.00000	0.00000

Table 14: Abbreviated Output from Program 10

As expected, the numerical and analytical solutions are the same at t = 0. They are both computed from $\phi(0,x)$. The agreement between the two solutions is generally good, as indicated in Figure 67. Also, Figure 67 indicates how the solution front sharpens with increasing t, as discussed previously (note that u for small x is greater than u for large x initially, the condition for front sharpening). If the front cannot be resolved with sufficient accuracy, additional grid points may be required, or an adaptive grid that concentrates the grid points where the solution is changing rapidly in space may be required.

In summary, Program 10 demonstrates:

(1) The ease with which numerical solutions of nonlinear PDEs can be computed (the implementation of the nonlinear term $-uu_x$ is straightforward).

(2) The detailed examination of all of the terms in a PDE (e.g., via the printout in DO loop 1 of PRINT) which can be used to determine the behavior of the individual terms, and which terms dominate in the computation of the solution.

(12) Conclusion

We have investigated numerically a series of approximation methods for PDEs with strongly convective properties. In each case, the PDE had a known analytical solution that could be used to evaluate the numerical solution. The results of the tests of the various approximations are summarized in Figures 1 to 67. We conclude from these tests that the five point biased upwind approximations in subroutine DSS020 gave the best performance for the selected test problems, and can easily be used in the soluition of systems of PDEs, both linear and nonlinear. We have included all of our code so that other investigators can understand and replicate what we have done, and hopefully, improve on our

methods. In any case, we think the testing has been reported here in sufficient detail that there is no question about what we have done. If we have incorrectly applied numerical methods reported by others, we ask that our errors be reported so we can correct our code and then repeat the tests. For this purpose, we ask that communications concerning this work be directed to W. E. Schiesser.

We hope that the details of using the various numerical methods will be useful to others. In particular, the series of spatial differention routines and finite element routines can be applied to a spectrum of PDE problems with dominant convective properties. We hope that this software is useful to scientists and engineers who need to produce solutions to PDEs with these properties, and we welcome inquiries about our results. The software is available as Fortran 77 source code on DOS-formatted 3.5 inch 1.44 mb or 5.25 inch 1.2 mb diskettes.

References

(1) Forsythe, G. E., M. A. Malcolm and C. B. Moler, Computer Methods for Mathematical Computations, Prentice Hall, Inc., Englewood Cliffs, 1977

(2) Schiesser, W. E., The Numerical Method of Lines, Academic Press, New York, 1991

(3) Carver, Michael B., and H. W. Hinds, "The Method of Lines and the Advection Equation", Simulation, 31, no. 2 pp. 59-69, August, 1978

(4) Leonard, B. P., "A Stable and Accurate Convective Modeling Procedure Based on Quadratic Upstream Interpolation", Computer Methods in Applied Mechanics and Engineering, 19, no. 1, June, 1979. Republished in Computer Methods in Applied Mechanics and Engineering, Special Edition, pp 59-98, 1990

(5) Myint-U, Tyn, and Lokenath Debnath, Partial Differential Equations for Scientists and Engineers", 3rd ed., North Holland, New York, 1987, p454

(6) Press, W. H., B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, Numerical Recipes, Cambridge University Press, Cambridge, 1986, pp 40-41

(7) Brooks, A. N., and T. J. R. Hughes, "Streamline upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations", Computer Methods in Applied Mechanics and Engineering, 32, nos. 1-3, September, 1982. Republished in Computer Methods in Applied Mechanics and Engineering, Special Edition, pp 199-259, 1990

(8) Patankar, S. V., Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing Corporation, New York, 1980

(9) Lick, W. J., Difference Equations from Differential Equations, Lecture Notes in Engineering No. 41, C. A. Brebbia and S. A. Orszag (eds.), Springer-Verlag, Berlin, 1989

(10) Bateman, H., Partial Differential Equations of Mathematical Physics, Cambridge at the University Press, 1932

(11) Strang, G., Introduction to Applied Mathematics, Wellesley-Cambridge Press, Wellesley, MA, 1986

(12) Carslaw, H. S., and J. C. Jaeger, Conduction of Heat in Solids, Oxford Science Publications and Oxford University Press, Clarendon Press - Oxford, Second Edition, 1959

(13) Beyer, W. H., CRC Handbook of Mathematical Sciences, 5th ed., CRC Press, Inc., West Palm Beach, FL, 1978

(14) Press, W. H., B. P. Flannery, S. A. Teukolsy and W. T. Vetterling, Numerical Recipes, Cambridge University Press, Cambridge, 1986

(15) Schiesser, W. E., DSS/2 Introductory Programming Manual, Lehigh University, Bethlehem, PA

(16) Madsen, N. K. and R. F. Sincovec, "Software for Partial Differential Equations", in Numerical Methods and Software for Differential Systems", L. Lapidus and W. E. Schiesser, eds., Academic Press, NY, 1976

Appendix 1: Solution of the Homogeneous Advection Equation

We derive here the analytical solution to the problem

$$u_t + vu_x = 0 \tag{a.1}$$

$$u(x,0) = g(x),\ u(0,t) = f(t) \tag{a.2)(a.3}$$

which is just a restatement of equations (7) to (9).

If the Laplace transform of u(x,t) with respect to t is defined as

$$L_t\left\{u(x,t)\right\} = \int_0^\infty u(x,t)e^{-st}dt = \bar{u}(x,s) \tag{a.4}$$

equations (a.1) to (a.3) transform to

$$s\bar{u}(x,s) - g(x) + v d\bar{u}(x,s)/dx = 0 \tag{a.5}$$

$$\bar{u}(0,s) = \bar{f}(s) \tag{a.6}$$

If the Laplace transform of $\bar{u}(x,s)$ with respect to x is defined as

$$L_x\left\{\bar{u}(x,s)\right\} = \int_0^\infty \bar{u}(x,s)e^{-px}dx = \bar{\bar{u}}(p,s) \tag{a.7}$$

equations (a.5) and (a.6) transform to

$$s\bar{\bar{u}}(p,s) - \bar{g}(p) + v\{p\bar{\bar{u}}(p,s) - \bar{f}(s)\} = 0 \tag{a.8}$$

Solution of equation (a.8) for $\bar{\bar{u}}(p,s)$ gives

$$\bar{\bar{u}}(p,s) = (1/v)\frac{1}{(s/v + p)}\bar{g}(p) + \frac{1}{(s/v + p)}\bar{f}(s) \tag{a.9}$$

Equation (a.9) can now be inverted, first with respect to p

$$\bar{u}(x,s) = L^{-1}\left\{\bar{\bar{u}}(p,s)\right\} = (1/v)\int_0^x e^{-(s/v)(x-\lambda)}g(\lambda)d\lambda + e^{-(s/v)x}\,\bar{f}(s)$$

$$= (1/v)\,e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}g(\lambda)d\lambda + e^{-(s/v)x}\,\bar{f}(s)$$

$$= (1/v)\,\bar{G}(x,s) + e^{-(s/v)x}\,\bar{f}(s) \tag{a.10}$$

where

$$\bar{G}(x,s) = e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}g(\lambda)d\lambda \tag{a.11}$$

Then, inversion with respect to s gives

$$u(x,t) = L^{-1}\left\{\bar{u}(x,s)\right\} = (1/v)G(x,t) + f(t - x/v)h(t - x/v) \tag{a.12}$$

where

$$G(x,t) = L^{-1}\left\{\bar{G}(x,s)\right\} \tag{a.13}$$

$$h(t) = \begin{cases} 0, & t < 0 \\ 1, & t > 0 \end{cases} \tag{a.14}$$

Some special cases can now be considered:

(1) $g(x) = 0$ and equation (a.12) reduces to

$$u(x,t) = f(t - x/v)h(t - x/v) \tag{a.15}$$

which is equation (10). Two other special cases follow which could be used as test problems for numerical approximations.

(2) $g(x) = 1$ and equations (a.11) to (a.13) give

$$\bar{G}(x,s) = e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}d\lambda = (v/s)\{1 - e^{-(s/v)x}\} \tag{a.16}$$

and

$$G(x,t) = v\left\{h(t) - h(t - x/v)\right\}$$

and from equation (a.12)

$$u(x,t) = \left\{h(t) - h(t - x/v)\right\} + f(t - x/v)h(t - x/v) \tag{a.17}$$

For $t > x/v$, this reduces to

$$u(x,t) = \left\{1 - 1\right\} + f(t - x/v) = f(t - x/v) \tag{a.18}$$

(3) $g(x) = x$ and equations (a.11) to (a.13) give

$$\overline{G}(x,s) = e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}\lambda d\lambda = e^{-(s/v)x}\left\{(v/s)e^{(s/v)\lambda}\lambda\Big|_0^x - (v/s)\int_0^x e^{(s/v)\lambda}d\lambda\right\}$$

$$= e^{-(s/v)x}\left\{(v/s)e^{(s/v)x}x - (v/s)^2[e^{(s/v)x} - 1]\right\}$$

$$= (v/s)x - (v/s)^2[1 - e^{-(s/v)x}] \tag{a.19}$$

and

$$G(x,t) = vxh(t) - v^2[t - (t - x/v)h(t - x/v)] \tag{a.20}$$

$$u(x,t) = xh(t) - v[t - (t - x/v)h(t - x/v)] + f(t - x/v)h(t - x/v) \tag{a.21}$$

For $t > x/v$, this reduces to

$$u(x,t) = x - v[t - (t - x/v)] + f(t - x/v) = f(t - x/v) \tag{a.22}$$

Thus, in all three cases, the solution is the same for $t > x/v$ indicating that by this time, the initial condition $u(x,0) = g(x)$ has no effect (i.e., the initial condition has "left or flowed out of the system").

Appendix 2: Listing of Subroutine DSS012

```
      SUBROUTINE DSS012(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE DSS012 IS AN APPLICATION OF FIRST-ORDER DIRECTIONAL
C...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
C...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
C...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS WITH THE
C...  SIMPLEST FORM
C...
C...                                U   + V*U   = 0                    (1)
C...                                 T       X
C...
C...  THE FIRST FIVE PARAMETERS, XL, XU, N, U AND UX, ARE THE SAME
C...  AS FOR SUBROUTINES DSS002 TO DSS010 AS DEFINED IN THOSE ROUTINES.
C...  THE SIXTH PARAMETER, V, MUST BE PROVIDED TO DSS012 SO THAT THE
C...  DIRECTION OF FLOW IN EQUATION (1) CAN BE USED TO SELECT THE
C...  APPROPRIATE FINITE DIFFERENCE APPROXIMATION FOR THE FIRST-ORDER
C...  SPATIAL DERIVATIVE IN EQUATION (1), U .  THE CONVENTION FOR THE
C...  SIGN OF V IS                          X
C...
C...     FLOW LEFT TO RIGHT                  V GT 0
C...     (I.E., IN THE DIRECTION             (I.E., THE SIXTH ARGUMENT IS
C...     OF INCREASING X)                    POSITIVE IN CALLING DSS012)
C...
C...     FLOW RIGHT TO LEFT                  V LT 0
C...     (I.E., IN THE DIRECTION             (I.E., THE SIXTH ARGUMENT IS
C...     OF DECREASING X)                    NEGATIVE IN CALLING DSS012)
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFERENCE
C...  APPROXIMATION DEPENDING ON THE SIGN OF V IN EQUATION (1).  THE
C...  ORIGIN OF THE FINITE DIFFERENCE APPROXIMATIONS USED BELOW IS GIVEN
C...  AT THE END OF SUBROUTINE DSS012.
      DX=(XU-XL)/FLOAT(N-1)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
              UX(1)=(U(2)-U(1))/DX
              DO 1 I=2,N
              UX(I)=(U(I)-U(I-1))/DX
1             CONTINUE
              RETURN
C...
C...     (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
10            NM1=N-1
              DO 2 I=1,NM1
              UX(I)=(U(I+1)-U(I))/DX
2             CONTINUE
```

```
      UX(N)=(U(N)-U(N-1))/DX
      RETURN
C...
C...  THE BACKWARD DIFFERENCES IN SECTION (1) ABOVE ARE BASED ON THE
C...  TAYLOR SERIES
C...
C...                                          2            3
C...  UI-1 = UI + UI  (-DX) + UI   (-DX)  + UI   (-DX)  + ...
C...                X  1F       2X  2F       3X  3F
C...
C...                                                 2
C...  IF THIS SERIES IS TRUNCATED AFTER THE DX   TERM AND THE RESULTING
C...  EQUATION SOLVED FOR U ,  WE OBTAIN IMMEDIATELY
C...                       X
C...
C...  UI   = (UI - UI-1)/DX + O(DX)
C...    X
C...
C...  WHICH IS THE FIRST-ORDER BACKWARD DIFFERENCE USED IN DO LOOP 1.
C...  THE DERIVATIVE U1   IS COMPUTED BY USING THE POINT TO THE RIGHT OF
C...                  X
C...  U1, I.E., U2, SINCE THIS IS THE ONLY POINT AVAILABLE IF FICTITIOUS
C...  POINTS TO THE LEFT OF U1 ARE TO BE AVOIDED.
C...
C...  THE FORWARD DIFFERENCES IN SECTION (2) ABOVE ARE BASED ON THE
C...  TAYLOR SERIES
C...
C...                                          2            3
C...  UI+1 = UI + UI  ( DX) + UI   ( DX)  + UI   ( DX)  + ...
C...                X  1F       2X  2F       3X  3F
C...
C...                                                 2
C...  IF THIS SERIES IS TRUNCATED AFTER THE DX   TERM AND THE RESULTING
C...  EQUATION SOLVED FOR U ,  WE OBTAIN IMMEDIATELY
C...                       X
C...
C...  UI   = (UI+1 - UI)/DX + O(DX)
C...    X
C...
C...  WHICH IS THE FIRST-ORDER FORWARD DIFFERENCE USED IN DO LOOP 2.
C...  THE DERIVATIVE UN   IS COMPUTED BY USING THE POINT TO THE LEFT OF
C...                  X
C...  UN (UN-1), SINCE THIS IS THE ONLY POINT AVAILABLE IF FICTITIOUS
C...  POINTS TO THE RIGHT OF UN ARE TO BE AVOIDED.
      END
```

Appendix 3: Numerical Diffusion in the Two Point Upwind Approximation of the Advection Equation

Consider again equation (16)

$$\frac{du_i}{dt} = -v\left(\frac{u_i - u_{i-1}}{\Delta x}\right) \tag{a.23}$$

which is the two point upwind approximation of the advection equation (7). Typically, in the method of lines, du_i/dt is integrated by an ODE library routine such as RKF45 which automatically adjusts the integration step size in t, Δt, to meet a user-specified error criterion. However, for the purpose of analyzing the effect of the two point upwind approximation in the RHS of equation (a.23), we approximate the derivative by a first order forward difference, which would correspond to a fixed step integration in t

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -v\left(\frac{u_i^n - u_{i-1}^n}{\Delta x}\right) \tag{a.24}$$

where n and i are the time and space indices, respectively.

u_i^{n+1} can be expanded in a Taylor series as

$$u_i^{n+1} = u_i^n + \frac{\partial u_i^n}{\partial t}\Delta t + \frac{\partial^2 u_i^n}{\partial t^2}\frac{\Delta t^2}{2!} + \frac{\partial^3 u_i^n}{\partial t^3}\frac{\Delta t^3}{3!} + \ldots \tag{a.25}$$

Similarly, u_{i-1}^n can be expanded in a Taylor series as

$$u_{i-1}^n = u_i^n + \frac{\partial u_i^n}{\partial x}(-\Delta x) + \frac{\partial^2 u_i^n}{\partial x^2}\frac{(-\Delta x)^2}{2!} + \frac{\partial^3 u_i^n}{\partial x^3}\frac{(-\Delta x)^3}{3!} + \ldots \tag{a.26}$$

From equation (a.25), the LHS of equation (a.24) can be written as

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\partial u_i^n}{\partial t} + \frac{\partial^2 u_i^n}{\partial t^2}\frac{\Delta t}{2!} + \frac{\partial^3 u_i^n}{\partial t^3}\frac{\Delta t^2}{3!} + \ldots \tag{a.27}$$

Similarly, from equation (a.26), the RHS of equation (a.24) can be written as

$$-v\left\{\frac{u_i^n - u_{i-1}^n}{\Delta x}\right\} = -v\left\{\frac{\partial u_i^n}{\partial x} + \frac{\partial^2 u_i^n}{\partial x^2}\frac{(-\Delta x)}{2!} + \frac{\partial^3 u_i^n}{\partial x^3}\frac{(-\Delta x)^2}{3!} + \ldots\right\} \tag{a.28}$$

Subtraction of equation (a.28) from equation (a.27) then gives

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + v\left\{\frac{u_i^n - u_{i-1}^n}{\Delta x}\right\} =$$

$$= \frac{\partial u_i^n}{\partial t} + \frac{\partial^2 u_i^n}{\partial t^2}\frac{\Delta t}{2!} + \frac{\partial^3 u_i^n}{\partial t^3}\frac{\Delta t^2}{3!} + \ldots$$

$$+ v\left\{\frac{\partial u_i^n}{\partial x} + \frac{\partial^2 u_i^n}{\partial x^2}\frac{(-\Delta x)}{2!} + \frac{\partial^3 u_i^n}{\partial x^3}\frac{(-\Delta x)^2}{3!} + \ldots\right\}$$

$$= \left\{\frac{\partial u_i^n}{\partial t} + v\frac{\partial u_i^n}{\partial x}\right\} + \left\{\frac{\partial^2 u_i^n}{\partial t^2}\frac{\Delta t}{2!} + v\frac{\partial^2 u_i^n}{\partial x^2}\frac{(-\Delta x)}{2!}\right\}$$

$$+ \left\{\frac{\partial^3 u_i^n}{\partial t^3}\frac{\Delta t^2}{3!} + v\frac{\partial^3 u_i^n}{\partial x^3}\frac{\Delta x^2}{3!}\right\} + \ldots \tag{a.29}$$

Considering the RHS terms of equation (a.29),

$$\left\{\frac{\partial u_i^n}{\partial t} + v\frac{\partial u_i^n}{\partial x}\right\}$$

is just the advection group at grid points i and n. Thus, the additional terms represent departures from the advection group due to the finite differencing of the LHS, i.e., errors resulting from the approximation of the advection group by

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + v\left\{\frac{u_i^n - u_{i-1}^n}{\Delta x}\right\}$$

The second RHS group

$$\left\{\frac{\partial^2 u_i^n}{\partial t^2}\frac{\Delta t}{2!} + v\,\frac{\partial^2 u_i^n}{\partial x^2}\frac{(-\Delta x)}{2!}\right\} \tag{a.30}$$

can be rewritten by differentiating the advection equation (6)

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} \tag{a.31}$$

Differentiation of equation (a.31) with respect to t gives

$$\frac{\partial^2 u}{\partial t^2} = -v\frac{\partial^2 u}{\partial x \partial t} \tag{a.32}$$

Differentiation of equation (a.31) with respect to x gives

$$\frac{\partial^2 u}{\partial t \partial x} = -v\frac{\partial^2 u}{\partial x^2} \tag{a.33}$$

If the mixed partials of equations (a.32) and (a.33) are assumed to be equal, then equations (a.32) and (a.33) can be combined to give the second order wave equation.

$$\frac{\partial^2 u}{\partial t^2} = v^2\frac{\partial^2 u}{\partial x^2} \tag{a.34}$$

$\frac{\partial^2 u}{\partial t^2}$ from equation (a.34), evaluated at grid points i and n, can now be substituted in group (a.30)

$$\left\{v^2\frac{\partial^2 u_i^n}{\partial x^2}\frac{\Delta t}{2!} + v\,\frac{\partial^2 u_i^n}{\partial x^2}\frac{(-\Delta x)}{2!}\right\} = \left\{\frac{v^2\Delta t}{2!} - \frac{v\Delta x}{2!}\right\}\frac{\partial^2 u_i^n}{\partial x^2} \tag{a.35}$$

If equation (a.35) is substituted in equation (a.29), and the third derivatives terms are considered as higher order terms, the finite difference approximation of the advection equation becomes

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + v\left\{\frac{u_i^n - u_{i-1}^n}{\Delta x}\right\} =$$

$$\left\{\frac{\partial u_i^n}{\partial t} + v\frac{\partial u_i^n}{\partial x}\right\} + \left\{\frac{v^2\Delta t}{2!} - \frac{v\Delta x}{2!}\right\}\frac{\partial^2 u_i^n}{\partial x^2} + \ldots$$

$$= \frac{\partial u_i^n}{\partial t} + v\frac{\partial u_i^n}{\partial x} - D\frac{\partial^2 u_i^n}{\partial x^2} + \ldots \tag{a.36}$$

where D is a numerical diffusivity, i.e.,

$$D = \left\{\frac{v\Delta x}{2!} - \frac{v^2\Delta t}{2!}\right\} \tag{a.37}$$

In other words, the finite difference approximation of equation (a.24) includes a numerical diffusion term which accounts for the numerical diffusion observed in the solutions produced by the two point upwind approximations in subroutine DSS012.

The effect of this diffusion term can be negated if the RHS of equation (a.37) is set to zero, i.e.,

$$\frac{v\Delta x}{2!} = \frac{v^2\Delta t}{2!} \tag{a.38}$$

or $\frac{v\Delta t}{\Delta x} = 1$. This is the Courant-Friedrichs-Lewy (CFL) stabilty condition [Myint-U et al (5)]. In other words, setting $D = 0$ places the finite difference solution just at the border of instability. Practically, this cannot be done, and we require $\frac{v\Delta t}{\Delta x} < 1$ for stability, but this means the diffusion term in equation (a.36) remains.

Similar diffusion terms are introduced by the nine other approximations considered previously, which accounts for the observed numerical diffusion in all of the solutions in Figures 1 to 50. An analysis of the numerical diffusion of some of these approximations, particularly of the five point biased upwind approximations, is given by Carver and Hinds (3).

Appendix 4: Main Program SRKF45

```
      PROGRAM SRKF45
C...
C...  PROGRAM SRKF45 CALLS: (1) SUBROUTINE INITAL TO DEFINE THE ODE
C...  INITAL CONDITIONS, (2) SUBROUTINE RKF45 TO INTEGRATE THE ODES,
C...  AND (3) SUBROUTINE PRINT TO PRINT THE SOLUTION.
C...
C...  THE FOLLOWING CODING IS FOR 450 ODES.  IF MORE ODES ARE TO BE INTE-
C...  GRATED, ALL OF THE 450'S SHOULD BE CHANGED TO THE REQUIRED NUMBER
      COMMON/T/          T,      NSTOP,      NORUN
     1       /Y/     Y(450)
     2       /F/     F(450)
C...
C...  THE NUMBER OF DIFFERENTIAL EQUATIONS IS IN COMMON/N/ FOR USE IN
C...  SUBROUTINE FCN
      COMMON/N/       NEQN
C...
C...  COMMON AREA TO PROVIDE THE INPUT/OUTPUT UNIT NUMBERS TO OTHER
C...  SUBROUTINES
      COMMON/IO/        NI,         NO
C...
C...  ABSOLUTE DIMENSIONING OF THE ARRAYS REQUIRED BY RKF45
      DIMENSION YV(450), WORK(3000), IWORK(5)
C...
C...  EXTERNAL THE DERIVATIVE ROUTINE CALLED BY RKF45
      EXTERNAL FCN
C...
C...  ARRAY FOR THE TITLE (FIRST LINE OF DATA), CHARACTERS  END OF RUNS
      CHARACTER TITLE(20)*4, ENDRUN(3)*4
C...
C...  DEFINE THE CHARACTERS  END OF RUNS
      DATA ENDRUN/'END ','OF R','UNS '/
C...
C...  DEFINE THE INPUT/OUTPUT UNIT NUMBERS
      NI=5
      NO=6
C...
C...  OPEN INPUT AND OUTPUT FILES
      OPEN(NI,FILE=  'DATA',STATUS='OLD')
      OPEN(NO,FILE='OUTPUT',STATUS='NEW')
C...
C...  INITIALIZE THE RUN COUNTER
      NORUN=0
C...
C...  BEGIN A RUN
1     NORUN=NORUN+1
C...
C...  INITIALIZE THE RUN TERMINATION VARIABLE
      NSTOP=0
```

```
C...
C...  READ THE FIRST LINE OF DATA
      READ(NI,1000,END=999)(TITLE(I),I=1,20)
C...
C...  TEST FOR  END OF RUNS  IN THE DATA
      DO 2 I=1,3
      IF(TITLE(I).NE.ENDRUN(I))GO TO 3
2     CONTINUE
C...
C...  AN END OF RUNS HAS BEEN READ, SO TERMINATE EXECUTION
999   STOP
C...
C...  READ THE SECOND LINE OF DATA
3     READ(NI,1001,END=999)TO,TF,TP
C...
C...  READ THE THIRD LINE OF DATA
      READ(NI,1002,END=999)NEQN,ERROR
C...
C...  PRINT A DATA SUMMARY
      WRITE(NO,1003)NORUN,(TITLE(I),I=1,20),
     1                    TO,TF,TP,
     2                    NEQN,ERROR
C...
C...  INITIALIZE TIME
      T=TO
C...
C...  SET THE INITIAL CONDITIONS
      CALL INITAL
C...
C...  SET THE INITIAL DERIVATIVES (FOR POSSIBLE PRINTING)
      CALL DERV
C...
C...  PRINT THE INITIAL CONDITIONS
      CALL PRINT(NI,NO)
C...
C...  SET THE INITIAL CONDITIONS FOR SUBROUTINE RKF45
      TV=TO
      DO 5 I=1,NEQN
      YV(I)=Y(I)
5     CONTINUE
C...
C...  SET THE PARAMETERS FOR SUBROUTINE RKF45
      RELERR=ERROR
      ABSERR=ERROR
      IFLAG=-1
      TOUT=TO+TP
C...
C...  CALL SUBROUTINE RKF45 TO START THE SOLUTION FROM THE INITIAL
C...  CONDITION (IFLAG = 1 OR COMPUTE THE SOLUTION TO THE NEXT PRINT
C...  POINT (IFLAG = 2)
C...
```

```
C...   BECAUSE OF THE MANY OUTPUT POINTS, THE INTEGRATION INTERVAL
C...   OF RKF45 IS SO SEVERELY RESTRICTED THE USUAL FREE-RUNNING MODE
C...   MAY NOT OPERATE CORRECTLY (RKF45 REPORTS AN ERROR IFLAG = 7;
C...   SEE THE DOCUMENTATION IN SUBROUTINE RKF45 FOR THE MEANING OF
C...   THE VARIOUS VALUES OF IFLAG).  THEREFORE THE ONE-STEP MODE IS
C...   USED BY SETTING IFLAG = -1 (RATHER THAN THE USUAL INITIALIZATION
C...   IFLAG = 1)
4      IFLAG=-1
       CALL RKF45(FCN,NEQN,YV,TV,TOUT,RELERR,ABSERR,IFLAG,WORK,IWORK)
C...
C...   PRINT THE SOLUTION AT THE NEXT PRINT POINT
       T=TV
       DO 6 I=1,NEQN
       Y(I)=YV(I)
6      CONTINUE
       CALL DERV
       CALL PRINT(NI,NO)
C...
C...   TEST FOR AN ERROR CONDITION
       IF(IFLAG.NE.2)THEN
C...
C...      PRINT A MESSAGE INDICATING AN ERROR CONDITION
          WRITE(NO,1004)IFLAG
C...
C...      GO ON TO THE NEXT RUN
          GO TO 1
       END IF
C...
C...   CHECK FOR A RUN TERMINATION
       IF(NSTOP.NE.0)GO TO 1
C...
C...   CHECK FOR THE END OF THE RUN
       TOUT=TV+TP
       IF(TV.LT.(TF-0.5*TP))GO TO 4
C...
C...   THE CURRENT RUN IS COMPLETE, SO GO ON TO THE NEXT RUN
       GO TO 1
C...
C...   ***********************************************************************
C...
C...   FORMATS
C...
1000   FORMAT(20A4)
1001   FORMAT(3E10.0)
1002   FORMAT(I5,20X,E10.0)
1003   FORMAT(1H1,
     1 ' RUN NO. - ',I3,2X,20A4,//,
     2 ' INITIAL T - ',E10.3,//,
     3 '   FINAL T - ',E10.3,//,
     4 '   PRINT T - ',E10.3,//,
     5 ' NUMBER OF DIFFERENTIAL EQUATIONS - ',I3,//,
```

```
     6 ' MAXIMUM INTEGRATION ERROR - ',E10.3,//,
     7 1H1)
1004  FORMAT(1H ,//,' IFLAG = ',I3,//,
     1 ' INDICATING AN INTEGRATION ERROR, SO THE CURRENT RUN'      ,/,
     2 ' IS TERMINATED.  PLEASE REFER TO THE DOCUMENTATION FOR'    ,/,
     3 ' SUBROUTINE',//,25X,'RKF45',//,
     4 ' FOR AN EXPLANATION OF THESE ERROR INDICATORS'              )
      END

      SUBROUTINE FCN(TV,YV,YDOT)
C...
C...  SUBROUTINE FCN IS AN INTERFACE ROUTINE BETWEEN SUBROUTINES RKF45
C...  AND DERV
C...
C...  NOTE THAT THE SIZE OF ARRAYS Y AND F IN THE FOLLOWING COMMON AREA
C...  IS ACTUALLY SET BY THE CORRESPONDING COMMON STATEMENT IN MAIN
C...  PROGRAM PRO1P3
      COMMON/T/          T,      NSTOP,      NORUN
     1       /Y/       Y(1)
     2       /F/       F(1)
C...
C...  THE NUMBER OF DIFFERENTIAL EQUATIONS IS AVAILABLE THROUGH COMMON
C...  /N/
      COMMON/N/       NEQN
C...
C...  ABSOLUTE DIMENSION THE DEPENDENT VARIABLE, DERIVATIVE VECTORS
      REAL YV(450), YDOT(450)
C...
C...  TRANSFER THE INDEPENDENT VARIABLE, DEPENDENT VARIABLE VECTOR
C...  FOR USE IN SUBROUTINE DERV
      T=TV
      DO 1 I=1,NEQN
      Y(I)=YV(I)
1     CONTINUE
C...
C...  EVALUATE THE DERIVATIVE VECTOR
      CALL DERV
C...
C...  TRANSFER THE DERIVATIVE VECTOR FOR USE BY SUBROUTINE RKF45
      DO 2 I=1,NEQN
      YDOT(I)=F(I)
2     CONTINUE
      RETURN
      END
```

Appendix 5: Listing of Subroutine DSS014

```
      SUBROUTINE DSS014(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE DSS014 IS AN APPLICATION OF SECOND-ORDER DIRECTIONAL
C...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
C...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
C...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS AS DIS-
C...  CUSSED IN SUBROUTINE DSS012.  THE COEFFICIENTS OF THE FINITE
C...  DIFFERENCE APPROXIMATIONS USED HEREIN ARE TAKEN FROM BICKLEY, W.
C...  G., FORMULAE FOR NUMERICAL DIFFERENTIATION, THE MATHEMATICAL
C...  GAZETTE, PP. 19-27, 1941, N = 2, M = 1, P = 0, 1, 2.
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE COMMON FACTOR FOR EACH FINITE DIFFERENCE APPROXIMATION
C...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
C...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      DX=(XU-XL)/FLOAT(N-1)
      R2FDX=1./(2.*DX)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
      UX(1)=R2FDX*
     1(     -3.    *U(  1)      +4.    *U(  2)      -1.    *U(  3))
      UX(2)=R2FDX*
     1(     -1.    *U(  1)      +0.    *U(  2)      +1.    *U(  3))
      DO 1 I=3,N
      UX(I)=R2FDX*
     1(      1.    *U(I-2)      -4.    *U(I-1)      +3.    *U(  I))
1     CONTINUE
      RETURN
C...
C...     (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
10    NM2=N-2
      DO 2 I=1,NM2
      UX(I)=R2FDX*
     1(     -3.    *U(  I)      +4.    *U(I+1)      -1.    *U(I+2))
2     CONTINUE
      UX(N-1)=R2FDX*
     1(     -1.    *U(N-2)      +0.    *U(N-1)      +1.    *U(  N))
      UX(N)=R2FDX*
     1(      1.    *U(N-2)      -4.    *U(N-1)      +3.    *U(  N))
      RETURN
      END
```

Appendix 6: Listing of Subroutine DSS018

```
      SUBROUTINE DSS018(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE DSS018 IS AN APPLICATION OF THIRD-ORDER DIRECTIONAL
C...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
C...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
C...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS AS DIS-
C...  CUSSED IN SUBROUTINE DSS012.  THE COEFFICIENTS OF THE FINITE
C...  DIFFERENCE APPROXIMATIONS USED HEREIN ARE TAKEN FROM BICKLEY, W.
C...  G., FORMULAE FOR NUMERICAL DIFFERENTIATION, THE MATHEMATICAL
C...  GAZETTE, PP. 19-27, 1941, N = 3, M = 1, P = 0, 1, 2, 3.  THE
C...  IMPLEMENTATION IS THE **FOUR-POINT BIASED UPWIND FORMULA** OF
C...  M. B. CARVER AND H. W. HINDS, THE METHOD OF LINES AND THE
C...  ADVECTION EQUATION, SIMULATION, VOL. 31, NO. 2, PP. 59-69,
C...  AUGUST, 1978
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE COMMON FACTOR FOR EACH FINITE DIFFERENCE APPROXIMATION
C...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
C...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      DX=(XU-XL)/FLOAT(N-1)
      R3FDX=1./(6.*DX)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
      UX(  1)=R3FDX*
     1( -11.*U(  1) +18.*U(  2)  -9.*U(  3)  +2.*U(  4))
      UX(  2)=R3FDX*
     1(  -2.*U(  1)  -3.*U(  2)  +6.*U(  3)  -1.*U(  4))
      NM1=N-1
      DO 1 I=3,NM1
      UX(  I)=R3FDX*
     1(  +1.*U(I-2)  -6.*U(I-1)  +3.*U(  I)  +2.*U(I+1))
1     CONTINUE
      UX(  N)=R3FDX*
     1(  -2.*U(N-3)  +9.*U(N-2) -18.*U(N-1) +11.*U(  N))
      RETURN
C...
C...     (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
10    UX(  1)=R3FDX*
     1( -11.*U(  1) +18.*U(  2)  -9.*U(  3)  +2.*U(  4))
      NM2=N-2
      DO 2 I=2,NM2
      UX(  I)=R3FDX*
     1(  -2.*U(I-1)  -3.*U(  I)  +6.*U(I+1)  -1.*U(I+2))
2     CONTINUE
      UX(N-1)=R3FDX*
     1(  +1.*U(N-3)  -6.*U(N-2)  +3.*U(N-1)  +2.*U(  N))
```

```
      UX(  N)=R3FDX*
     1(  -2.*U(N-3)  +9.*U(N-2) -18.*U(N-1) +11.*U(  N))
      RETURN
      END
```

Appendix 7: Listing of Subroutine DSS019

```
      SUBROUTINE DSS019(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE DSS019 IS AN APPLICATION OF THIRD-ORDER DIRECTIONAL
C...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
C...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
C...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS AS DIS-
C...  CUSSED IN SUBROUTINE DSS012.  THE COEFFICIENTS OF THE FINITE
C...  DIFFERENCE APPROXIMATIONS USED HEREIN ARE TAKEN FROM BICKLEY, W.
C...  G., FORMULAE FOR NUMERICAL DIFFERENTIATION, THE MATHEMATICAL
C...  GAZETTE, PP. 19-27, 1941, N = 3, M = 1, P = 0, 1, 2, 3.  THE
C...  IMPLEMENTATION IS THE **FOUR-POINT BIASED UPWIND FORMULA** OF
C...  M. B. CARVER AND H. W. HINDS, THE METHOD OF LINES AND THE
C...  ADVECTION EQUATION, SIMULATION, VOL. 31, NO. 2, PP. 59-69,
C...  AUGUST, 1978
C...
C...  SUBROUTINE DSS019 DIFFERS FROM DSS018 IN THE USE OF TWO POINT
C...  APPROXIMATIONS AT THE BOUNDARIES (THE DETAILS ARE OBVIOUS FROM
C...  THE FOLLOWING CODE).
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE COMMON FACTOR FOR EACH FINITE DIFFERENCE APPROXIMATION
C...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
C...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      DX=(XU-XL)/FLOAT(N-1)
      R3FDX=1./(6.*DX)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
      UX(  1)=(1.0/DX)*
     1(  -1.*U(  1)  +1.*U(  2))
      UX(  2)=(1.0/DX)*
     1(  -1.*U(  1)  +1.*U(  2))
      NM1=N-1
      DO 1 I=3,NM1
      UX(  I)=R3FDX*
     1(  +1.*U(I-2)  -6.*U(I-1)  +3.*U(  I)  +2.*U(I+1))
1     CONTINUE
      UX(N  )=(1.0/DX)*
     1(  -1.*U(N-1)  +1.*U(N  ))
      RETURN
C...
```

```
C...      (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
10     UX(  1)=(1.0/DX)*
      1(  -1.*U(  1)  +1.*U(  2))
       NM2=N-2
       DO 2 I=2,NM2
       UX(  I)=R3FDX*
      1(  -2.*U(I-1)  -3.*U(  I)  +6.*U(I+1)  -1.*U(I+2))
2      CONTINUE
       UX(N-1)=(1.0/DX)*
      1(  -1.*U(N-1)  +1.*U(N  ))
       UX(N  )=(1.0/DX)*
      1(  -1.*U(N-1)  +1.*U(N  ))
       RETURN
       END
```

Appendix 8: Listing of Subroutine DSS020

```
      SUBROUTINE DSS020(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE DSS020 IS AN APPLICATION OF FOURTH-ORDER DIRECTIONAL
C...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
C...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
C...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS AS DIS-
C...  CUSSED IN SUBROUTINE DSS012.  THE COEFFICIENTS OF THE FINITE
C...  DIFFERENCE APPROXIMATIONS USED HEREIN ARE TAKEN FROM BICKLEY, W.
C...  G., FORMULAE FOR NUMERICAL DIFFERENTIATION, THE MATHEMATICAL
C...  GAZETTE, PP. 19-27, 1941, N = 4, M = 1, P = 0, 1, 2, 3, 4.  THE
C...  IMPLEMENTATION IS THE **FIVE-POINT BIASED UPWIND FORMULA** OF
C...  M. B. CARVER AND H. W. HINDS, THE METHOD OF LINES AND THE
C...  ADVECTION EQUATION, SIMULATION, VOL. 31, NO. 2, PP. 59-69,
C...  AUGUST, 1978
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE COMMON FACTOR FOR EACH FINITE DIFFERENCE APPROXIMATION
C...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
C...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      DX=(XU-XL)/FLOAT(N-1)
      R4FDX=1./(12.*DX)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
      UX(  1)=R4FDX*
     1( -25.*U(  1) +48.*U(  2) -36.*U(  3) +16.*U(  4)  -3.*U(  5))
      UX(  2)=R4FDX*
     1(  -3.*U(  1) -10.*U(  2) +18.*U(  3)  -6.*U(  4)  +1.*U(  5))
      UX(  3)=R4FDX*
     1(  +1.*U(  1)  -8.*U(  2)  +0.*U(  3)  +8.*U(  4)  -1.*U(  5))
      NM1=N-1
      DO 1 I=4,NM1
      UX(  I)=R4FDX*
     1(  -1.*U(I-3)  +6.*U(I-2) -18.*U(I-1) +10.*U(  I)  +3.*U(I+1))
1     CONTINUE
      UX(  N)=R4FDX*
     1(   3.*U(N-4) -16.*U(N-3) +36.*U(N-2) -48.*U(N-1) +25.*U(  N))
      RETURN
C...
C...     (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
10    UX(  1)=R4FDX*
     1( -25.*U(  1) +48.*U(  2) -36.*U(  3) +16.*U(  4)  -3.*U(  5))
      NM3=N-3
      DO 2 I=2,NM3
      UX(  I)=R4FDX*
     1(  -3.*U(I-1) -10.*U(  I) +18.*U(I+1)  -6.*U(I+2)  +1.*U(I+3))
2     CONTINUE
```

```
      UX(N-2)=R4FDX*
     1(   +1.*U(N-4)   -8.*U(N-3)   +0.*U(N-2)   +8.*U(N-1)   -1.*U(   N))
      UX(N-1)=R4FDX*
     1(   -1.*U(N-4)   +6.*U(N-3)  -18.*U(N-2)  +10.*U(N-1)   +3.*U(   N))
      UX(   N)=R4FDX*
     1(    3.*U(N-4)  -16.*U(N-3)  +36.*U(N-2)  -48.*U(N-1)  +25.*U(   N))
      RETURN
      END
```

Appendix 9: Listing of Subroutine DSS021

```
      SUBROUTINE DSS021(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE DSS021 IS AN APPLICATION OF FOURTH-ORDER DIRECTIONAL
C...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
C...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
C...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS AS DIS-
C...  CUSSED IN SUBROUTINE DSS012.  THE COEFFICIENTS OF THE FINITE
C...  DIFFERENCE APPROXIMATIONS USED HEREIN ARE TAKEN FROM BICKLEY, W.
C...  G., FORMULAE FOR NUMERICAL DIFFERENTIATION, THE MATHEMATICAL
C...  GAZETTE, PP. 19-27, 1941, N = 4, M = 1, P = 0, 1, 2, 3, 4.  THE
C...  IMPLEMENTATION IS THE **FIVE-POINT BIASED UPWIND FORMULA** OF
C...  M. B. CARVER AND H. W. HINDS, THE METHOD OF LINES AND THE
C...  ADVECTION EQUATION, SIMULATION, VOL. 31, NO. 2, PP. 59-69,
C...  AUGUST, 1978.
C...
C...  SUBROUTINE DSS021 DIFFERS FROM DSS020 IN THE USE OF TWO POINT
C...  APPROXIMATIONS AT THE BOUNDARIES (THE DETAILS ARE OBVIOUS FROM
C...  THE FOLLOWING CODE).
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE COMMON FACTOR FOR EACH FINITE DIFFERENCE APPROXIMATION
C...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
C...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      DX=(XU-XL)/FLOAT(N-1)
      R4FDX=1./(12.*DX)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
      UX(  1)=(1.0/DX)*
     1(  -1.*U(  1)  +1.*U(  2))
      UX(  2)=(1.0/DX)*
     1(  -1.*U(  1)  +1.*U(  2))
      UX(  3)=R4FDX*
     1(  +1.*U(  1)  -8.*U(  2)  +0.*U(  3)  +8.*U(  4)  -1.*U(  5))
      NM1=N-1
      DO 1 I=4,NM1
      UX(  I)=R4FDX*
     1(  -1.*U(I-3)  +6.*U(I-2) -18.*U(I-1) +10.*U(  I)  +3.*U(I+1))
1     CONTINUE
      UX(N  )=(1.0/DX)*
     1(  -1.*U(N-1)  +1.*U(N  ))
      RETURN
C...
C...     (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
10    UX(  1)=(1.0/DX)*
     1(  -1.*U(  1)  +1.*U(  2))
      NM3=N-3
```

```
      DO 2 I=2,NM3
      UX(   I)=R4FDX*
     1(   -3.*U(I-1) -10.*U(   I) +18.*U(I+1)   -6.*U(I+2)   +1.*U(I+3))
2     CONTINUE
      UX(N-2)=R4FDX*
     1(   +1.*U(N-4)   -8.*U(N-3)   +0.*U(N-2)   +8.*U(N-1)   -1.*U(   N))
      UX(N-1)=(1.0/DX)*
     1(   -1.*U(N-1)   +1.*U(N   ))
      UX(N   )=(1.0/DX)*
     1(   -1.*U(N-1)   +1.*U(N   ))
      RETURN
      END
```

Appendix 10: Listing of Subroutine LEO1

```
      SUBROUTINE LEO1(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE LEO1 IS AN APPLICATION OF LEONARD*S FOUR POINT
C...  APROXIMATIONS.  THE COEFFICIENTS OF THE APPROXIMATIONS ARE
C...  GIVEN IN LEONARD, B. P., **A STABLE AND ACCURATE CONVECTIVE
C...  MODELLING PROCEDURE BASED ON QUADRATIC UPSTREAM INTERPOLATION**,
C...  COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING, VOL. 19,
C...  NO. 1, JUNE, 1979.  REPUBLISHED IN COMPUTER METHODS IN APPLIED
C...  MECHANICS AND ENGINEERING, SPECIAL EDITION, PP 59-98, AUGUST,
C...  1990.  FOUR POINT APPROXIMATIONS ARE USED AT THE BOUNDARIES.
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE COMMON FACTOR FOR EACH SPATIAL APPROXIMATION
C...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
C...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      DX=(XU-XL)/FLOAT(N-1)
      R8DX=1./(8.*DX)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  APPROXIMATIONS FOR POSITIVE V
C...
C...  FINITE DIFFERENCES
      UX(  1)=(1.0/(6.0*DX))*
     1( -11.*U(  1) +18.*U(  2)  -9.*U(  3)  +2.*U(  4))
      UX(  2)=(1.0/(6.0*DX))*
     1(  -2.*U(  1)  -3.*U(  2)  +6.*U(  3)  -1.*U(  4))
C...
C...  LEONARD*S COEFFICIENTS
      NM1=N-1
      DO 1  I=3,NM1
      UX(  I)=R8DX*
     1(  +1.*U(I-2)  -7.*U(I-1)  +3.*U(  I)  +3.*U(I+1))
1     CONTINUE
C...
C...  FINITE DIFFERENCE
      UX(N  )=(1.0/(6.0*DX))*
     1(  -2.*U(N-3)  +9.*U(N-2) -18.*U(N-1) +11.*U(  N))
      RETURN
C...
C...     (2)  APPROXIMATION FOR NEGATIVE V
C...
C...  FINITE DIFFERENCE
10    UX(  1)=(1.0/(6.0*DX))*
     1( -11.*U(  1) +18.*U(  2)  -9.*U(  3)  +2.*U(  4))
C...
C...  LEONARD*S COEFFICIENTS
      NM2=N-2
```

```
      DO 2 I=2,NM2
      UX(   I)=R3FDX*
     1(   +3.*U(I-1)   +3.*U(   I)   -7.*U(I+1)   +1.*U(I+2))
2     CONTINUE
C...
C...  FINITE DIFFERENCES
      UX(N-1)=(1.0/(6.0*DX))*
     1(   +1.*U(N-3)   -6.*U(N-2)   +3.*U(N-1)   +2.*U(   N))
      UX(N   )=(1.0/(6.0*DX))*
     1(   -2.*U(N-3)   +9.*U(N-2) -18.*U(N-1) +11.*U(   N))
      RETURN
      END
```

Appendix 11: Listing of Subroutine LEO2

```
      SUBROUTINE LEO2(XL,XU,N,U,UX,V)
C...
C...  SUBROUTINE LEO2 IS AN APPLICATION OF LEONARD*S FOUR POINT
C...  APROXIMATIONS.  THE COEFFICIENTS OF THE APPROXIMATIONS ARE
C...  GIVEN IN LEONARD, B. P., **A STABLE AND ACCURATE CONVECTIVE
C...  MODELLING PROCEDURE BASED ON QUADRATIC UPSTREAM INTERPOLATION**,
C...  COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING, VOL. 19,
C...  NO. 1, JUNE, 1979.  REPUBLISHED IN COMPUTER METHODS IN APPLIED
C...  MECHANICS AND ENGINEERING, SPECIAL EDITION, PP 59-98, AUGUST,
C...  1990.  TWO POINT APPROXIMATIONS ARE USED AT THE BOUNDARIES.
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE COMMON FACTOR FOR EACH SPATIAL APPROXIMATION
C...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
C...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      DX=(XU-XL)/FLOAT(N-1)
      R8DX=1./(8.*DX)
      IF(V.LT.0.)GO TO 10
C...
C...     (1)  APPROXIMATIONS FOR POSITIVE V
C...
C...  FINITE DIFFERENCES
      UX(  1)=(1.0/DX)*
     1(  -1.*U(  1)  +1.*U(  2))
      UX(  2)=(1.0/DX)*
     1(  -1.*U(  1)  +1.*U(  2))
C...
C...  LEONARD*S COEFFICIENTS
      NM1=N-1
      DO 1 I=3,NM1
      UX(  I)=R8DX*
     1(  +1.*U(I-2)  -7.*U(I-1)  +3.*U(  I)  +3.*U(I+1))
1     CONTINUE
C...
C...  FINITE DIFFERENCE
      UX(N  )=(1.0/DX)*
     1(  -1.*U(N-1)  +1.*U(N  ))
      RETURN
C...
C...     (2)  APPROXIMATION FOR NEGATIVE V
C...
C...  FINITE DIFFERENCE
10    UX(  1)=(1.0/DX)*
     1(  -1.*U(  1)  +1.*U(  2))
C...
C...  LEONARD*S COEFFICIENTS
      NM2=N-2
```

```
      DO 2 I=2,NM2
      UX(   I)=R3FDX*
     1(  +3.*U(I-1)  +3.*U(   I)  -7.*U(I+1)  +1.*U(I+2))
2     CONTINUE
C...
C...  FINITE DIFFERENCES
      UX(N-1)=(1.0/DX)*
     1(  -1.*U(N-1)  +1.*U(N   ))
      UX(N   )=(1.0/DX)*
     1(  -1.*U(N-1)  +1.*U(N   ))
      RETURN
      END
```

```
      SUBROUTINE DSS002(XL,XU,N,U,UX)
C...
C...  SUBROUTINE DSS002 COMPUTES THE FIRST DERIVATIVE, U  , OF A
C...                                                   X
C...  VARIABLE U OVER THE SPATIAL DOMAIN XL LE X LE XU
C...
C...  ARGUMENT LIST
C...
C...     XL       LOWER BOUNDARY VALUE OF X (INPUT)
C...
C...     XU       UPPER BOUNDARY VALUE OF X (INPUT)
C...
C...     N        NUMBER OF GRID POINTS IN THE X DOMAIN INCLUDING THE
C...              BOUNDARY POINTS (INPUT)
C...
C...     U        ONE-DIMENSIONAL ARRAY CONTAINING THE VALUES OF U AT
C...              THE N GRID POINT POINTS FOR WHICH THE DERIVATIVE IS
C...              TO BE COMPUTED (INPUT)
C...
C...     UX       ONE-DIMENSIONAL ARRAY CONTAINING THE NUMERICAL
C...              VALUES OF THE DERIVATIVES OF U AT THE N GRID POINTS
C...              (OUTPUT)
C...
C...  SUBROUTINE DSS002 COMPUTES THE FIRST DERIVATIVE, U  , OF A
C...                                                   X
C...  VARIABLE U OVER THE SPATIAL DOMAIN XL LE X LE XU FROM THE
C...  CLASSICAL THREE-POINT, SECOND-ORDER FINITE DIFFERENCE APPROXI-
C...  TIONS
C...
C...                                          2
C...  U1   = (1/2DX)(-3U1 + 4U2 - U3) + O(DX ) (LEFT BOUNDARY,      (1)
C...    X                                          X = XL)
C...
C...                                        2
C...  UI   = (1/2DX)(UI+1 - UI-1) + O(DX ) (INTERIOR POINT,        (2)
C...    X                                     X NE XL, XU)
C...
C...                                          2
C...  UN   = (1/2DX)(3UN - 4UN-1 + UN-2) + O(DX ) (RIGHT BOUNDARY, (3)
C...    X                                              X = XU)
C...
C...  EQUATIONS (1) TO (3) APPLY OVER A GRID IN X WITH CORRESPONDING
C...  VALUES OF THE FUNCTION U(X) REPRESENTED AS
C...
C...   U1       U2        U3          UI        UN-2       UN-1     UN
C...
C...  X=XL  X=XL+DX  X=XL+2DX ... X=XI ... X=XU-2DX  X=XU-DX  X=XU
C...
```

```
C...   THE ORIGIN OF EQUATIONS (1) TO (3) IS OUTLINED BELOW.
C...
C...   CONSIDER THE FOLLOWING POLYNOMIAL IN X OF ARBITRARY ORDER
C...
C...                                          2              3
C...   U(X) = A0 + A1(X - X0) + A2(X - X0)  + A3(X - X0)  + ....    (4)
C...
C...   WE SEEK THE VALUES OF THE COEFFICIENTS A0, A1, A2, ... FOR A
C...   PARTICULAR FUNCTION U(X).  IF X = X0 IS SUBSTITUTED IN EQUATION
C...   (4), WE HAVE IMMEDIATELY A0 = U(X0).  NEXT, IF EQUATION (4) IS
C...   DIFFERENTIATED WITH RESPECT TO X,
C...
C...                                                        2
C...   DU(X)/DX = U  (X) = A1 + 2A2(X - X0) + 3A3(X - X0)  + ...     (5)
C...               X
C...
C...   AGAIN, WITH X = X0, A1 = DU(X0)/DX = U  (X0).  DIFFERENTIATION
C...                                        X
C...   OF EQUATION (5) IN TURN GIVES
C...
C...   D2U(X)/DX2 = U   (X) = 2A2 + 6A3(X - X0) + ...
C...                 2X
C...
C...   AND FOR X = X0, A2 = U   (X0)/2F (2F = 1*2, I.E., 2 FACTORIAL).
C...                       2X
C...
C...   WE CAN CONTINUE THIS PROCESS OF DIFFERENTIATION FOLLOWED BY THE
C...   SUBSTITUTION X = X0 TO OBTAIN THE SUCCESSIVE COEFFICIENTS IN
C...   EQUATION (4), A3, A4, ...  FINALLY, SUBSTITUTION OF THESE CO-
C...   EFFICIENTS IN EQUATION (4) GIVES
C...
C...                                                        2
C...   U(X) = U(X0) + U  (X0)(X - X0) + U   (X0)(X - X0)  +
C...                  X        1F        2X        2F
C...                                                                (6)
C...                                   3                    4
C...                  U   (X0)(X - X0)  + U   (X0)(X - X0)  + ...
C...                   3X        3F        4X        4F
C...
C...   THE CORRESPONDENCE BETWEEN EQUATION (6) AND THE WELL-KNOWN
C...   TAYLOR SERIES SHOULD BE CLEAR.  THUS THE EXPANSION OF A
C...   FUNCTION, U(X), AROUND A NEIGHBORING POINT X0 IN TERMS OF U(X0)
C...   AND THE DERIVATIVES OF U(X) AT X = X0 IS EQUIVALENT TO APPROXI-
C...   MATING U(X) NEAR X0 BY A POLYNOMIAL.
C...
C...   EQUATION (6) IS THE STARTING POINT FOR THE DERIVATION OF THE
C...   CLASSICAL FINITE DIFFERENCE APPROXIMATIONS OF DERIVATIVES SUCH
C...   AS THE THREE-POINT FORMULAS OF EQUATIONS (1), (2) AND (3).  WE
C...   WILL NOW CONSIDER THE DERIVATION OF THESE THREE-POINT FORMULAS
C...   IN A STANDARD FORMAT WHICH CAN THEN BE EXTENDED TO HIGHER
C...   MULTI-POINT FORMULAS IN OTHER SUBROUTINES, E.G., FIVE-POINT
```

```
C...  FORMULAS IN SUBROUTINE DSS004.
C...
C...  THREE-POINT FORMULAS
C...
C...     (1)  LEFT END, POINT I = 1
C...
C...  IF EQUATION (6) IS WRITTEN AROUND THE POINT X = XL FOR X = XL +
C...  DX AND X = XL + 2DX, FOR WHICH THE CORRESPONDING VALUES OF U(X)
C...  ARE U1, U2 AND U3 (U1 AND U2 ARE SEPARATED WITH RESPECT TO X BY
C...  DISTANCE DX AS ARE U2 AND U3, I.E., WE ASSUME A UNIFORM GRID
C...  SPACING, DX, FOR INDEPENDENT VARIABLE X)
C...
C...                                      2            3
C...  U2 = U1 + U1 ( DX) + U1  ( DX)  + U1  ( DX)  + ...          (7)
C...             X  1F      2X  2F       3X  3F
C...
C...                                      2            3
C...  U3 = U1 + U1 (2DX) + U1  (2DX)  + U1  (2DX)  + ...          (8)
C...             X  1F      2X  2F       3X  3F
C...
C...  WE CAN NOW TAKE A LINEAR COMBINATION OF EQUATIONS (7) AND (8)
C...  BY FIRST MULTIPLYING EQUATION (7) BY A CONSTANT, A, AND EQUA-
C...  TION (8) BY CONSTANT B
C...
C...                                        2            3
C...  A(U2 = U1 + U1 ( DX) + U1  ( DX) + U1  ( DX) + ...)          (9)
C...             X  1F      2X  2F      3X  3F
C...
C...                                        2            3
C...  B(U3 = U1 + U1 (2DX) + U1  (2DX) + U1  (2DX) + ...)         (10)
C...             X  1F      2X  2F      3X  3F
C...
C...  CONSTANTS A AND B ARE THEN SELECTED SO THAT THE COEFFICIENTS OF
C...  THE U1  TERMS SUM TO ONE (SINCE WE ARE INTERESTED IN OBTAINING
C...       X
C...  A FINITE DIFFERENCE APPROXIMATION FOR THIS FIRST DERIVATIVE).
C...  ALSO, WE SELECT A AND B SO THAT THE COEFFICIENTS OF THE U1
C...                                                            2X
C...  TERMS SUM TO ZERO IN ORDER TO DROP OUT THE CONTRIBUTION OF THIS
C...  SECOND DERIVATIVE (THE BASIC IDEA IS TO DROP OUT AS MANY OF THE
C...  DERIVATIVES AS POSSIBLE IN THE TAYLOR SERIES BEYOND THE DERI-
C...  VATIVE OF INTEREST, IN THIS CASE U1  , IN ORDER TO PRODUCE A
C...                                    X
C...  FINITE DIFFERENCE APPROXIMATION FOR THE DERIVATIVE OF MAXIMUM
C...  ACCURACY).  IN THIS CASE WE HAVE ONLY TWO CONSTANTS, A AND B,
C...  TO SELECT SO WE CAN DROP OUT ONLY THE SECOND DERIVATIVE, U1  ,
C...                                                            2X
C...  IN THE TAYLOR SERIES (IN ADDITION TO RETAINING THE FIRST DERI-
C...  VATIVE).  THIS PROCEDURE LEADS TO TWO LINEAR ALGEBRAIC EQUA-
C...  TIONS IN THE TWO CONSTANTS
C...
```

```
C...  A + 2B = 1
C...
C...  A + 4B = 0
C...
C...  SOLUTION OF THESE EQUATIONS FOR A AND B GIVES
C...
C...  A = 2, B = -1/2
C...
C...  SOLUTION OF EQUATIONS (9) AND (10) FOR U1  WITH THESE VALUES OF
C...  A AND B GIVES EQUATION (1)                X
C...
C...                                          2
C...  U1 = (1/2DX)(-3U1 + 4U2 - U3) + O(DX )                          (1)
C...    X
C...                 2
C...  THE TERM O(DX ) INDICATES A PRINCIPAL ERROR TERM DUE TO TRUNCA-
C...                                                          2
C...  TION OF THE TAYLOR SERIES WHICH IS OF ORDER DX .  THIS TERM IN
C...                      2
C...  FACT EQUALS U1  DX /3F, WHICH IS EASILY OBTAINED IN DERIVING
C...                 3X
C...  EQUATION (1).
C...
C...  THIS SAME BASIC PROCEDURE CAN NOW BE APPLIED TO THE DERIVATION
C...  OF EQUATIONS (2) AND (3).
C...
C...     (2)  INTERIOR POINT I
C...
C...                                          2            3
C...  A(UI-1 = UI + UI (-DX) + UI   (-DX) + UI   (-DX) + ...)
C...                  X  1F      2X  2F      3X  3F
C...
C...                                          2            3
C...  B(UI+1 = UI + UI ( DX) + UI   ( DX) + UI   ( DX) + ...)
C...                  X  1F      2X  2F      3X  3F
C...
C...  -A + B = 1
C...
C...   A + B = 0
C...
C...  A = 1/2, B = -1/2
C...                                          2
C...  UI   = (1/2DX)(UI+1 - UI-1) + O(DX )                           (2)
C...    X
C...
C...     (3)  RIGHT END, POINT I = N
C...
C...                                          2             3
C...  A(UN-2 = UN + UN (-2DX) + UN   (-2DX) + UN   (-2DX) + ...)
C...                  X   1F      2X   2F      3X   3F
C...
```

```
C...                                                 2              3
C...  B(UN-1 = UN + UN ( -DX) + UN   ( -DX) + UN   ( -DX) + ...)
C...                  X   1F       2X   2F       3X   3F
C...
C...  -2A - B = 1
C...
C...   4A + B = 0
C...
C...   A = -2, B = 1/2
C...                                                 2
C...  UN   = (1/2DX)(3UN - 4UN-1 + UN-2) + O(DX )                    (3)
C...    X
C...
C...  THE WEIGHTING COEFFICIENTS FOR EQUATIONS (1), (2) AND (3) CAN
C...  BE SUMMARIZED AS
C...
C...          -3   4  -1
C...
C...     1/2  -1   0   1
C...
C...           1  -4   3
C...
C...  WHICH ARE THE COEFFICIENTS REPORTED BY BICKLEY FOR N = 2, M =
C...  1, P = 0, 1, 2 (BICKLEY, W. G., FORMULAE FOR NUMERICAL DIFFER-
C...  ENTIATION, MATH. GAZ., VOL. 25, 1941).
C...
C...  EQUATIONS (1), (2) AND (3) CAN NOW BE PROGRAMMED TO GENERATE
C...  THE DERIVATIVE U (X) OF FUNCTION U(X) (ARGUMENTS U AND UX OF
C...                 X
C...  SUBROUTINE DSS002, RESPECTIVELY).
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE SPATIAL INCREMENT
      DX=(XU-XL)/FLOAT(N-1)
      R2FDX=1./(2.*DX)
      NM1=N-1
C...
C...  EQUATION (1) (NOTE - THE RHS OF THE FINITE DIFFERENCE APPROXI-
C...  TIONS, EQUATIONS (1), (2) AND (3) HAVE BEEN FORMATTED SO THAT
C...  THE NUMERICAL WEIGHTING COEFFICIENTS CAN BE MORE EASILY ASSOCI-
C...  ATED WITH THE BICKLEY MATRIX LISTED ABOVE)
      UX(1)=R2FDX*
     1(      -3.    *U(  1)      +4.    *U(  2)      -1.    *U(  3))
C...
C...  EQUATION (2)
      DO 1 I=2,NM1
      UX(I)=R2FDX*
     1(      -1.    *U(I-1)      +0.    *U(  I)      +1.    *U(I+1))
1     CONTINUE
C...
```

```
C...  EQUATION (3)
      UX(N)=R2FDX*
     1(       1.    *U(N-2)       -4.    *U(N-1)       +3.    *U(   N))
      RETURN
      END
```

```
      SUBROUTINE DSS004(XL,XU,N,U,UX)
C...
C...  SUBROUTINE DSS004 COMPUTES THE FIRST DERIVATIVE, U  , OF A
C...                                                    X
C...  VARIABLE U OVER THE SPATIAL DOMAIN XL LE X LE XU FROM CLASSICAL
C...  FIVE-POINT, FOURTH-ORDER FINITE DIFFERENCE APPROXIMATIONS
C...
C...  ARGUMENT LIST
C...
C...     XL       LOWER BOUNDARY VALUE OF X (INPUT)
C...
C...     XU       UPPER BOUNDARY VALUE OF X (INPUT)
C...
C...     N        NUMBER OF GRID POINTS IN THE X DOMAIN INCLUDING THE
C...              BOUNDARY POINTS (INPUT)
C...
C...     U        ONE-DIMENSIONAL ARRAY CONTAINING THE VALUES OF U AT
C...              THE N GRID POINT POINTS FOR WHICH THE DERIVATIVE IS
C...              TO BE COMPUTED (INPUT)
C...
C...     UX       ONE-DIMENSIONAL ARRAY CONTAINING THE NUMERICAL
C...              VALUES OF THE DERIVATIVES OF U AT THE N GRID POINTS
C...              (OUTPUT)
C...
C...  THE MATHEMATICAL DETAILS OF THE FOLLOWING TAYLOR SERIES (OR
C...  POLYNOMIALS) ARE GIVEN IN SUBROUTINE DSS002.
C...
C...  FIVE-POINT FORMULAS
C...
C...     (1)  LEFT END, POINT I = 1
C...
C...                                          2             3             4
C...  A(U2 = U1 + U1  ( DX) + U1  ( DX)  + U1  ( DX)  + U1  ( DX)
C...                X   1F      2X  2F       3X  3F       4X  4F
C...
C...                         5             6             7
C...           + U1  ( DX)  + U1  ( DX)  + U1  ( DX)  + ...)
C...               5X  5F      6X  6F       7X  7F
C...
C...                                          2             3             4
C...  B(U3 = U1 + U1  (2DX) + U1  (2DX)  + U1  (2DX)  + U1  (2DX)
C...                X   1F      2X  2F       3X  3F       4X  4F
C...
C...                         5             6             7
C...           + U1  (2DX)  + U1  (2DX)  + U1  (2DX)  + ...)
C...               5X  5F      6X  6F       7X  7F
C...
C...                                          2             3             4
```

```
C...  C(U4 = U1 + U1   (3DX) + U1   (3DX)   + U1   (3DX)   + U1   (3DX)
C...                X    1F       2X   2F        3X   3F         4X   4F
C...
C...                          5              6              7
C...            + U1   (3DX)   + U1   (3DX)   + U1   (3DX)   + ...)
C...                5X   5F        6X   6F        7X   7F
C...
C...                                         2              3              4
C...  D(U5 = U1 + U1   (4DX) + U1   (4DX)   + U1   (4DX)   + U1   (4DX)
C...                X    1F       2X   2F        3X   3F         4X   4F
C...
C...                          5              6              7
C...            + U1   (4DX)   + U1   (4DX)   + U1   (4DX)   + ...)
C...                5X   5F        6X   6F        7X   7F
C...
C...  CONSTANTS A, B, C AND D ARE SELECTED SO THAT THE COEFFICIENTS
C...  OF THE U1   TERMS SUM TO ONE AND THE COEFFICIENTS OF THE U1   ,
C...           X                                                  2X
C...  U1    AND U1    TERMS SUM TO ZERO
C...    3X        4X
C...
C...  A +   2B +   3C +   4D = 1
C...
C...  A +   4B +   9C +  16D = 0
C...
C...  A +   8B +  27C +  64D = 0
C...
C...  A +  16B +  81C + 256D = 0
C...
C...  SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C...  TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE U
C...                                                              4X
C...  TERMS, FOR U1   GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C...                X
C...                                                          4
C...  U1   = (1/12DX)(-25U1 + 48U2 - 36U3 + 16U4 - 3U5) + O(DX )    (1)
C...    X
C...
C...     (2)  INTERIOR POINT, I = 2
C...
C...                                         2              3              4
C...  A(U1 = U2 + U2   (-DX) + U2   (-DX)   + U2   (-DX)   + U2   (-DX)
C...                X    1F       2X   2F        3X   3F         4X   4F
C...
C...                          5              6              7
C...            + U2   (-DX)   + U2   (-DX)   + U2   (-DX)   + ...)
C...                5X   5F        6X   6F        7X   7F
C...
C...                                         2              3              4
C...  B(U3 = U2 + U2   ( DX) + U2   ( DX)   + U2   ( DX)   + U2   ( DX)
C...                X    1F       2X   2F        3X   3F         4X   4F
```

```
C...
C...                                  5                6                7
C...            + U2   ( DX)   + U2   ( DX)   + U2   ( DX)   + ...)
C...                5X   5F         6X   6F         7X   7F
C...
C...                                                  2                3                4
C...   C(U4 = U2 + U2   (2DX) + U2   (2DX)   + U2   (2DX)   + U2   (2DX)
C...                 X    1F       2X   2F         3X   3F         4X   4F
C...
C...                                  5                6                7
C...            + U2   (2DX)   + U2   (2DX)   + U2   (2DX)   + ...)
C...                5X   5F         6X   6F         7X   7F
C...
C...                                                  2                3                4
C...   D(U5 = U2 + U2   (3DX) + U2   (3DX)   + U2   (3DX)   + U2   (3DX)
C...                 X    1F       2X   2F         3X   3F         4X   4F
C...
C...                                  5                6                7
C...            + U2   (3DX)   + U2   (3DX)   + U2   (3DX)   + ...)
C...                5X   5F         6X   6F         7X   7F
C...
C...   -A +    B +  2C +  3D = 1
C...
C...    A +    B +  4C +  9D = 0
C...
C...   -A +    B +  8C + 27D = 0
C...
C...    A +    B + 16C + 81D = 0
C...
C...   SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C...   TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE U
C...                                                                    4X
C...   TERMS, FOR U1   GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C...                 X
C...                                                                  4
C...   U2   = (1/12DX)(-3U1 - 10U2 + 18U3 -  6U4 +  U5) + O(DX )      (2)
C...     X
C...
C...      (3)  INTERIOR POINT I, I NE 2, N-1
C...
C...                                                  2                3
C...   A(UI-2 = UI + UI   (-2DX)   + UI   (-2DX)   + UI   (-2DX)
C...                  X     1F        2X    2F        3X    3F
C...
C...                                                  4                5                6
C...              + UI   (-2DX)   + UI   (-2DX)   + UI   (-2DX)   + ...)
C...                  4X    4F        5X    5F        6X    6F
C...
C...                                                  2                3
C...   B(UI-1 = UI + UI   ( -DX)   + UI   ( -DX)   + UI   ( -DX)
C...                  X     1F        2X    2F        3X    3F
```

```
C...
C...                                   4                5                6
C...                 + UI   ( -DX)  + UI   ( -DX)  + UI   ( -DX)  + ...)
C...                     4X   4F        5X   5F        6X   6F
C...
C...                                                   2                3
C...   C(UI+1 = UI + UI   (  DX)  + UI   (  DX)  + UI   (  DX)
C...                     X    1F        2X   2F        3X   3F
C...
C...                                   4                5                6
C...                 + UI   (  DX)  + UI   (  DX)  + UI   (  DX)  + ...)
C...                     4X   4F        5X   5F        6X   6F
C...
C...                                                   2                3
C...   D(UI+2 = UI + UI   ( 2DX)  + UI   ( 2DX)  + UI   ( 2DX)
C...                     X    1F        2X   2F        3X   3F
C...
C...                                   4                5                6
C...                 + UI   ( 2DX)  + UI   ( 2DX)  + UI   ( 2DX)  + ...)
C...                     4X   4F        5X   5F        6X   6F
C...
C...    -2A -    B +    C +   2D = 1
C...
C...     4A +    B +    C +   4D = 0
C...
C...    -8A -    B +    C +   8D = 0
C...
C...    16A +    B +    C + 16D = 0
C...
C...   SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C...   TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE U
C...                                                                 4X
C...   TERMS, FOR U1  GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C...                X
C...                                                                  4
C...   UI   = (1/12DX)(UI-2 - 8UI-1 + 0UI + 8UI+1 - UI+2) + O(DX )   (3)
C...     X
C...
C...      (4)   INTERIOR POINT, I = N-1
C...
C...                                                   2                3
C...   A(UN-4 = UN-1 + UN-1   (-3DX)  + UN-1   (-3DX)  + UN-1   (-3DX)
C...                       X    1F          2X   2F          3X   3F
C...
C...                                   4                5                6
C...          + UN-1   (-3DX)  + UN-1   (-3DX)  + UN-1   (-3DX)  + ...
C...                 4X   4F          5X   5F          6X   6F
C...
C...                                                   2                3
C...   B(UN-3 = UN-1 + UN-1   (-2DX)  + UN-1   (-2DX)  + UN-1   (-2DX)
C...                       X    1F          2X   2F          3X   3F
```

```
C...
C...                                 4                    5                    6
C...          + UN-1   (-2DX)   + UN-1   (-2DX)   + UN-1   (-2DX)   + ...
C...                4X    4F          5X    5F          6X    6F
C...
C...                                                     2                    3
C...   C(UN-2 = UN-1 + UN-1   ( -DX)   + UN-1   (- -X)   + UN-1   ( -DX)
C...                         X     1F          2X    2F          3X    3F
C...
C...                                 4                    5                    6
C...          + UN-1   ( -DX)   + UN-1   ( -DX)   + UN-1   ( -DX)   + ...
C...                4X    4F          5X    5F          6X    6F
C...
C...                                                     2                    3
C...   D(UN    = UN-1 + UN-1   (  DX)   + UN-1   (  DX)   + UN-1   (  DX)
C...                         X     1F          2X    2F          3X    3F
C...
C...                                 4                    5                    6
C...          + UN-1   (  DX)   + UN-1   (  DX)   + UN-1   (  DX)   + ...
C...                4X    4F          5X    5F          6X    6F
C...
C...   -3A -  2B -   C +   D = 1
C...
C...    9A +  4B +   C +   D = 0
C...
C... -27A -  8B -   C +   D = 0
C...
C...   81A + 16B +   C +   D = 0
C...
C...   SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C...   TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE U
C...                                                                  4X
C...   TERMS, FOR U1  GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C...                X
C...                                                                      4
C...   UN-1   = (1/12DX)(-UN-4 + 6UN-3 - 18UN-2 + 10UN-1 + 3UN) + O(DX )
C...      X
C...                                                                     (4)
C...
C...     (5)  RIGHT END, POINT I = N
C...
C...                                                     2                    3
C...   A(UN-4 = UN + UN (-4DX)   + UN   (-4DX)   + UN   (-4DX)
C...                    X    1F         2X    2F         3X    3F
C...
C...                                 4                    5                    6
C...              + UN   (-4DX)   + UN   (-4DX)   + UN   (-4DX)   + ...)
C...                  4X    4F         5X    5F         6X    6F
C...
C...                                                     2                    3
C...   B(UN-3 = UN + UN (-3DX)   + UN   (-3DX)   + UN   (-3DX)
```

```
C...                    X    1F        2X    2F        3X    3F
C...
C...                                 4              5              6
C...              + UN   (-3DX)  + UN   (-3DX)  + UN   (-3DX)  + ...)
C...                 4X    4F        5X    5F        6X    6F
C...
C...                                           2              3
C...  C(UN-2 = UN + UN (-2DX)  + UN   (-2DX)  + UN   (-2DX)
C...                    X    1F        2X    2F        3X    3F
C...
C...                                 4              5              6
C...              + UN   (-2DX)  + UN   (-2DX)  + UN   (-2DX)  + ...)
C...                 4X    4F        5X    5F        6X    6F
C...
C...                                           2              3
C...  D(UN-1 = UN + UN ( -DX)  + UN   ( -DX)  + UN   ( -DX)
C...                    X    1F        2X    2F        3X    3F
C...
C...                                 4              5              6
C...              + UN   ( -DX)  + UN   ( -DX)  + UN   ( -DX)  + ...)
C...                 4X    4F        5X    5F        6X    6F
C...
C...   -4A -  3B -  2C -   D = 1
C...
C...   16A +  9B +  4C +   D = 0
C...
C...  -64A - 27B -  8C -   D = 0
C...
C...  256A + 81B + 16C +   D = 0
C...
C...  SIMULTANEOUS SOLUTION FOR A, B, C AND D FOLLOWED BY THE SOLU-
C...  TION OF THE PRECEDING TAYLOR SERIES, TRUNCATED AFTER THE U
C...                                                            4X
C...  TERMS, FOR U1  GIVES THE FOLLOWING FIVE-POINT APPROXIMATION
C...                X
C...                                                              4
C...  UN   = (1/12DX)(3UN-4 - 16UN-3 + 36UN-2 - 48UN-1 + 25UN) + O(DX )
C...    X
C...                                                                (5)
C...
C...  THE WEIGHTING COEFFICIENTS FOR EQUATIONS (1) TO (5) CAN BE
C...  SUMMARIZED AS
C...
C...              -25   48  -36   16   -3
C...
C...               -3  -10   18   -6    1
C...
C...       1/12     1   -8    0    8   -1
C...
C...               -1    6  -18   10    3
C...
```

```
C...                3  -16   36  -48   25
C...
C...  WHICH ARE THE COEFFICIENTS REPORTED BY BICKLEY FOR N = 4, M =
C...  1, P = 0, 1, 2, 3, 4 (BICKLEY, W. G., FORMULAE FOR NUMERICAL
C...  DIFFERENTIATION, MATH. GAZ., VOL. 25, 1941.  NOTE - THE BICKLEY
C...  COEFFICIENTS HAVE BEEN DIVIDED BY A COMMON FACTOR OF TWO).
C...
C...  EQUATIONS (1) TO (5) CAN NOW BE PROGRAMMED TO GENERATE THE
C...  DERIVATIVE U (X) OF FUNCTION U(X) (ARGUMENTS U AND UX OF SUB-
C...              X
C...  ROUTINE DSS004 RESPECTIVELY).
C...
      DIMENSION U(N),UX(N)
C...
C...  COMPUTE THE SPATIAL INCREMENT
      DX=(XU-XL)/FLOAT(N-1)
      R4FDX=1./(12.*DX)
      NM2=N-2
C...
C...  EQUATION (1) (NOTE - THE RHS OF EQUATIONS (1), (2), (3), (4)
C...  AND (5) HAVE BEEN FORMATTED SO THAT THE NUMERICAL WEIGHTING
C...  COEFFICIENTS CAN BE MORE EASILY ASSOCIATED WITH THE BICKLEY
C...  MATRIX ABOVE)
      UX(  1)=R4FDX*
     1( -25.*U(  1) +48.*U(  2) -36.*U(  3) +16.*U(  4)  -3.*U(  5))
C...
C...  EQUATION (2)
      UX(  2)=R4FDX*
     1(  -3.*U(  1) -10.*U(  2) +18.*U(  3)  -6.*U(  4)  +1.*U(  5))
C...
C...  EQUATION (3)
      DO 1 I=3,NM2
      UX(  I)=R4FDX*
     1(  +1.*U(I-2)  -8.*U(I-1)  +0.*U(  I)  +8.*U(I+1)  -1.*U(I+2))
1     CONTINUE
C...
C...  EQUATION (4)
      UX(N-1)=R4FDX*
     1(  -1.*U(N-4)  +6.*U(N-3) -18.*U(N-2) +10.*U(N-1)  +3.*U(  N))
C...
C...  EQUATION (5)
      UX(  N)=R4FDX*
     1(   3.*U(N-4) -16.*U(N-3) +36.*U(N-2) -48.*U(N-1) +25.*U(  N))
      RETURN
      END
```

Appendix 14: Formulas for A_0, A_1, A_2 in Equations (56) and (57)

The QU/PG element of Figure 7.1 consists of two parts for $x < 0$ and $x > 0$, each of which consists of the quadratic

$$\phi_i(x) = a_0 + a_1 x + a_2 x^2 \tag{a.39}$$

From $\phi_i(0) = 1$, $a_0 = 1$.

Also, for $x < 0$

$$\phi_i(-\Delta x/2) = \phi_L = 1 + a_1(-\Delta x/2) + a_2(-\Delta x/2)^2$$

$$\phi_i(-\Delta x) = 0 = 1 + a_1(-\Delta x) + a_2(-\Delta x)^2 \tag{a.40}$$

If equation (a.40) is multiplied by (-1/2)

$$\phi_i(-\Delta x/2) = \phi_L = 1 - a_1(\Delta x/2) + a_2(\Delta x/2)^2$$

$$(-1/2)\phi_i(-\Delta x) = 0 = -1/2 + a_1(\Delta x/2) - (1/2)a_2(\Delta x)^2 \tag{a.41}$$

Then solution of equations (a.41) for a_2, followed by a_1, gives

$$\phi_L = 1/2 - (1/4)a_2\Delta x^2$$

$$a_2 = (-4/\Delta x^2)(\phi_L - 1/2)$$

$$a_1\Delta x = 1 - (4/\Delta x^2)(\phi_L - 1/2)(\Delta x^2)$$

$$a_1 = (1/\Delta x)[1 - 4(\phi_L - 1/2)]$$

Thus, the equation for the QU/PG element for $x < 0$ is

$$\phi_i(x) = 1 + (1/\Delta x)[1 - 4(\phi_L - 1/2)]x - (4/\Delta x^2)(\phi_L - 1/2)x^2 \tag{a.42}$$

which can be checked by at the three points $x = 0$, $-\Delta x/2$ and $-\Delta x$:

$$\phi_i(0) = 1$$

$$\phi_i(-\Delta x/2) = 1 + (1/\Delta x)[1 - 4(\phi_L - 1/2)](-\Delta x/2) - (4/\Delta x^2)(\phi_L - 1/2)(-\Delta x/2)^2 =$$

$$= 1 - (1/2)[1 - 4(\phi_L - 1/2)] - (\phi_L - 1/2) = \phi_L$$

$$\phi_i(-\Delta x) = 1 + (1/\Delta x)[1 - 4(\phi_L - 1/2)](-\Delta x) - (4/\Delta x^2)(\phi_L - 1/2)(-\Delta x)^2$$

$$= 1 - [1 - 4(\phi_L - 1/2)] - (4)(\phi_L - 1/2) = 0$$

Also, we have for $\phi_L = 0.5$, $a_1 = 1/\Delta x$, $a_2 = 0$, $\phi_i(x) = 1 + x/\Delta x$ as expected.

For the section $x > 0$, we can arrive at the final result from equation (a.42) merely by replacing $-x \rightarrow x$, $\phi_R \rightarrow \phi_L$, i.e.,

$$\phi_i(x) = 1 - (1/\Delta x)[1 - 4(\phi_R - 1/2)]x - (4/\Delta x^2)(\phi_R - 1/2)x^2 \qquad \text{(a.43)}$$

which can be checked at the three points $x = 0$, $\Delta x/2$ and Δx:

$$\phi_i(0) = 1$$

$$\phi_i(\Delta x/2) = 1 - (1/\Delta x)[1 - 4(\phi_R - 1/2)](\Delta x/2) - (4/\Delta x^2)(\phi_R - 1/2)(\Delta x/2)^2 =$$

$$= 1 - (1/2)[1 - 4(\phi_R - 1/2)] - (\phi_R - 1/2) = \phi_R$$

$$\phi_i(\Delta x) = 1 - (1/\Delta x)[1 - 4(\phi_R - 1/2)](\Delta x) - (4/\Delta x^2)(\phi_R - 1/2)(\Delta x)^2$$

$$= 1 - [1 - 4(\phi_R - 1/2)] - (4)(\phi_R - 1/2) = 0$$

Also, we have for $\phi_R = 0.5$, $a_1 = -1/\Delta x$, $a_2 = 0$, $\phi_i(x) = 1 - x/\Delta x$ as expected.

Equations (a.42) and (a.43) are coded in functions PHIM and PHIP of Program 8 in Appendix 15. Also, the derivatives of $\phi_i(x)$ from equations (a.42) and (a.43) are programmed in functions PHIDM and PHIDP of Program 8 since these derivatives are required in the evaluation of integrals 2.1 to 2.3 and 4.1 to 4.3.

Appendix 15: Subroutines INITAL, DERV and PRINT, and Subordinate Routines for the QU/PG Solution of Advection Equation (7)

The following points should be noted about this program:

(1) Integrals 1.1 to 4.4 are evaluated in subroutine COEFF1 via the numerical quadrature routine QUANC8.

(2) The integrands of integrals 1.1 to 4.4 are evaluated in functions F1P1 to F4P4, respectively, which are called by QUANC8. In each of these functions, two integrands can be computed for: (a) the linear finite element of Figure 6.1 (ITYPE = 1) (the solution from this numerical integration agrees closely with the solution from the analytical integration in Program 7, and (b) the QU/PG element of Figure 7.1 (ITYPE = 2).

(3) Equations (a.42) and (a.43) are evaluated in functions PHIM and PHIP respectively, and their derivatives are evaluated in functions PHIDM and PHIDP, respectively. Note that in each of these four functions, ϕ_L or ϕ_R is defined numerically (as PHIL and PHIR, respectively).

(4) Main program SRKF45 of Appendix 4, in combination with ODE integrator RKF45, was used, along with the data of Table 7 for the five test functions.

```
      SUBROUTINE INITAL
C...
C...  PURE CONVECTION THROUGH A TUBE WITH A UNIFORM VELOCITY PROFILE CAN
C...  BE DEPICTED AS
C...
C...
C...          ..............................................
C...                         V......+.   .
C...                         V......+.   .
C...                         V......+.DX.  U(X,T)
C...                         V......+.   .
C...                         V......+.   .
C...          ..............................................
C...        X=0                                            X=XL
C...
C...  IF A MASS OR ENERGY BALANCE IS WRITTEN FOR A DIFFERENTIAL SECTION
C...  OF THE TUBE, THE WELL-KNOWN ADVECTION EQUATION RESULTS
C...
C...     U  + V*U  = 0                                                (1)
```

```
C...      T       X
C...
C...  SINCE EQUATION (1) IS FIRST-ORDER IN THE SPATIAL VARIABLE, X, AND
C...  THE TEMPORAL VARIABLE, T, IT REQUIRES ONE BOUNDARY CONDITION AND
C...  ONE INITIAL CONDITION
C...
C...     U(0,T) = F(T), U(X,0) = G(X)                                (2)(3)
C...
C...  FOR WHICH THE EXACT SOLUTION TO EQUATIONS (1) TO (3) IS
C...
C...     U(X,T) = 0, X GT V*T
C...                                                                  (4)
C...     U(X,T) = F(T - X/V), X LT V*T
C...
C...  THE SPATIAL DERIVATIVE IN EQUATION (1), U  IS REPLACED BY A FINITE
C...                                          X
C...  ELEMENT APPROXIMATION OVER A GRID OF N (= 21) POINTS IN X.  A
C...  SYSTEM OF N INITIAL-VALUE ORDINARY DIFFERENTIAL EQUATIONS IN T
C...  RESULTS, ONE DIFFERENTIAL EQUATION FOR EACH GRID POINT.  THIS
C...  SYSTEM OF ODES IS THEN INTEGRATED SIMULTANEOUSLY TO OBTAIN THE
C...  NUMERICAL SOLUTION TO EQUATIONS (1) TO (3).
C...
      PARAMETER (N=21)
      COMMON/T/       T,   NSTOP,   NORUN
     1      /Y/    U(N)
     2      /F/   UT(N)
     3     /SD/   UX(N)
     4      /C/       V,     SSE,      FT
     5     /FE/      DX,   ITYPE,CW(3,3),
     6   AL(N),   BM(N),   CU(N),DRHS(N)
     7      /I/      IP
C...
C...  COMMON AREA FOR I/O UNIT NUMBERS
      COMMON/IO/    NI,      NO
C...
C...  SET THE VELOCITY, LENGTH OF THE SYSTEM, NUMBER OF SPATIAL GRID
C...  POINTS, LENGTH OF THE GRID INTERVAL, SUM OF SQUARES OF ERRORS OF
C...  THE NUMERICAL SOLUTION, COUNTER FOR THE PLOTTED SOLUTION USED IN
C...  SUBROUTINE PRINT
      V=1.
      X1=0.
      XN=1.
      DX=(XN-X1)/FLOAT(N-1)
      SSE=0.
      IP=0
C...
C...  SELECT THE BASIS FUNCTIONS
C...
C...     ITYPE = 1, LINEAR FINITE ELEMENTS (HAT FUNCTIONS)
C...
C...     ITYPE = 2, QUADRATIC UPWIND/PETROV-GALERKIN (QU/PG)
```

```
C...
      ITYPE=2
C...
C...  SET UP THE COEFFICIENT MATRIX IN BAND STORAGE MODE. THIS IS
C...  DONE ONLY ONCE SINCE THE COEFFICIENT MATRIX IS CONSTANT
      CALL COEFF1
C...
C...  INITIAL CONDITION (5)
      DO 1 I=1,N
      U(I)=0.
1     CONTINUE
C...
C...  INITIALIZE THE TEMPORAL DERIVATIVES IN COMMON/F/
      CALL DERV
C...
C...  PRINT THE COEFFICIENT MATRIX FOR VERIFICATION
      IF(NORUN.EQ.1)THEN
         DO 2 I=1,N
            WRITE(NO,3)I,AL(I),BM(I),CU(I)
3           FORMAT(I5,3F12.4)
2        CONTINUE
      END IF
      RETURN
      END

      SUBROUTINE DERV
      PARAMETER (N=21)
      COMMON/T/       T,   NSTOP,   NORUN
     1      /Y/    U(N)
     2      /F/   UT(N)
     3     /SD/   UX(N)
     4      /C/       V,     SSE,      FT
     5     /FE/      DX,   ITYPE,CW(3,3),
     6   AL(N),   BM(N),   CU(N),DRHS(N)
     7      /I/      IP
C...
C...  UNIT STEP FUNCTION
      IF(NORUN.EQ.1)THEN
         FT=1.0E0
C...
C...  TRUNCATED RAMP FUNCTION
      ELSE IF(NORUN.EQ.2)THEN
         S=5.0
         IF(T.LT.0.)FT=0.
         IF(T.GT.(1./S))FT=1.
         IF((T.GE.0.).AND.(T.LE.(1./S)))FT=S*T
C...
C...  COSINE STEP FUNCTION
      ELSE IF(NORUN.EQ.3)THEN
         W=5.
         PI=4.*ATAN(1.)
```

```
         IF(T.LT.0.)FT=0.
         IF((W*T).GT.(PI/2.))FT=1.
         IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI/2.)))FT=1.-COS(W*T)
C...
C...  SMOOTH COSINE STEP FUNCTION
      ELSE IF(NORUN.EQ.4)THEN
         W=5.
         PI=4.*ATAN(1.)
         IF(T.LT.0.)FT=0.
         IF((W*T).GT.(PI))FT=1.
         IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI)))
     +      FT=(1.-COS(W*T))/2.0
C...
C...  COSINE PULSE FUNCTION
      ELSE IF(NORUN.EQ.5)THEN
         W=5.
         PI=4.*ATAN(1.)
         IF(T.LT.0.)FT=0.
         IF((W*T).GT.(PI))FT=0.
         IF(((W*T).GE.(0.0)).AND.((W*T).LE.(PI/2.)))FT=1.-COS(W*T)
         IF(((W*T).GT.(PI/2.)).AND.((W*T).LE.(PI))) FT=1.+COS(W*T)
      END IF
C...
C...  APPLY BOUNDARY CONDITION (4) AS A CONSTRAINT
         U(1)=FT
C...
C...  RIGHT HAND SIDE VECTOR
      DO 1 I=1,N
         IF(I.EQ.1)THEN
            DRHS(1)=-V*(CW(1,1)*U(1)+
     +                  CW(1,2)*U(2))
         ELSE
     +   IF(I.EQ.N)THEN
            DRHS(N)=-V*(CW(3,1)*U(N-1)+
     +                  CW(3,2)*U(N))
         ELSE
            DRHS(I)=-V*(CW(2,1)*U(I-1)+
     +                  CW(2,2)*U(I)+
     +                  CW(2,3)*U(I+1))
         END IF
1     CONTINUE
C...
C...  SOLVE THE LINEAR ALGEBRAIC EQUATIONS BY SUBROUTINE TRIDAG, WHICH
C...  RETURNS THE DERIVATIVE VECTOR UT IN COMMON/F/
      CALL TRIDAG(AL,BM,CU,DRHS,UT,N)
      RETURN
      END

      SUBROUTINE PRINT(NI,NO)
      PARAMETER (N=21)
      COMMON/T/       T,   NSTOP,   NORUN
```

```
     1       /Y/   U(N)
     2       /F/  UT(N)
     3      /SD/  UX(N)
     4       /C/       V,     SSE,      FT
     5      /FE/      DX,  ITYPE,CW(3,3),
     6   AL(N),  BM(N),  CU(N),DRHS(N)
     7       /I/      IP
C...
C...  DIMENSION THE ARRAYS FOR PLOTTING THE SOLUTION
      DIMENSION TPE(201),TPN(201),UPE(201),UPN(201)
C...
C...  PRINT A HEADING FOR NUMERICAL BSOLUTION
      IF(IP.EQ.0)WRITE(NO,1)
1     FORMAT(9X,1HT,6X,6HU(0,T),6X,6HU(1,T),7X,5HEXACT,11X,1HE)
C...
C...  STORE THE EXACT SOLUTION IN ARRAY UPE FOR PLOTTING
      X=1.0
      TXV=T-X/V
      DO 3 I=1,NP
C...
C...  UNIT STEP FUNCTION
      IF(NORUN.EQ.1)THEN
         IF(TXV.LE.0.)EXACT=0.
         IF(TXV.GT.0.)EXACT=1.0
C...
C...  TRUNCATED RAMP FUNCTION
      ELSE IF(NORUN.EQ.2)THEN
         S=5.0
         IF(TXV.LT.0.)EXACT=0.
         IF(TXV.GT.(1./S))EXACT=1.
         IF((TXV.GE.0.).AND.(TXV.LE.(1./S)))EXACT=S*TXV
C...
C...  COSINE STEP FUNCTION
      ELSE IF(NORUN.EQ.3)THEN
         W=5.0
         PI=4.*ATAN(1.)
         IF(TXV.LT.0.)EXACT=0.
         IF((W*TXV).GT.(PI/2.))EXACT=1.
         IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI/2.)))
     +      EXACT=1.-COS(W*TXV)
C...
C...  SMOOTH COSINE STEP FUNCTION
      ELSE IF(NORUN.EQ.4)THEN
         W=5.0
         PI=4.*ATAN(1.)
         IF(TXV.LT.0.)EXACT=0.
         IF((W*TXV).GT.(PI))EXACT=1.
         IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI)))
     +      EXACT=(1.-COS(W*TXV))/2.
C...
C...  COSINE PULSE FUNCTION
```

```
      ELSE IF(NORUN.EQ.5)THEN
            W=5.0
            PI=4.*ATAN(1.)
            IF(TXV.LT.0.)EXACT=0.
            IF((W*TXV).GT.(PI))EXACT=0.
            IF(((W*TXV).GE.(0.0)).AND.((W*TXV).LE.(PI/2.)))
     +         EXACT=1.-COS(W*TXV)
            IF(((W*TXV).GT.(PI/2.)).AND.((W*TXV).LE.(PI)))
     +         EXACT=1.+COS(W*TXV)
      END IF
3     CONTINUE
C...
C...  COMPUTE THE ERROR AT X = XN, FIGURE OF MERIT, SSE
         E=U(N)-EXACT
         SSE=SSE+E**2
C...
C...  PRINT THE NUMERICAL AND EXACT SOLUTIONS, ERROR
      IF((IP/10*10).EQ.IP)
     1 WRITE(NO,20)T,FT,U(N).EXACT,E
20    FORMAT(F10.2,4F12.3)
C...
C...  STORE THE NUMERICAL SOLUTION FOR SUBSEQUENT PLOTTING VIA TOP
C...  DRAWER
      IP=IP+1
      UPN(IP)=U(N)
C...
C...  STORE THE EXACT SOLUTION IN ARRAY UPE FOR PLOTTING
      UPE(IP)=EXACT
C...
C...  STORE THE INDEPENDENT VARIABLE FOR SUBSEQUENT PLOTTING VIA TOP
C...  DRAWER
      NP=201
      TPE(IP)=2.0*FLOAT(IP-1)/FLOAT(NP-1)
      TPN(IP)=TPE(IP)
C...
C...  TEST FOR THE END OF A RUN
      IF(IP.LT.NP)RETURN
C...
C...  OPEN FILE FOR TOP DRAWER PLOTTING
         OPEN(4,FILE='T.TOP',STATUS='NEW')
C...
C...  WRITE TOP DRAW FILE FOR PLOTTING
         WRITE(4,17)
17       FORMAT(' SET LIMITS X FROM 0 TO 2 Y FROM -0.5 TO 1.5',/,
     1          ' SET FONT DUPLEX')
         WRITE(4,13)
13       FORMAT(' SET WINDOW X 2 TO 6 Y 2 TO 8')
      WRITE(4,14)(TPE(I),UPE(I),I=1,NP)
14    FORMAT(2F10.4)
      WRITE(4,16)
16    FORMAT(' JOIN 1')
```

```
      WRITE(4,14)(TPN(I),UPN(I),I=1,NP)
      WRITE(4,16)
      WRITE(4,18)NORUN+55,SSE
18    FORMAT(
     1' Title 3.5 9.0 "Fig. ',I2,'"'
     2,/,' TITLE LEFT "u(1,t)" '
     3,/,' TITLE BOTTOM "t"'
     4,/,'Title 3.0, 0.75 "SSE = ',F6.3,'"')
      WRITE(4,15)
15    FORMAT(' NEW FRAME')
C...
C...  LEGEND OF RUNS AT THE END
      IF(NORUN.EQ.5)THEN
      WRITE(4,19)
19    FORMAT(
     1' Title 3.25 9.0 "QU/PG Basis Functions, L = 0.75, R = 0.25"',/,
     1' Title 3.25 8.5 "Fig. 56: Heaviside Unit Step Function"',/,
     1' Title 3.25 8.0 "Fig. 57: Truncated Ramp"',/,
     1' Title 3.25 7.5 "Fig. 58: Cosine Step"',/,
     1' Title 3.25 7.0 "Fig. 59: Smooth Cosine Step"',/,
     1' Title 3.25 6.5 "Fig. 60: Cosine Pulse"',/,
     1' Title 3.25 6.0 "All solutions are for a 21-point grid"')
C...
C...  PLOT THE BASIS FUNCTION AND ITS DERIVATIVE (ARRAYS TPE, UPE
C...  AND UPN ARE REUSED)
22    DO 21 I=1,NP
         TPE(I)=-DX+2.0*DX*(FLOAT(I-1)/FLOAT(NP-1))
         IF(TPE(I).LE.0.0)UPE(I)=PHIM (TPE(I))
         IF(TPE(I).LE.0.0)UPN(I)=PHIDM(TPE(I))
         IF(TPE(I).GT.0.0)UPE(I)=PHIP (TPE(I))
         IF(TPE(I).GT.0.0)UPN(I)=PHIDP(TPE(I))
21    CONTINUE
         WRITE(4,23)
23       FORMAT(' SET LIMITS X FROM -0.05 TO 0.05
     1   Y FROM -0.5 TO 1.5',/,' SET FONT DUPLEX')
         WRITE(4,24)
24       FORMAT(' SET WINDOW X 2 TO 8 Y 2 TO 8')
C...
C...  BASIS FUNCTION
      WRITE(4,15)
      WRITE(4,26)(TPE(I),UPE(I),I=1,NP)
26    FORMAT(2F10.5)
      WRITE(4,16)
C...
C...  DERIVATIVE OF BASIS FUNCTION
      WRITE(4,15)
      WRITE(4,26)(TPE(I),UPN(I),I=1,NP)
      WRITE(4,16)
      END IF
      RETURN
      END
```

```
      SUBROUTINE COEFF1
C...
C...  SUBROUTINE COEFF EVALUATES THE COEFFICIENT MATRIX FOR THE
C...  TRIDIAGONAL ODES (WHICH REQUIRES NUMERICAL QUADRATURE FOR
C...  A SERIES OF INTEGRALS)
C...
      PARAMETER (N=21)
      COMMON/T/       T,   NSTOP,   NORUN
     1      /Y/    U(N)
     2      /F/   UT(N)
     3     /SD/   UX(N)
     4      /C/       V,     SSE,      FT
     5     /FE/      DX,   ITYPE,CW(3,3),
     6   AL(N),   BM(N),   CU(N),DRHS(N)
     7      /I/      IP
C...
C...  COMMON FOR I/O UNIT NUMBERS
      COMMON/IO/     NI,      NO
C...
C...  EXTERNAL THE FUNCTIONS USED BY SUBROUTINE QUANC8 (A NUMERICAL
C...  QUADRATURE ROUTINE) TO EVALAUTE THE INTEGRANDS OF INTEGRALS
C...  5.1 TO 8.4
      EXTERNAL F1P1, F1P2, F1P3,
     +         F2P1, F2P2, F2P3,
     +         F3P1, F3P2, F3P3, F3P4,
     +         F4P1, F4P2, F4P3, F4P4
C...
C...  INTEGRAL 1.1
         CALL QUANC8(F1P1,-DX,DX,1.0E-05,1.0E-05,C1P1,EST,NFUN,FLAG)
         WRITE(NO,110)C1P1,EST
110      FORMAT(' C1P1 = ',F8.5,' EST = ',E11.3)
         IF(FLAG.NE.0.0)WRITE(NO,111)FLAG
111      FORMAT(' C1P1 MAY BE UNRELIABLE, FLAG = ',F6.2)
CC...
C...  INTEGRAL 1.2
         CALL QUANC8(F1P2,0.0,DX,1.0E-05,1.0E-05,C1P2,EST,NFUN,FLAG)
         WRITE(NO,112)C1P2,EST
112      FORMAT(' C1P2 = ',F8.5,' EST = ',E11.3)
         IF(FLAG.NE.0.0)WRITE(NO,113)FLAG
113      FORMAT(' C1P2 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 1.3
         CALL QUANC8(F1P3,-DX,0.0,1.0E-05,1.0E-05,C1P3,EST,NFUN,FLAG)
         WRITE(NO,114)C1P3,EST
114      FORMAT(' C1P3 = ',F8.5,' EST = ',E11.3)
         IF(FLAG.NE.0.0)WRITE(NO,115)FLAG
115      FORMAT(' C1P3 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 2.1
         CALL QUANC8(F2P1,-DX,DX,1.0E-05,1.0E-05,C2P1,EST,NFUN,FLAG)
```

```
      WRITE(NO,116)C2P1,EST
116   FORMAT(' C2P1 = ',F8.5,' EST = ',E11.3)
      IF(FLAG.NE.0.0)WRITE(NO,117)FLAG
117   FORMAT(' C2P1 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 2.2
      CALL QUANC8(F2P2,0.0,DX,1.0E-05,1.0E-05,C2P2,EST,NFUN,FLAG)
      WRITE(NO,118)C2P2,EST
118   FORMAT(' C2P2 = ',F8.5,' EST = ',E11.3)
      IF(FLAG.NE.0.0)WRITE(NO,119)FLAG
119   FORMAT(' C2P2 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 2.3
      CALL QUANC8(F2P3,-DX,0.0,1.0E-05,1.0E-05,C2P3,EST,NFUN,FLAG)
      WRITE(NO,120)C2P3,EST
120   FORMAT(' C2P3 = ',F8.5,' EST = ',E11.3)
      IF(FLAG.NE.0.0)WRITE(NO,121)FLAG
121   FORMAT(' C2P3 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 3.1
      CALL QUANC8(F3P1,0.0,DX,1.0E-05,1.0E-05,C3P1,EST,NFUN,FLAG)
      WRITE(NO,122)C3P1,EST
122   FORMAT(' C3P1 = ',F8.5,' EST = ',E11.3)
      IF(FLAG.NE.0.0)WRITE(NO,123)FLAG
123   FORMAT(' C3P1 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 3.2
      CALL QUANC8(F3P2,0.0,DX,1.0E-05,1.0E-05,C3P2,EST,NFUN,FLAG)
      WRITE(NO,124)C3P2,EST
124   FORMAT(' C3P2 = ',F8.5,' EST = ',E11.3)
      IF(FLAG.NE.0.0)WRITE(NO,125)FLAG
125   FORMAT(' C3P2 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 3.3
      CALL QUANC8(F3P3,-DX,0.0,1.0E-05,1.0E-05,C3P3,EST,NFUN,FLAG)
      WRITE(NO,126)C3P3,EST
126   FORMAT(' C3P3 = ',F8.5,' EST = ',E11.3)
      IF(FLAG.NE.0.0)WRITE(NO,127)FLAG
127   FORMAT(' C3P3 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 3.4
      CALL QUANC8(F3P4,-DX,0.0,1.0E-05,1.0E-05,C3P4,EST,NFUN,FLAG)
      WRITE(NO,128)C3P4,EST
128   FORMAT(' C3P4 = ',F8.5,' EST = ',E11.3)
      IF(FLAG.NE.0.0)WRITE(NO,129)FLAG
129   FORMAT(' C3P4 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 4.1
      CALL QUANC8(F4P1,0.0,DX,1.0E-05,1.0E-05,C4P1,EST,NFUN,FLAG)
      WRITE(NO,130)C4P1,EST
130   FORMAT(' C4P1 = ',F8.5,' EST = ',E11.3)
```

```
         IF(FLAG.NE.0.0)WRITE(NO,131)FLAG
131      FORMAT(' C4P1 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 4.2
         CALL QUANC8(F4P2,0.0,DX,1.0E-05,1.0E-05,C4P2,EST,NFUN,FLAG)
         WRITE(NO,132)C4P2,EST
132      FORMAT(' C4P2 = ',F8.5,' EST = ',E11.3)
         IF(FLAG.NE.0.0)WRITE(NO,133)FLAG
133      FORMAT(' C4P2 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 4.3
         CALL QUANC8(F4P3,-DX,0.0,1.0E-05,1.0E-05,C4P3,EST,NFUN,FLAG)
         WRITE(NO,134)C4P3,EST
134      FORMAT(' C4P3 = ',F8.5,' EST = ',E11.3)
         IF(FLAG.NE.0.0)WRITE(NO,135)FLAG
135      FORMAT(' C4P3 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  INTEGRAL 4.4
         CALL QUANC8(F4P4,-DX,0.0,1.0E-05,1.0E-05,C4P4,EST,NFUN,FLAG)
         WRITE(NO,136)C4P4,EST
136      FORMAT(' C4P4 = ',F8.5,' EST = ',E11.3)
         IF(FLAG.NE.0.0)WRITE(NO,137)FLAG
137      FORMAT(' C4P4 MAY BE UNRELIABLE, FLAG = ',F6.2)
C...
C...  LOWER DIAGONAL
      DO 23 I=1,N
         IF(I.EQ.1)THEN
            AL(1)=0.0E0
         ELSE
     +   IF(I.EQ.N)THEN
            AL(N)=C3P4
         ELSE
            AL(I)=C1P3
         END IF
23    CONTINUE
C...
C...  MAIN DIAGONAL
      DO 24 I=1,N
         IF(I.EQ.1)THEN
            BM(1)=C3P1
         ELSE
     +   IF(I.EQ.N)THEN
            BM(N)=C3P3
         ELSE
            BM(I)=C1P1
         END IF
24    CONTINUE
C...
C...  UPPER DIAGONAL
      DO 25 I=1,N
         IF(I.EQ.1)THEN
```

```
            CU(1)=C3P2
         ELSE
     +   IF(I.EQ.N)THEN
            CU(N)=0.0E0
         ELSE
            CU(I)=C1P2
         END IF
25    CONTINUE
C...
C...  WEIGHTING COEFFICIENTS IN CONVECTIVE TERM (TRANSFERRED THROUGH
C...  ARRAY CW(3,3) FOR USE IN SUBROUTINE DERV)
C...
C...  EQUATION 1
         CW(1,1)=C4P1
         CW(1,2)=C4P2
C...
C...  EQUATIONS 2 TO N-1
         CW(2,1)=C2P3
         CW(2,2)=C2P1
         CW(2,3)=C2P2
C...
C...  EQUATION N
         CW(3,1)=C4P4
         CW(3,2)=C4P3
      RETURN
      END

      REAL FUNCTION F1P1(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 1.1
C...
      COMMON/FE/     DX,   ITYPE
C...
C...  LINEAR BASIS FUNCTIONS
      IF(ITYPE.EQ.1)THEN
         IF(X.LE.0.0)F1P1=((X+DX)/DX)**2
         IF(X.GT.0.0)F1P1=((DX-X)/DX)**2
C...
C...  QUADRATIC BASIS FUNCTIONS
      ELSE IF(ITYPE.EQ.2)THEN
         IF(X.LE.0.0)F1P1=PHIM(X)**2
         IF(X.GT.0.0)F1P1=PHIP(X)**2
      END IF
      RETURN
      END

      REAL FUNCTION F1P2(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 1.2
C...
      COMMON/FE/     DX,   ITYPE
```

```
C...
C...  LINEAR BASIS FUNCTIONS
      IF(ITYPE.EQ.1)THEN
         F1P2=((DX-X)/DX)*(X/DX)
C...
C...  QUADRATIC BASIS FUNCTIONS
      ELSE IF(ITYPE.EQ.2)THEN
         F1P2=PHIP(X)*PHIM(X-DX)
      END IF
      RETURN
      END

      REAL FUNCTION F1P3(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 1.3
C...
      COMMON/FE/     DX,  ITYPE
C...
C...  LINEAR BASIS FUNCTIONS
      IF(ITYPE.EQ.1)THEN
         F1P3=((X+DX)/DX)*((-X)/DX)
C...
C...  QUADRATIC BASIS FUNCTIONS
      ELSE IF(ITYPE.EQ.2)THEN
         F1P3=PHIM(X)*PHIP(X+DX)
      END IF
      RETURN
      END

      REAL FUNCTION F2P1(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 2.1
C...
      COMMON/FE/     DX,  ITYPE
C...
C...  LINEAR BASIS FUNCTIONS
      IF(ITYPE.EQ.1)THEN
         IF(X.LE.0.0)F2P1=((X+DX)/DX)*( 1.0/DX)
         IF(X.GT.0.0)F2P1=((DX-X)/DX)*(-1.0/DX)
C...
C...  QUADRATIC BASIS FUNCTIONS
      ELSE IF(ITYPE.EQ.2)THEN
         IF(X.LE.0.0)F2P1=PHIM(X)*PHIDM(X)
         IF(X.GT.0.0)F2P1=PHIP(X)*PHIDP(X)
      END IF
      RETURN
      END

      REAL FUNCTION F2P2(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 2.2
```

```
C...
      COMMON/FE/      DX,   ITYPE
C...
C...  LINEAR BASIS FUNCTIONS
      IF(ITYPE.EQ.1)THEN
         F2P2=((DX-X)/DX)*(1.0/DX)
C...
C...  QUADRATIC BASIS FUNCTIONS
      ELSE IF(ITYPE.EQ.2)THEN
         F2P2=PHIP(X)*PHIDM(X-DX)
      END IF
      RETURN
      END

      REAL FUNCTION F2P3(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 2.3
C...
      COMMON/FE/      DX,   ITYPE
C...
C...  LINEAR BASIS FUNCTIONS
      IF(ITYPE.EQ.1)THEN
         F2P3=((X+DX)/DX)*(-1.0/DX)
C...
C...  QUADRATIC BASIS FUNCTIONS
      ELSE IF(ITYPE.EQ.2)THEN
         F2P3=PHIM(X)*PHIDP(X+DX)
      END IF
      RETURN
      END

      REAL FUNCTION F3P1(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 3.1,
C...  WHICH IS THE SAME AS THE INTEGRAND OF INTEGRAL 1.1
C...
      F3P1=F1P1(X)
      RETURN
      END

      REAL FUNCTION F3P2(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 3.2,
C...  WHICH IS THE SAME AS THE INTEGRAND OF INTEGRAL 1.2
C...
      F3P2=F1P2(X)
      RETURN
      END

      REAL FUNCTION F3P3(X)
C...
```

```
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 3.3,
C...  WHICH IS THE SAME AS THE INTEGRAND OF INTEGRAL 1.1
      F3P3=F1P1(X)
      RETURN
      END

      REAL FUNCTION F3P4(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 3.4,
C...  WHICH IS THE SAME AS THE INTEGRAND OF INTEGRAL 1.3
      F3P4=F1P3(X)
      RETURN
      END

      REAL FUNCTION F4P1(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 4.1,
C...  WHICH IS THE SAME AS THE INTEGRAND OF INTEGRAL 2.1
      F4P1=F2P1(X)
      RETURN
      END

      REAL FUNCTION F4P2(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 4.2,
C...  WHICH IS THE SAME AS THE INTEGRAND OF INTEGRAL 2.2
C...
      F4P2=F2P2(X)
      RETURN
      END

      REAL FUNCTION F4P3(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 4.3,
C...  WHICH IS THE SAME AS THE INTEGRAND OF INTEGRAL 2.1
      F4P3=F2P1(X)
      RETURN
      END

      REAL FUNCTION F4P4(X)
C...
C...  FUNCTION COMPUTES THE INTEGRAND OF INTEGRAL 4.4,
C...  WHICH IS THE SAME AS THE INTEGRAND OF INTEGRAL 2.3
C...
      F4P4=F2P3(X)
      RETURN
      END

      REAL FUNCTION PHIM(X)
C...
C...  FUNCTION PHIM COMPUTES THE QU/PG FUNCTION FOR X LE 0 WHICH GOES
```

```
C...   THROUGH THE THREE POINTS
C...
C...      PHI(-DX) = 0
C...
C...      PHI(-DX/2) = PHIL
C...
C...      PHI(0) = 1
C...
      COMMON/FE/      DX
         PHIL=0.75
         A0=1.0
         A1=(1.0/DX)*(1.0-4.0*(PHIL-1.0/2.0))
         A2=(-4.0/DX**2)*(PHIL-1.0/2.0)
         PHIM=A0+A1*X+A2*X**2
      RETURN
      END

      REAL FUNCTION PHIDM(X)
C...
C...   FUNCTION PHIDM COMPUTES THE DERIVATIVE OF PHIM
C...
      COMMON/FE/      DX
         PHIL=0.75
         A1=(1.0/DX)*(1.0-4.0*(PHIL-1.0/2.0))
         A2=(-4.0/DX**2)*(PHIL-1.0/2.0)
         PHIDM=A1+2.0*A2*X
      RETURN
      END

      REAL FUNCTION PHIP(X)
C...
C...   FUNCTION PHIP COMPUTES THE QU/PG FUNCTION FOR X GT 0 WHICH GOES
C...   THROUGH THE THREE POINTS
C...
C...      PHI(0) = 1
C...
C...      PHI(DX/2) = PHIR
C...
C...      PHI(DX) = 0
C...
      COMMON/FE/      DX
         PHIR=0.25
         A0=1.0
         A1=-(1.0/DX)*(1.0-4.0*(PHIR-1.0/2.0))
         A2=(-4.0/DX**2)*(PHIR-1.0/2.0)
         PHIP=A0+A1*X+A2*X**2
      RETURN
      END

      REAL FUNCTION PHIDP(X)
C...
```

```
C...   FUNCTION PHIDP COMPUTES THE DERIVATIVE OF PHIP
C...
       COMMON/FE/      DX
          PHIR=0.25
          A1=-(1.0/DX)*(1.0-4.0*(PHIR-1.0/2.0))
          A2=(-4.0/DX**2)*(PHIR-1.0/2.0)
          PHIDP=A1+2.0*A2*X
       RETURN
       END
```

Appendix 16: Analytical Solution of the Nonhomogeneous Advection Equation

The nonhomogensous advection equation is

$$u_t + vu_x = c(x,t) \tag{a.44}$$

which requires an initial condition and a boundary condition

$$u(x,0) = g(x),\ u(0,t) = f(t) \tag{a.45)(a.46}$$

Equations (a.44) to (a.46) are a restatement of equations (58) to (60). We now proceed to an analytical solution of equations (a.44) to (a.46) which can be used as a test of various numerical procedures.

If the Laplace transform of u(x,t) and c(x,t) with respect to t are defined as

$$L_t\left\{u(x,t)\right\} = \int_0^\infty u(x,t)e^{-st}dt = \bar{u}(x,s) \tag{a.47}$$

$$L_t\left\{c(x,t)\right\} = \int_0^\infty c(x,t)e^{-st}dt = \bar{c}(x,s) \tag{a.48}$$

equations (a.44) to (a.46) transform to

$$s\bar{u}(x,s) - g(x) + vd\bar{u}(x,s)/dx = \bar{c}(x,s) \tag{a.49}$$

$$\bar{u}(0,s) = \bar{f}(s) \tag{a.50}$$

If the Laplace transform of $\bar{u}(x,s)$ and $\bar{c}(x,s)$ and with respect to x are defined as

$$L_x\left\{\bar{u}(x,s)\right\} = \int_0^\infty \bar{u}(x,s)e^{-px}dx = \bar{\bar{u}}(p,s) \tag{a.51}$$

$$L_x\left\{\bar{c}(x,s)\right\} = \int_0^\infty \bar{c}(x,s)e^{-px}dx = \bar{\bar{c}}(p,s) \tag{a.52}$$

equations (a.49) and (a.50) transform to

$$s\bar{\bar{u}}(p,s) - \bar{g}(p) + v\{p\bar{\bar{u}}(p,s) - \bar{f}(s)\} = \bar{\bar{c}}(p,s) \tag{a.53}$$

Solution of equation (a.53) for $\bar{\bar{u}}(p,s)$ gives

$$\bar{\bar{u}}(p,s) = (1/v)\frac{1}{(s/v + p)}\bar{g}(p) + \frac{1}{(s/v + p)}\bar{f}(s) + (1/v)\frac{1}{(s/v + p)}\bar{\bar{c}}(p,s) \tag{a.54}$$

Equation (a.54) can now be inverted, first with respect to p

$$\bar{u}(x,s) = L^{-1}\left\{\bar{\bar{u}}(p,s)\right\} = (1/v)\int_0^x e^{-(s/v)(x - \lambda)}g(\lambda)d\lambda + e^{-(s/v)x}\,\bar{f}(s)$$

$$+ (1/v)\int_0^x e^{-(s/v)(x - \lambda)}\bar{c}(\lambda,s)d\lambda$$

$$= (1/v)\,e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}g(\lambda)d\lambda + e^{-(s/v)x}\,\bar{f}(s) + (1/v)\,e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}\bar{c}(\lambda,s)d\lambda$$

$$= (1/v)\,\bar{G}(x,s) + e^{-(s/v)x}\,\bar{f}(s) + (1/v)\,\bar{C}(x,s) \tag{a.55}$$

where

$$\bar{G}(x,s) = e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}g(\lambda)d\lambda \tag{a.56}$$

$$\bar{C}(x,s) = e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}\bar{c}(\lambda,s)d\lambda \tag{a.57}$$

Then, inversion with respect to s gives

$$u(x,t) = L^{-1}\left\{\bar{u}(x,s)\right\} = (1/v)G(x,t) + f(t - x/v)h(t - x/v) + (1/v)C(x,t) \tag{a.58}$$

where

$$G(x,t) = L^{-1}\left\{\bar{G}(x,s)\right\} \tag{a.59}$$

$$C(x,t) = L^{-1}\left\{\bar{C}(x,s)\right\} \tag{a.60}$$

and again

$$h(t) = \begin{matrix} 0, & t < 0 \\ 1, & t > 0 \end{matrix} \tag{a.61}$$

Some special cases can now be considered:

(1) $c(x,t) = d(x)$ and equations (a.48), (a.52) and (a.54) give

$$L_t\left\{d(x)\right\} = \int_0^\infty d(x)e^{-st}dt = d(x)/s \tag{a.62}$$

$$L_x\left\{d(x)/s\right\} = \int_0^\infty [d(x)/s]e^{-px}dx = \bar{d}(p)/s \tag{a.63}$$

$$\bar{\bar{u}}(p,s) = (1/v)\frac{1}{(s/v + p)}\bar{g}(p) + \frac{1}{(s/v + p)}\bar{f}(s) + (1/v)\frac{1}{s(s/v + p)}\bar{d}(p) \tag{a.64}$$

Inversion with respect to p then proceeds as before

$$\bar{u}(x,s) = L^{-1}\left\{\bar{\bar{u}}(p,s)\right\} = (1/v)\int_0^x e^{-(s/v)(x - \lambda)}g(\lambda)d\lambda + e^{-(s/v)x}\,\bar{f}(s)$$

$$+ (1/v)\int_0^x e^{-(s/v)(x - \lambda)}[d(\lambda)/s]d\lambda$$

$$= (1/v)\,e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}g(\lambda)d\lambda + e^{-(s/v)x}\,\bar{f}(s) + (1/v)(1/s)\,e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}d(\lambda)d\lambda$$

$$= (1/v)\,\bar{G}(x,s) + e^{-(s/v)x}\,\bar{f}(s) + (1/v)\,\bar{D}(x,s) \tag{a.65}$$

where

$$\bar{G}(x,s) = e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}g(\lambda)d\lambda \tag{a.66}$$

$$\overline{D}(x,s) = (1/s)\, e^{-(s/v)x} \int_0^x e^{(s/v)\lambda} d(\lambda) d\lambda \tag{a.67}$$

Then, inversion with respect to s gives

$$u(x,t) = L^{-1}\left\{\bar{u}(x,s)\right\} = (1/v)G(x,t) + f(t - x/v)h(t - x/v) + (1/v)D(x,t) \tag{a.68}$$

where

$$G(x,t) = L^{-1}\left\{\overline{G}(x,s)\right\} \tag{a.69}$$

$$D(x,t) = L^{-1}\left\{\overline{D}(x,s)\right\} \tag{a.70}$$

We can also consider the limiting condition for u(x,t) as $t \to \infty$ using the final value theorem of the Laplace transform

$$\lim_{t \to \infty} u(x,t) = \lim_{s \to 0} s\bar{u}(x,s) = (1/v)\int_0^x d(\lambda) d\lambda \tag{a.71}$$

where we have assumed

$$\lim_{t \to \infty} f(t) = \lim_{s \to 0} s\tilde{f}(s) = 0 \tag{a.72}$$

Thus, for this limiting condition, the solution is merely the integral of d(x) with respectt to x, which can be used as a convenient check of the steady state numerical solution. This follows also from equations (a.44) and (a.45) written at steady state

$$vu_x = d(x) \tag{a.73}$$

$$u(0) = 0 \tag{a.74}$$

(2) c(x,t) = e(t) and equations (a.51), (a.52) and (a.64) give

$$L_t\left\{e(t)\right\} = \int_0^\infty e(t)e^{-st}dt = \bar{e}(s) \tag{a.75}$$

$$L_x\left\{\bar{e}(s)\right\} = \int_0^\infty \bar{e}(s)e^{-px}dx = \bar{e}(s)/p \tag{a.76}$$

$$\bar{\bar{u}}(p,s) = (1/v)\frac{1}{(s/v + p)}\bar{g}(p) + \frac{1}{(s/v + p)}\bar{f}(s) + (1/v)\frac{1}{p(s/v + p)}\bar{e}(s) \tag{a.77}$$

Inversion with respect to p then proceeds as before

$$\bar{u}(x,s) = L^{-1}\left\{\bar{\bar{u}}(p,s)\right\} = (1/v)\int_0^x e^{-(s/v)(x - \lambda)}g(\lambda)d\lambda + e^{-(s/v)x}\,\bar{f}(s)$$
$$+ (1/v)\bar{e}(s)\int_0^x e^{-(s/v)\lambda}d\lambda$$

$$= (1/v)\,e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}g(\lambda)d\lambda + e^{-(s/v)x}\,\bar{f}(s) + (1/v)\bar{e}(s)(-v/s)[e^{-(s/v)x} - 1]$$

$$= (1/v)\,\bar{G}(x,s) + e^{-(s/v)x}\,\bar{f}(s) + (1/v)\,\bar{E}(x.s) \tag{a.78}$$

where

$$\bar{G}(x,s) = e^{-(s/v)x}\int_0^x e^{(s/v)\lambda}g(\lambda)d\lambda \tag{a.79}$$

$$\bar{E}(x,s) = \bar{e}(s)(-v/s)[e^{-(s/v)x} - 1] \tag{a.80}$$

Then, inversion with respect to s gives

$$u(x,t) = L^{-1}\left\{\bar{u}(x,s)\right\} = (1/v)G(x,t) + f(t - x/v)h(t - x/v) + (1/v)E(x,t) \tag{a.81}$$

where

$$G(x,t) = L^{-1}\left\{\bar{G}(x,s)\right\} \tag{a.82}$$

$$E(x,t) = L^{-1}\left\{\bar{E}(x,s)\right\} \tag{a.83}$$

We can also consider the limiting condition for $u(x,t)$ as $t \to \infty$ using the final value theorem of the Laplace transform applied to equation (a.78)

$$\lim_{t \to \infty} u(x,t) = \lim_{s \to 0} s\bar{u}(x,s) = \lim_{s \to 0} \bar{e}(s)[1 - e^{-(s/v)x}] \tag{a.84}$$

where we have again assumed equation (a.72) applies.

If $e(t) = h(t)$, $\bar{e}(s) = 1/s$, by l'Hospital's rule

$$\lim_{t \to \infty} u(x,t) = \lim_{s \to 0} s\bar{u}(x,s) = \lim_{s \to 0} (1/s)[1 - e^{-(s/v)x}] = x/v \tag{a.85}$$

which is the solution to the problem

$$vu_x = 1 \tag{a.86}$$

$$u(0) = 0 \tag{a.87}$$

as expected.

Appendix 17: Analytical Solution of the Simultaneous PDEs

We consider here the solution of equations (62) to (66), which are renumbered here

$$v\left(\frac{\partial V}{\partial t} + c\frac{\partial V}{\partial x}\right) = k(U - V) \tag{a.88}$$

$$u\frac{\partial U}{\partial t} = k(V - U) \tag{a.89}$$

$$V(x,0),\ V(0,t) \text{ and } U(x,0) \qquad \text{(a.90)(a.91)(a.92)}$$

If Laplace transforms of V(x,t) and U(x,t) with respect t are defined as

$$L_t[V(x,t)] = \int_0^\infty V(x,t)e^{-st}dt = \overline{V}(x,s) \tag{a.93}$$

$$L_t[U(x,t)] = \int_0^\infty U(x,t)e^{-st}dt = \overline{U}(x,s) \tag{a.94}$$

Application of these transforms to equations (a.88) to (a.92) gives

$$v\left\{s\overline{V}(x,s) - V(x,0) + c\frac{d\overline{V}(x,s)}{dx}\right\} = k(\overline{U}(x,s) - \overline{V}(x,s)) \tag{a.95}$$

$$u\left\{s\overline{U}(x,s) - U(x,0)\right\} = -\,k(\overline{U}(x,s) - \overline{V}(x,s)) \tag{a.96}$$

Equation (a.96) can be solved for $\overline{U}(x,s)$

$$\overline{U}(x,s) = \frac{k\overline{V}(x,s)}{us + k} + \frac{uU(x,0)}{us + k} \tag{a.97}$$

Substitution of equation (a.97) in equation (a.95) then gives

$$vc\frac{d\overline{V}(x,s)}{dx} + (vs + k)\overline{V}(x,s) = k\overline{U}(x,s) + vV(x,0) \tag{a.98}$$

Equation (a.98) is easily rearranged to

$$vc\frac{d\overline{V}(x,s)}{dx} + (vs + k - \frac{k^2}{us + k})\overline{V}(x,s) = \frac{kuU(x,0)}{us + k} + vV(x,0) \tag{a.99}$$

or if f(s) is defined as

$$f(s) = \left(vs + k - \frac{k^2}{us + k}\right)/vc \tag{a.100}$$

equation (a.99) becomes

$$\frac{d\overline{V}(x,s)}{dx} + f(s)\overline{V}(x,s) = \frac{(\frac{ku}{vc})U(x,0)}{us + k} + (1/c)V(x,0) \tag{a.101}$$

Equation (a.101) is a first order ordinary differential equation defining $\overline{V}(x,s)$ as a function of x. If both sides of equation (a.101) are multiplied by $e^{f(s)x}$, which introduces an integrating factor [Strang (11), pp 472-473]

$$e^{f(s)x}\left(\frac{d\overline{V}(x,s)}{dx} + f(s)\overline{V}(x,s)\right) = e^{f(s)x}\left(\frac{(\frac{ku}{vc})U(x,0)}{us + k} + (1/c)V(x,0)\right) \tag{a.102}$$

integration of both sides of equation (a.102) with respect to x gives

$$e^{f(s)x}\overline{V}(x,s) - V(0,s) = \int_0^x e^{f(s)\lambda}\left(\frac{(\frac{ku}{vc})U(\lambda,0)}{us + k} + (1/c)V(\lambda,0)\right)d\lambda$$

or

$$\overline{V}(x,s) = V(0,s)e^{-f(s)x} + \int_0^x e^{f(s)(\lambda - x)}\left(\frac{(\frac{ku}{vc})U(\lambda,0)}{us + k} + (1/c)V(\lambda,0)\right)d\lambda \tag{a.103}$$

V(x,t) can now be obtained by an inverse Laplace transform

$$V(x,t) = L^{-1}\left\{\overline{V}(x,s)\right\}$$

$$= \frac{1}{2\pi i}\int_{\gamma - i\infty}^{\gamma + i\infty}\left\{V(0,s)e^{-f(s)x} + \int_0^x e^{f(s)(\lambda - x)}\left(\frac{(\frac{ku}{vc})U(\lambda,0)}{us + k} + (1/c)V(\lambda,0)\right)d\lambda\right\}e^{st}ds \tag{a.104}$$

Equation (a.104) is the final solution we seek. However, in order to produce a solution which

can be evaluated numerically with reasonable effort, we take the initial conditions, equations (a.90) and (a.92), to be homogeneous

$$V(x,0) = U(x,0) = 0 \qquad \text{(a.105)(a.106)}$$

and equation (a.104) reduces to

$$V(x,t) = \frac{1}{2\pi i} \int_{\gamma - i\infty}^{\gamma + i\infty} \left\{ V(0,s) e^{-\left((vs + k - \frac{k^2}{us + k})/(vc) \right) x} \right\} e^{st} ds$$

which rearranges to

$$V(x,t) = \frac{1}{2\pi i} \int_{\gamma - i\infty}^{\gamma + i\infty} e^{-(k/(vc))x} \left\{ V(0,s) e^{-(s/c)x} e^{\left(\frac{k^2/(uvc)}{s + k/u}\right)x} \right\} e^{st} ds \qquad \text{(a.107)}$$

At this point, we can consider two types of boundary conditions (specifications of $V(0,x)$):

(1) A boundary condition which is consistent with the initial condition, and thereby avoids a discontinuity. An example is

$$V(0,t) = 1 - e^{-(k/u)t} \qquad \text{(a.108)}$$

for which the transform is

$$\overline{V}(0,s) = \frac{1}{s} \frac{k/u}{s + k/u} \qquad \text{(a.109)}$$

(2) A boundary condition which is not consistent with the initial condition, and therefore introduces a discontinuity. Examples are

$$V(0,t) = e^{-(k/u)t} \qquad \text{(a.110)}$$

$$\overline{V}(0,s) = \frac{1}{s + k/u} \qquad \text{(a.111)}$$

and

$$V(0,t) = h(t) = \begin{cases} 0, & t < 0 \\ 1, & t > 0 \end{cases} \qquad \text{(a.112)}$$

$$\bar{V}(0,s) = 1/s \tag{a.113}$$

We now consider each of these examples.

We first note the following transform pairs

$$L^{-1}\left\{\frac{1}{s}e^{k/s}\right\} = I_o(2\sqrt{kt}) \tag{a.114}$$

[Carslaw and Jaeger, (12), p 495]

$$L^{-1}\left\{\bar{f}(s + a)\right\} = e^{-at}f(t) \tag{a.115}$$

$$L^{-1}\left\{e^{-as}\bar{f}(s)\right\} = h(t - a)f(t - a) \tag{a.116}$$

$$L^{-1}\left\{\frac{1}{s}\bar{f}(s)\right\} = \int_0^t f(\lambda)d\lambda \tag{a.117}$$

[Beyer (13), p 600]

V(x,t) for boundary condition (a.108) is therefore (from substituting equation (a.109) in equation (a.107))

$$V(x,t) = e^{-(k/(vc))x}\, L^{-1}\left\{ \frac{1}{s}\,\frac{k/u}{s + k/u}e^{-(s/c)x}e^{\left(\frac{k^2/(uvc)}{s + k/u}\right)x}\right\} \tag{a.118}$$

or

$$V(x,t) = e^{-(k/(vc))x}\,(k/u)\int_0^t h(\lambda - x/c)e^{-(k/u)(\lambda - x/c)}\, I_o\left\{2k\sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\}d\lambda \tag{a.119}$$

A special case of equation (a.119) can be checked by applying the final value theorem of the Laplace transform to equation (a.118)

$$\lim_{t\to\infty} V(x,t) = \lim_{s\to 0} s\,V(x,s) = \lim_{s\to 0} e^{-(k/(vc))x}\left\{\frac{k/u}{s + k/u}e^{-(s/c)x}e^{\left(\frac{k^2/(uvc)}{s + k/u}\right)x}\right\}$$

$$e^{-(k/(vc))x}\, e^{(k/(vc))x} = 1$$

as expected (i.e., $V(0,t) = 1 - e^{-(k/u)t}$ at $x = 0$ gives $V(x,\infty) = 1$).

For boundary condition (a.110) we have (from substituting equation (a.111) in equation (a.107))

$$V(x,t) = e^{-(k/(vc))x} L^{-1}\left\{ \frac{1}{s + k/u} e^{-(s/c)x} e^{\left(\frac{k^2/(uvc)}{s + k/u}\right)x} \right\} \tag{a.120}$$

or

$$V(x,t) = e^{-(k/(vc))x} h(\lambda - x/c) e^{-(k/u)(\lambda - x/c)} I_o\left\{2k \sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\} \tag{a.121}$$

The final value theorem of the Laplace transform applied to equation (a.120) gives

$$\lim_{t \to \infty} V(x,t) = \lim_{s \to 0} s\, V(x,s) = \lim_{s \to 0} e^{-(k/(vc))x} s \left\{ \frac{1}{s + k/u} e^{-(s/c)x} e^{\left(\frac{k^2/(uvc)}{s + k/u}\right)x} \right\} = 0$$

as expected (i.e., $V(0,t) = e^{-(k/u)t}$ gives $V(x,\infty) = 0$).

For boundary condition (a.112) we have (from substituting equation (a.113) in equation (a.107))

$$V(x,t) = e^{-(k/(vc))x} L^{-1}\left\{ \frac{1}{s} e^{-(s/c)x} e^{\left(\frac{k^2/(uvc)}{s + k/u}\right)x} \right\} \tag{a.122}$$

or

$$V(x,t) = e^{-(k/(vc))x} L^{-1}\left\{ \left(\frac{1}{s + k/u} + \frac{1}{s} \frac{k/u}{s + k/u} \right) e^{-(s/c)x} e^{\left(\frac{k^2/(uvc)}{s + k/u}\right)x} \right\} \tag{a.123}$$

Equation (a.123) is just a superposition of equations (a.118) and (a.120). Thus, from equation (a.123),

$$\begin{aligned} V(x,t) = {} & e^{-(k/(vc))x} (k/u) \int_0^t h(\lambda - x/c) e^{-(k/u)(\lambda - x/c)} I_o\left\{2k \sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\} d\lambda \\ & + \left\{ e^{-(k/(vc))x} h(\lambda - x/c) e^{-(k/u)(\lambda - x/c)} I_o\left\{2k \sqrt{\frac{x}{uvc}(\lambda - x/c)}\right\} \right\} \end{aligned} \tag{a.124}$$

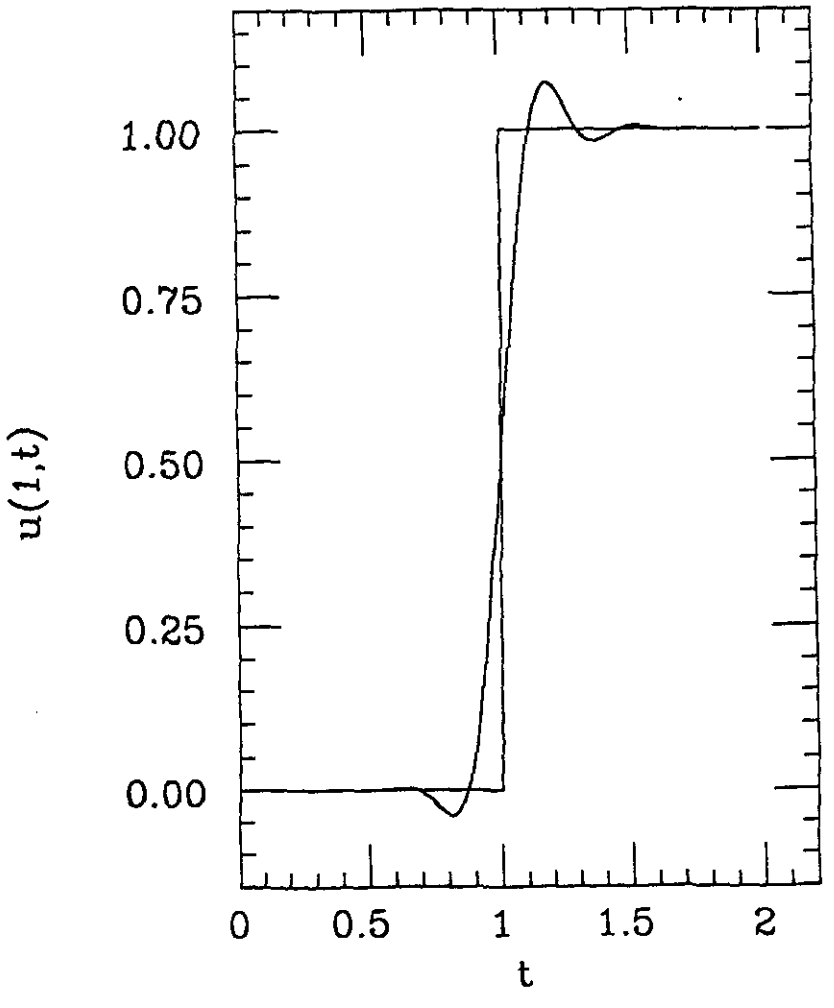
The final value theorem of the Laplace transform applied to equation (a.123) gives

$$\lim_{t \to \infty} V(x,t) = \lim_{s \to 0} s\, V(x,s) = 1$$

as expected (i.e., $V(0,t) = h(t)$ gives $V(x,\infty) = 1$).

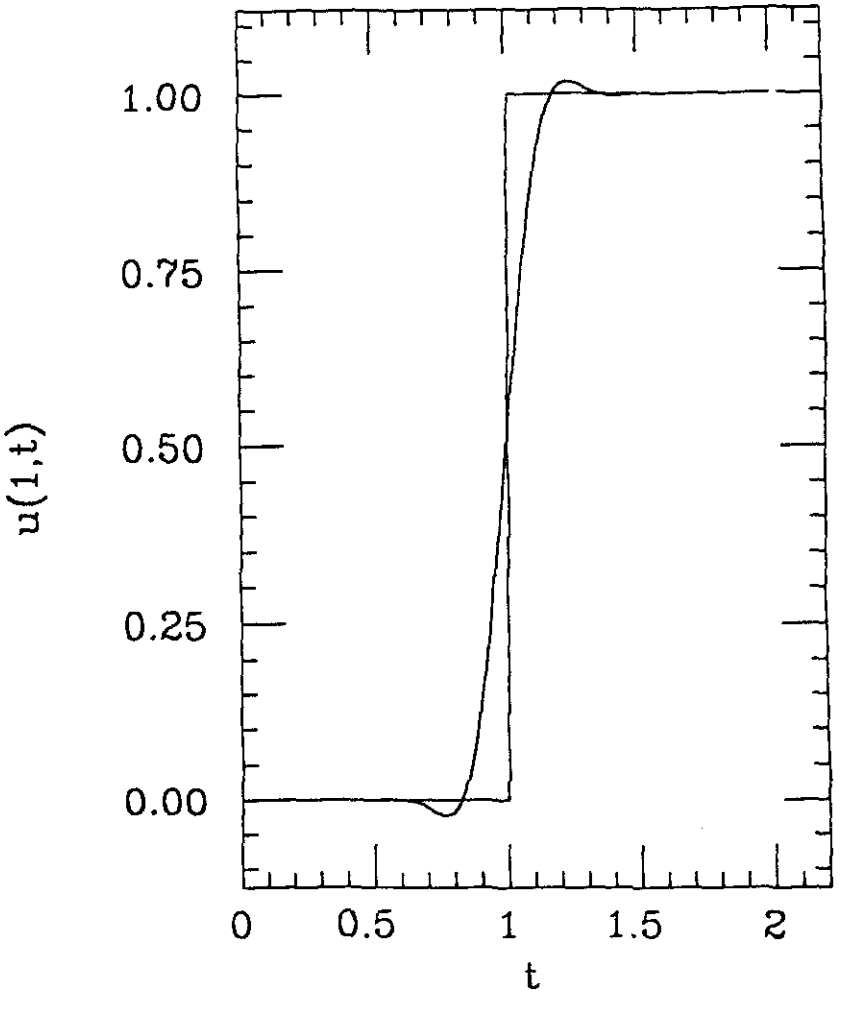
These three examples provide a spectrum of test problems since, by varying k/u, the rate of change of $V(0,t)$ can be varied for both the continuous and discontinuous cases. These examples can therefore be used to test a computer code which implements a numerical solution of equations (a.88) and (a.89) (or equations (63) and (64)).

Fig. 3



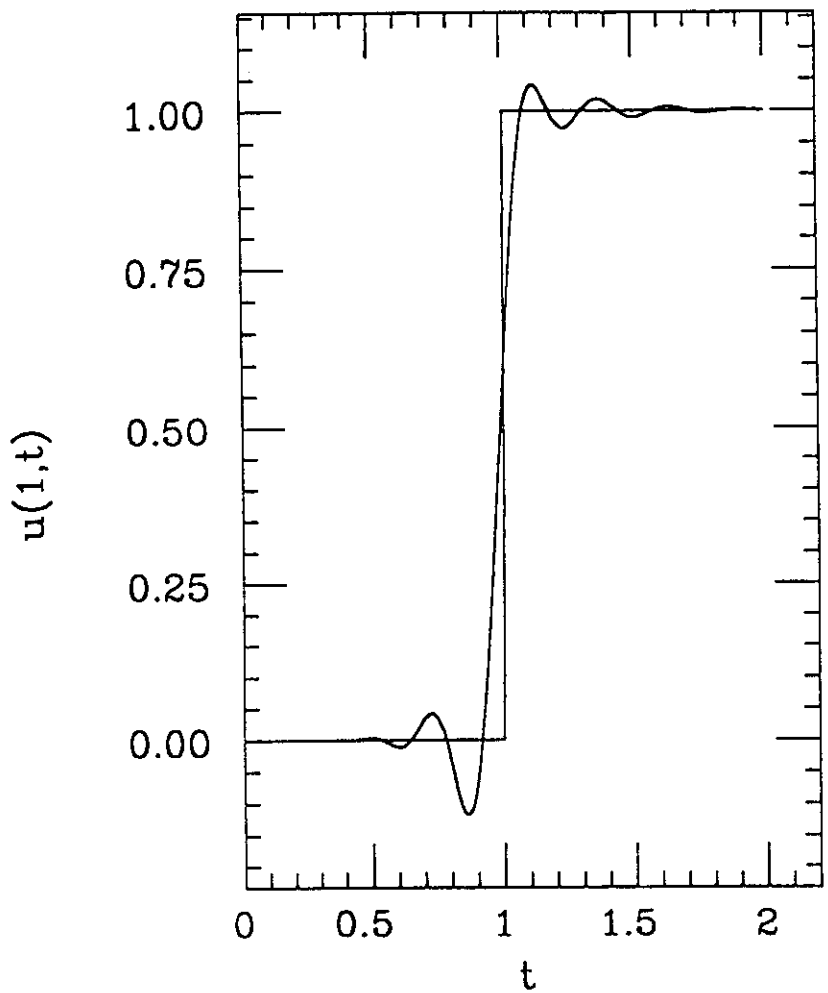
SSE = 1.711

Fig. 4



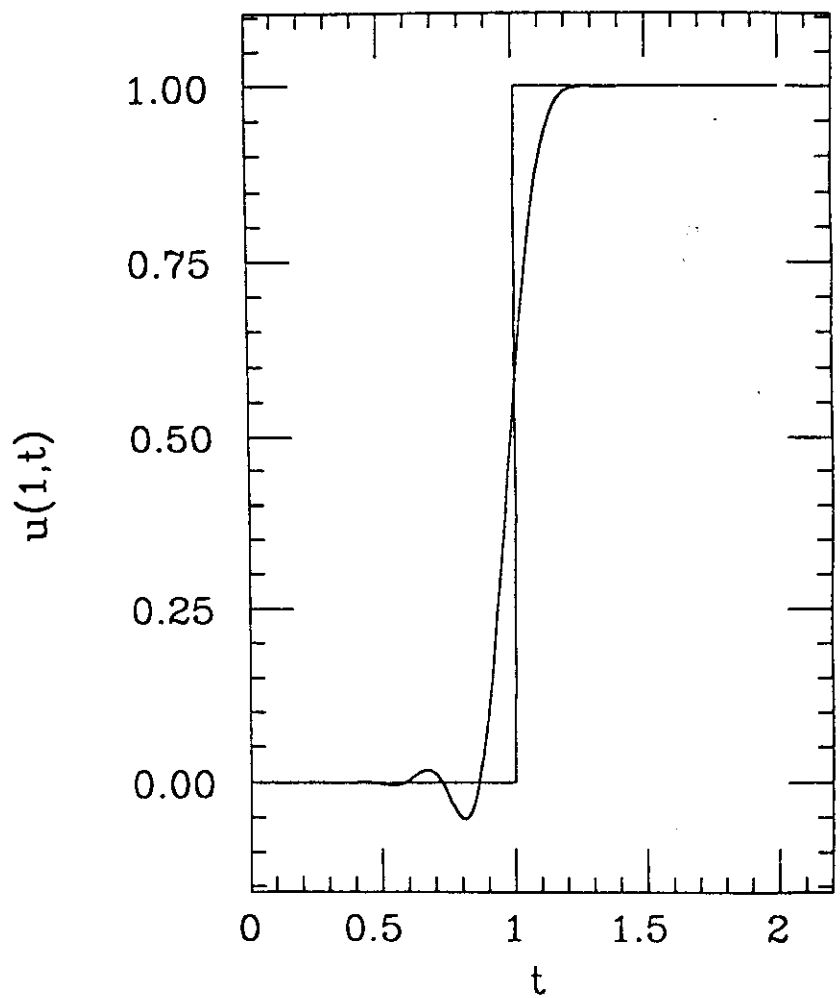
SSE = 2.157

Fig. 5



SSE = 1.392

Fig. 6



SSE = 2.036

Heaviside Unit Step Funtion

Fig. 1: Two Point Upwind

Fig. 2: Three Point Upwind

Fig. 3: Four Point Biased Upwind, 4PB

Fig. 4: Four Point Biased Upwind, 2PB

Fig. 5: Five Point Biased Upwind, 5PB

Fig. 6: Five Point Biased Upwind, 2PB

Fig. 7: Leonard, 4PB

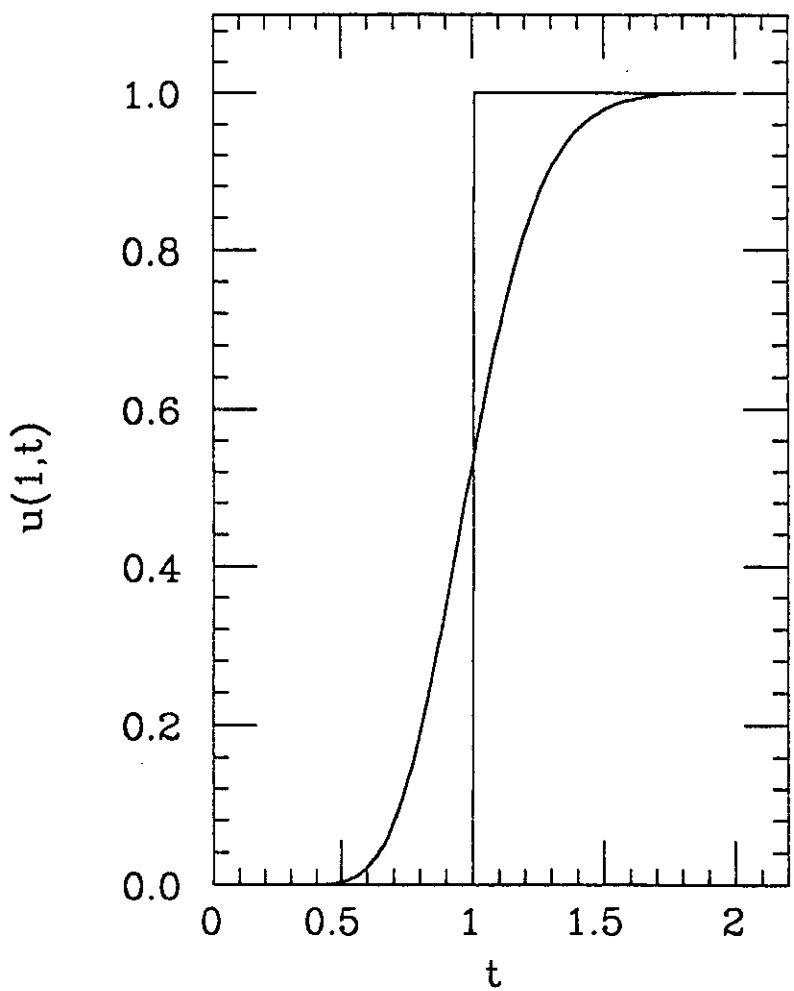
Fig. 8: Leonard, 2PB

Fig. 9: Three Point Centered

Fig. 10: Five Point Centered

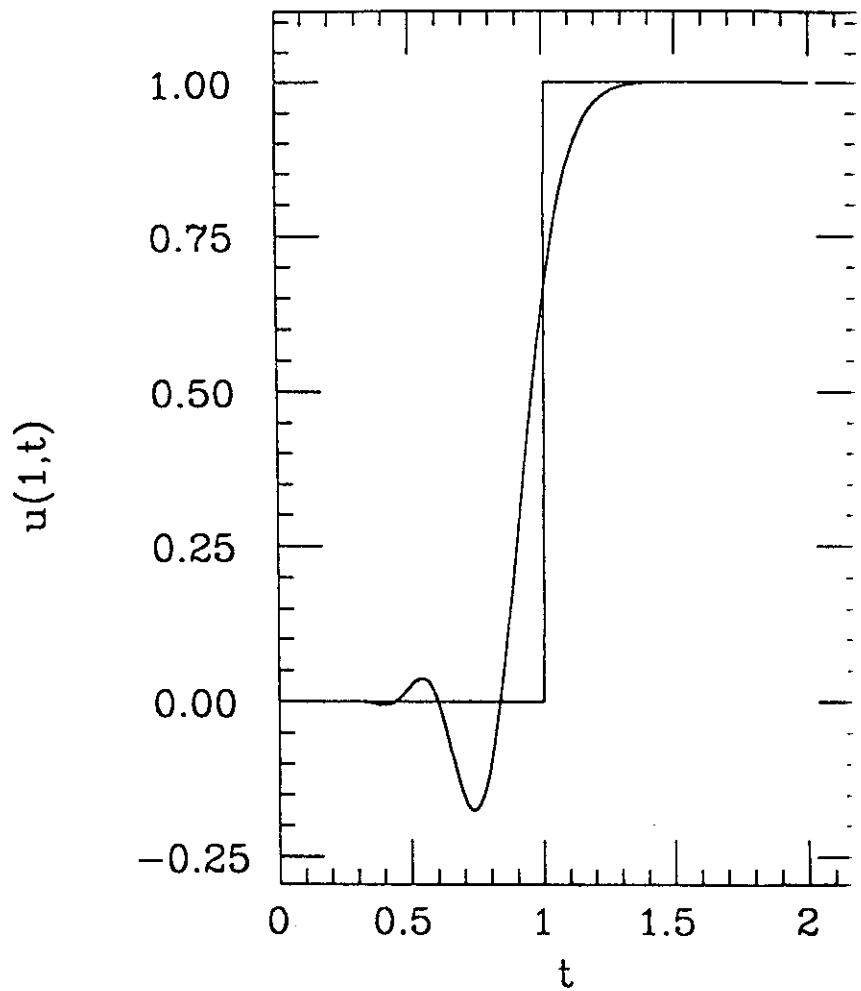
All solutions are for a 21-point grid

Fig. 1



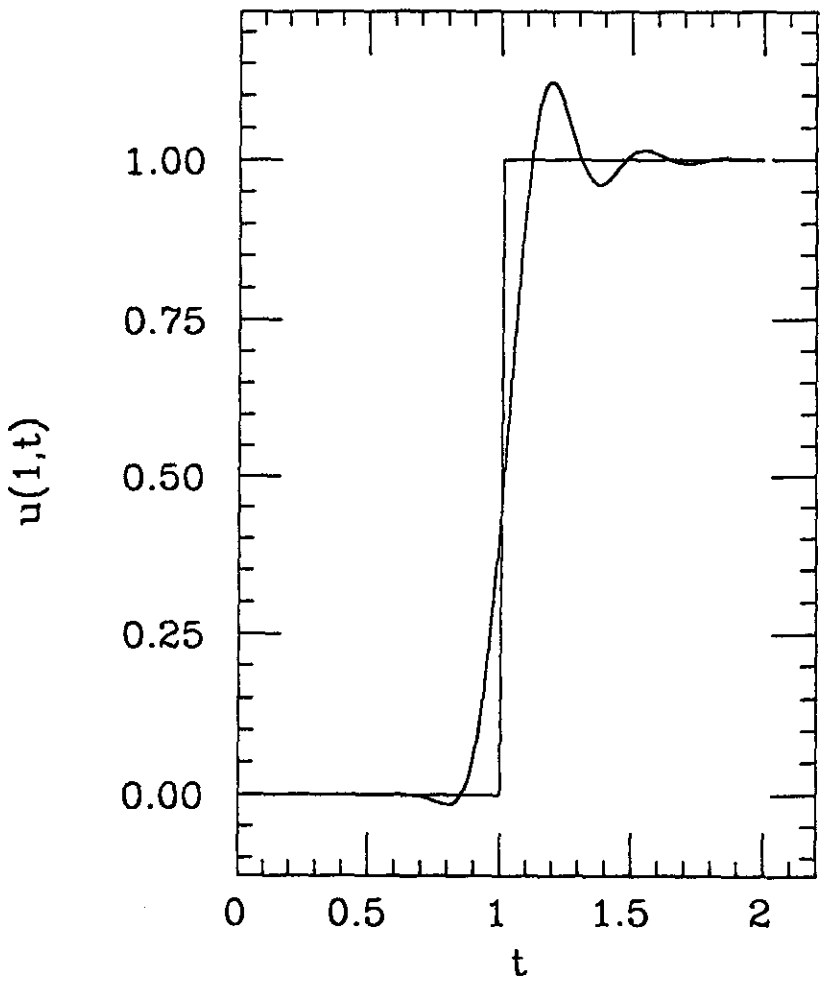
SSE = 5.263

Fig. 2



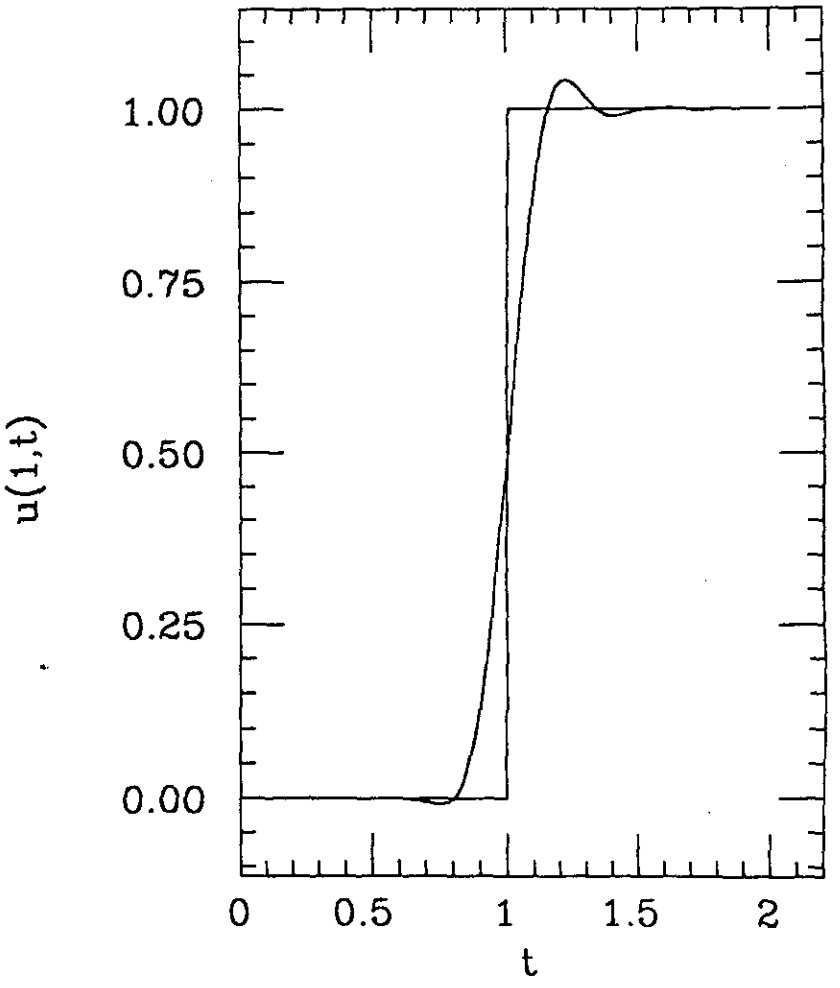
SSE = 3.494

Fig. 7



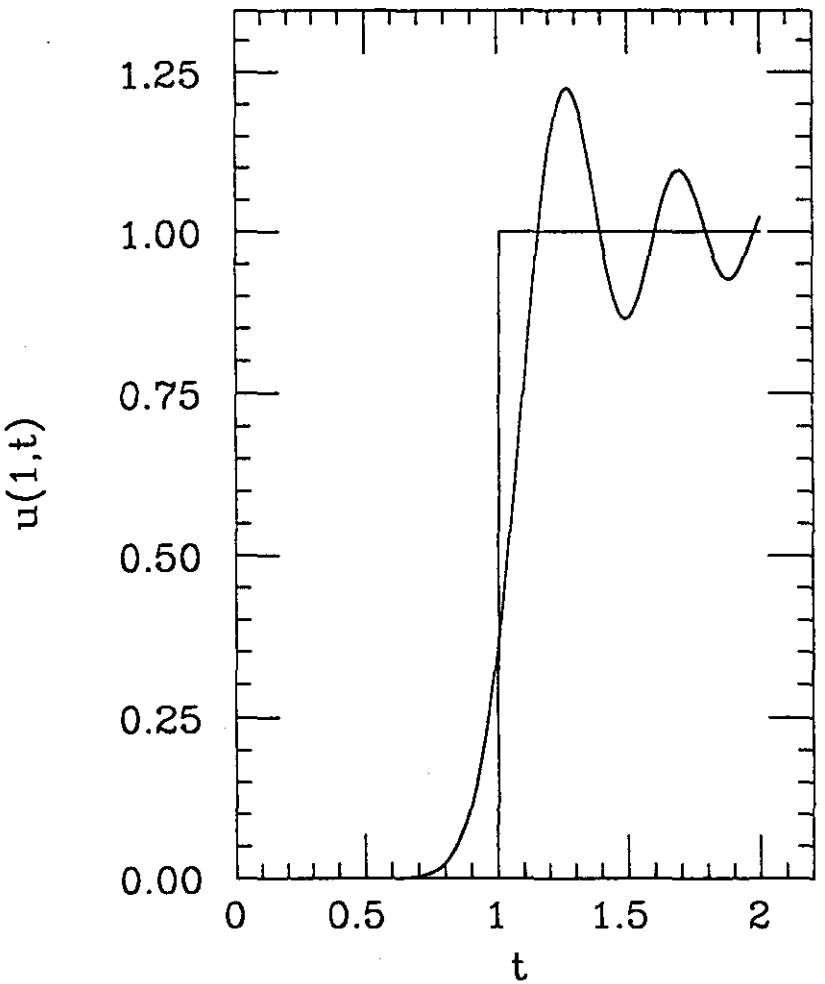
SSE = 1.881

Fig. 8



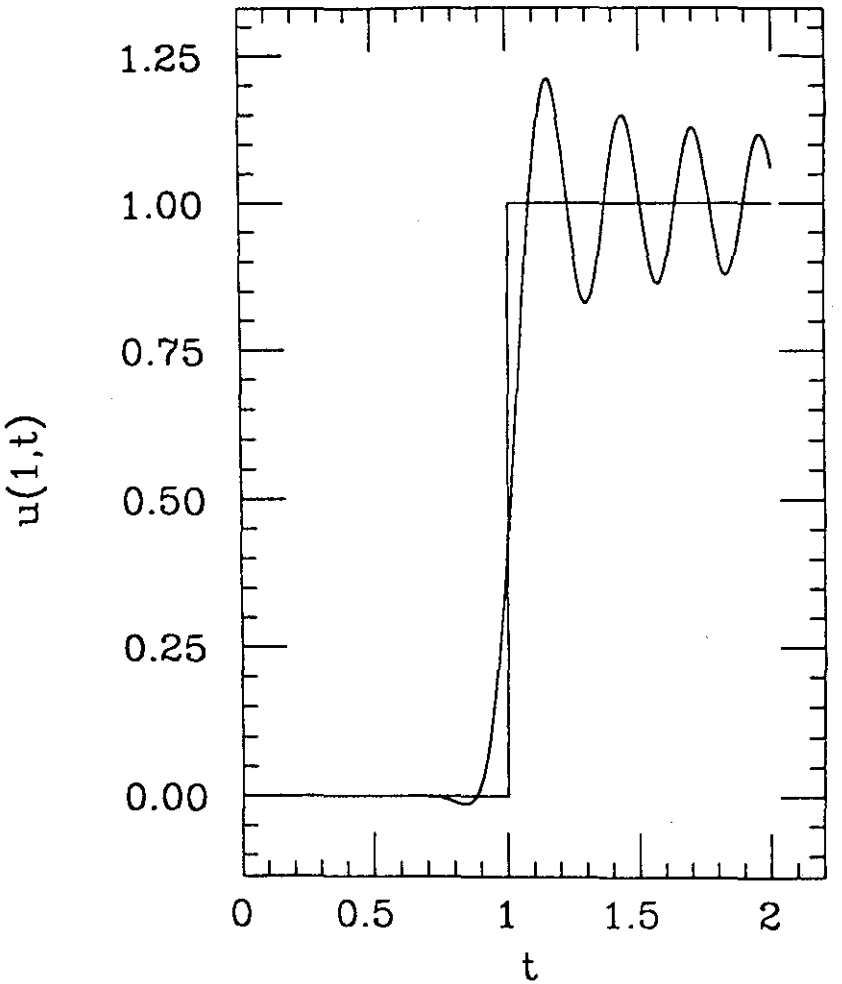
SSE = 2.133

Fig. 9



SSE = 3.654

Fig. 10



SSE = 2.456

Truncated Ramp Funtion

Fig. 11: Two Point Upwind

Fig. 12: Three Point Upwind

Fig. 13: Four Point Biased Upwind, 4PB

Fig. 14: Four Point Biased Upwind, 2PB

Fig. 15: Five Point Biased Upwind, 5PB

Fig. 16: Five Point Biased Upwind, 2PB

Fig. 17: Leonard, 4PB

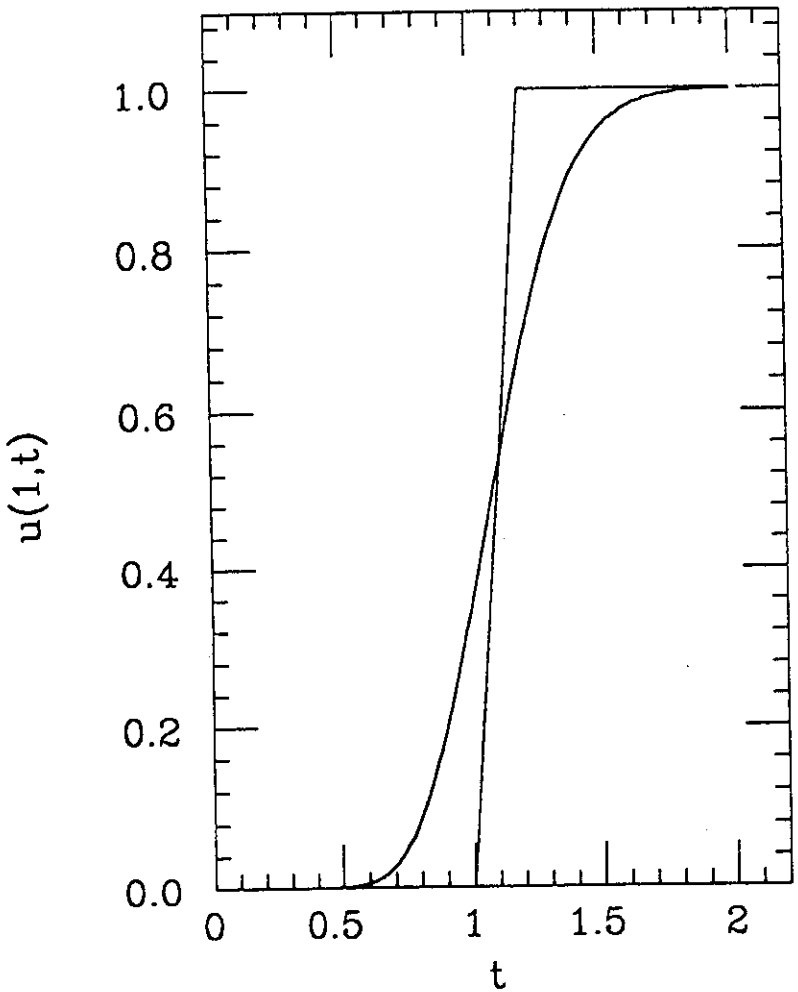
Fig. 18: Leonard, 2PB

Fig. 19: Three Point Centered

Fig. 20: Five Point Centered

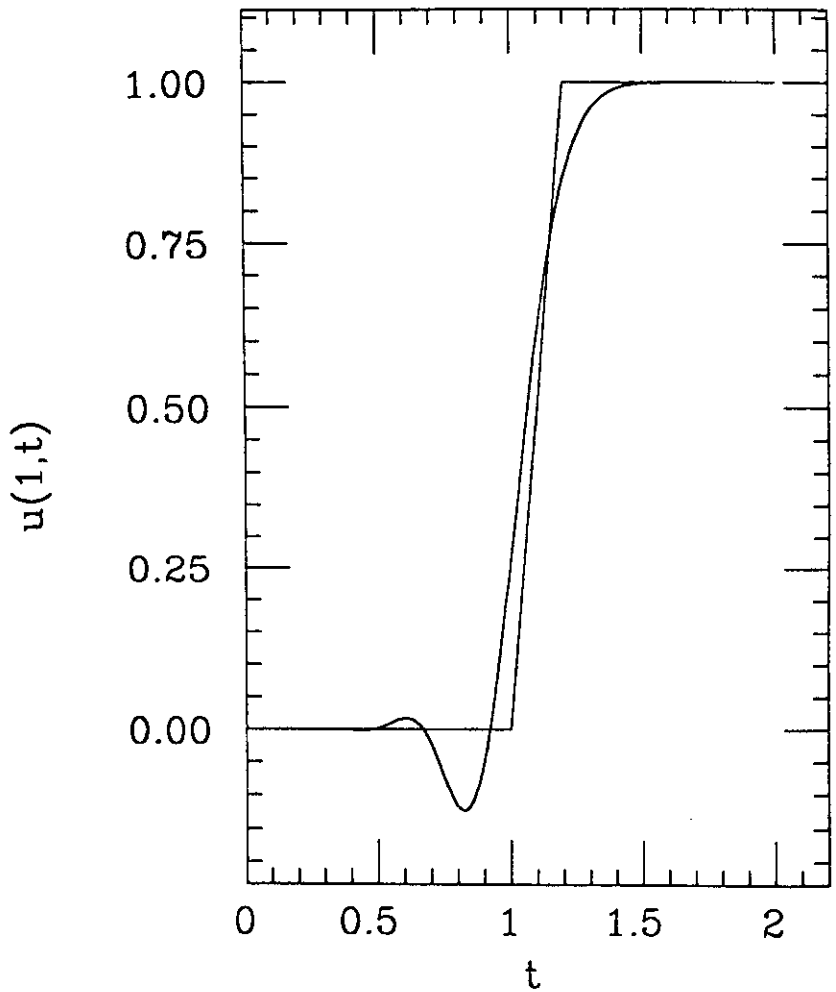
All solutions are for a 21-point grid

Fig. 11



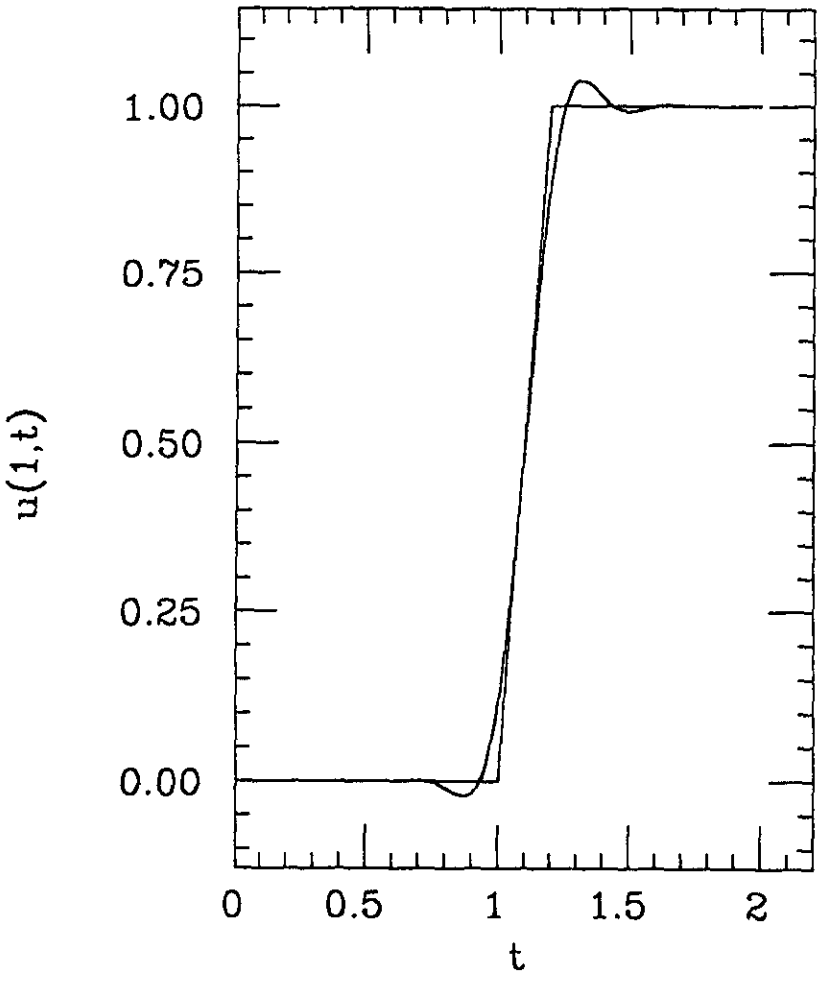
SSE = 2.636

Fig. 12



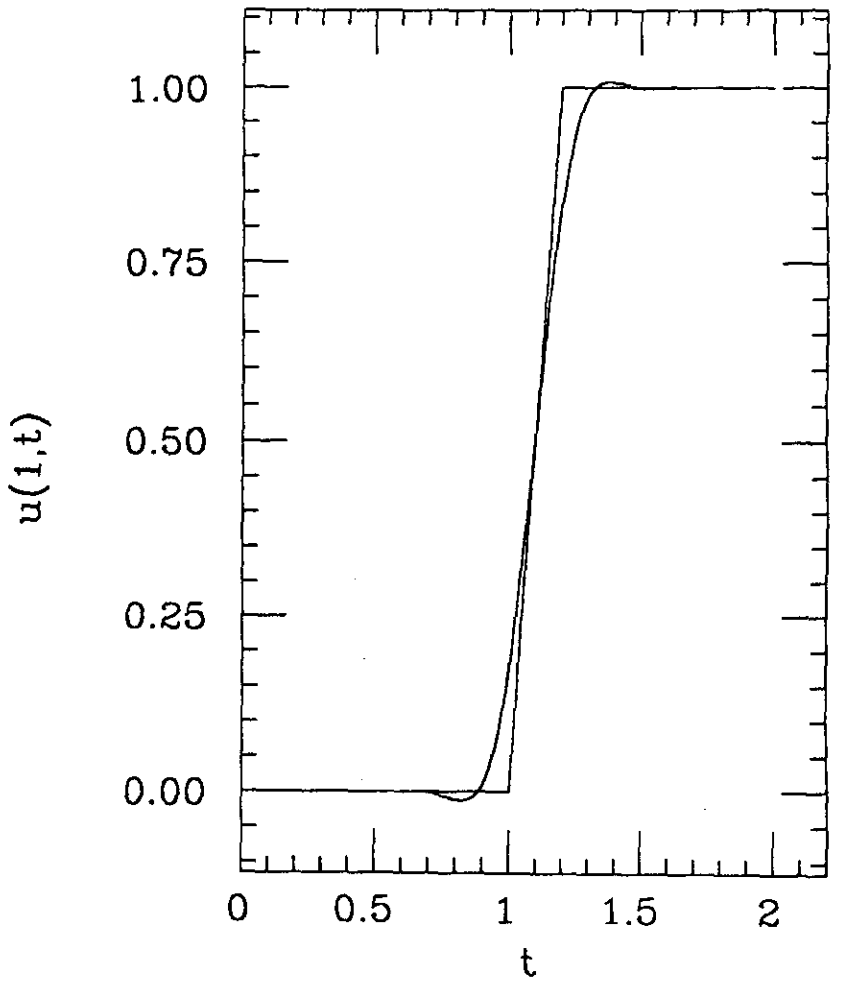
SSE = 0.892

Fig. 13



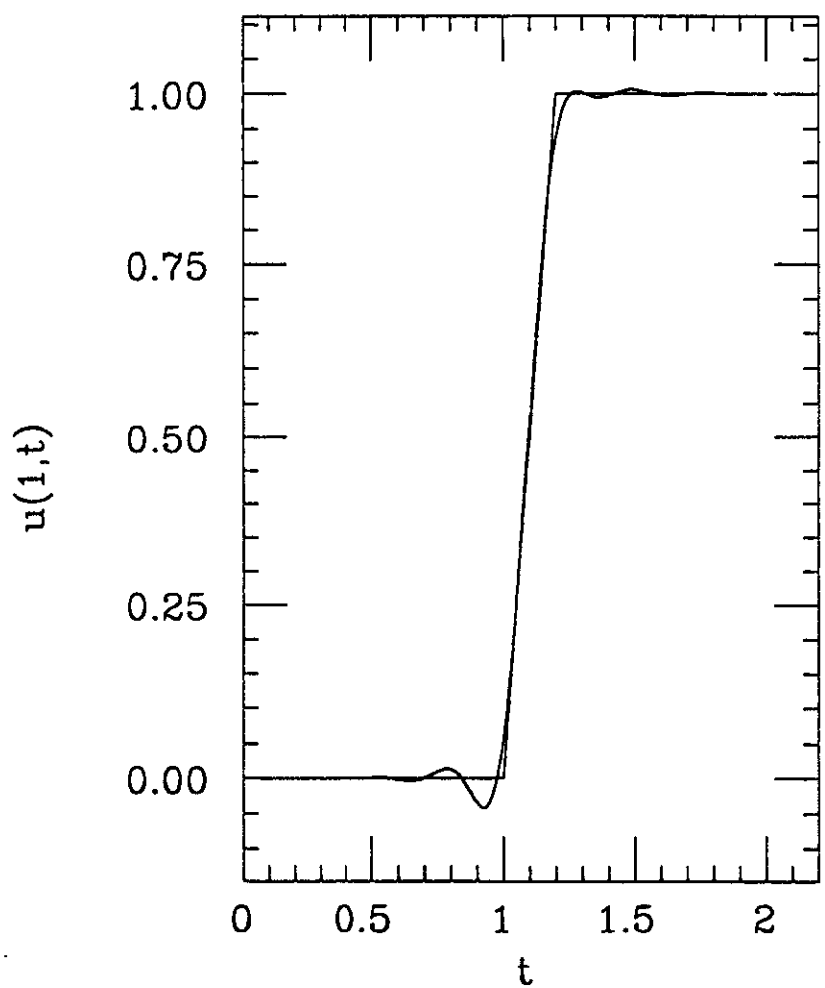
SSE = 0.121

Fig. 14



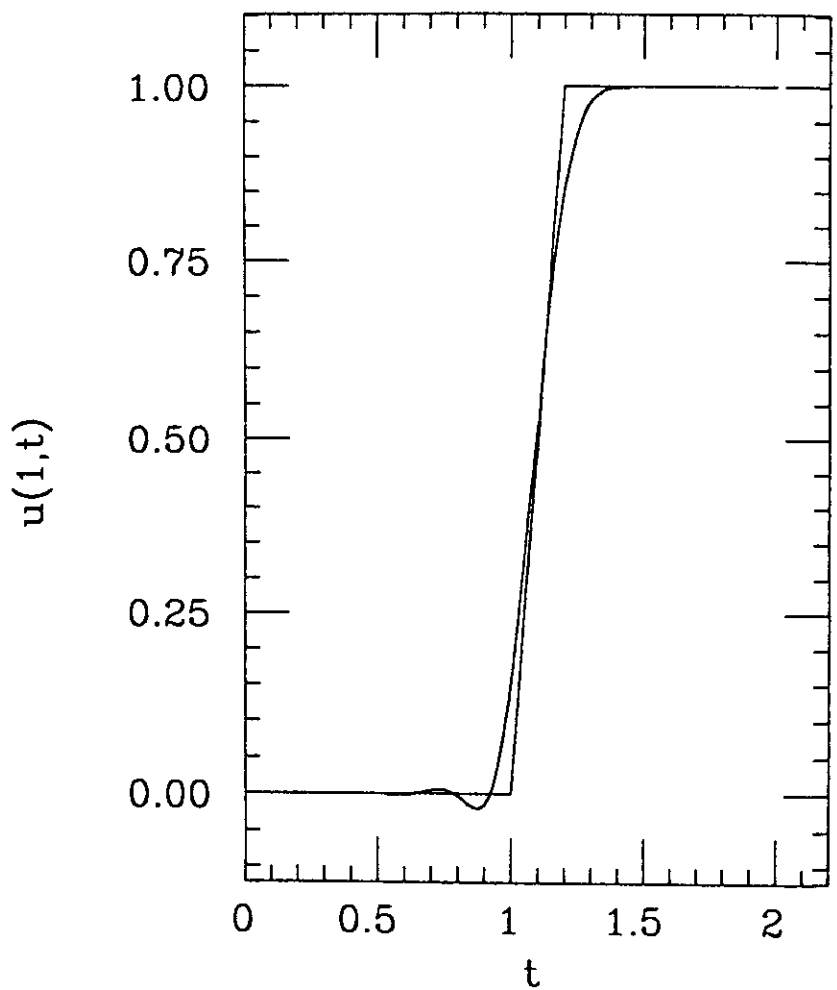
SSE = 0.326

Fig. 15



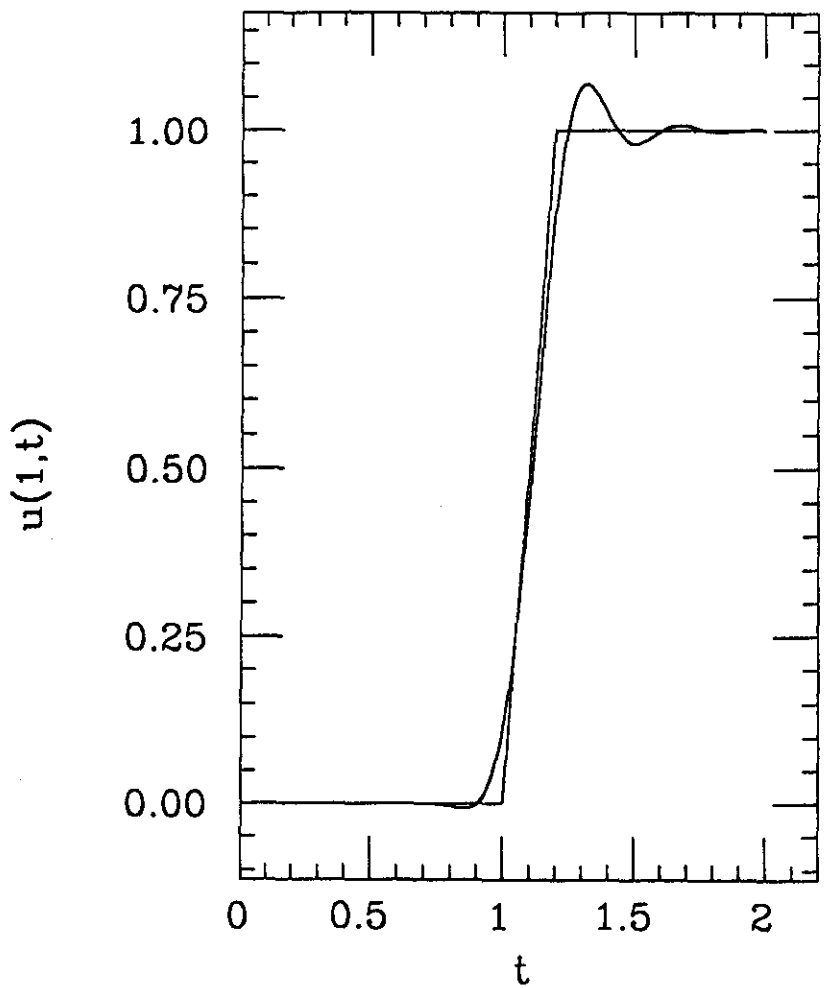
SSE = 0.037

Fig. 16



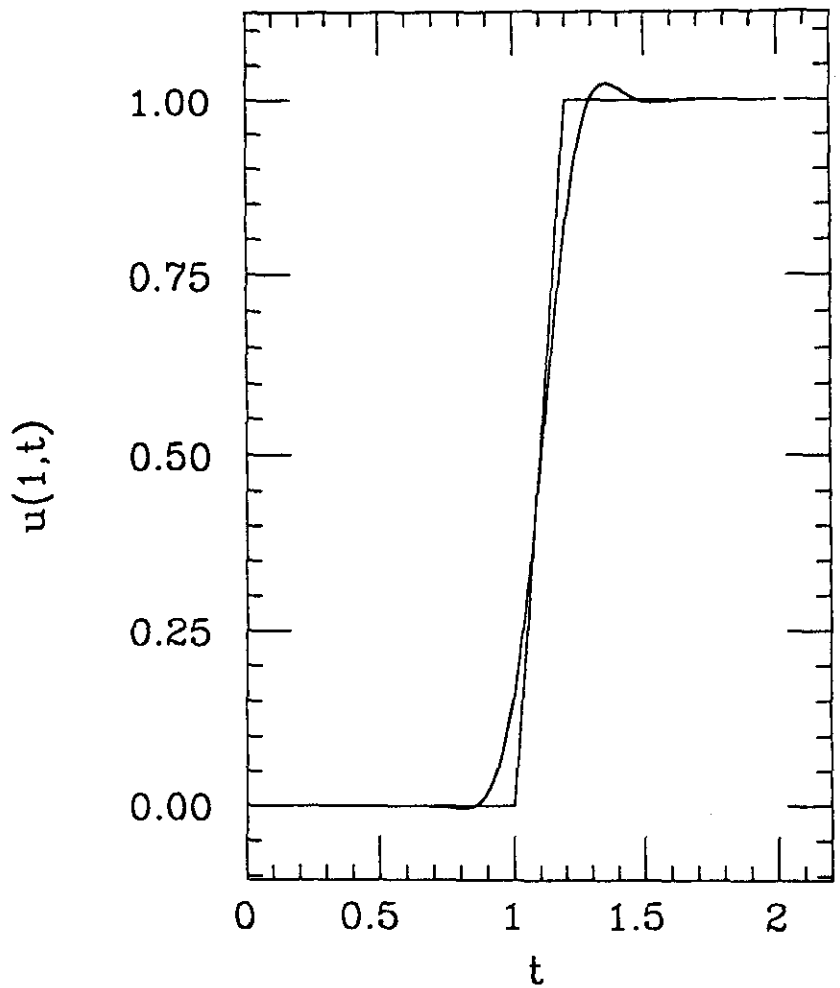
SSE = 0.242

Fig. 17



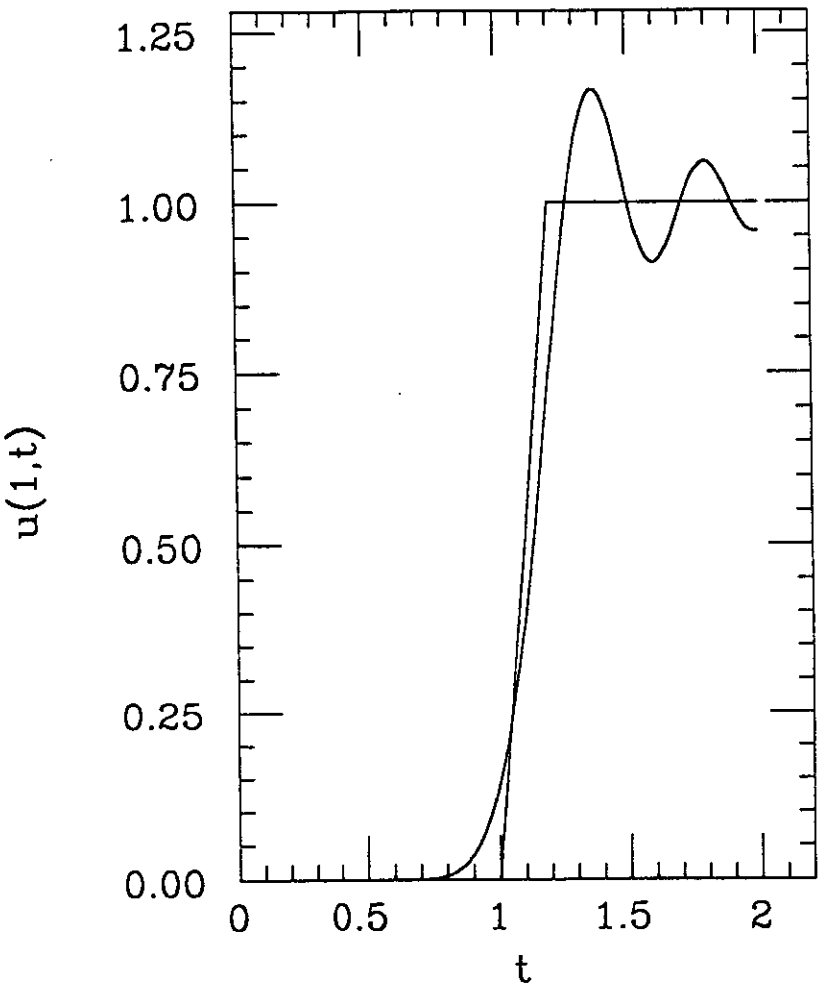
SSE = 0.192

Fig. 18



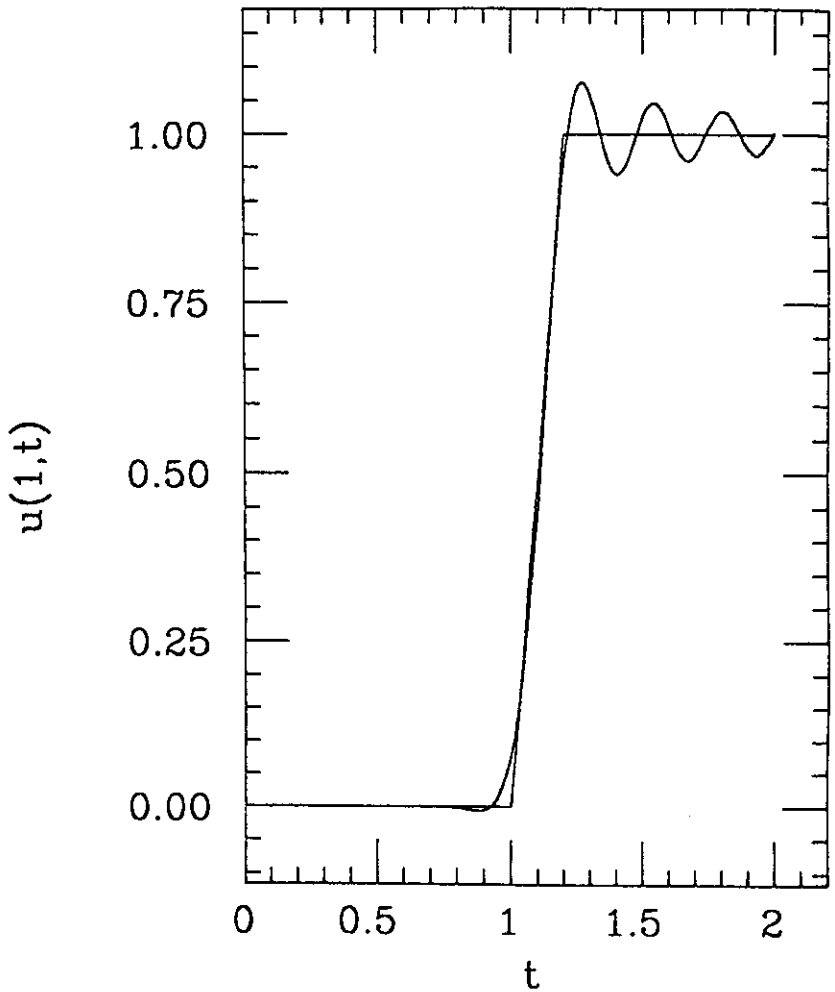
SSE = 0.323

Fig. 19



SSE = 1.069

Fig. 20



SSE = 0.141

Cosine Step Funtion

Fig. 21: Two Point Upwind

Fig. 22: Three Point Upwind

Fig. 23: Four Point Biased Upwind, 4PB

Fig. 24: Four Point Biased Upwind, 2PB

Fig. 25: Five Point Biased Upwind, 5PB

Fig. 26: Five Point Biased Upwind, 2PB

Fig. 27: Leonard, 4PB

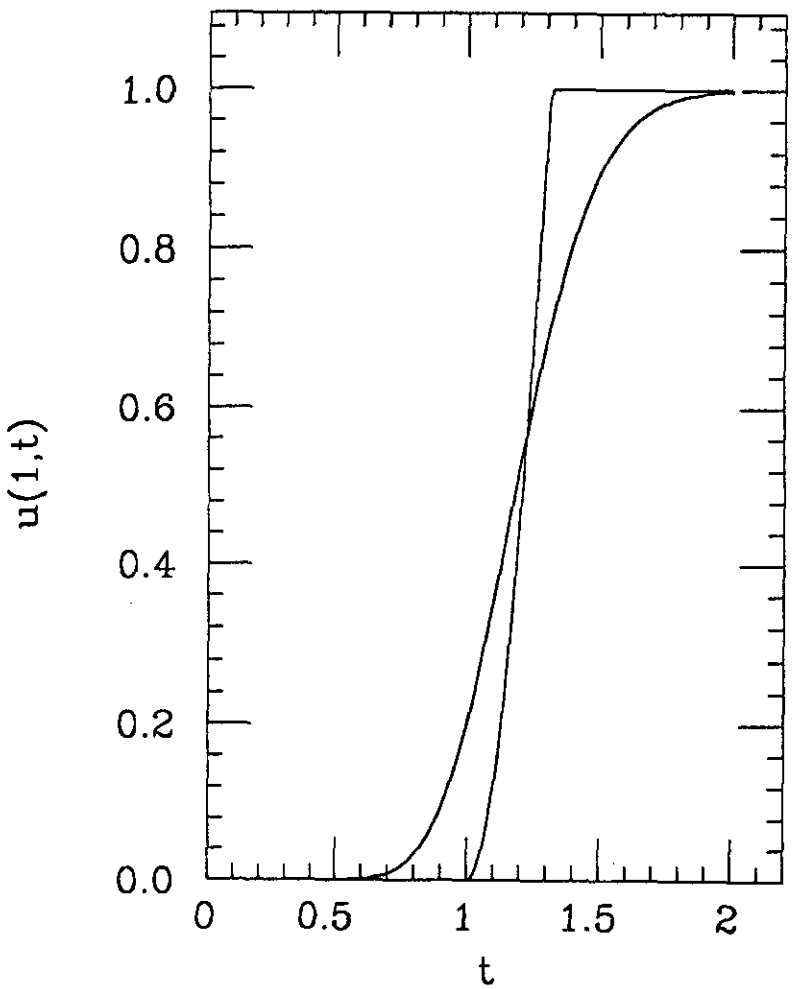
Fig. 28: Leonard, 2PB

Fig. 29: Three Point Centered

Fig. 30: Five Point Centered

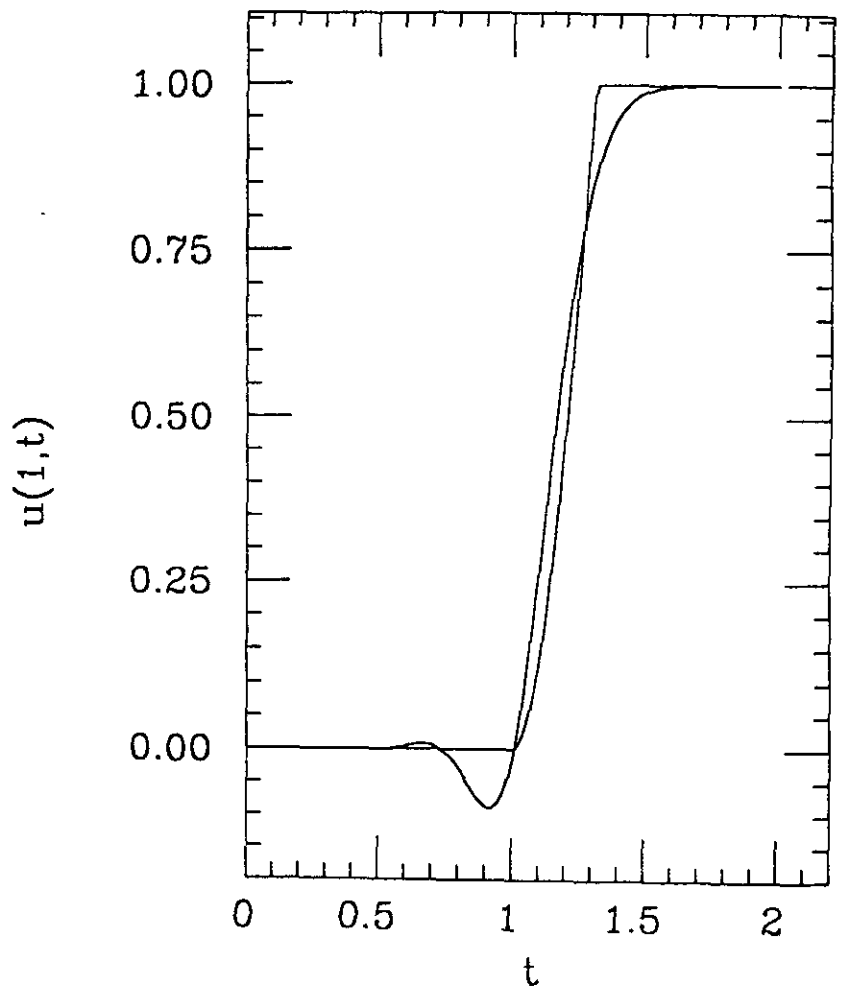
All solutions are for a 21-point grid

Fig. 21



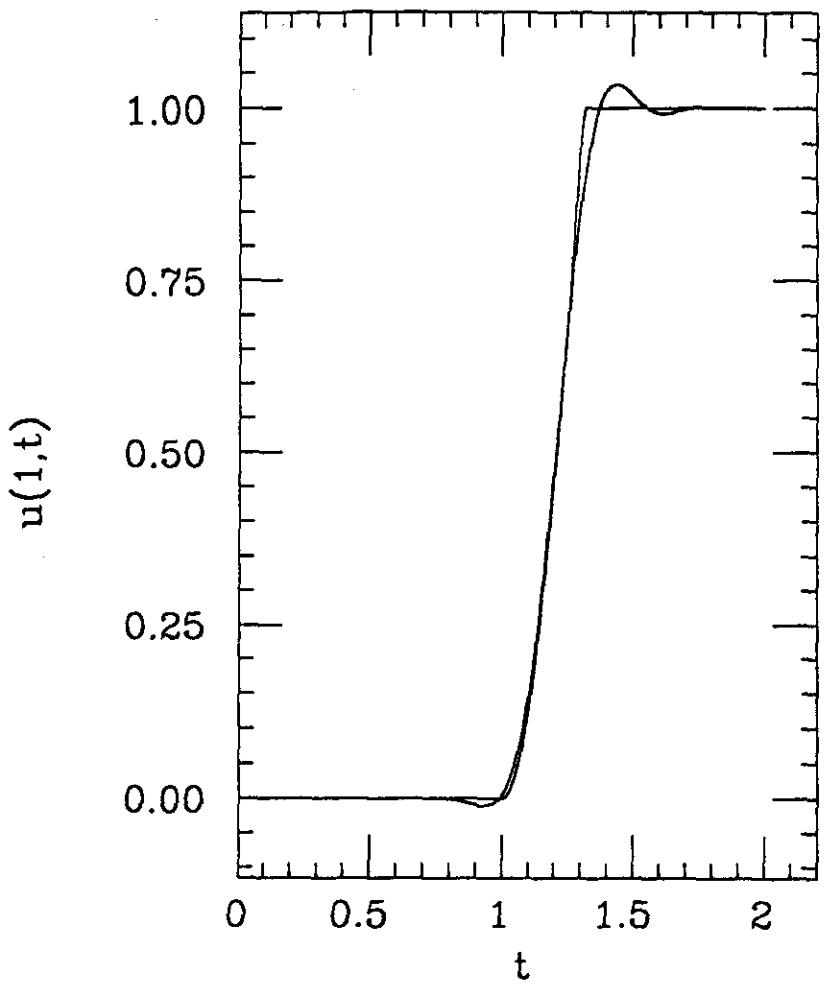
SSE = 2.155

Fig. 22



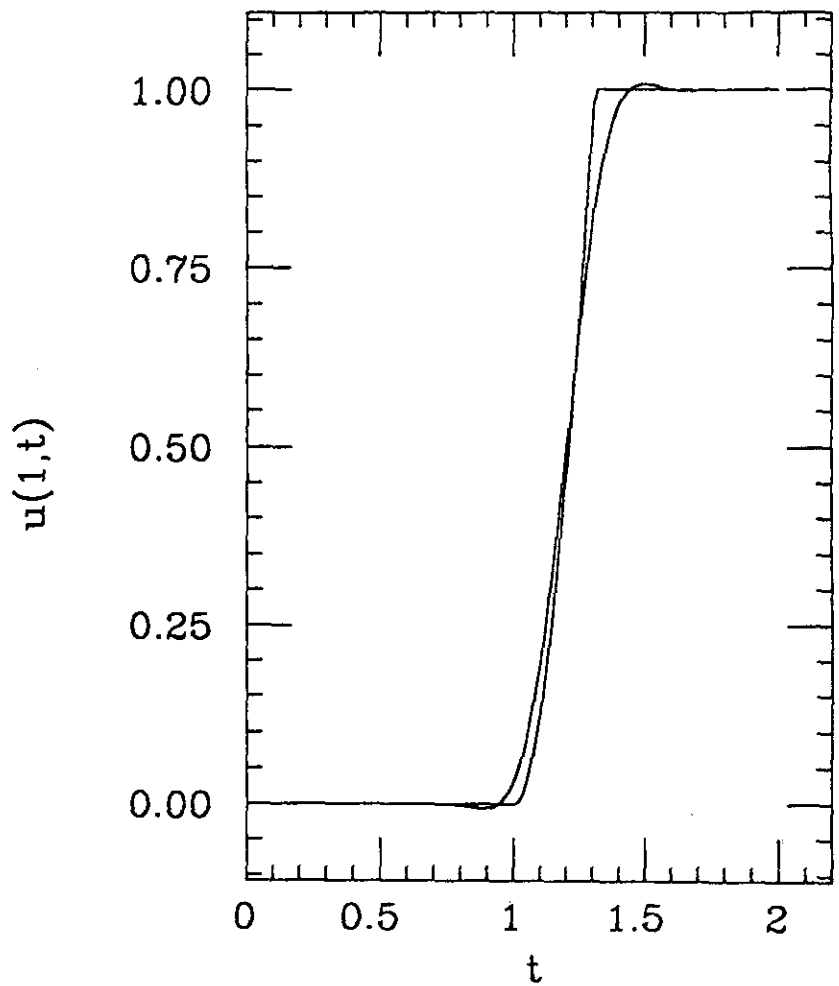
SSE = 0.564

Fig. 23



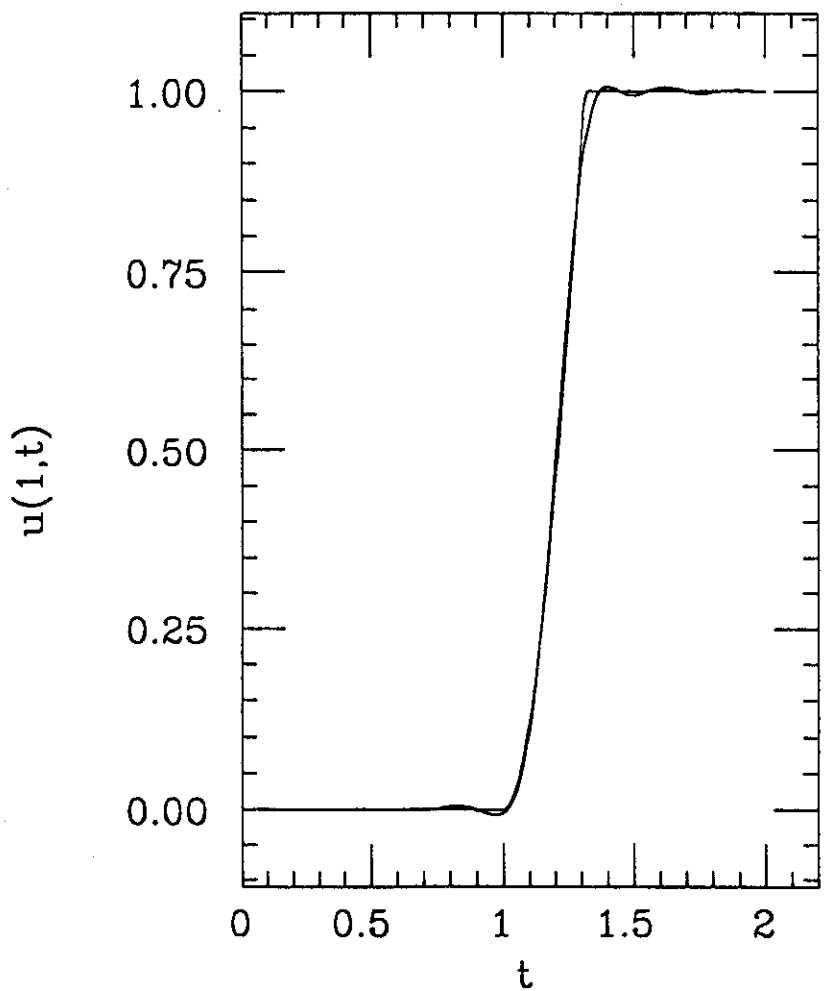
SSE = 0.063

Fig. 24



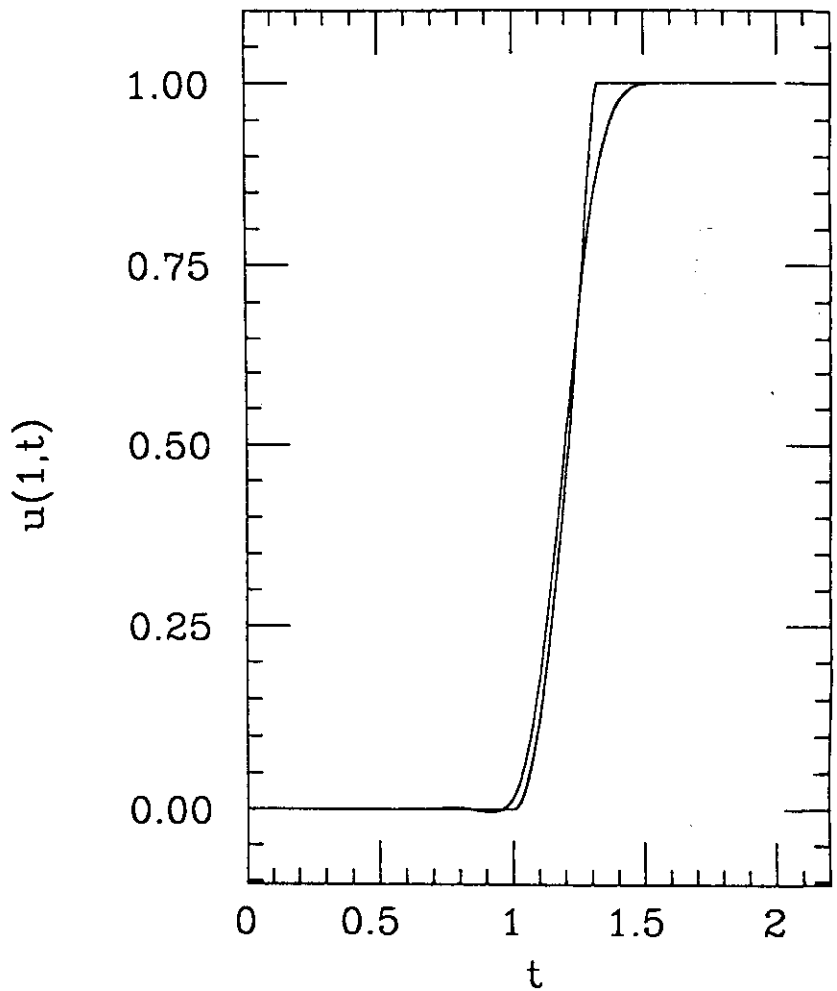
SSE = 0.201

Fig. 25



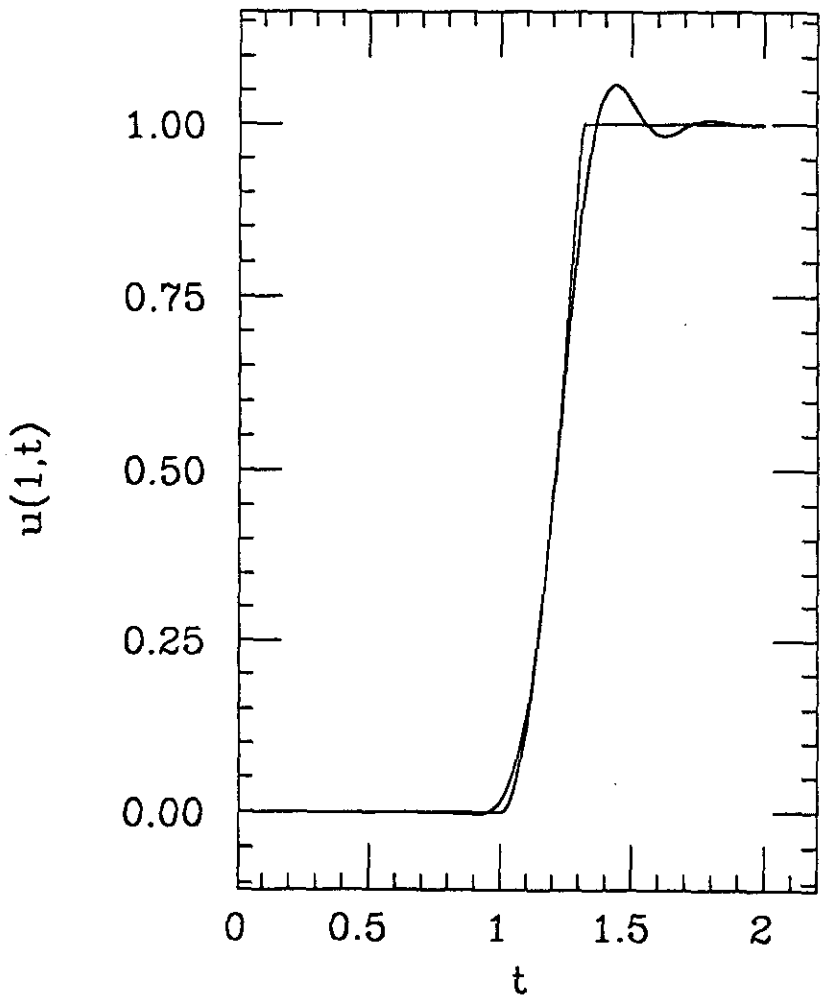
SSE = 0.024

Fig. 26



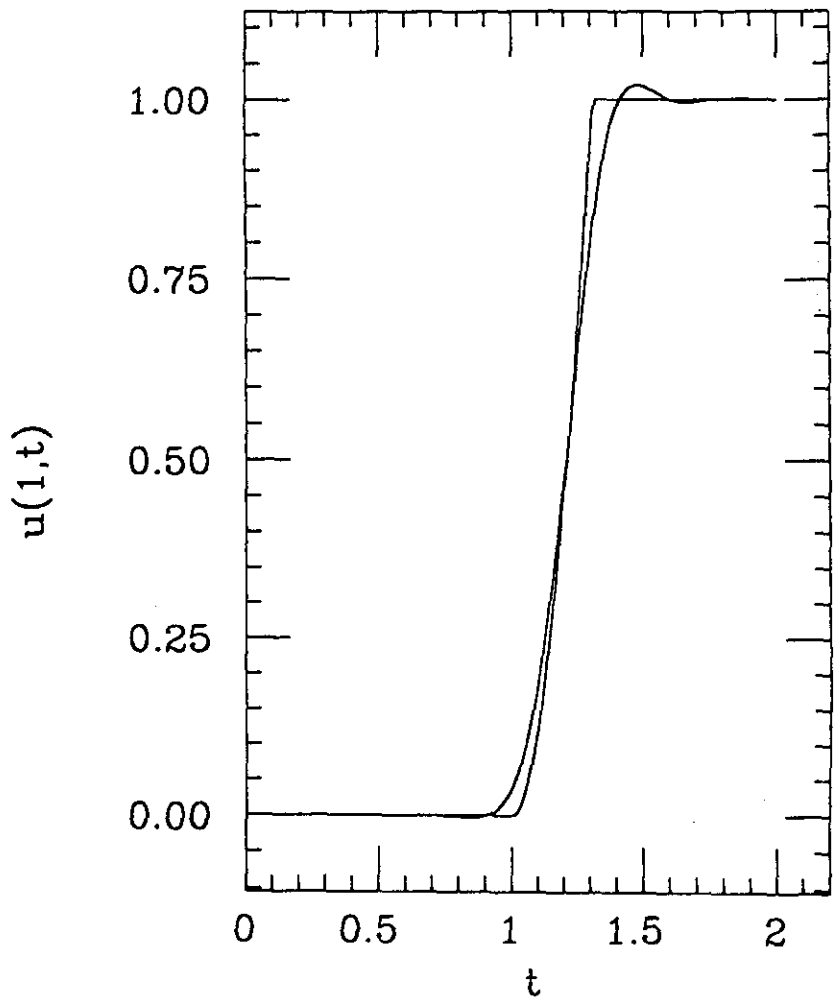
SSE = 0.148

Fig. 27



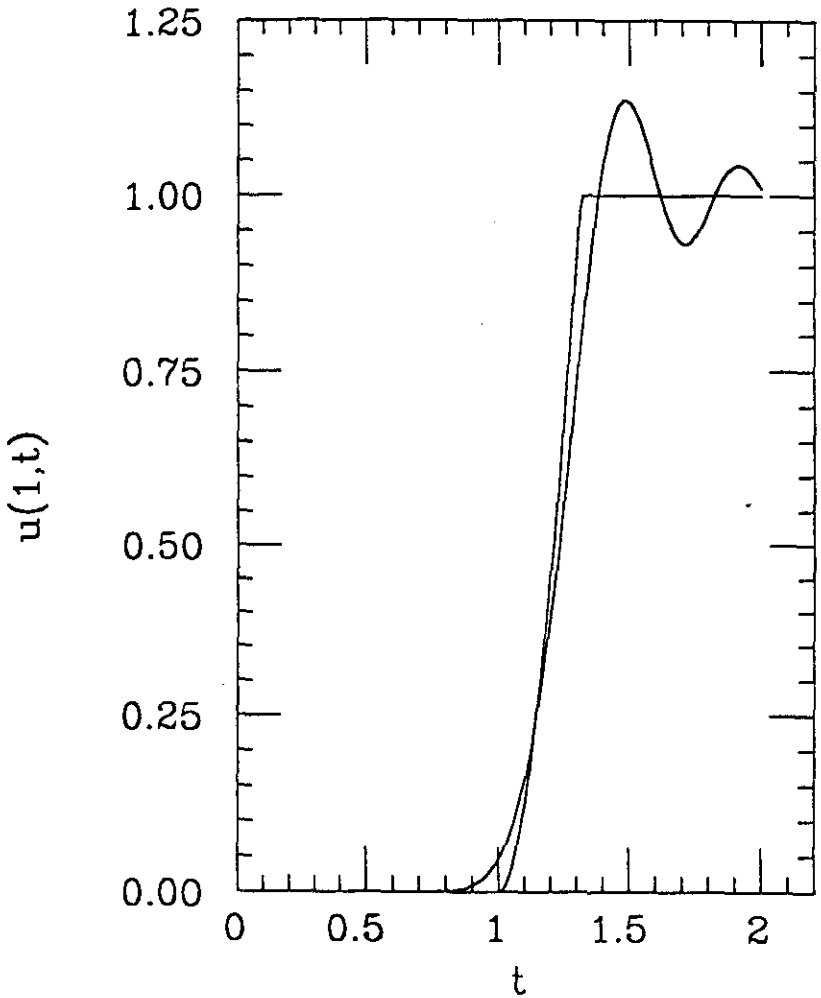
SSE = 0.101

Fig. 28



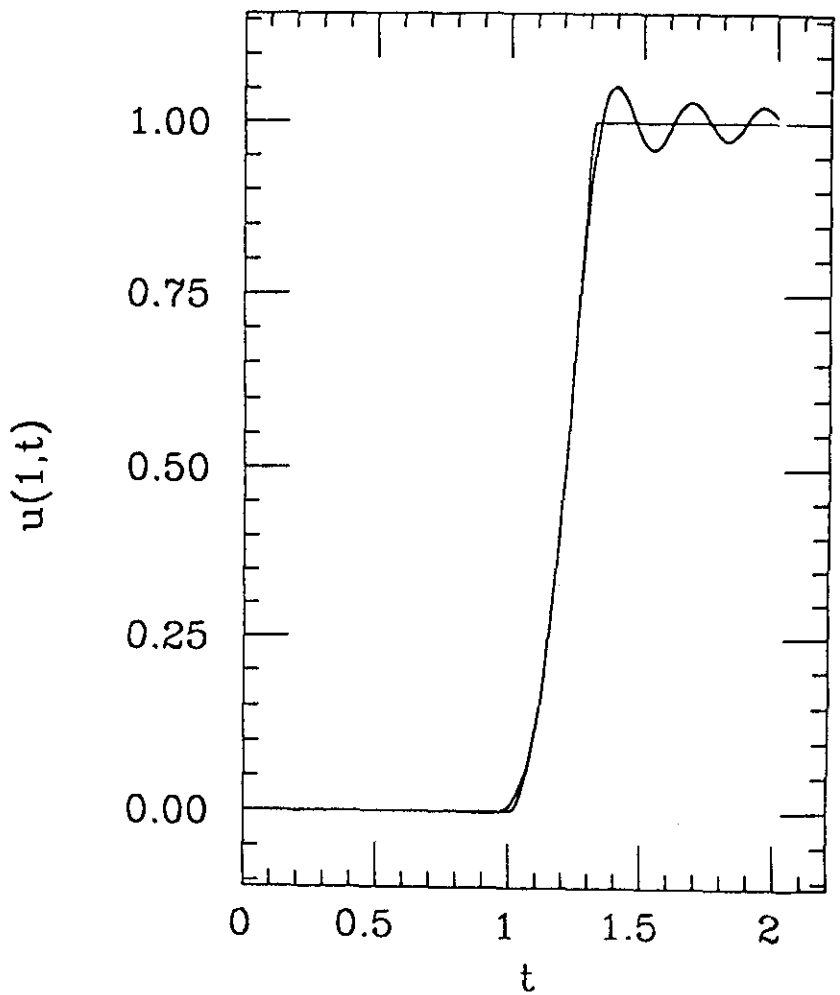
SSE = 0.196

Fig. 29



SSE = 0.637

Fig. 30



SSE = 0.058

Smooth Cosine Step Funtion

Fig. 31: Two Point Upwind

Fig. 32: Three Point Upwind

Fig. 33: Four Point Biased Upwind, 4PB

Fig. 34: Four Point Biased Upwind, 2PB

Fig. 35: Five Point Biased Upwind, 5PB

Fig. 36: Five Point Biased Upwind, 2PB

Fig. 37: Leonard, 4PB

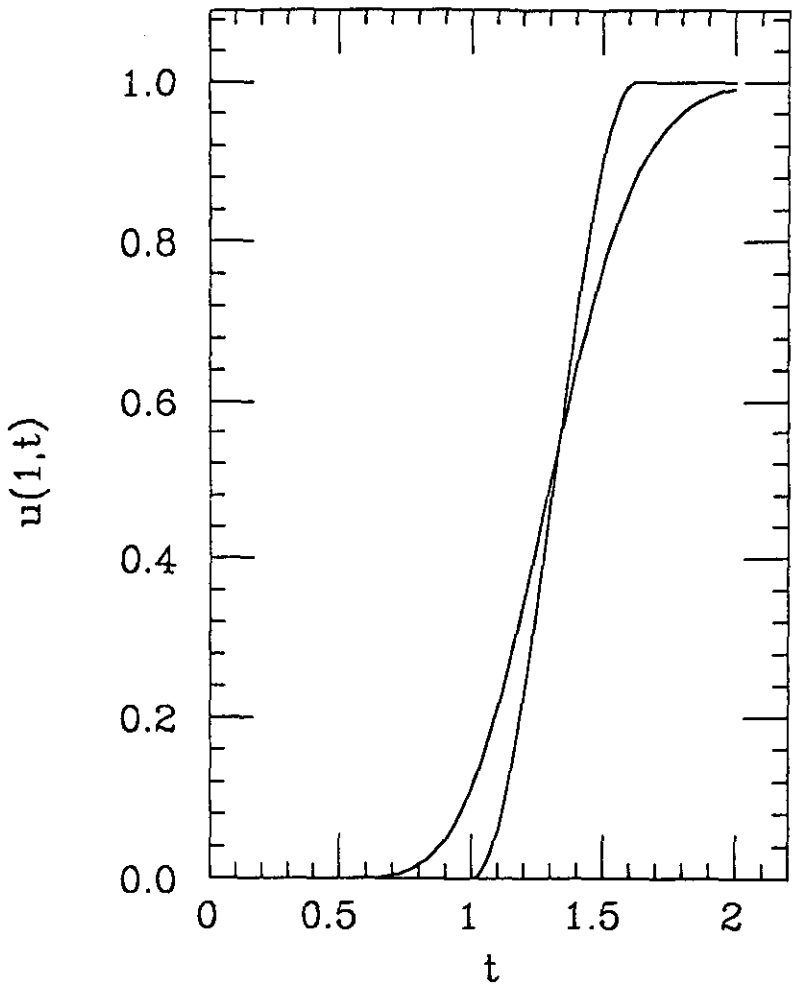
Fig. 38: Leonard, 2PB

Fig. 39: Three Point Centered

Fig. 40: Five Point Centered

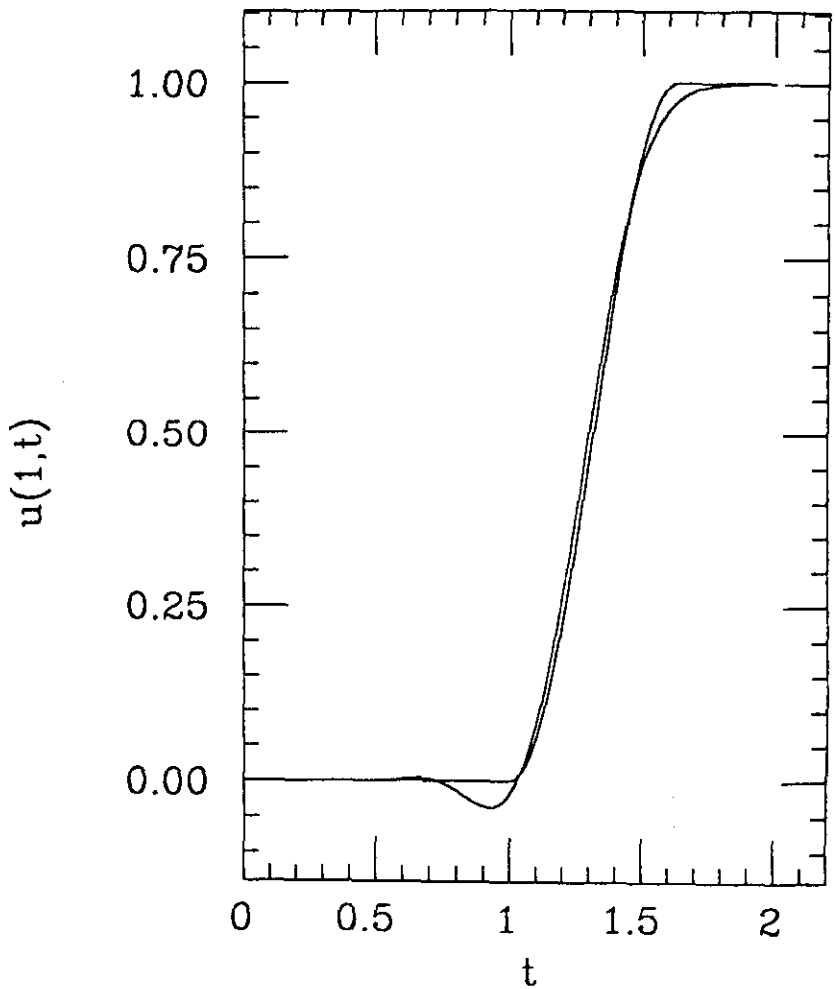
All solutions are for a 21-point grid

Fig. 31



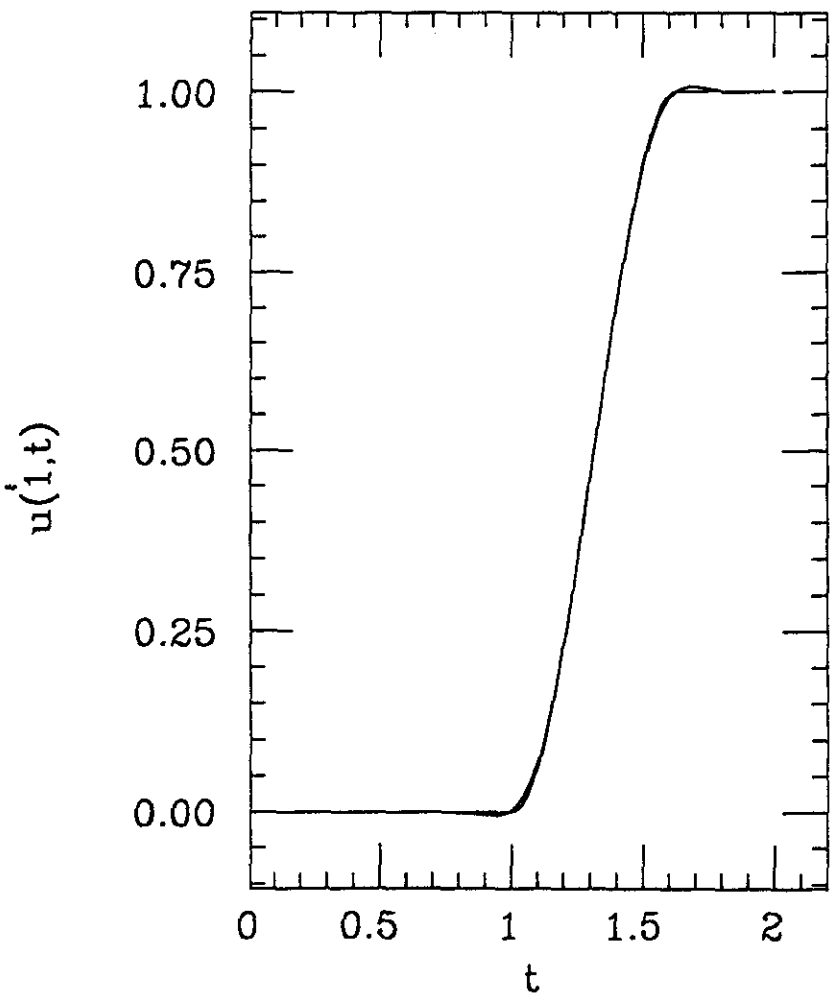
SSE = 0.99303

Fig. 32



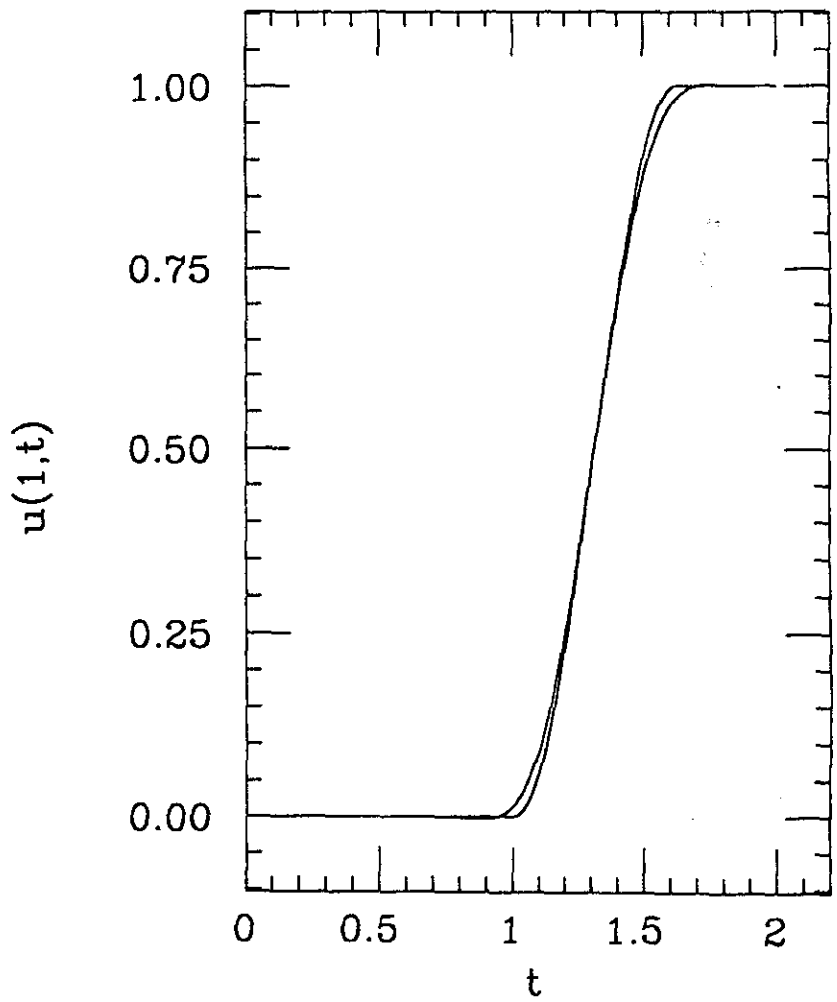
SSE = 0.07744

Fig. 33



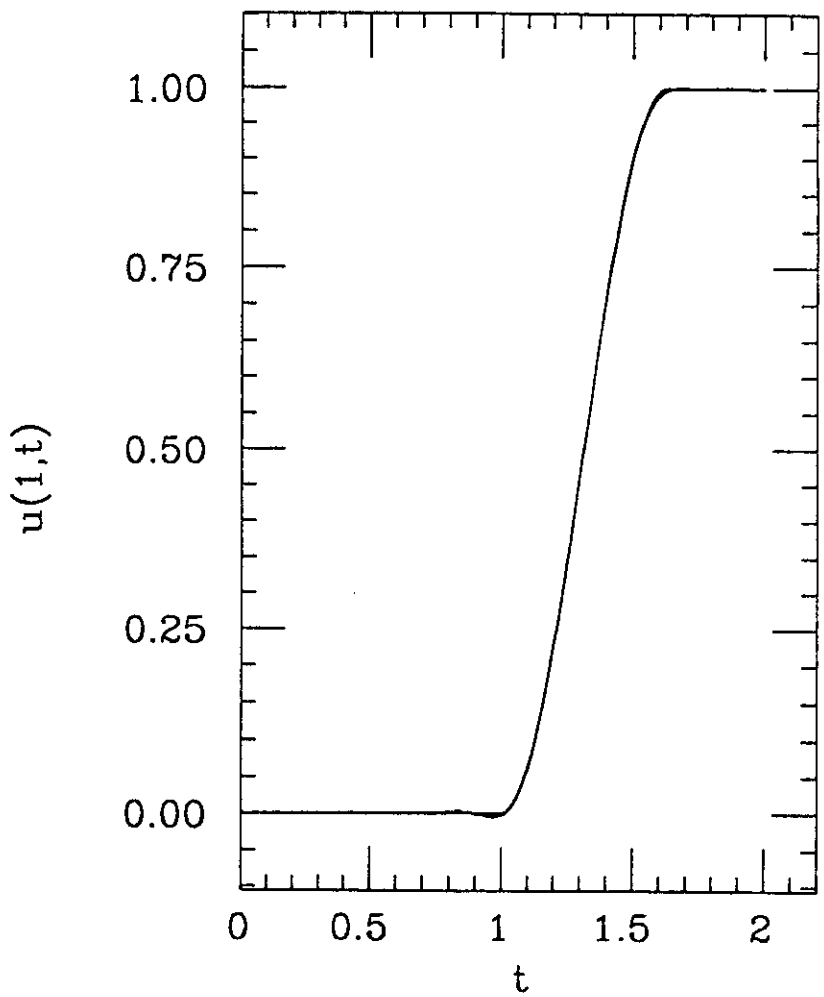
SSE = 0.00224

Fig. 34



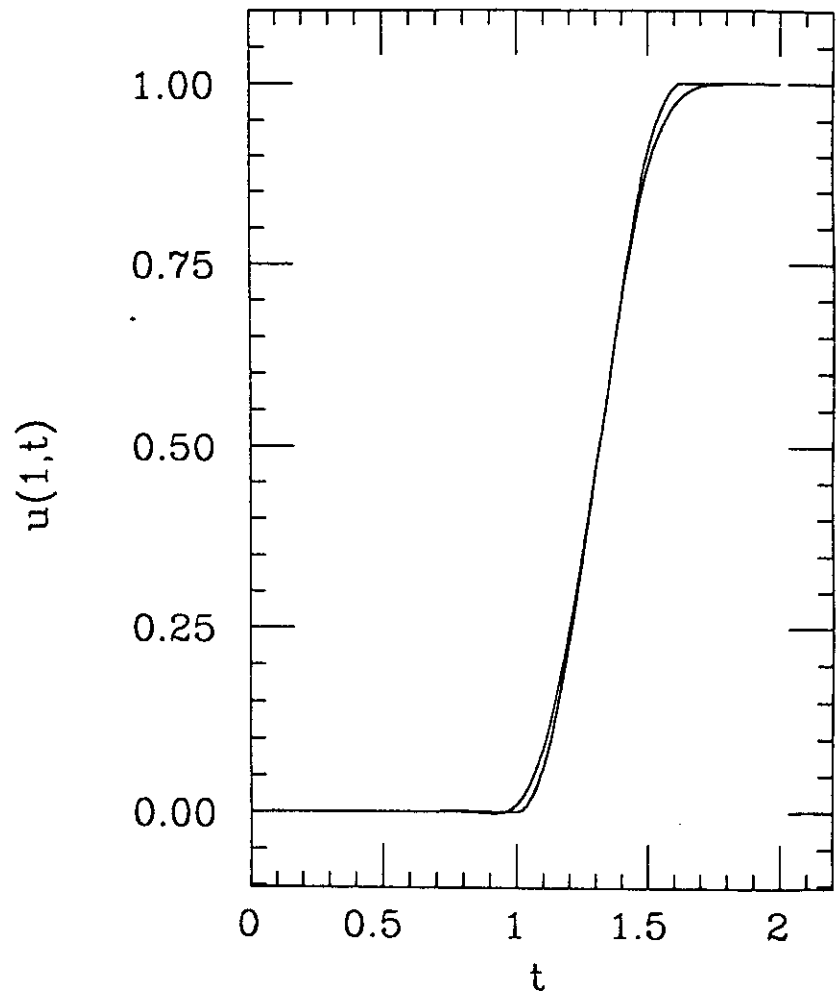
SSE = 0.03109

Fig. 35



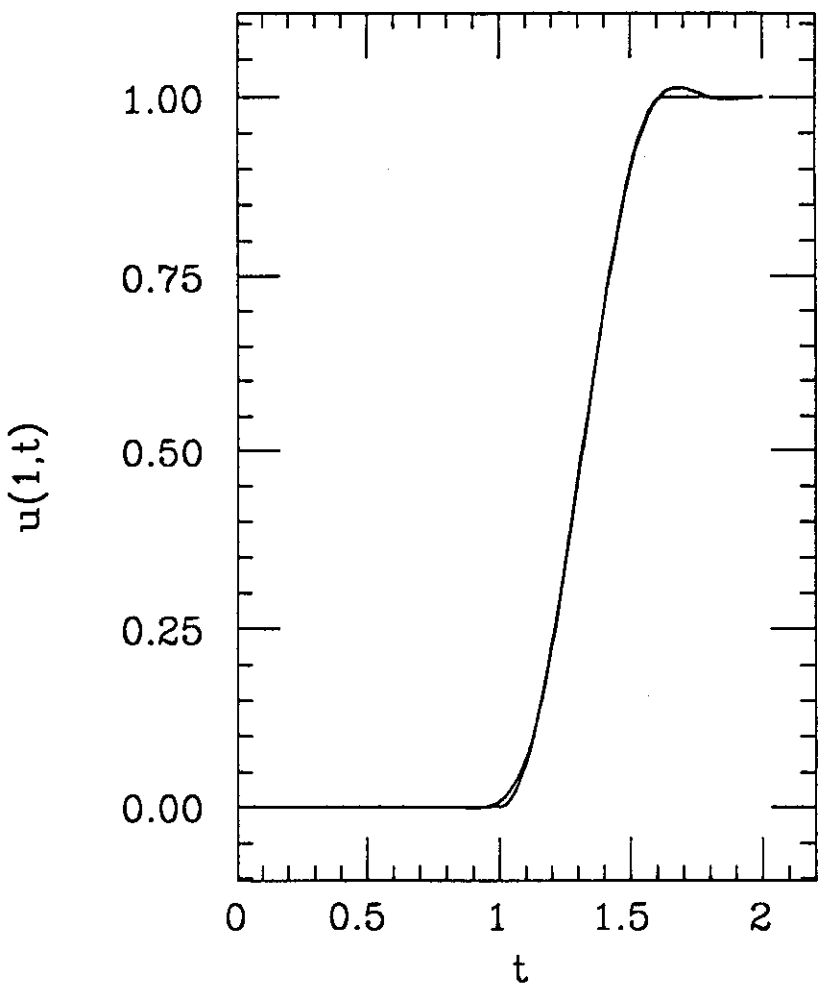
SSE = 0.00049

Fig. 36



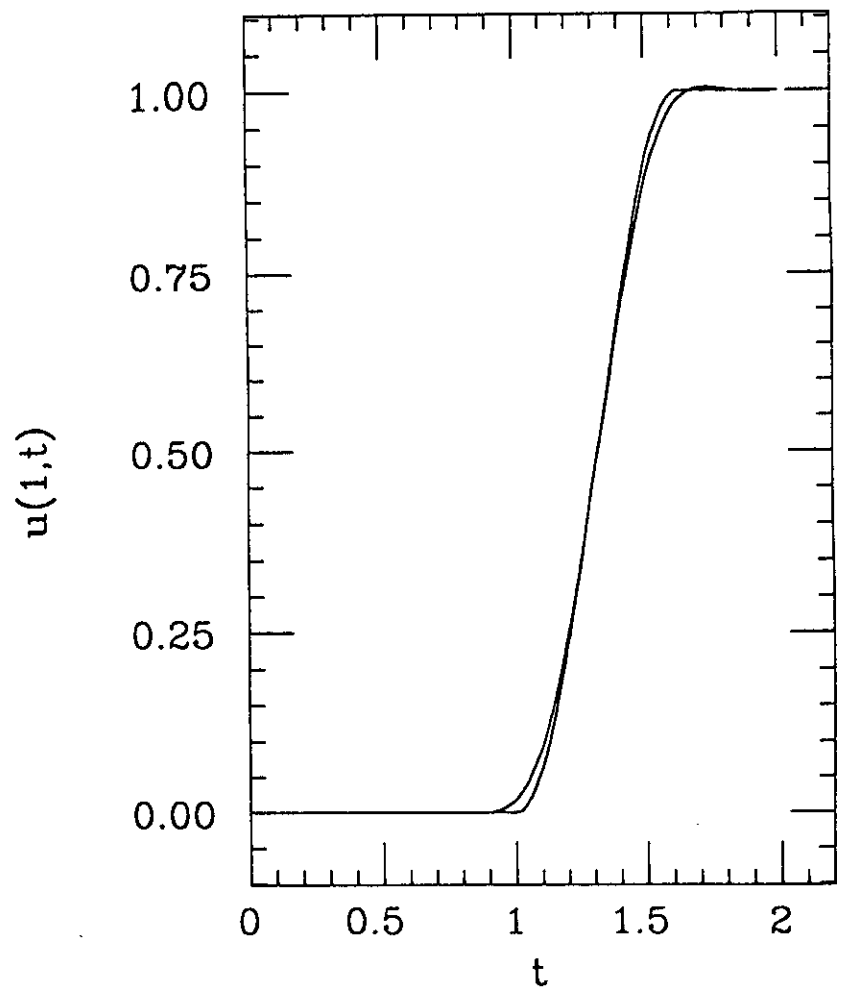
SSE = 0.02384

Fig. 37



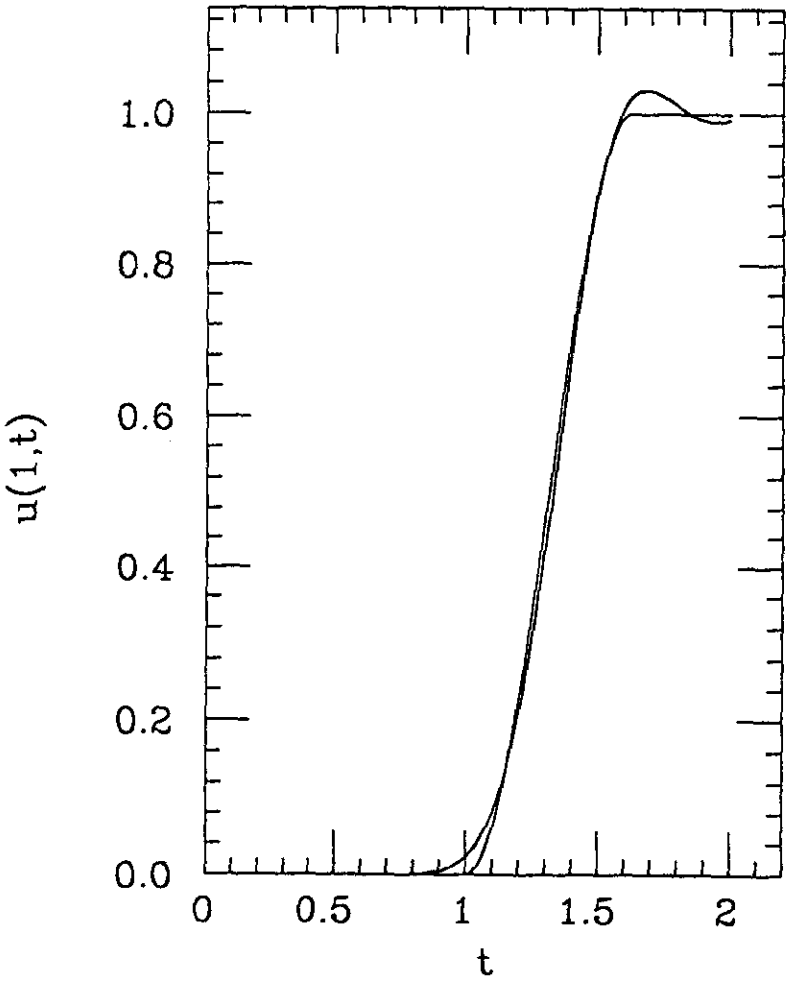
SSE = 0.00487

Fig. 38



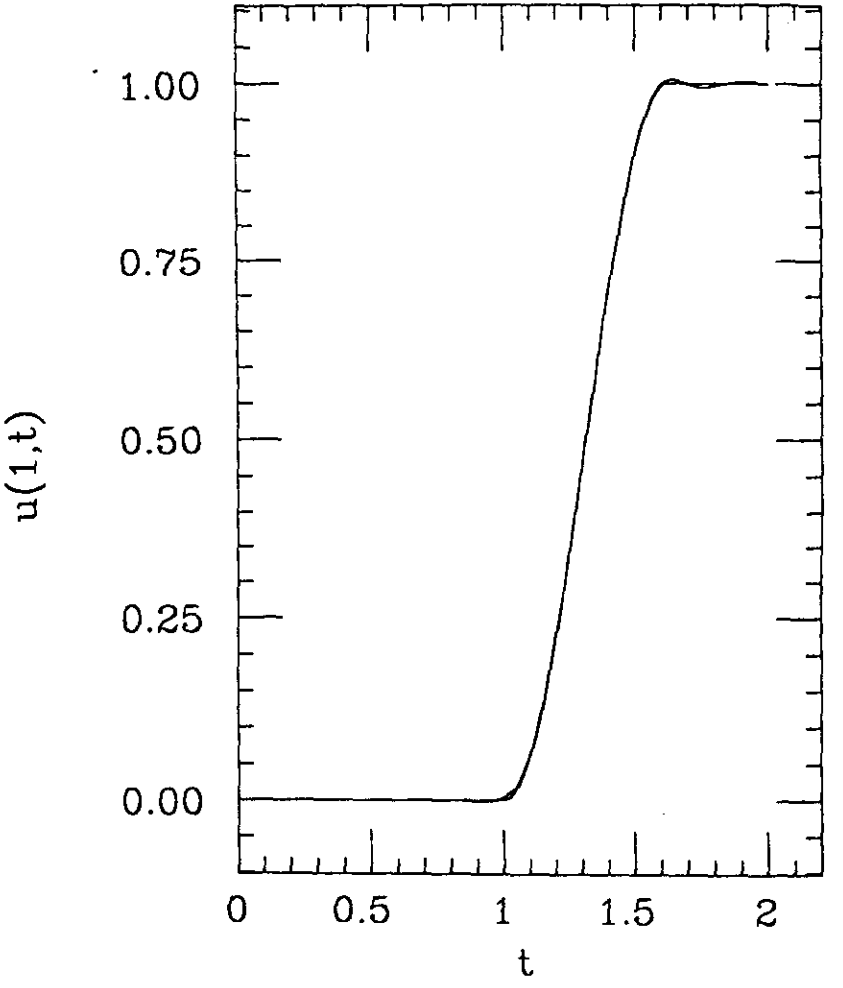
SSE = 0.02851

Fig. 39



SSE = 0.04865

Fig. 40



SSE = 0.00118

Cosine Pulse Function

Fig. 41: Two Point Upwind

Fig. 42: Three Point Upwind

Fig. 43: Four Point Biased Upwind, 4PB

Fig. 44: Four Point Biased Upwind, 2PB

Fig. 45: Five Point Biased Upwind, 5PB

Fig. 46: Five Point Biased Upwind, 2PB

Fig. 47: Leonard, 4PB

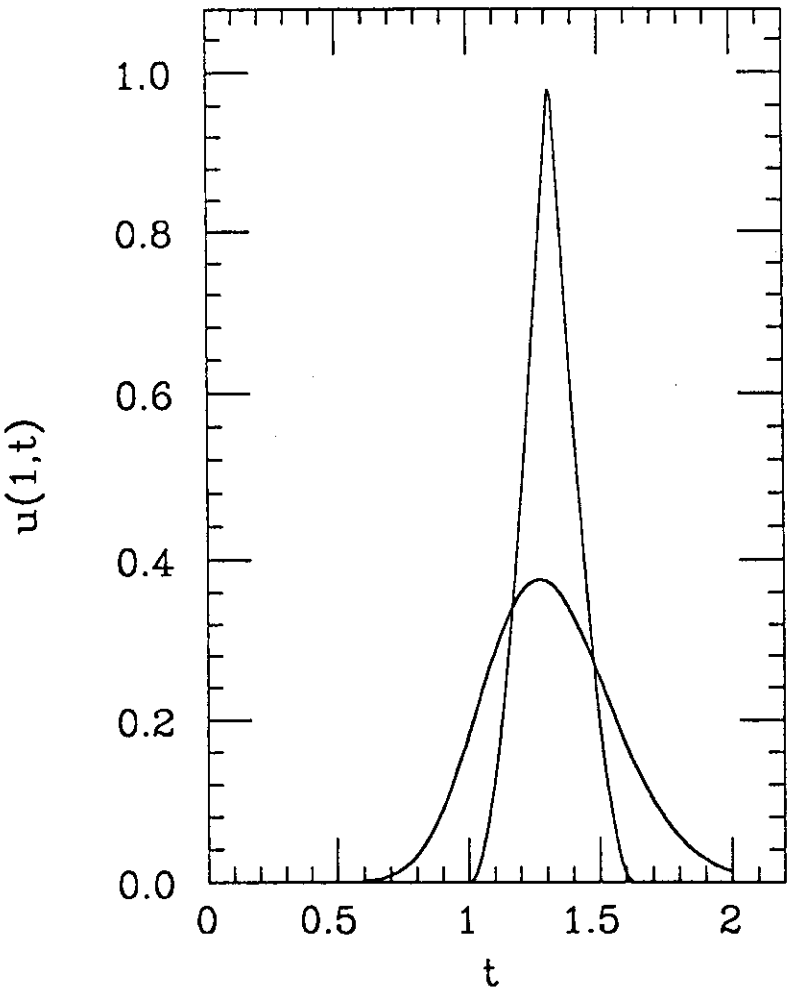
Fig. 48: Leonard, 2PB

Fig. 49: Three Point Centered

Fig. 50: Five Point Centered

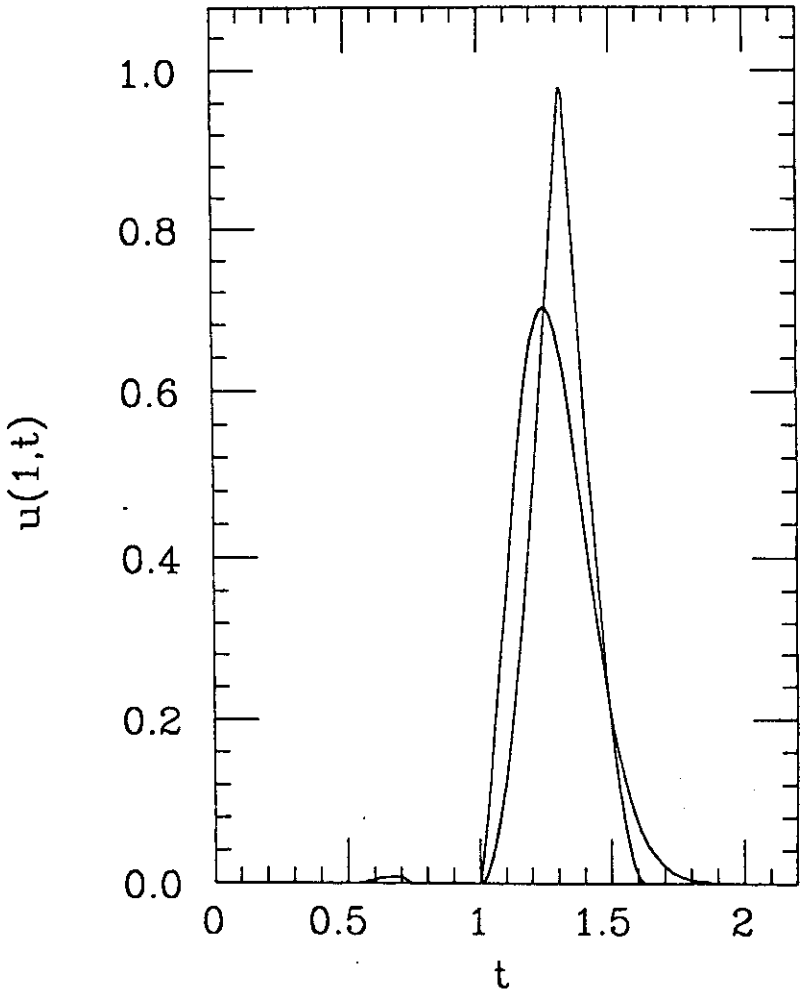
All solutions are for a 21-point grid

Fig. 41



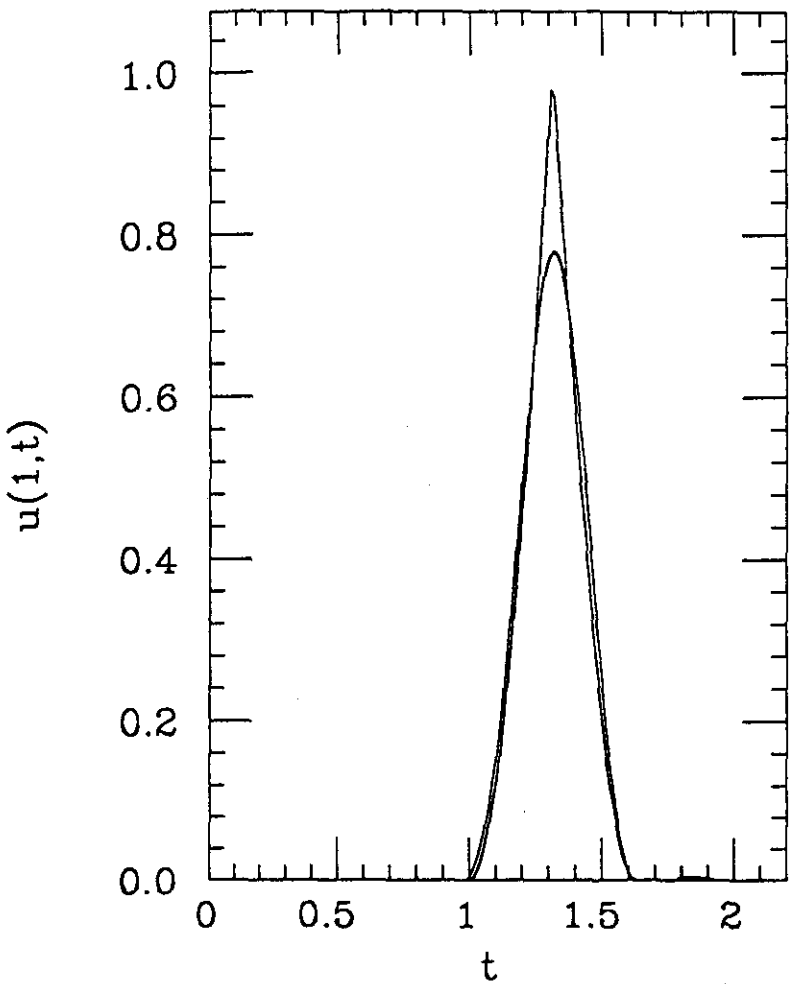
SSE = 4.645

Fig. 42



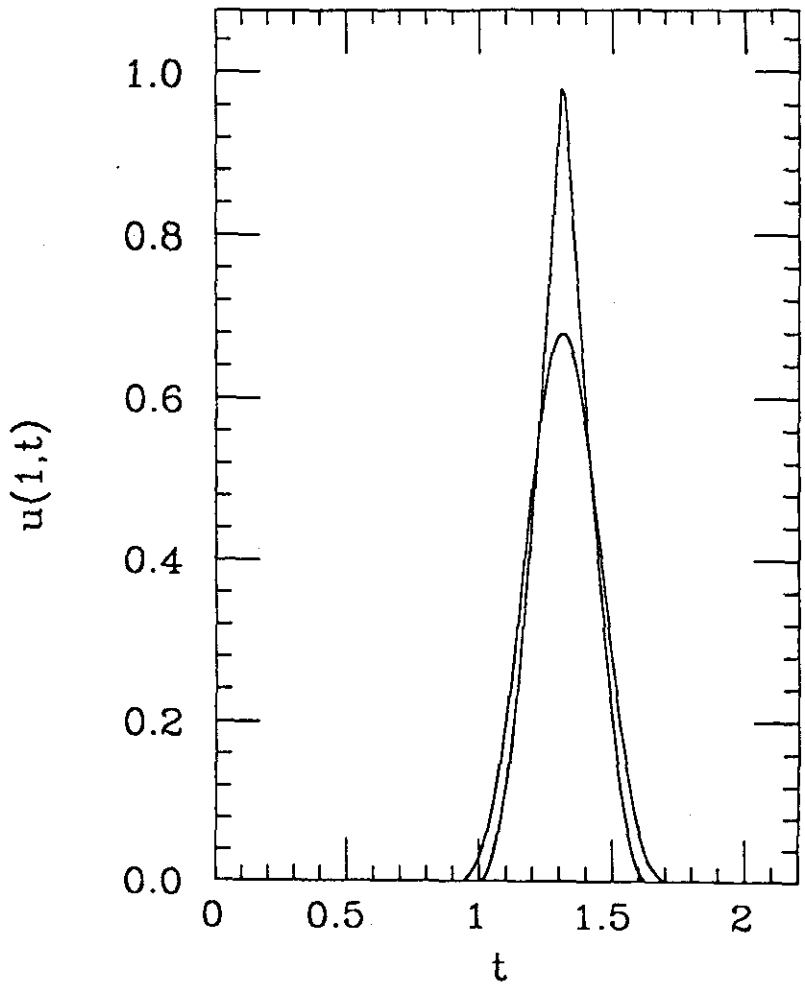
SSE = 1.947

Fig. 43



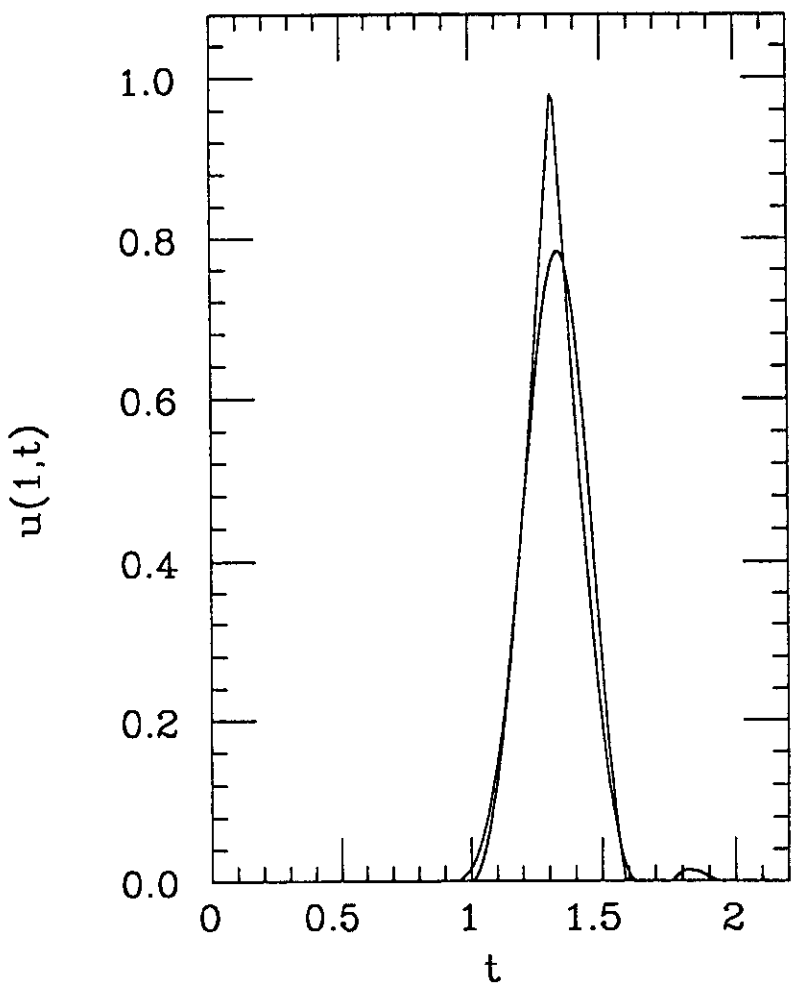
SSE = 0.244

Fig. 44



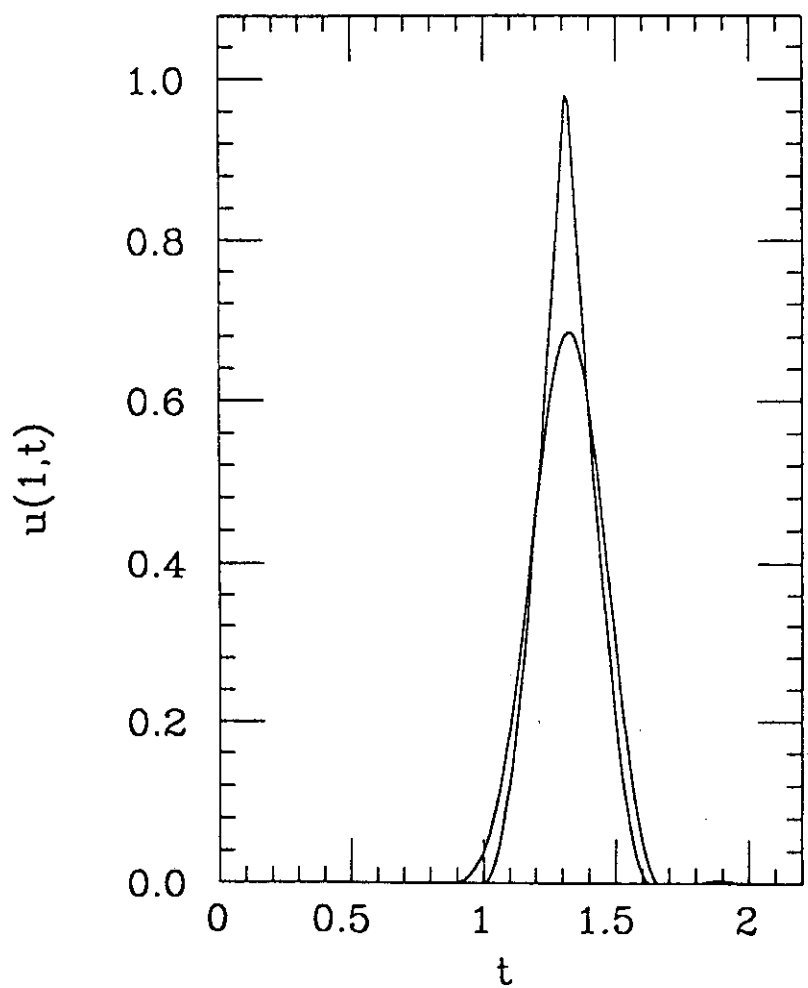
SSE = 0.680

Fig. 47



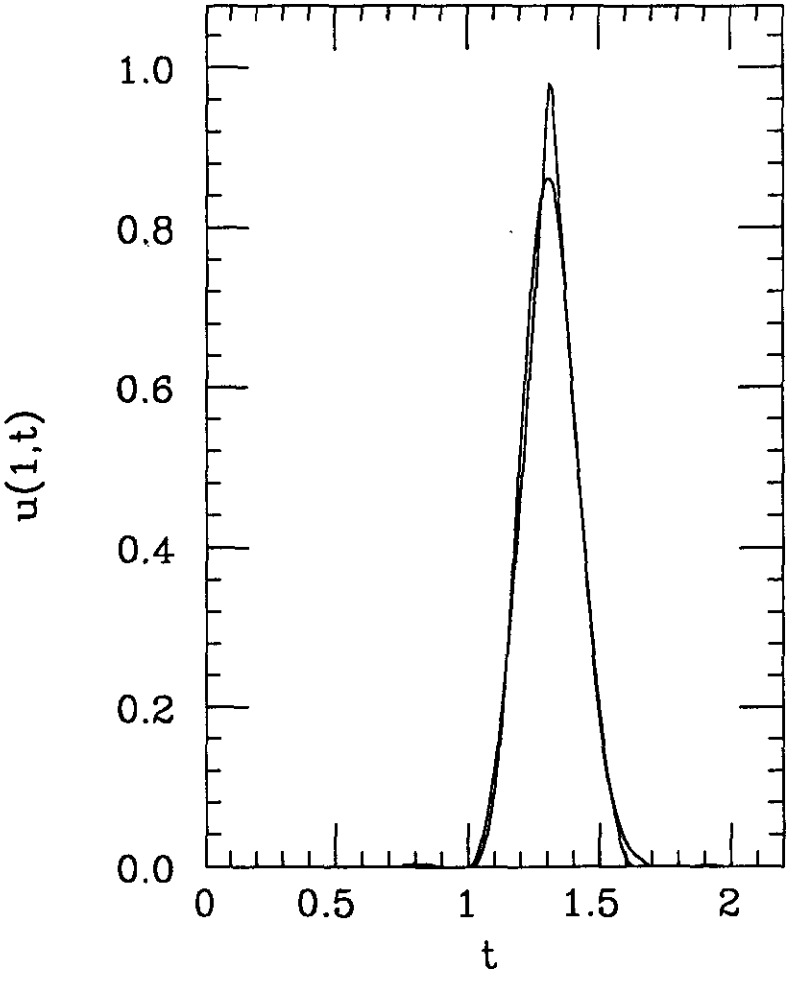
SSE = 0.384

Fig. 48



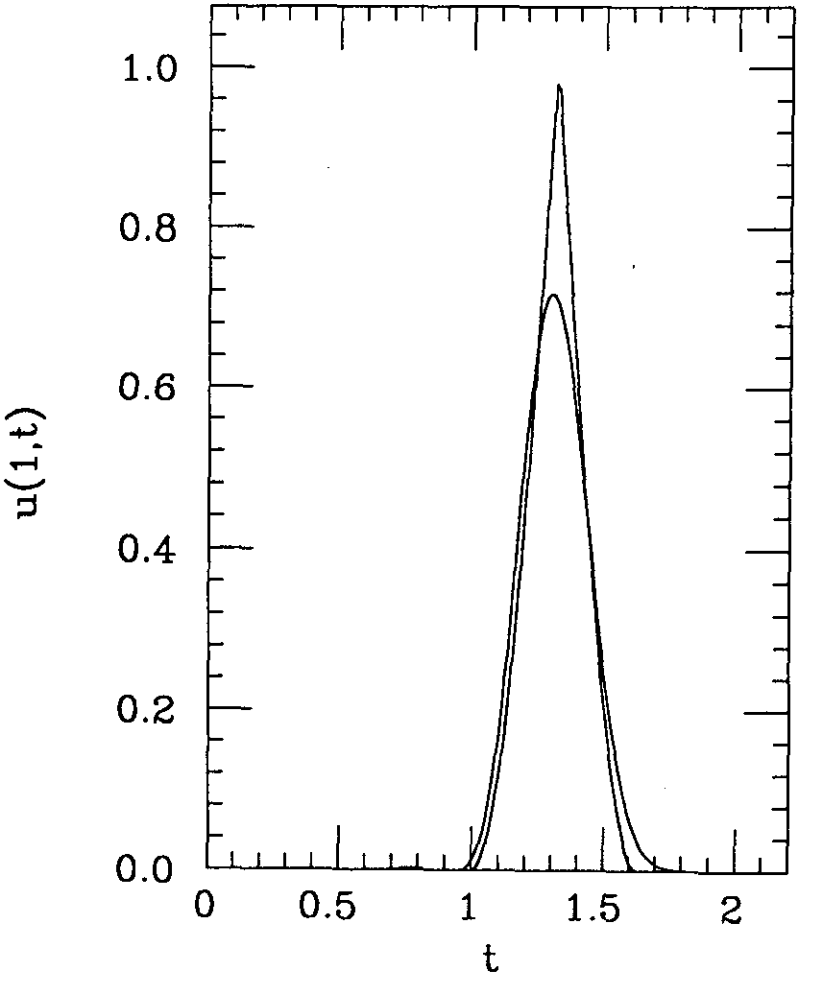
SSE = 0.671

Fig. 45



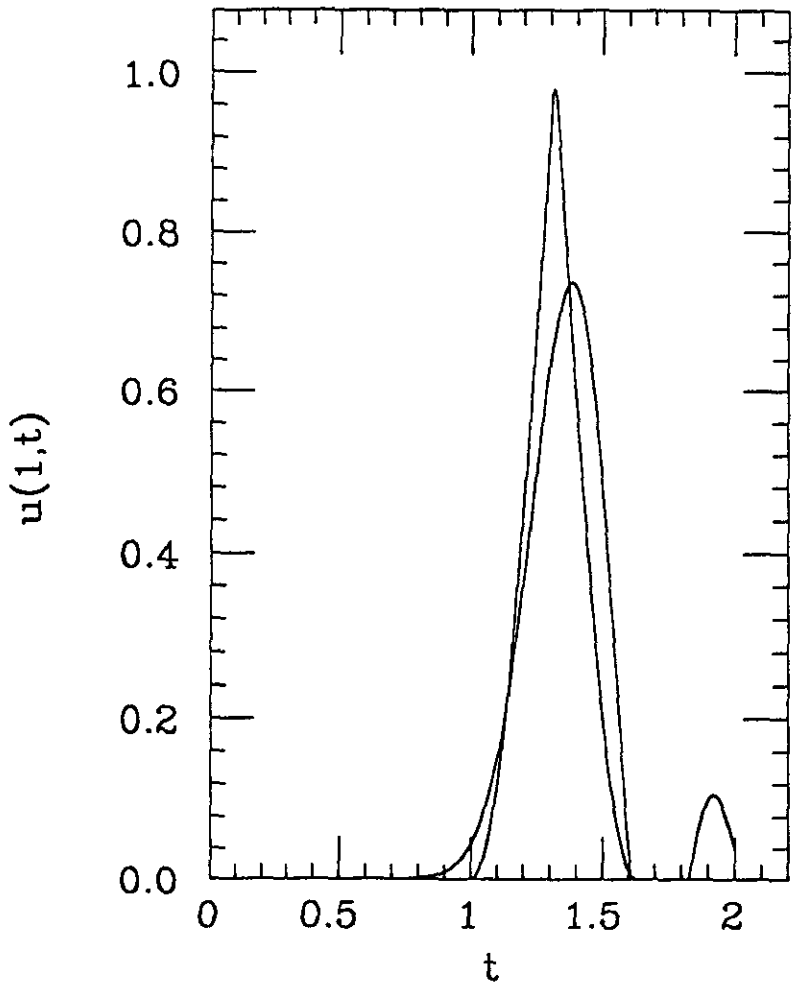
SSE = 0.094

Fig. 46



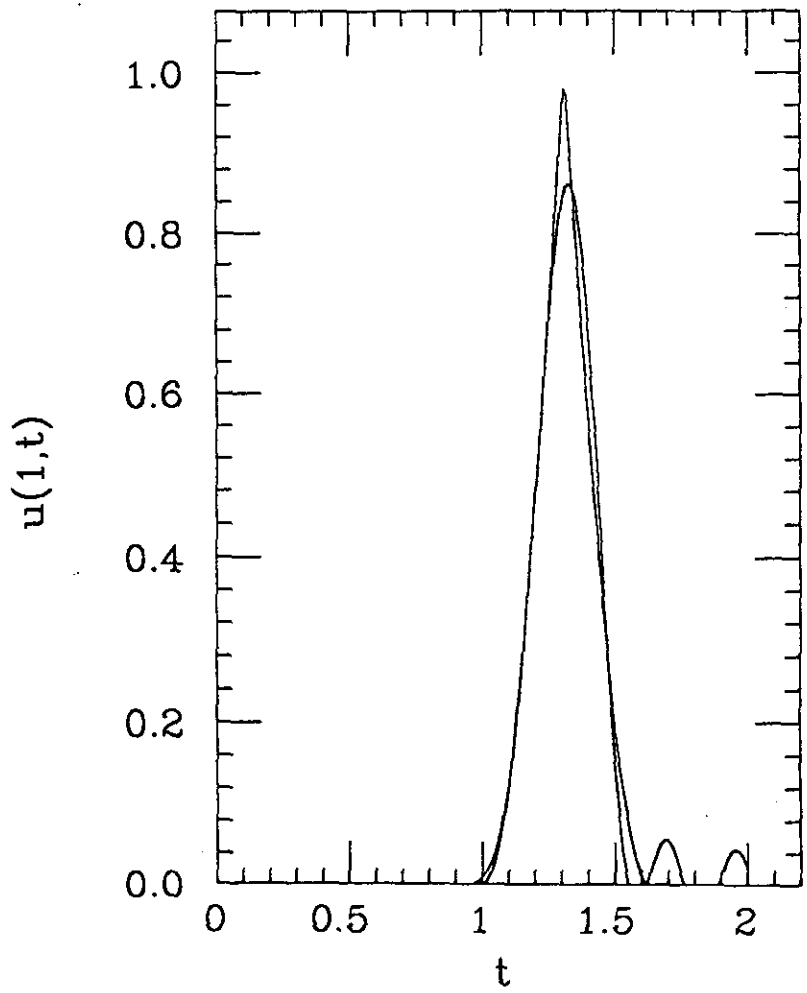
SSE = 0.498

Fig. 49



SSE = 2.335

Fig. 50



SSE = 0.235

Galerkin with Linear Finite Elements

Fig. 51: Heaviside Unit Step Function

Fig. 52: Truncated Ramp

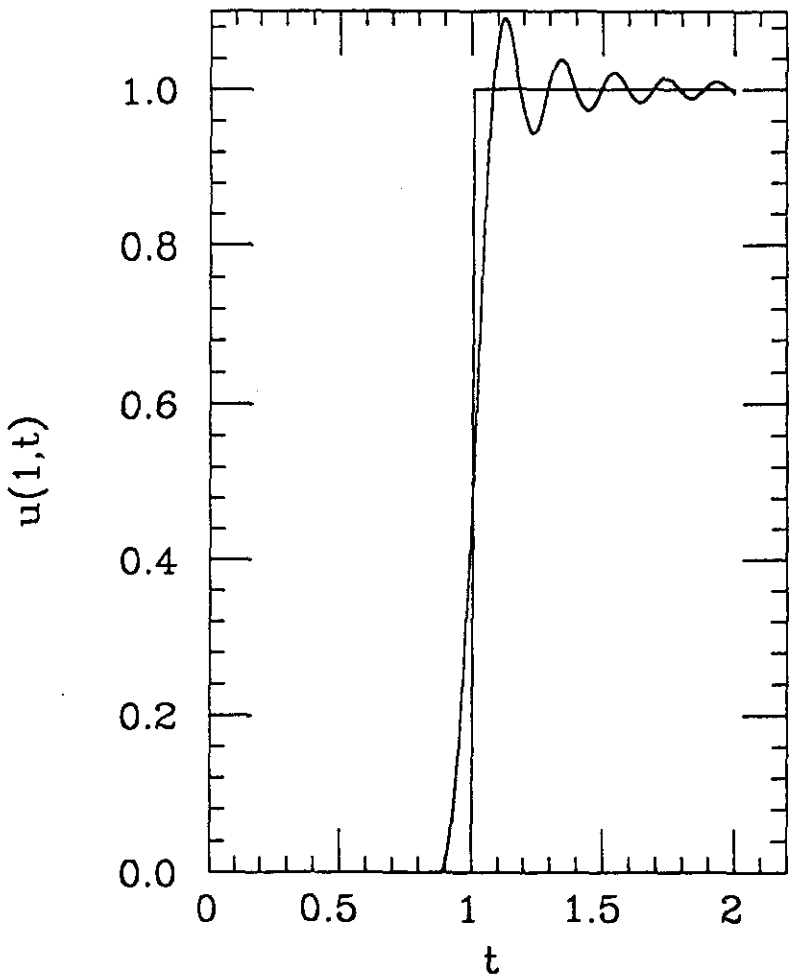
Fig. 53: Cosine Step

Fig. 54: Smooth Cosine Step

Fig. 55: Cosine Pulse

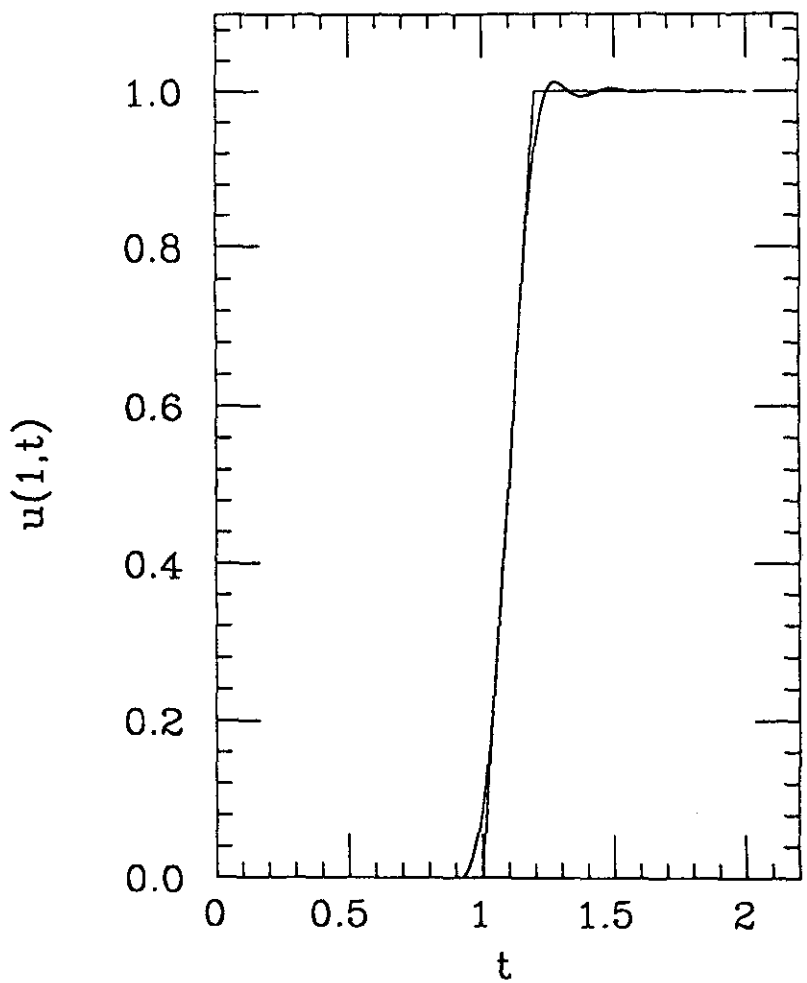
All solutions are for a 21-point grid

Fig. 51



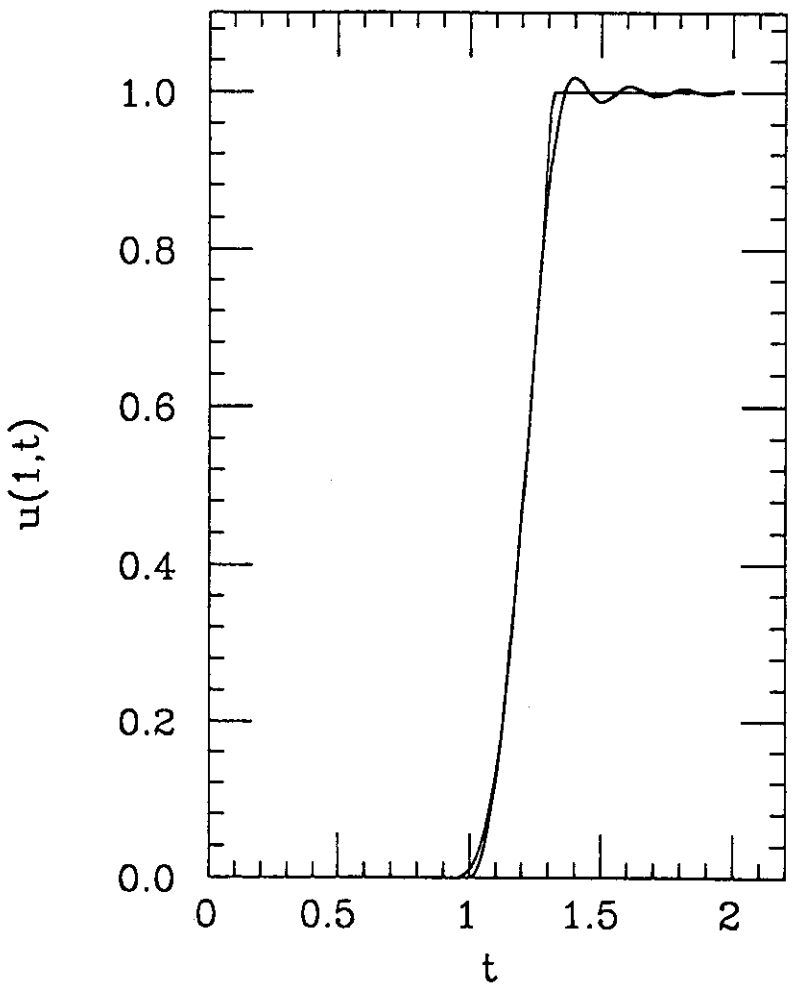
SSE = 1.271

Fig. 52



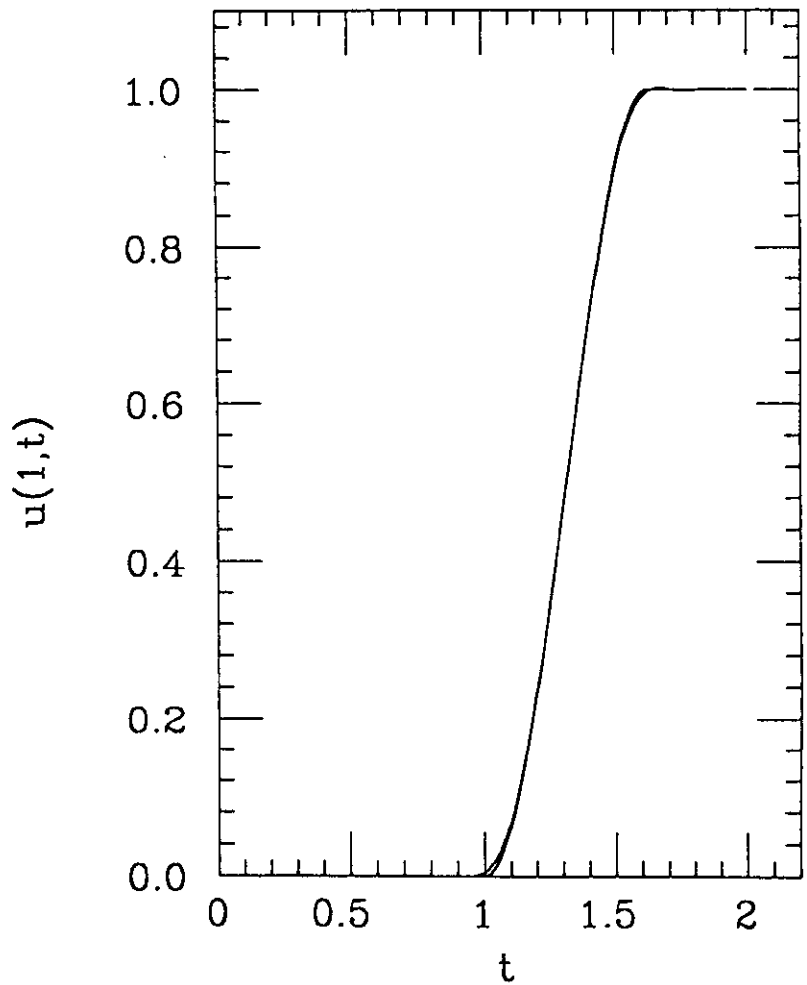
SSE = 0.037

Fig. 53



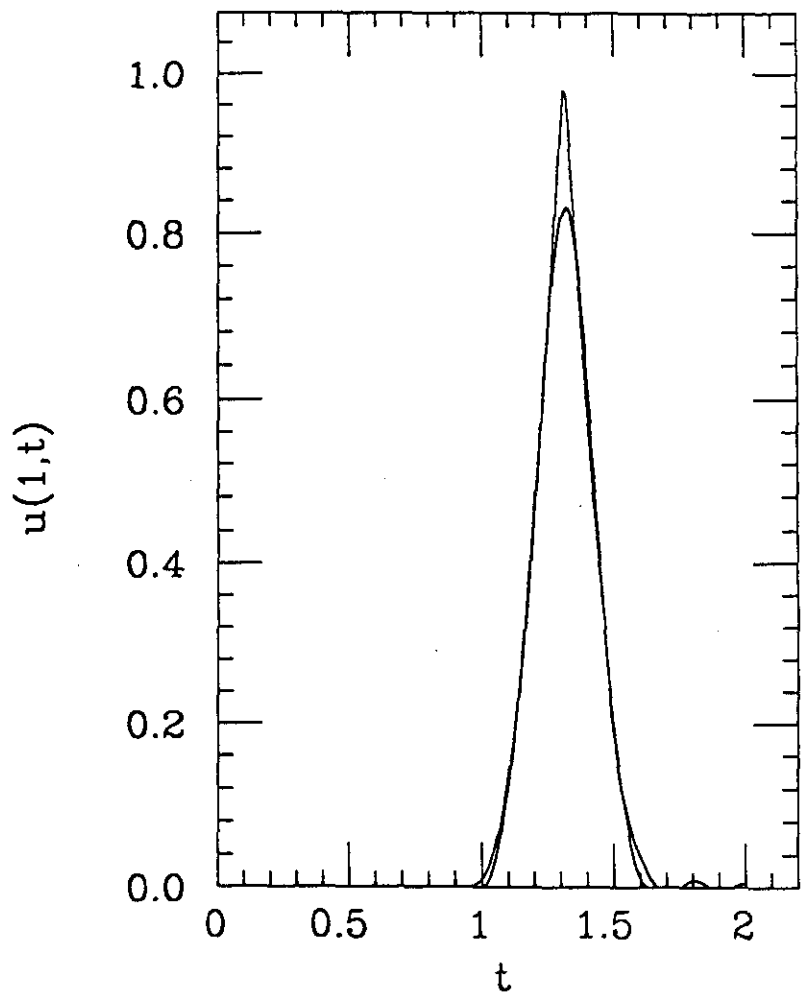
SSE = 0.027

Fig. 54



SSE = 0.002

Fig. 55



SSE = 0.099

QU/PG Basis Functions, L = 0.75, R = 0.25

Fig. 56: Heaviside Unit Step Function

Fig. 57: Truncated Ramp

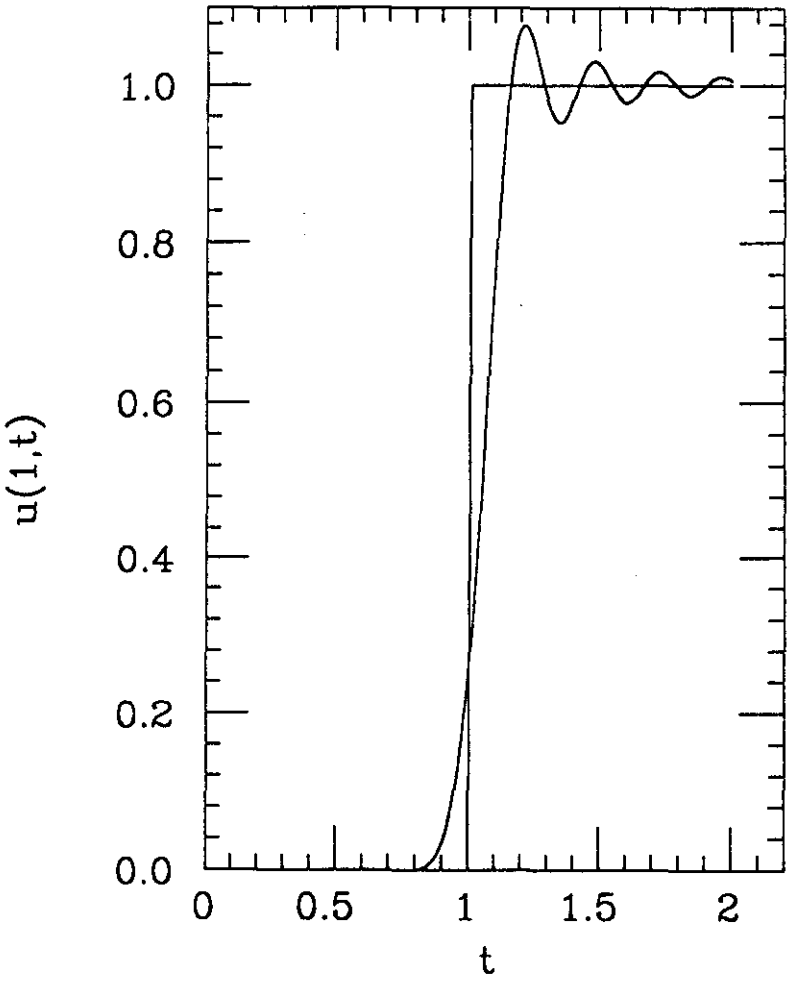
Fig. 58: Cosine Step

Fig. 59: Smooth Cosine Step

Fig. 60: Cosine Pulse

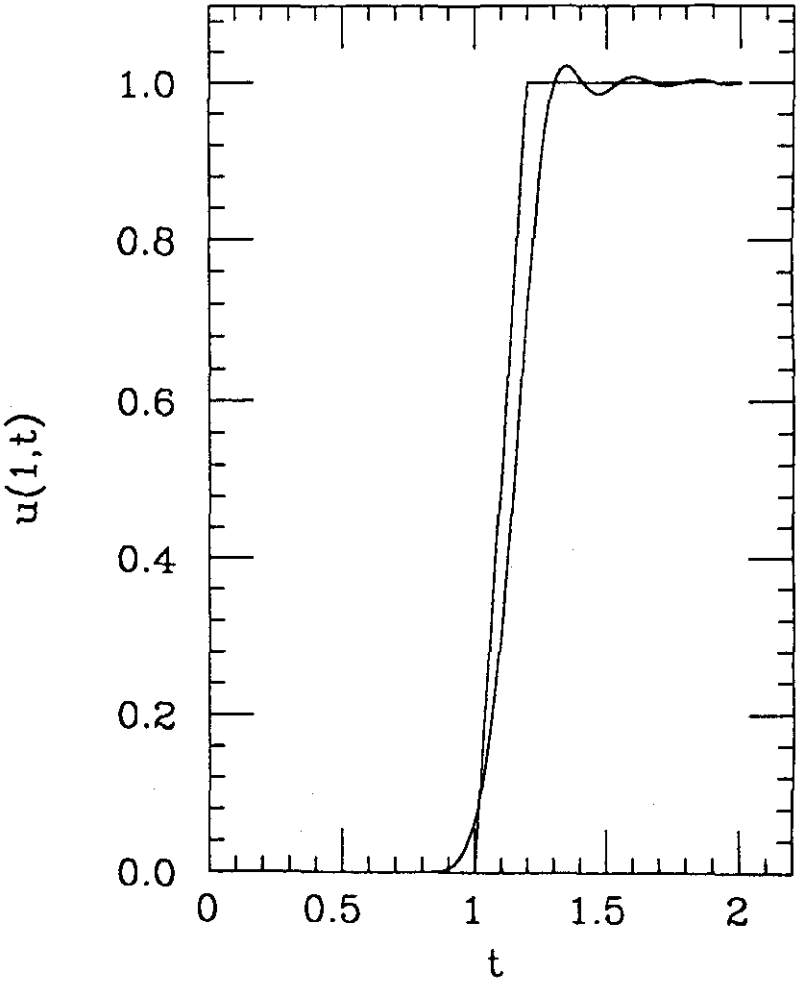
All solutions are for a 21-point grid

Fig. 56



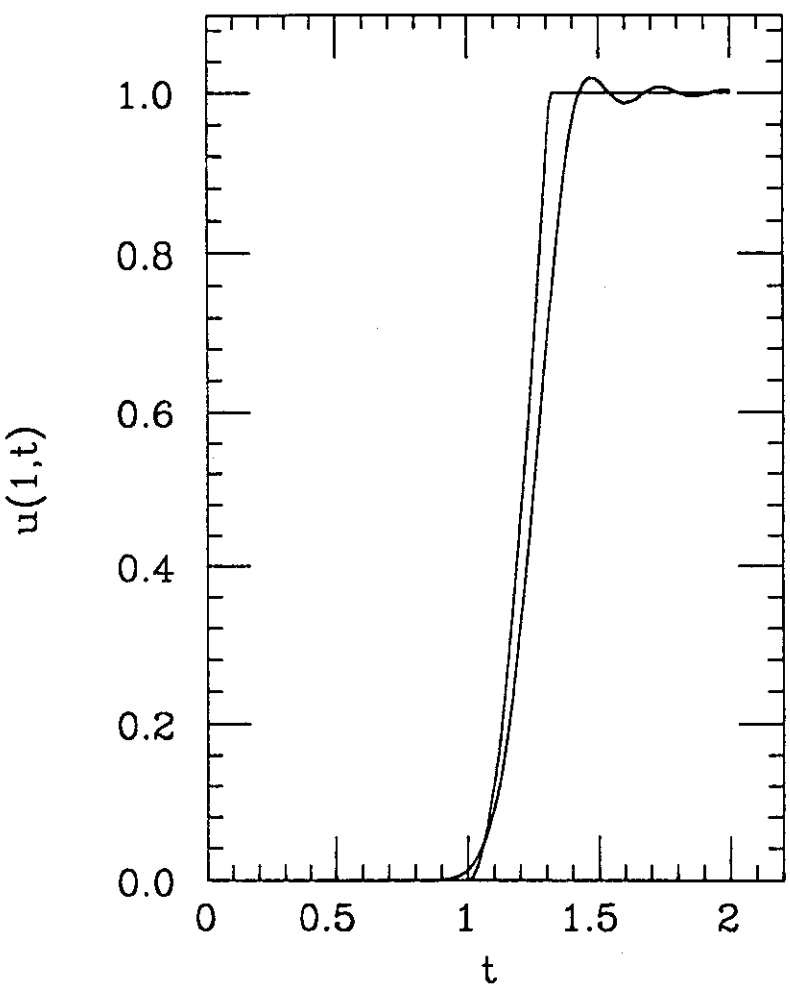
SSE = 2.952

Fig. 57



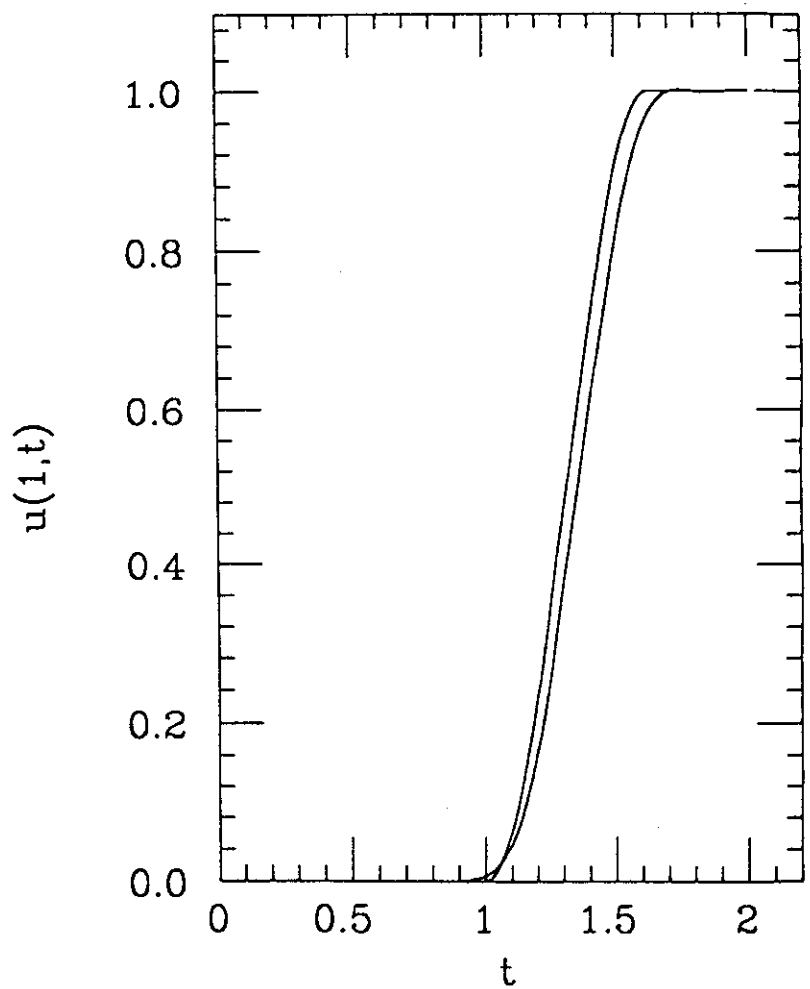
SSE = 0.954

Fig. 58



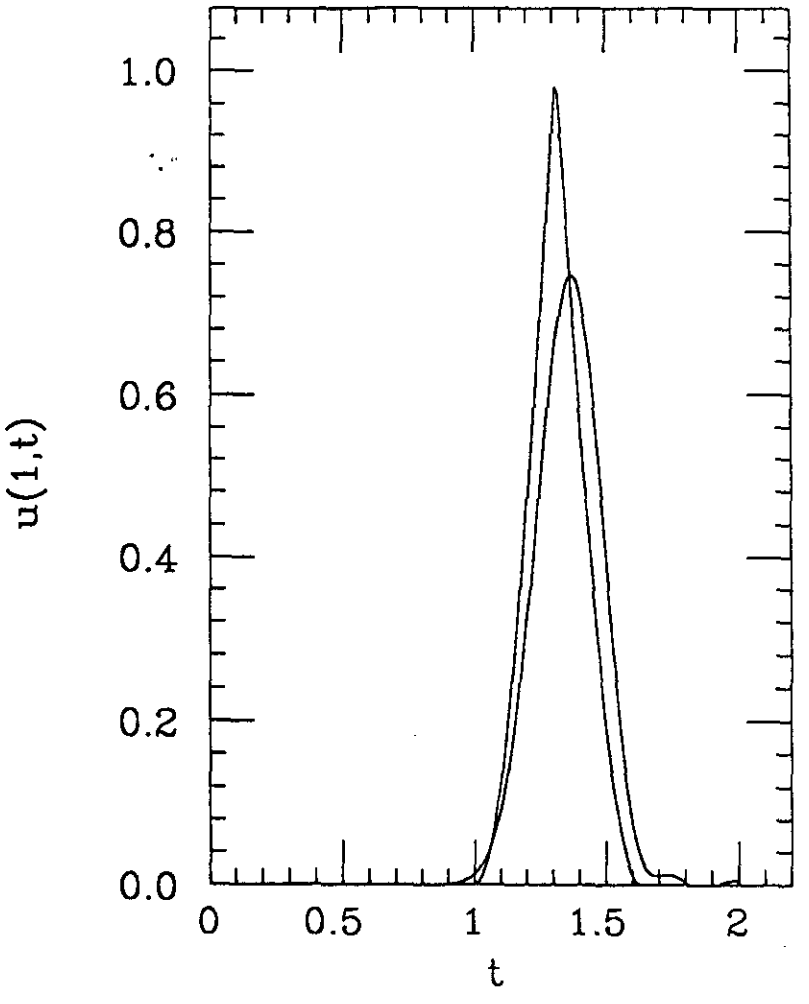
SSE = 0.731

Fig. 59



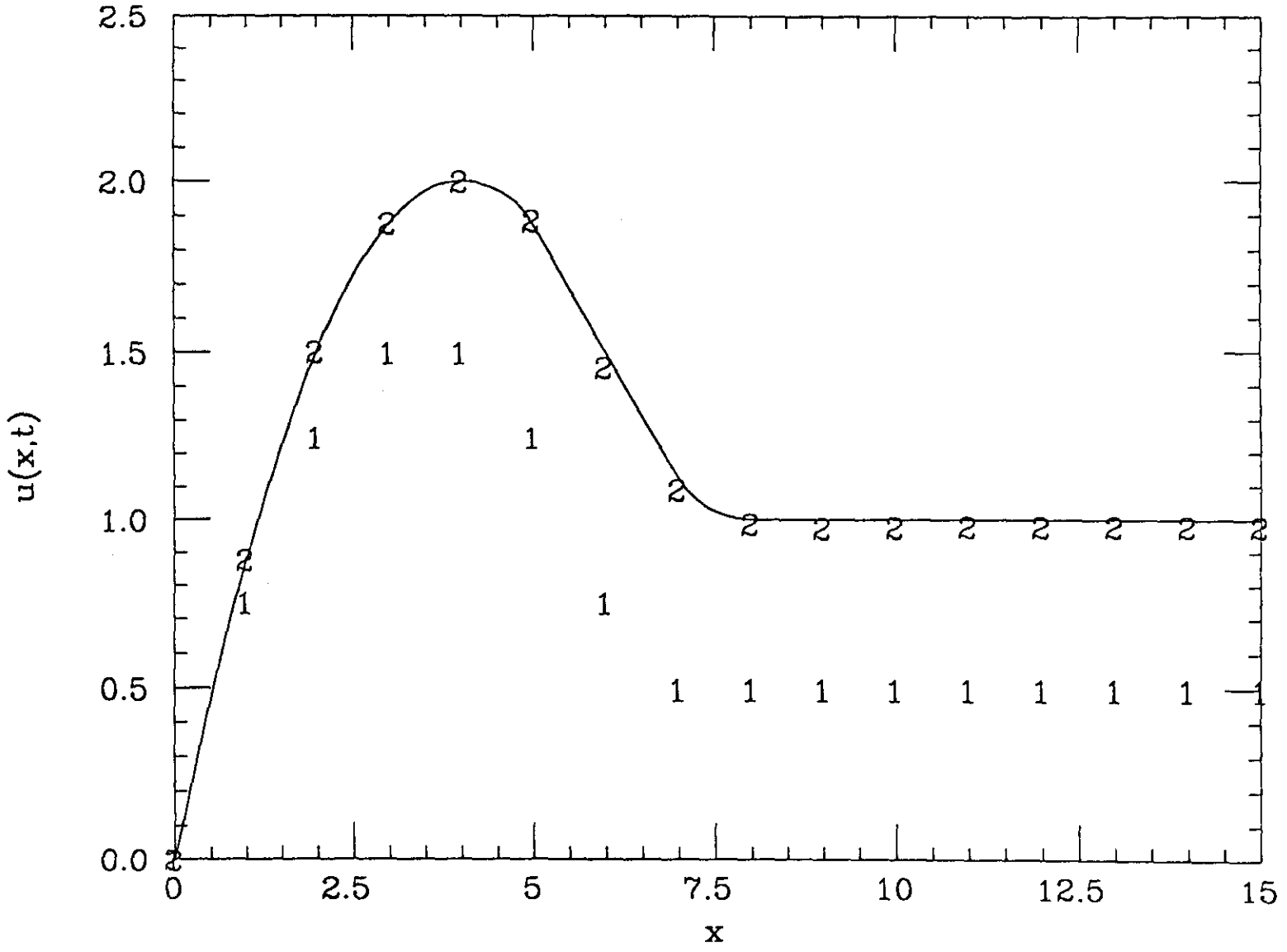
SSE = 0.355

Fig. 60



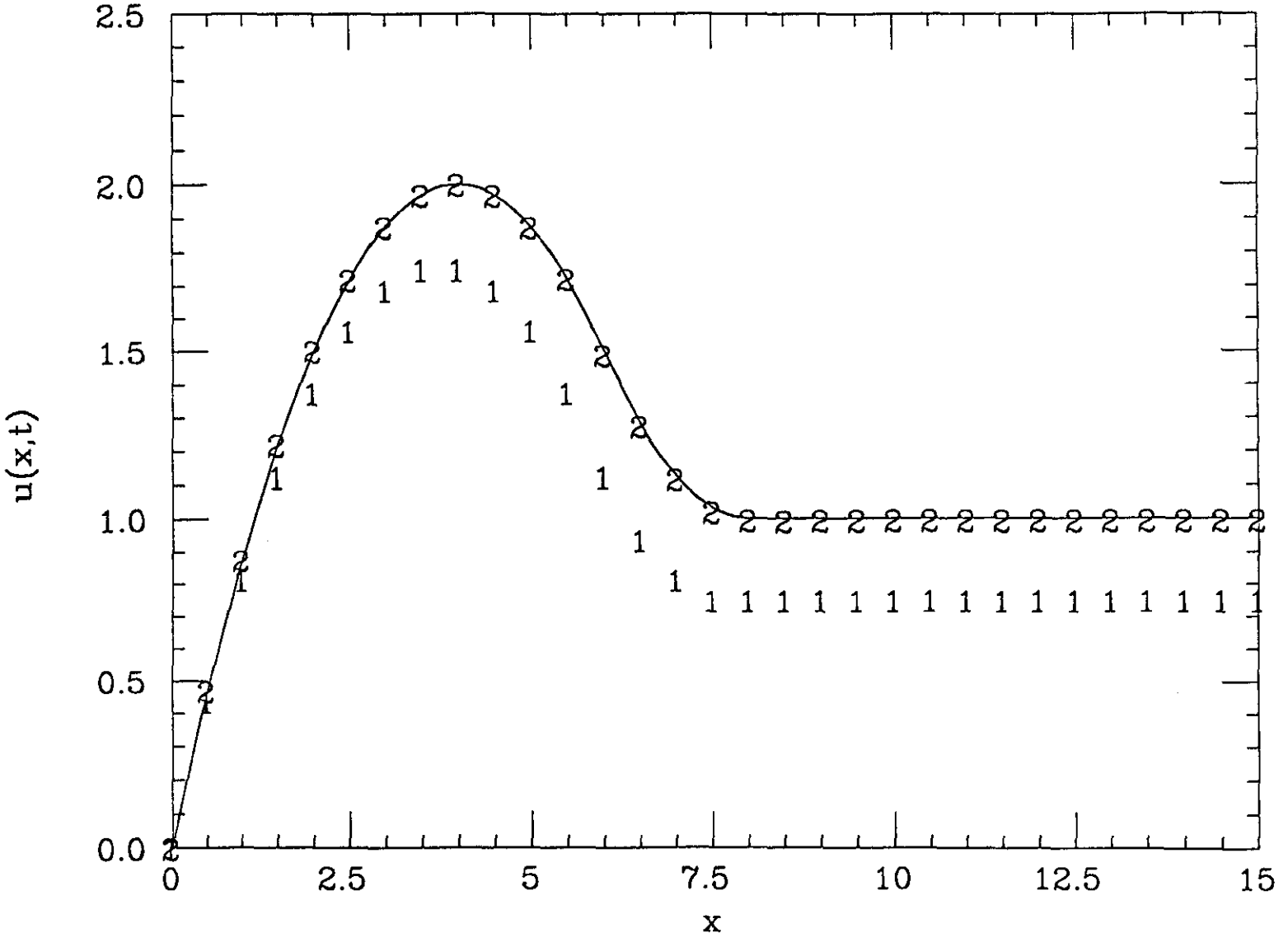
SSE = 1.505

Fig. 61: u(x,t) vs x at steady state



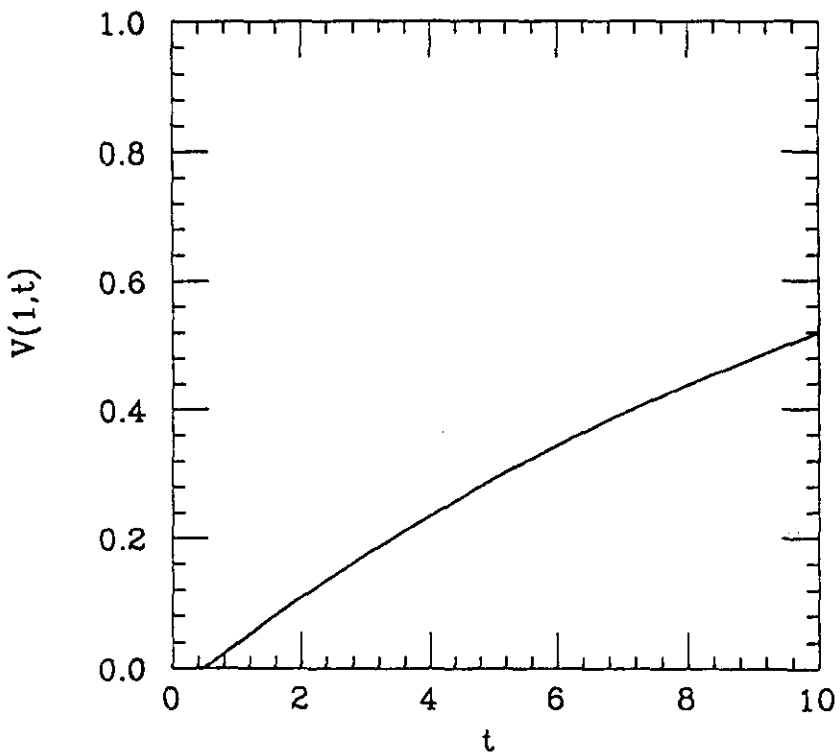
Solid – exact, 1 – 2PUW, 2 – 5PBUW

Fig. 62: u(x,t) vs x at steady state



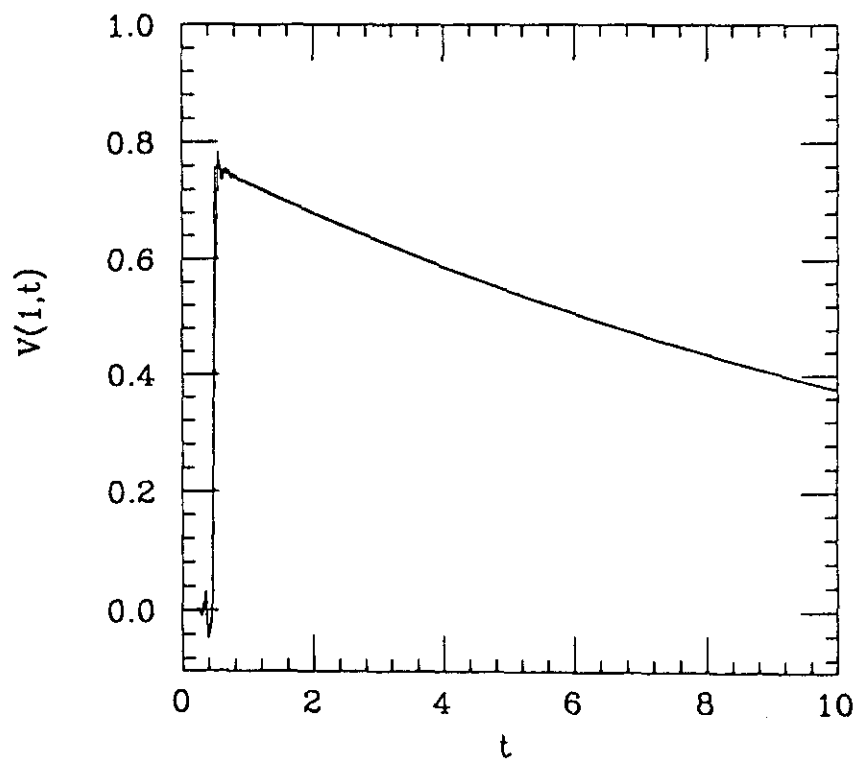
Solid – exact, 1 – 2PUW, 2 – 5PBUW

Fig. 63: Two First Order PDEs



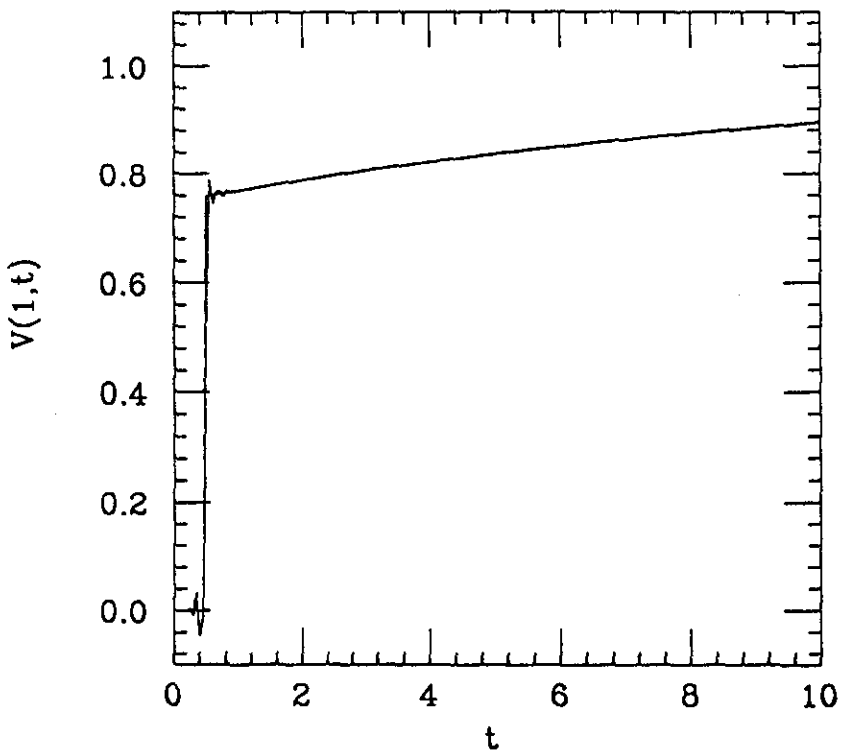
V(1,t) vs t, V(0,t) = 1 − exp(−(k/u)t)

Fig. 64: Two First Order PDEs



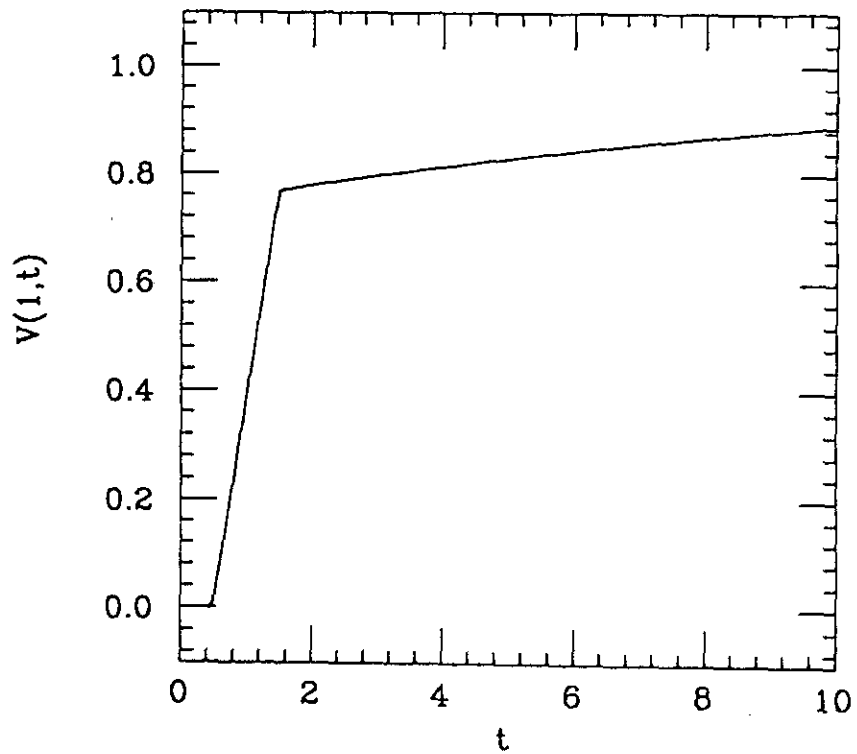
V(1,t) vs t, V(0,t) = exp(−(k/u)t)

Fig. 65: Two First Order PDEs



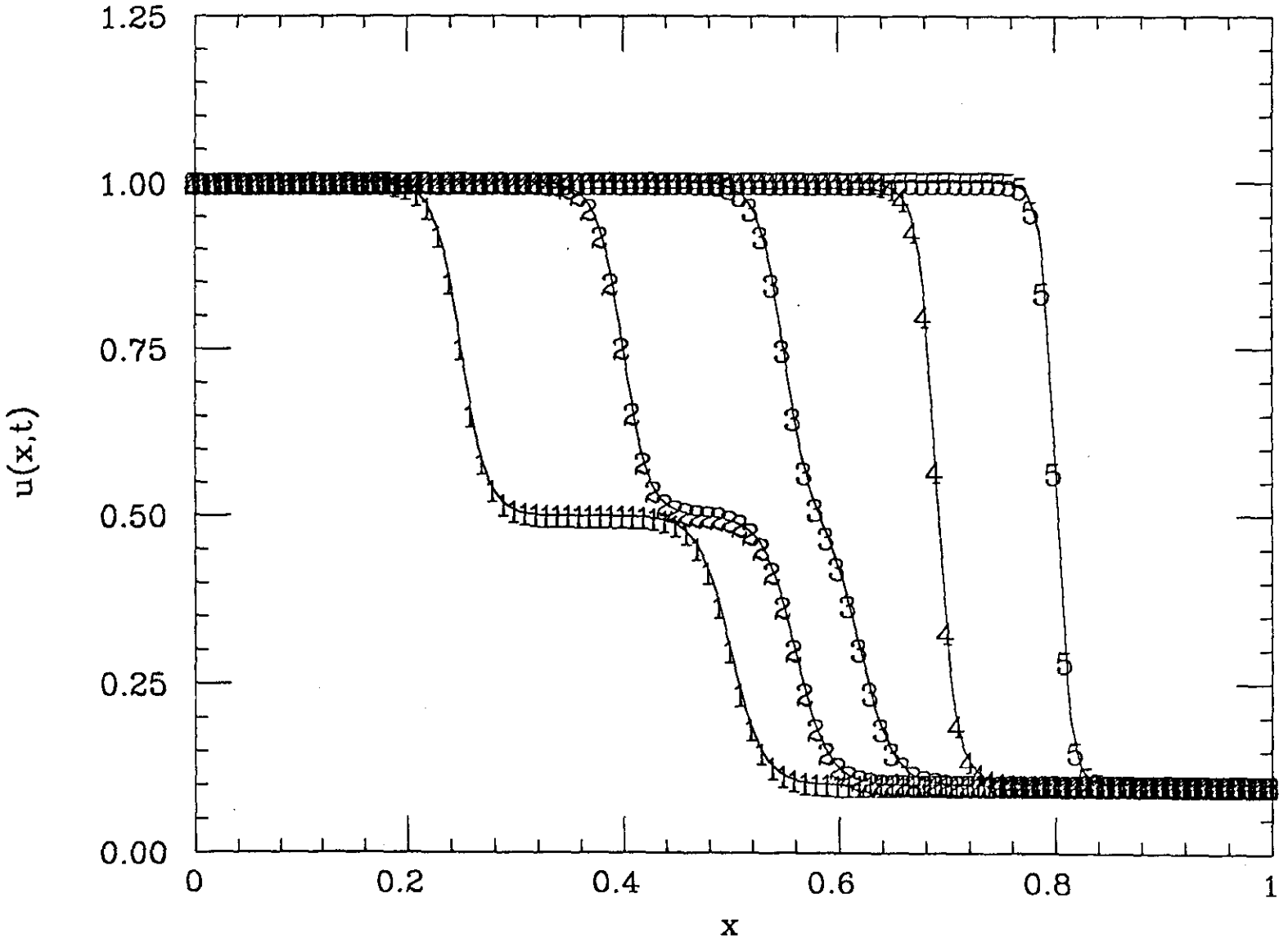
V(1,t) vs t, V(0,t) = unit step

Fig. 66: Two First Order PDEs



V(1,t) vs t, V(0,t) = unit ramp

Fig. 67: u(x,t) vs x, t = 0, 0.2, 0.4, 0.6, 0.8



Solid – exact; 1, 2, 3, 4, 5 – numerical