SSCL-N-882

An Introduction to Front-sharpening Nonlinear Convective Systems

W. Schiesser

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.

An Introduction to Front-sharpening Nonlinear Convective Systems

W. E. Schiesser
SSC Laboratory, MS 4003
2550 Beckleymeade Avenue
Dallas, TX 75237

We consider here some fundamental characteristics of convective systems which complicate the calculation of numerical solutions of the descriptive PDEs for these systems. We start with the momentum balance (the precursor of the Navier-Stokes equations) [Bird, et al (1960), eq. (3.2-20)]

$$\frac{\partial}{\partial t}\rho v_x + \frac{\partial}{\partial x}\rho v_x v_x + \frac{\partial}{\partial y}\rho v_y v_x + \frac{\partial}{\partial z}\rho v_z v_x + \frac{\partial}{\partial x}\tau_{xx} + \frac{\partial P}{\partial x} - \rho g_x = 0 \tag{1.1}$$

where

v_x, v_y, v_z	components of the velocity vector
ρ	density
P	pressure
τ_{xx}	normal component of the stress tensor
g_x	x-component of the gravitational acceleration vector

Eq. (1.1) is the x-component of the vector momentum balance (expressing Newton's second law in the x direction).

To focus on the terms of particular interest in eq. (1.1), we consider one dimensional flow ($v_y = v_z = 0$) and drop the pressure and gravitational terms

$$\frac{\partial}{\partial t}\rho v_x + \frac{\partial}{\partial x}\rho v_x v_x + \frac{\partial}{\partial x}\tau_{xx} = 0 \tag{1.2}$$

Also, if the shear is expressed in term of a visocity (through Newton's law for fluids), eq. (1.2) becomes

$$\frac{\partial}{\partial t}\rho v_x + \frac{\partial}{\partial x}\rho v_x v_x = \mu\frac{\partial^2 v_x}{\partial x^2} \tag{1.3}$$

The next step that is usually followed in analyzing eq. (1.3) is to expand the LHS

$$\rho\frac{\partial}{\partial t}v_x + v_x\frac{\partial}{\partial t}\rho + \rho v_x\frac{\partial}{\partial x}v_x + v_x\frac{\partial}{\partial x}\rho v_x = \mu\frac{\partial^2 v_x}{\partial x^2}$$

or

$$\rho\frac{\partial}{\partial t}v_x + \rho v_x\frac{\partial}{\partial x}v_x + v_x\left\{\frac{\partial}{\partial t}\rho + \frac{\partial}{\partial x}\rho v_x\right\} = \mu\frac{\partial^2 v_x}{\partial x^2} \tag{1.4}$$

The term in brackets is zero from the continuity equation (mass balance)

$$\frac{\partial}{\partial t}\rho + \frac{\partial}{\partial x}\rho v_x = 0 \tag{1.5}$$

and therefore eq. (1.4) reduces to

$$\frac{\partial}{\partial t}v_x + v_x\frac{\partial}{\partial x}v_x = \nu\frac{\partial^2 v_x}{\partial x^2} \tag{1.6}$$

where $\nu = \mu/\rho$ (the kinematic viscosity).

Eq. (1.6) is the well-known *Burgers' equation*, which has been studied extensively, both analytically and numerically (therefore, we might consider Burgers' equation as a special case of the Navier-Stokes equations, see for example Schiesser (1993)). For the purpose of the subsequent analysis, we consider the case $\nu = 0$, and in conformity with the usual convention of the numerical analysis literature, we denote the dependent variable as u, i.e., $u \Leftrightarrow v_x$. The equation that is the starting point for the subsequent analysis, the *inviscid Burgers' equation*, is therefore from eq. (1.6)

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \tag{1.7}$$

Clearly eq. (1.7) is a highly simplified form of the general momentum balance, eq. (1.1). However, for the purpose of the subsequent analysis, it has the *nonlinear advection group*, $\partial u/\partial t + u\partial u/\partial x$, which is of primary interest (and, of course, appears in computational fluid dynamics problems in general, when both the dynamic momentum and convective momentum terms, e.g., $\partial u/\partial t$ and $u\partial u/\partial x$, must be taken into consideration).

The intention of the following analysis is to:

(1) Compare analytical and numerical solutions to eq. (1.7) in order to illustrate and evaluate the numerical methods.

(2) Illustrate some important properties of the nonlinear advection group, e.g., front flattening and front sharpening, which can have a major effect on the performance of numerical methods used to integrate eq. (1.7).

Thus, we proceed first to an analytical solution of eq. (1.7).

If we assume a product solution of the form

$$u(x,t) = f(x)g(t) \tag{1.8}$$

substitution in eq. (1.7) gives

$$f(x)g'(t) + f(x)g(t)^2 f'(x) = 0 \tag{1.9}$$

which can be separated into

$$g'(t)/g^2(t) = - f'(x) = \lambda \tag{1.10}$$

where λ is a separation constant (to be determined). Thus, we have two ordinary differential equations (ODEs)

$$g'(t) - \lambda g^2(t) = 0 \tag{1.11}$$

$$f'(x) + \lambda = 0 \tag{1.12}$$

Eq. (1.11) has a solution

$$g(t) = \frac{1}{a + bt} \tag{1.13}$$

where a and b are constants (to be determined) and $\lambda = -b$. Then, from eq. (1.12)

$$f(x) = bx + c \tag{1.14}$$

where c is a constant (to be determined). Thus, a solution to eq. (1.7) is

$$u(x,t) = \frac{c + bx}{a + bt} \tag{1.15}$$

In the subsequent analysis, we take the one initial condition and the one boundary condition required by eq. (1.7) to be consistent with eq. (1.15).

We now consider a method of lines solution to eq. (1.7), subject to the initial and boundary conditions from eq. (1.15). The first case is for: (1) a > and b > 0 and (2) a > 0 and b < 0, i.e., two executions or "runs" of the program. Subroutine INITAL to define the initial condition for eq. (1.7) (from eq. (1.15), u(x,0) = c + bx/a) is listed below

```
      SUBROUTINE INITAL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /I/       N,   NCASE,      IP,      NT
C...
C...  SELECT CASE
      NCASE=1
C...
C...  SELECT INITIALIZATION ROUTINE
      IF(NCASE.EQ.1)CALL INIT1
      IF(NCASE.EQ.2)CALL INIT2
      IF(NCASE.EQ.3)CALL INIT3
      RETURN
      END
```

We can note in subroutine INITAL that three cases are programmed, NCASE = 1, 2 or 3, for which three initialization routines are called, INIT1 for NCASE = 1, etc. In the present INITAL, NCASE = 1.

Subroutine INIT1 is listed below:

```
      SUBROUTINE INIT1
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NX=101)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /Y/  U(NX)
     +       /F/ UT(NX)
     +       /S/ UX(NX)
     +       /X/  X(NX)
     +       /C/       A,       B,       C,      XL,      DX
     +       /I/       N,   NCASE,      IP,      NT
C...
C...  SELECT THE SOLUTION PARAMETERS
C...
C...     FRONT FLATTENING
         IF(NORUN.EQ.1)THEN
         A=1.0D0
         B=1.0D0
```

```
          C=0.0D0
          NT=11
C...
C...      FRONT SHARPENING
          ELSE
     +    IF(NORUN.EQ.2)THEN
          A= 1.01D0
          B=-1.0D0
          C= 1.0D0
          NT=11
          END IF
C...
C...   NUMBER OF GRID POINTS
       N=11
C...
C...   TOTAL LENGTH IN X, GRID SPACING
       XL=1.0D0
       DX=XL/DFLOAT(N-1)
C...
C...   INITIAL CONDITION
       DO 1 I=1,N
          X(I)=XL*DFLOAT(I-1)/DFLOAT(N-1)
          U(I)=ANAL(X(I),0.0D0)
1      CONTINUE
C...
C...   INITIAL DERIVATIVES
       CALL DERV
       IP=0
       RETURN
       END
```

We can note the following points about subroutine INIT1:

(1) The coding is in double precision. A maximum of 101 grid points in x is specified.

```
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       PARAMETER(NX=101)
       COMMON/T/       T,   NSTOP,   NORUN
     +       /Y/  U(NX)
     +       /F/ UT(NX)
     +       /S/ UX(NX)
     +       /X/  X(NX)
     +       /C/       A,       B,       C,      XL,      DX
     +       /I/       N,   NCASE,      IP,      NT
```

The following variables are used throughout the code:

U(NX) u(x,t) in eq. (1.7)

UT(NX)	$\partial u/\partial t$ in eq. (1.7), (which is subsequently expressed in subscript notation as u_t)
UX(NX)	$\partial u/\partial x$ in eq. (1.7), (in subscript notation, u_x)
X(NX)	spatial variable x in eq. (1.7), defined on a grid of NX points
A,B,C	a, b and c in eq. (1.15)
XL	length of the x axis
DX	interval used in the finite difference approximation of $\partial u/\partial x$ in eq. (1.7)
N	number of spatial grid points actually used ($\leq$ NX)
NCASE	case implemented in the computer code (subsequently explained)
IP	integer index used in the plotting routine *PLOTTD*
NT	total number of output points used in *PLOTTD*

(2) Two runs are programmed

```
C...
C...   SELECT THE SOLUTION PARAMETERS
C...
C...       FRONT FLATTENING
           IF(NORUN.EQ.1)THEN
           A=1.0D0
           B=1.0D0
           C=0.0D0
           NT=11
C...
C...       FRONT SHARPENING
           ELSE
      +    IF(NORUN.EQ.2)THEN
           A= 1.01D0
           B=-1.0D0
           C= 1.0D0
           NT=11
           END IF
```

For the first run (NORUN = 1), a = 1, b = 1 and c = 0 in eq. (1.15); the number of output points, NT, is 11. For the second run (NORUN = 2), a = 1.01, b = -1 and c = 1 in eq. (1.15), and again the number of output points is 11.

(3) In both runs, the number of spatial grid points, N, is 11

```
C...
C...  NUMBER OF GRID POINTS
      N=11
```

A small number of grid points (i.e., N $\ll$ NX) can be used since the solution is quite smooth for both runs (as discussed subsequently).

(4) The total length of the system, XL, is unity, and the grid spacing for the finite difference approximation of $\partial u/\partial x$ in eq. (1.7), DX, is then computed for this value of the length

```
C...
C...  TOTAL LENGTH IN X, GRID SPACING
      XL=1.0D0
      DX=XL/DFLOAT(N-1)
```

(5) The spatial grid, X(I), and the initial condition from eq. (1.15), implemented via function ANAL, are defined in DO loop 1

```
C...
C...  INITIAL CONDITION
      DO 1 I=1,N
         X(I)=XL*DFLOAT(I-1)/DFLOAT(N-1)
         U(I)=ANAL(X(I),0.0D0)
1     CONTINUE
C...
C...  INITIAL DERIVATIVES
```

Function ANAL is a straightforward implementation of eq. (1.15)

```
      DOUBLE PRECISION FUNCTION ANAL(X,T)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/C/      A,      B,      C,      XL,      DX
      ANAL=(C+B*X)/(A+B*T)
      RETURN
      END
```

Note the use of a, b and c in eq. (1.15) available through COMMON/C/.

(6) A call to DERV then computes the initial derivatives in eq. (1.7). Finally, the index for plotting used in subroutine PLOTTD is intialized

```
C...
C...  INITIAL DERIVATIVES
```

```
      CALL DERV
      IP=0
```

Subroutine DERV for the calculation of the derivatives in eq. (1.7) is listed below

```
      SUBROUTINE DERV
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /I/       N,   NCASE,      IP,      NT
C...
C...  SELECT DERIVATIVE SUROUTINE
      IF(NCASE.EQ.1)CALL DERV1
      IF(NCASE.EQ.2)CALL DERV2
      IF(NCASE.EQ.3)CALL DERV3
      RETURN
      END
```

Note the COMMON area is the same as in INITAL, and three derivative subroutines are called, depending on NCASE (currently, NCASE = 1, so DERV1 is called).

Subroutine DERV1 is listed below

```
      SUBROUTINE DERV1
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NX=101)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /Y/  U(NX)
     +       /F/ UT(NX)
     +       /S/ UX(NX)
     +       /X/  X(NX)
     +       /C/       A,       B,       C,      XL,      DX
     +       /I/       N,   NCASE,      IP,      NT
C...
C...  BOUNDARY CONDITION AT X = 0
       U(1)=ANAL(0.0D0,T)
      UT(1)=0.0D0
C...
C...  PDE
      DO 1 I=2,N
         UT(I)=-U(I)*(U(I)-U(I-1))/DX
1     CONTINUE
      RETURN
      END
```

We can note the following points about DERV1:

(1) The COMMON area is the same as in INIT1.

(2) The boundary condition for eq. (1.7), u(0,t), is defined by analytical solution eq. (1.15) via function ANAL

```
C...
C...   BOUNDARY CONDITION AT X = 0
        U(1)=ANAL(0.0D0,T)
       UT(1)=0.0D0
```

Also, since u(0,t) is defined, its temporal derivative, $\partial u(0,t)/\partial t$ is set to zero so that the ODE integrator does not move u(0,t) away from its prescribed value. Of course, mathematically, this is not correct since this temporal derivative is not zero (from eq. (1.15), $\partial u(0,t)/\partial t = -bc/(a + bt)^2$). However, the next time DERV1 is called, the boundary value, u(0,t), will again be reset according to eq. (1.15) (via the call to ANAL). By zeroing the derivative, $\partial u(0,t)/\partial t$, the ODE integration at x = 0 is facilitated. We should note, however, that this procedure may not work correctly with some ODE integrators, e.g., the BDF integrators of ODEPACK such as LSODE and LSODES, and the DAE integrator DASSL; for such integrators, the correct temporal derivative at x = 0 may be required, and the setting of the dependent variable at x = 0 would then be dropped. In other words, the following alternative programming could be used

```
C...
C...   BOUNDARY CONDITION AT X = 0
       UT(1)=ANALT(0.0D0,T)
```

where the $\partial u(x,t)/\partial t$ is programmed in function ANALT as

```
      DOUBLE PRECISION FUNCTION ANALT(X,T)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/C/       A,       B,       C,       XL,       DX
      ANALT=-B*(C+B*X)/(A+B*T)**2
      RETURN
      END
```

Now we are actually integrating the correct ODE at x = 0, and the initial condition for this ODE, u(0,0), would be set in INIT1 as u(0,0) = c/a (from eq. (1.15)).

(3) The method of lines approximation of eq. (1.7) is finally programmed over points 2 to N

```
C...
```

```
C...   PDE
       DO 1 I=2,N
          UT(I)=-U(I)*(U(I)-U(I-1))/DX
1      CONTINUE
```

Here we have used the first order, backward finite difference approximation

$$\frac{\partial u(x,t)}{\partial x} \approx \frac{u(x,t) - u(x-\Delta x,t)}{\Delta x} \Leftrightarrow (U(I)-U(I-1))/DX \tag{1.16}$$

This finite difference representation of the spatial derivative in eq. (1.7) is usually termed a *two point, upwind approximation* since the "upwind" point, $u(x-\Delta x,t)$ is used. Note, also, that this approximation is first order correct, i.e.,

$$\frac{\partial u(x,t)}{\partial x} \approx \frac{u(x,t) - u(x-\Delta x,t)}{\Delta x} + O(\Delta x) \tag{1.17}$$

where "O" (capital oh) is interpreted as "of order". In other words, the approximation of eq. (1.17) is exact (or "superaccurate") for constant and linear functions (in x), as we shall observe in the subsequent numerical solutions.

Subroutine DERV1 defines the temporal derivatives for all of the method of lines ODEs, in this case N = 11 ODEs. This derivative vector, UT(N), is sent through COMMON/F/ to an ODE integrator, in this case RKF45 [Forsythe, et al (1977)]. The ODE integrator then returns the dependent variable vector, U(N), through COMMON/Y/ for the programming of the temporal derivatives in DERV1. The dependent variable vector, U(N), which is the numerical solution to eq. (1.7), is printed via subroutine PRINT, listed below

```
      SUBROUTINE PRINT(NI,NO)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /I/      N,   NCASE,      IP,      NT
C...
C...  SELECT PRINT ROUTINE
      IF(NCASE.EQ.1)CALL PRINT1(NI,NO)
      IF(NCASE.EQ.2)CALL PRINT2(NI,NO)
      IF(NCASE.EQ.3)CALL PRINT3(NI,NO)
      RETURN
      END
```

PRINT is similar to INITAL and DERV. For NCASE = 1, PRINT1 is called to print and plot the numerical solution resulting from the integration of the ODEs programmed in DERV1. PRINT1 is listed below

```
      SUBROUTINE PRINT1(NI,NO)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NX=101)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /Y/  U(NX)
     +       /F/ UT(NX)
     +       /S/ UX(NX)
     +       /X/  X(NX)
     +       /C/       A,       B,       C,      XL,      DX
     +       /I/       N,   NCASE,      IP,      NT
C...
C...  MONITOR OUTPUT ON SCREEN
      WRITE(*,*)NORUN,T
C...
C...  PRINT NUMERICAL AND ANALYTICAL SOLUTIONS
      WRITE(NO,3)T
3        FORMAT(/,' t = ',F5.2,/,9X,'x',4X,'u(x,t)',3X,'ua(x,t)')
         DO 1 I=1,N
         UA=ANAL(X(I),T)
         WRITE(NO,2)X(I),U(I),UA
2        FORMAT(F10.2,2F10.5)
1        CONTINUE
C...
C...  PLOT THE SOLUTION
      IP=IP+1
      CALL PLOTTD
      RETURN
      END
```

We can note the following points about subroutine PRINT1:

(1) The COMMON area is the same as in INIT1 and DERV1.

(2) The numerical solution, UA, is computed via function ANAL, then printed along with the numerical solution, U(I), as a function of x (at the values of t corresponding to when PRINT1 is called, as defined by the data file discussed subsequently)

```
C...
C...  PRINT NUMERICAL AND ANALYTICAL SOLUTIONS
      WRITE(NO,3)T
3        FORMAT(/,' t = ',F5.2,/,9X,'x',4X,'u(x,t)',3X,'ua(x,t)')
         DO 1 I=1,N
         UA=ANAL(X(I),T)
         WRITE(NO,2)X(I),U(I),UA
2        FORMAT(F10.2,2F10.5)
1        CONTINUE
```

(3) Finally, the counter IP is incremented each time PRINT1 is called to control the storing and plotting of the solution by subroutine PLOTTD

```
C...
C...   PLOT THE SOLUTION
       IP=IP+1
       CALL PLOTTD
```

PLOTTD writes a computer file with the solution to be plotted. This file is then read by a plotting system, Top Drawer, to draw the plots discussed subsequently. Subroutine PLOTTD is listed in Appendix A. The data file read by the main program that calls ODE integrator RKF45 is listed below

```
Inviscid Burgers equation, front flattening
0.        1.0       0.1
   11                     0.0001
Inviscid Burgers equation, front sharpening
0.        1.0       0.1
   11                     0.0001
END OF RUNS
```

This file defines:

(1) Line 1: A title for the run.

(2) Line 2: The initial, final and print interval values of t. Thus, PRINT1 is call at t = 0, 0.1, 0.2,. . ., 1 (i.e., 11 times).

(3) Line 3: 11 ODEs to be integrated by RKF45 with an accuracy of 0.0001.

Three lines of data are read for each run. At the end of the second run, the characters END OF RUNS are read to terminate execution of the main program.

Abbreviated output from the main program and subroutine PRINT1 is listed below

```
RUN NO. -   1  Inviscid Burgers equation, front flattening
INITIAL T -  0.000D+00
```

```
  FINAL T -  0.100D+01

  PRINT T -  0.100D+00

NUMBER OF DIFFERENTIAL EQUATIONS -  11

MAXIMUM INTEGRATION ERROR -  0.100D-03

t =  0.00
        x     u(x,t)    ua(x,t)
     0.00   0.00000   0.00000
     0.10   0.10000   0.10000
     0.20   0.20000   0.20000
     0.30   0.30000   0.30000
     0.40   0.40000   0.40000
     0.50   0.50000   0.50000
     0.60   0.60000   0.60000
     0.70   0.70000   0.70000
     0.80   0.80000   0.80000
     0.90   0.90000   0.90000
     1.00   1.00000   1.00000

t =  0.10
        x     u(x,t)    ua(x,t)
     0.00   0.00000   0.00000
     0.10   0.09091   0.09091
     0.20   0.18182   0.18182
     0.30   0.27273   0.27273
     0.40   0.36364   0.36364
     0.50   0.45455   0.45455
     0.60   0.54545   0.54545
     0.70   0.63636   0.63636
     0.80   0.72727   0.72727
     0.90   0.81818   0.81818
     1.00   0.90909   0.90909
      .
      .
t =  0.20
      .
      .

t =  0.30
      .
      .
      .

t =  0.80
      .
      .

t =  0.90
```

x	u(x,t)	ua(x,t)
0.00	0.00000	0.00000
0.10	0.05263	0.05263
0.20	0.10526	0.10526
0.30	0.15789	0.15789
0.40	0.21053	0.21053
0.50	0.26316	0.26316
0.60	0.31579	0.31579
0.70	0.36842	0.36842
0.80	0.42105	0.42105
0.90	0.47368	0.47368
1.00	0.52632	0.52632

t = 1.00

x	u(x,t)	ua(x,t)
0.00	0.00000	0.00000
0.10	0.05000	0.05000
0.20	0.10000	0.10000
0.30	0.15000	0.15000
0.40	0.20000	0.20000
0.50	0.25000	0.25000
0.60	0.30000	0.30000
0.70	0.35000	0.35000
0.80	0.40000	0.40000
0.90	0.45000	0.45000
1.00	0.50000	0.50000

RUN NO. - 2 Inviscid Burgers equation, front sharpening

INITIAL T - 0.000D+00

FINAL T - 0.100D+01

PRINT T - 0.100D+00

NUMBER OF DIFFERENTIAL EQUATIONS - 11

MAXIMUM INTEGRATION ERROR - 0.100D-03

t = 0.00

x	u(x,t)	ua(x,t)
0.00	0.99010	0.99010
0.10	0.89109	0.89109
0.20	0.79208	0.79208
0.30	0.69307	0.69307
0.40	0.59406	0.59406
0.50	0.49505	0.49505
0.60	0.39604	0.39604
0.70	0.29703	0.29703
0.80	0.19802	0.19802

```
      0.90    0.09901    0.09901
      1.00    0.00000    0.00000

t =   0.10
         x     u(x,t)    ua(x,t)
      0.00    1.09890    1.09890
      0.10    0.98900    0.98901
      0.20    0.87915    0.87912
      0.30    0.76921    0.76923
      0.40    0.65934    0.65934
      0.50    0.54945    0.54945
      0.60    0.43956    0.43956
      0.70    0.32967    0.32967
      0.80    0.21978    0.21978
      0.90    0.10989    0.10989
      1.00    0.00000    0.00000
       .
       .
t =   0.20
       .
       .

t =   0.30
       .
       .
       .

t =   0.80
       .
       .
t =   0.90
         x     u(x,t)    ua(x,t)
      0.00    9.09091    9.09091
      0.10    8.18169    8.18182
      0.20    7.27312    7.27273
      0.30    6.36346    6.36364
      0.40    5.45462    5.45455
      0.50    4.54549    4.54545
      0.60    3.63640    3.63636
      0.70    2.72731    2.72727
      0.80    1.81822    1.81818
      0.90    0.90911    0.90909
      1.00    0.00000    0.00000

t =   1.00
         x     u(x,t)    ua(x,t)
      0.00  100.00000  100.00000
      0.10   89.99937   90.00000
      0.20   80.00154   80.00000
      0.30   69.99988   70.00000
      0.40   60.00046   60.00000
```

```
0.50   50.00040   50.00000
0.60   40.00043   40.00000
0.70   30.00044   30.00000
0.80   20.00045   20.00000
0.90   10.00042   10.00000
1.00    0.00000    0.00000
```

We can note the following points about this output:

(1) Considering first the output from the first run, the initial condition of eq. (1.15) with a = b = 1, c = 0 (set in INIT1) is

$$u(x,0) = x \tag{1.18}$$

This linear function is apparent in the output at t = 0

```
t =   0.00
        x     u(x,t)    ua(x,t)
    0.00    0.00000    0.00000
    0.10    0.10000    0.10000
    0.20    0.20000    0.20000
    0.30    0.30000    0.30000
    0.40    0.40000    0.40000
    0.50    0.50000    0.50000
    0.60    0.60000    0.60000
    0.70    0.70000    0.70000
    0.80    0.80000    0.80000
    0.90    0.90000    0.90000
    1.00    1.00000    1.00000
```

(checking the initial condition is, of course, always important in developing a dynamic simulation; also, the output of the solution is in accordance with the data file, i.e., the output is at t = 0, 0.1, 0.2,. . ., 1).

(2) The evolution of the solution is in accordance with eq. (1.15)

$$u(x,t) = \frac{x}{1+t} \tag{1.19}$$

Note that at t = 1, the analytical and numerical solutions are still in agreement to five figures, i.e., numerical integration errors did not accumulate significantly as the solution evolved. This agreement is not unexpected since the solution is linear in x (from eq. (1.19)), and therefore the finite difference

approximation of eq. (1.17) is exact. Also, the integration in t by RKF45 surpassed the error tolerance of 0.0001 (RKF45 uses this number in a combined absolute/error tolerance). Of course, we cannot generally expect an accurate numerical solution to a PDE to be computed so easily; for example, the first order approximation of eq. (1.17) will typically give solutions of limited accuracy since the variation of the solution with x will usually depart substantially from a linear relationship.

(3) Continuing on to the solution for the second run, we see again that the numerical solution is in five figure agreement with the analytical solution (with a = 1.01, b = -1 and c = 1 in eq. (1.15))

$$u(x,t) = \frac{1 - x}{1.01 - t} \tag{1.20}$$

Note in particular that the initial condition, $u(x,0) = (1 - x)/1.01$, is again linear in x

t = 0.00

x	u(x,t)	ua(x,t)
0.00	0.99010	0.99010
0.10	0.89109	0.89109
0.20	0.79208	0.79208
0.30	0.69307	0.69307
0.40	0.59406	0.59406
0.50	0.49505	0.49505
0.60	0.39604	0.39604
0.70	0.29703	0.29703
0.80	0.19802	0.19802
0.90	0.09901	0.09901
1.00	0.00000	0.00000

and at t = 1, the solution has increased to a maximum value of 100

t = 1.00

x	u(x,t)	ua(x,t)
0.00	100.00000	100.00000
0.10	89.99937	90.00000
0.20	80.00154	80.00000
0.30	69.99988	70.00000
0.40	60.00046	60.00000
0.50	50.00040	50.00000
0.60	40.00043	40.00000
0.70	30.00044	30.00000
0.80	20.00045	20.00000
0.90	10.00042	10.00000
1.00	0.00000	0.00000

The five figure agreement at t = 1 is particularly interesting since the solution of eq. (1.20) has a singularity at t = 1.01 (from eq. (1.20)).

The solutions for the two runs are plotted in Figures 1.1a and 1.1b; the linear character of these solutions is apparent (which also suggests that good accuracy in the solutions could have been achieved using only two points in the spatial grid in x). Also, the front flattening and front sharpening of the two solutions is clear. In the first solution, the front flattening occurs because *the velocity is relatively small at small values of x*, i.e., the left end is flowing more slowly than the right end. In the second solution, the front sharpens because *the velocity is relatively large at small values of x*, the left end is flowing more rapidly than the right end.

The preceding solutions are not typical of PDE solutions in the sense that they were easily calculated to good accuracy with a small number of spatial grid points. To investigate a more typical PDE problem, we repeat the preceding example, but keep the boundary value, u(0,t), fixed at its initial value, u(0,0). Physically, this might represent an entering velocity which does not change with time, i.e., a constant entering flow rate. The preceding programming is modified in a very minor way to include this revised boundary condition. However, to clarify just how this case was programmed, we now consider subroutines INIT2, DERV2 and PRINT2, which are called from INITAL, DERV and PRINT, respectively, by setting NCASE = 2 in INITAL.

Subroutine INIT2 is listed below

```
      SUBROUTINE INIT2
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NX=101)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /Y/   U(NX)
     +       /F/  UT(NX)
     +       /S/  UX(NX)
     +       /X/   X(NX)
     +       /C/       A,       B,       C,      XL,      DX
     +       /I/       N,   NCASE,      IP,      NT
C...
C...  SELECT THE SOLUTION PARAMETERS
C...
C...      FRONT FLATTENING
          IF(NORUN.EQ.1)THEN
          A=1.0D0
          B=1.0D0
          C=0.0D0
          NT=11
C...
C...      FRONT SHARPENING
```

```
         ELSE
     +   IF(NORUN.EQ.2)THEN
         A= 1.0D0
         B=-1.0D0
         C= 1.0D0
         NT=13
         END IF
C...
C...  NUMBER OF GRID POINTS
      N=101
C...
C...  TOTAL LENGTH IN X, GRID SPACING
      XL=1.0D0
      DX=XL/DFLOAT(N-1)
C...
C...  INITIAL CONDITION
      DO 1 I=1,N
         X(I)=XL*DFLOAT(I-1)/DFLOAT(N-1)
         U(I)=ANAL(X(I),0.0D0)
1     CONTINUE
C...
C...  INITIAL DERIVATIVES
      CALL DERV
      IP=0
      RETURN
      END
```

INIT2 differs from INIT1 in only two ways:

(1) The number of spatial grid points has been increased from 11 to 101 to improve the spatial resolution, which is required because the solution is no longer necessarily linear in x.

(2) The value of a in the second run was changed from 1.01 to 1 and the number of output points, NT, was increased from 11 to 13 (so that PRINT2 is called at $t = 0, 0.1, 0.2, \ldots, 1.2$). Note that this does not cause a problem with the singularity of eq. (1.20) since eq. (1.20) is not the analytical solution to this problem (in fact, the analytical solution is unknown). However, we still use eq. (1.15) to define the initial condition in DO loop 1.

Subroutine DERV2 is listed below

```
      SUBROUTINE DERV2
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NX=101)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /Y/  U(NX)
```

```
     +       /F/ UT(NX)
     +       /S/ UX(NX)
     +       /X/  X(NX)
     +       /C/       A,       B,       C,      XL,      DX
     +       /I/       N,   NCASE,      IP,      NT
C...
C...  BOUNDARY CONDITION AT X = 0
       U(1)=ANAL(0.0D0,0.0D0)
      UT(1)=0.0D0
C...
C...  PDE
      DO 1  I=2,N
         UT(I)=-U(I)*(U(I)-U(I-1))/DX
1     CONTINUE
      RETURN
      END
```

Subroutine DERV2 differs from subroutine DERV1 in only one way: The call to ANAL to define the boundary condition at x = 0 is

```
U(1)=ANAL(0.0D0,0.0D0)
```

rather than

```
U(1)=ANAL(0.0D0,T)
```

Thus, the boundary value, u(0,t), remains at the initial value, u(0,0).

Subroutine PRINT2 is identical to PRINT1 and therefore is not listed. The data file for NCASE = 2 is

```
Inviscid Burgers equation, front flattening, n = 101
0.        1.0       0.1
  101                     0.0001
Inviscid Burgers equation, front sharpening, n = 101
0.        1.2       0.1
  101                     0.0001
END OF RUNS
```

There are two differences in this data file with respect to the NCASE = 1 data file:

(1) The numbers of ODEs is now 101.

(2) For the second solution, t runs to 1.2.

Abbreviated output from these the two runs is listed below

```
RUN NO. -   1  Inviscid Burgers equation, front flattening, n = 101

INITIAL T -  0.000D+00

  FINAL T -  0.100D+01

  PRINT T -  0.100D+00

NUMBER OF DIFFERENTIAL EQUATIONS - 101

MAXIMUM INTEGRATION ERROR -  0.100D-03

t =  0.00
        x     u(x,t)
     0.00   0.00000
     0.01   0.01000
     0.02   0.02000
     0.03   0.03000
     0.04   0.04000
     0.05   0.05000
      .       .
      .       .
      .       .
     0.95   0.95000
     0.96   0.96000
     0.97   0.97000
     0.98   0.98000
     0.99   0.99000
     1.00   1.00000

t =  0.10
        x     u(x,t)
     0.00   0.00000
     0.01   0.00909
     0.02   0.01818
     0.03   0.02727
     0.04   0.03636
     0.05   0.04545
      .       .
      .       .
      .       .
     0.95   0.86364
     0.96   0.87273
     0.97   0.88182
     0.98   0.89091
     0.99   0.90000
     1.00   0.90909
      .
```

```
        .
t =  0.20
        .
        .
        .
t =  0.80
        .
        .
t =  0.90
        .
        .
t =  1.00
          x     u(x,t)
       0.00   0.00000
       0.01   0.00500
       0.02   0.01000
       0.03   0.01500
       0.04   0.02000
       0.05   0.02500
        .       .
        .       .
        .       .
       0.95   0.47500
       0.96   0.48000
       0.97   0.48500
       0.98   0.49000
       0.99   0.49500
       1.00   0.50000


RUN NO. -   2  Inviscid Burgers equation, front sharpening, n = 101

INITIAL T -  0.000D+00

  FINAL T -  0.120D+01

  PRINT T -  0.100D+00

NUMBER OF DIFFERENTIAL EQUATIONS - 101

MAXIMUM INTEGRATION ERROR -  0.100D-03

t =  0.00
          x     u(x,t)
       0.00   1.00000
       0.01   0.99000
       0.02   0.98000
       0.03   0.97000
       0.04   0.96000
       0.05   0.95000
        .
```

```
        .
        .
      0.95    0.05000
      0.96    0.04000
      0.97    0.03000
      0.98    0.02000
      0.99    0.01000
      1.00    0.00000
        .
        .
t =   0.20
        .
        .
        .
t =   1.00
        .
        .
t =   1.10
        .
        .
t =   1.20
         x     u(x,t)
      0.00    1.00000
      0.01    1.00000
      0.02    1.00000
      0.03    1.00000
      0.04    1.00000
      0.05    1.00000
        .       .
      0.95    0.99951
      0.96    0.99935
      0.97    0.99909
      0.98    0.99876
      0.99    0.99833
      1.00    0.00000
```

We can note the following points about the preceding output:

(1) The first solution demonstrates front flattening and remains linear throughout the interval $0 \leq t \leq 1$; these characteristics are apparent in the plot of the solution in Figure 1.2a.

(2) The second solution demonstates front sharpening; by the time it has progressed to $t = 1.2$, the solution decreases from 0.99833 to 0.00000 within an interval $\Delta x = 0.01$. This front sharpening is apparent in the plot of the solution in Figure 1.2b, and indicates the requirement for a finer spatial grid, e.g., 101 grid points rather than 11. The performance of the first order approximation of eq.

(1.17) is surprisingly good considering the sharp spatial variation in the solution; in fact, we cannot usually expect such good results from a low order approximation.

(3) An interesting feature of the second solution is u(1,t) = 0. In other words, the front does not "flow out of the system" at x = 1, as indicated in Figure 1.2b, but rather, merely sharpens at x = 1. The reason for this can be seen in considering DO loop 2 of DERV2

```
C...
C...   PDE
       DO 1 I=2,N
          UT(I)=-U(I)*(U(I)-U(I-1))/DX
1      CONTINUE
```

Note that initially, u(1,0) = 0. Therefore, at t = 0, U(N) = 0 and from DO loop 2, UT(N) = 0. Since the velocity at x = 1 starts at zero, and its temporal derivative therefore is zero, it cannot change with increasing t, i.e., it remains at zero, which is what we observe in the numerical solution. Thus, the case $u(1,0) \neq 0$ should be interesting, and can easily be run by a small change in the initial condition set in INIT2.

To investigate the effect of the order of the approximation for the spatial derivative, $\partial u/\partial x$, in eq. (1.7), we now consider using a *second order, centered approximation*

$$\frac{\partial u(x,t)}{\partial x} \approx \frac{u(x+\Delta x,t) - u(x-\Delta x,t)}{2\Delta x} + O(\Delta x^2) \tag{1.21}$$

Intuitively, it would seem that the centered approximation of eq. (1.21) should produce a better solution than the upwind approximation of eq. (1.17). To investigate this point, we again consider the preceding INITAL, DERV and PRINT, with NCASE = 3, for which INIT3, DERV3 and PRINT3 are called. INIT3 is essentially the same as INIT2, so it is not listed here. The only difference is that the second solution is terminated at t = 1 (i.e., NT = 11 in INIT3) rather than at t = 1.2 (NT = 13 in INIT2); the reason for the shorter run with INIT3 is explained subsequently. DERV3 is listed below

```
      SUBROUTINE DERV3
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NX=101)
      COMMON/T/      T,   NSTOP,   NORUN
     +       /Y/  U(NX)
     +       /F/ UT(NX)
     +       /S/ UX(NX)
     +       /X/  X(NX)
     +       /C/      A,       B,       C,       XL,       DX
```

```
     +       /I/       N,  NCASE,      IP,      NT
C...
C...  BOUNDARY CONDITION AT X = 0
       U(1)=ANAL(0.0D0,0.0D0)
      UT(1)=0.0D0
C...
C...  PDE
C...  X NE 0 OR XL
      DO 1 I=2,N-1
         UT(I)=-U(I)*(U(I+1)-U(I-1))/(2.0D0*DX)
1     CONTINUE
C...
C...  X = XL
      UT(N)=-U(N)*(U(N)-U(N-1))/DX
      RETURN
      END
```

We can note the following points about DERV3:

(1) The boundary condition at x = 0 is the same as in DERV2

```
C...
C...  BOUNDARY CONDITION AT X = 0
       U(1)=ANAL(0.0D0,0.0D0)
      UT(1)=0.0D0
```

(2) The temporal derivative $\partial u(1,t)/\partial t$ is computed with the approximation of eq. (1.17) in order to avoid the use of the fictitious value $u(1+\Delta x,t)$

```
C...
C...  X = XL
      UT(N)=-U(N)*(U(N)-U(N-1))/DX
```

(3) The interior temporal derivatives are computed with the centered approximation of eq. (1.21)

```
C...
C...  PDE
C...  X NE 0 OR XL
      DO 1 I=2,N-1
         UT(I)=-U(I)*(U(I+1)-U(I-1))/(2.0D0*DX)
1     CONTINUE
```

Subroutine PRINT3 is the same as PRINT2, so it is not listed here. The data file for the two solutions computed with DERV3 is listed below

```
Inviscid Burgers equation, front flattening, n = 101
0.        1.0       0.1
  101                     0.0001
Inviscid Burgers equation, front sharpening, n = 101
0.        1.0       0.1
  101                     0.0001
END OF RUNS
```

Note that the only difference between this data file and that for NCASE = 2 is the termination of the second run at t = 1 rather than 1.2.

Abbreviated output from INIT3, DERV3 and PRINT3 is listed below

```
RUN NO. -   1  Inviscid Burgers equation, front flattening, n = 101

INITIAL T -  0.000D+00

  FINAL T -  0.100D+01

  PRINT T -  0.100D+00

NUMBER OF DIFFERENTIAL EQUATIONS - 101

MAXIMUM INTEGRATION ERROR -  0.100D-03

t =  0.00
        x     u(x,t)
     0.00   0.00000
     0.01   0.01000
     0.02   0.02000
     0.03   0.03000
     0.04   0.04000
     0.05   0.05000
      .       .
      .       .
      .       .
     0.95   0.95000
     0.96   0.96000
     0.97   0.97000
     0.98   0.98000
     0.99   0.99000
     1.00   1.00000

t =  0.10
        x     u(x,t)
     0.00   0.00000
     0.01   0.00909
     0.02   0.01818
```

```
      0.03    0.02727
      0.04    0.03636
      0.05    0.04545
       .        .
       .        .
       .        .
      0.95    0.86364
      0.96    0.87273
      0.97    0.88182
      0.98    0.89091
      0.99    0.90000
      1.00    0.90909
       .
       .
t =   0.20
       .
       .
       .
t =   0.80
       .
       .
t =   0.90
       .
       .
t =   1.00
          x     u(x,t)
      0.00    0.00000
      0.01    0.00500
      0.02    0.01000
      0.03    0.01500
      0.04    0.02000
      0.05    0.02500
       .
       .
       .
      0.95    0.47495
      0.96    0.47996
      0.97    0.48504
      0.98    0.49004
      0.99    0.49499
      1.00    0.49998

RUN NO. -   2  Inviscid Burgers equation, front sharpening, n = 101

INITIAL T -  0.000D+00

  FINAL T -  0.100D+01

  PRINT T -  0.100D+00
```

NUMBER OF DIFFERENTIAL EQUATIONS - 101

MAXIMUM INTEGRATION ERROR - 0.100D-03

1

t = 0.00

x	u(x,t)
0.00	1.00000
0.01	0.99000
0.02	0.98000
0.03	0.97000
0.04	0.96000
0.05	0.95000
.	.
.	.
.	.
0.95	0.05000
0.96	0.04000
0.97	0.03000
0.98	0.02000
0.99	0.01000
1.00	0.00000

t = 0.10

x	u(x,t)
0.00	1.00000
0.01	1.00023
0.02	1.00060
0.03	0.99974
0.04	0.99861
0.05	0.99900
.	.
.	.
.	.
0.95	0.05556
0.96	0.04444
0.97	0.03333
0.98	0.02222
0.99	0.01111
1.00	0.00000

.
.

t = 0.20

.
.
.

t = 0.80

.
.

t = 0.90

```
        .
        .
t =   1.00
         x      u(x,t)
      0.00    1.00000
      0.01    1.00000
      0.02    1.00000
      0.03    1.00000
      0.04    1.00000
      0.05    1.00001
        .        .
        .        .
        .        .
      0.95    1.01462
      0.96    1.02332
      0.97    1.03823
      0.98    1.07794
      0.99    1.34886
      1.00    0.00000
```

We can note the following points about this output:

(1) The first solution is essentially the same as with NCASE = 2, which is not unexpected since the solution is linear, and we would therefore expect the second order approximation of eq. (1.21) to perform as well as the first order approximation of eq. (1.17). In fact, the only difference between the NCASE = 2 and NCASE = 3 solutions is due to the mixed use of eq. (1.17) (at $x = 1$) and eq. (1.21) (at $x \neq 1$) in DERV3. The linear character of the first solution is apparent in Figure 1.3a.

(2) The second solution has unrealistic oscillations (e.g., note $u(0.99,1) = 1.34886$), and these oscillations become larger with increasing t (which is the reason for terminating the second solution at $t = 1$; beyond $t = 1$, the oscillations become so large they cannot be plotted). The character of the second solution is evident in Figure 1.3b.

From these results, we arrive at the following conclusions:

(1) *Centered approximations do not work well for the dynamic analysis of strongly convective systems.*

(2) *Some form of upwinding is essential for the dynamic analysis of strongly convective systems.*

Of course, we have not proven either of these conclusions rigorously, but rather, we infer them from the results of the one test problem based on eq. (1.7). More generally, these conclusions have proven to be

correct based on accumulated experience.

Conclusion (2) has an important implication. In using upwinding, the direction of flow must be known (in order to know the upwind direction for the approximation). But, in general, during the analysis of a convective system, we may not know the direction of flow intuitively. Therefore, some test or analytical method must be included in the analysis to indicate the correct direction for upwinding; if this is done incorrectly and downwinding is actually used, the solution will in general be unstable.

Finally, we mention one other generalization based essentially on experience. If a system of PDEs for a strongly convective system is to be integrated numerically, upwinding in all of the PDEs may not be required. For example, if the continuity, momentum and energy equations are to be integrated, upwinding of the continuity and energy equations may be sufficient to compute a solution. The momentum balance can be approximated by centered differences which is an important advantage since the momentum balance in effect provides the fluid velocity (i.e., the direction of flow is not required in approximating the momentum balance, and the fluid velocity that comes from the momentum balance can be used with appropriate upwinding in the continuity and energy balances). Again, we emphasize that there is apparently no proof of this statement, and some analysis to at least support this conclusion would be reassuring.

References

Bird, R. B., W. E. Stewart and E. N. Lightfoot, (1960), Transport Phenomena, John Wiley and Sons, Inc., New York, 1960

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

Schiesser, W. E., (1993), Computational Mathematics in Engineering and Applied Science, CRC Press, Boca Raton, FL

Appendix A: Subroutine PLOTTD

Although Top Drawer is not a part of Fortran, and is generally not available on most computers, subroutine PLOTTD is listed below to indicate how the plots of Figures 1.1a to 1.3b were actually produced and to assist in the conversion to other, more widely available plotting systems, e.g., Matlab, a spreadsheet; this conversion is a straightforward process.

```
      SUBROUTINE PLOTTD
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER(NX=101)
      COMMON/T/       T,   NSTOP,   NORUN
     +       /Y/  U(NX)
     +       /F/ UT(NX)
     +       /S/ UX(NX)
     +       /X/  X(NX)
     +       /C/       A,       B,       C,      XL,      DX
     +       /I/       N,   NCASE,      IP,      NT
C...
C...  OPEN A FILE FOR TOP DRAWER PLOTTING
      IF((IP.EQ.1).AND.(NORUN.EQ.1))THEN
         OPEN(1,FILE='TD.TOP',STATUS='UNKNOWN')
      END IF
C...
C...  SCALE AXES AT THE BEGINNING OF EACH RUN
      IF(IP.EQ.1)THEN
         WRITE(1,1)
1        FORMAT(' SET LIMITS X FROM 0 TO 1 Y FROM -0.1 TO 1.2',/,
     +           ' SET FONT DUPLEX')
      END IF
C...
C...  WRITE THE NUMERICAL SOLUTION FOR SUBSEQUENT TOP DRAWER PLOTTING
      DO 2 I=1,N
         WRITE(1,3)X(I),U(I)
3        FORMAT(F10.3,F10.5)
2     CONTINUE
C...
C...  CONNECT POINTS
         WRITE(1,4)
4        FORMAT(' JOIN 1')
C...
C...  LEGEND, AXES LABELS
      IF(IP.EQ.NT)THEN
         WRITE(1,5)NCASE
5        FORMAT(
     +' TITLE 4.5 9.75 "Case ',I1,': Inviscid Burgers Equation"'
     +,/,' TITLE LEFT "  u(x,t)"'
     +,/,' TITLE BOTTOM "x"')
```

```
C...
C...   NEXT PLOT
       WRITE(1,6)
6      FORMAT(' NEW FRAME')
       END IF
       RETURN
       END
```

We can note the following points about PLOTTD:

(1) The COMMON area is the same as in INIT1, DERV1 and PRINT1 so that the solution to eq. (1.7) can be transferred to PLOTTD for plotting.

(2) At the beginning of the first run, when IP = 1, a file, TD.TOP, is opened to store the numerical solution

```
C...
C...   OPEN A FILE FOR TOP DRAWER PLOTTING
       IF((IP.EQ.1).AND.(NORUN.EQ.1))THEN
          OPEN(1,FILE='TD.TOP',STATUS='UNKNOWN')
       END IF
```

(3) At the beginning of each run, with IP = 1, the axes for the plot are scaled

```
C...
C...   SCALE AXES AT THE BEGINNING OF EACH RUN
       IF(IP.EQ.1)THEN
          WRITE(1,1)
1         FORMAT(' SET LIMITS X FROM 0 TO 1 Y FROM -0.1 TO 1.2',/,
     +           ' SET FONT DUPLEX')
       END IF
```

The Top Drawer commands "SET LIMITS" and "SET FONT" are programmed with Format 1.

(4) During each call to PRINT1 (and therefore PLOTTD), the numerical solution is written to file TD.PLOT

```
C...
C...   WRITE THE NUMERICAL SOLUTION FOR SUBSEQUENT TOP DRAWER PLOTTING
       DO 2 I=1,N
          WRITE(1,3)X(I),U(I)
3         FORMAT(F10.3,F10.5)
2      CONTINUE
```

```
C...
C...   CONNECT POINTS
          WRITE(1,4)
4         FORMAT(' JOIN 1')
```

After the N values of the solution are written by DO loop 2, the Top Drawer command "JOIN" is written to join the points in the plot to produce a curve for each value of t.

(5) At the end of each run, with IP = NT, labels are added to the plot

```
C...
C...   LEGEND, AXES LABELS
       IF(IP.EQ.NT)THEN
          WRITE(1,5)NCASE
5         FORMAT(
     +' TITLE 4.5 9.75 "Case ',I1,': Inviscid Burgers Equation"'
     +,/,' TITLE LEFT "  u(x,t)"'
     +,/,' TITLE BOTTOM "x"')
```

The label includes the value of NCASE.

(6) The plot for each run is now complete, so the Top Drawer command, "NEW FRAME", is used to initiate the next plot

```
C...
C...   NEXT PLOT
       WRITE(1,6)
6      FORMAT(' NEW FRAME')
       END IF
```