An Introduction to the Piecewise Parabolic Method (PPM) Library
II. I/O and Data Analysis

David H. Porter,
Sarah E. Anderson,
Paul R. Woodward,
and B Edgar,

Laboratory of Computational Science or Engineering,
Department of Astronomy, School of Physics and Astronomy
116 Church St. SE.
University of Minnesota
Minneapolis, Minnesota 55455

4. Compressed Dumps

a. Introduction

Obviously, the manner in which the results are sampled in the above examples (`dump`) is inadequate in general, as it is usually not possible to predict which quantities will be most useful to study. Therefore, a means is provided in PPMLIB to produce more detailed data samples. These samples are referred to as *compressed dumps* because the two most significant bytes of each number is retained. This reduces the storage requirements thus encouraging a greater frequency of sampling. These compressed dumps are matched to the PPMLIB analysis program **a3d** which will be described later. The user is permitted to include a short description of each quantity and assign a *keyword*. It is with hese keywords that **a3d** will reference the related stored quantities.

This is perhaps best illustrated by two examples which create synthetic data similar to that found in a typical hydrodynamic calculation. It should be noted that as with restart dumps, it is up to the user to decide which quantities must be retained for future examination. *When in doubt, write it out!*

```
c
define(IQX,59)
define(IQY,61)
define(NBDY,9)
define(NVAR,5)
define(MAXCOMMENTS,6)

      program cdumps2dexample
c
c   WHAT:  This program is designed to test and illustrate the PPMLIB
c          compressed dump routines suitable to two-dimensional data sets.
c          It constructs a data set similar to that found in a typical
c          PPMLIB hydrodynamic calculation.  That is:
c       a)  A picture frame of fake or boundary zones is appended to all
c           sides of the data.
c       b)  The computational grid is defined by the locations of the lefthand
c           zone interfaces, stored in 1-d arrays XINTRL AND YINTRL.  In
c           order to fully define the computational grid using lefthand zone
c           interfaces, one additional interface is required on the right of
c           these arrays.  They are therefore dimensioned one greater on the
c           right than would otherwise be expected.
c       c)  The two-dimensional arrays are stored in interleaved order in a
c           three-dimensional array DDD.  This interleaved ordering provides
c           for more efficient memory access on cache based machines.
c           It is assumed that the ordering of the values in DDD is: density
c           pressure, x-velocity, y-velocity, diagnostic.  Thus the array
c           elements for zone DDD(1,i,j),   DDD(2,i,j),  DDD(3,i,j),
c           DDD(4,i,j),  and DDD(5,i,j) refer to the values of density,
```

```
c           pressure, x-velocity, y-velocity, and the diagnostic for zone
c           (i,j), respectively.  This specific ordering is arbitrary and
c           determined by the programmer.
c
c   SPECIAL COMMENTS:  It is assumed that this file will be preprocessed using
c          the m4 macro processor with the option -Difdouble=0 for single
c          precision or -Difdouble=1 for double precision.  That is:
c     single precision:  m4 -Difdouble=0 cdumps2dexample.m4 > cdumps2dexample.f
c     double precision:  m4 -Difdouble=1 cdumps2dexample.m4 > cdumps2dexample.f


      ifelse(ifdouble,1,implicit real*8 (a-h,o-z), )
      parameter(nxzones=IQX)
      parameter(nyzones=IQY)
      parameter(nbdy=NBDY)
      parameter(nvar=NVAR)
      parameter(maxcomments=MAXCOMMENTS)
c
c   These are 1d arrays which contain information defining the grid,
c      that is, the locations of the left hand zone interfaces.  One
c      extra zone is required on the right to fully specify the grid.

      dimension xintrl(1-nbdy:nxzones+nbdy+1)
      dimension yintrl(1-nbdy:nyzones+nbdy+1)

c
      character*20 filename
c
c   This will hold some fascinating commentary describing the problem for
c      inclusion in the compressed dump file.
      character*80 comments(maxcomments)
c
c   This array contains zone averaged information of the state variables
c     in interleaved order, making it a 3d array for two spatial dimensions
c     They are:
c        ddd(1,*,*) = density
c        ddd(2,*,*) = pressure
c        ddd(3,*,*) = x-velocity
c        ddd(4,*,*) = y-velocity
c        ddd(5,*,*) = diagnostic (an arbitrary quantity)
      dimension ddd(nvar,1-nbdy:nxzones+nbdy,
     ^                   1-nbdy:nyzones+nbdy)

c
c   Initialize the disk file name
      filename = 'q42a0000'
c
c   Initialize the commentary
      comments(1)='  This is the result from the program cdumps2dexample'
      comments(2)='  Its purpose is to demonstrate the PPMLIB compressed'
      comments(3)='  dumps routines. The arrays are REAL*ifelse(ifdouble,1,8,4)'
      ncomments = 3

c
c   Initialize the grid
      call init_grid2d(xintrl, yintrl, nxzones, nyzones, nbdy)
c
c   Initialize the zone average array
      call init_ddd(ddd,nvar,nxzones,nyzones,nbdy)

c
c   Open the file, write the comments including the date of file
c     creation.
      iunit = 66
      call ppm98_binopen(filename,iunit)
      do i=1,ncomments
         call ppm98_write_comment(comments(i),iunit)
      enddo
      call ppm98_timestamp(iunit)
c
c   This illustrates adding a constant value to the compressed dump file
      gamma= 1.4ifelse(ifdouble,1,d,e)+00
      call ifelse(ifdouble,1,d_,)ppm98_var0d('ratio of specific heats',
     ^              'GAMMA',gamma,iunit)

c
c   Now record the grid information in the compressed dump routine.  The
c     two axes will be assigned the names 'X', and 'Y'.  These
c     must be capitol letters; their derivatives will be referenced
c     with the corresponding lower case letters by the analysis routine a3d.
      call ifelse(ifdouble,1,d_,)ppm98_mesho2
     ^                           ('X', 'Y', nbdy, nxzones, nbdy,
     ^                            nbdy, nyzones, nbdy,
     ^                            xintrl, yintrl, iunit )

c
c   Now add the individual zone average quantities to the compressed dump.
      call ifelse(ifdouble,1,d_,) ppm98_var2d
     ^                      ('Density', 'RHO', nbdy, nxzones, nbdy,
     ^                                         nbdy, nyzones, nbdy,
     ^                                    nvar, 1, ddd, 0, iunit)
      call ifelse(ifdouble,1,d_,) ppm98_var2d
     ^                          ('Pressure', 'P', nbdy, nxzones, nbdy,
     ^                                        nbdy, nyzones, nbdy,
     ^                                    nvar, 2, ddd, 0, iunit)
      call ifelse(ifdouble,1,d_,) ppm98_var2d
     ^                          ('X velocity', 'VX', nbdy, nxzones, nbdy,
     ^                                          nbdy, nyzones, nbdy,
```

```
     ^                                    nvar, 3, ddd, 0, iunit)
      call ifelse(ifdouble,1,d_,) ppm98_var2d
     ^                          ('Y velocity', 'VY', nbdy, nxzones, nbdy,
     ^                                           nbdy, nyzones, nbdy,
     ^                                    nvar, 4, ddd, 0, iunit)
      call ifelse(ifdouble,1,d_,) ppm98_var2d
     ^                          ('2D chess board diagnostic', 'Diag',
     ^                                           nbdy, nxzones, nbdy,
     ^                                           nbdy, nyzones, nbdy,
     ^                                    nvar, 5, ddd, 0, iunit)

c   Close the file and exit
      call ppm98_binclose(iunit)


      end  ! program test2d

      subroutine init_ddd(ddd,nvar,nx,ny,nbdy)
      ifelse(ifdouble,1,implicit real*8 (a-h,o-z), )
      dimension ddd(nvar, 1-nbdy:nx+nbdy,
     ^                    1-nbdy:ny+nbdy)
c
c    ddd(1,*,*) = density
c    ddd(2,*,*) = pressure
c    ddd(3,*,*) = x-velocity
c    ddd(4,*,*) = y-velocity
c    ddd(5,*,*) = a diagnostic
c
c   The domain will be divided into quadrants for density and pressure
c
c   The velocities shall consist of a sine wave
      twopi =  2.0*atan2(0.,-1.)
      xfac = twopi/float(nx)
      yfac = twopi/float(ny)
c
c   The diagnostic consists of alternating squares of 0.,1. in
c     a chess board pattern.
      lcube = 16
c
      ycube = 0.0
      do iy=1-nbdy,ny+nbdy
       if (iy.le.ny/2) then
          y = 1.0
       else
          y = 2.0
       endif
       yvel = sin(float(iy)*yfac)
       if (mod(iy,lcube) .eq. 0) ycube = 1.0 - ycube
       xcube = ycube
          do ix=1-nbdy,nx+nbdy
          if (ix.le.nx/2) then
             x = 3.0
          else
             x = 4.0
          endif
          xvel = sin(float(ix)*xfac)
          if (mod(ix,lcube) .eq. 0) xcube = 1.0 - xcube
             ddd(1,ix,iy) = x*y
             ddd(2,ix,iy) = x*y
             ddd(3,ix,iy) = xvel
             ddd(4,ix,iy) = yvel
             ddd(5,ix,iy) = xcube
          enddo
      enddo

      return
      end
      subroutine init_grid2d(x1intrl,x2intrl,n1zones,n2zones,nbdy)
      ifelse(ifdouble,1,implicit real*8 (a-h,o-z), )
      dimension x1intrl(1-nbdy:n1zones+nbdy+1)
      dimension x2intrl(1-nbdy:n2zones+nbdy+1)

c   Set up a uniform 2d grid where the non boundary portion
c      ranges from 0 to 1.

      dx = 1.0 / float(n1zones)
      do i=1-nbdy,n1zones+nbdy+1
        x1intrl(i) = float(i-1)*dx
      enddo

      dx = 1.0 / float(n2zones)
      do i=1-nbdy,n2zones+nbdy+1
        x2intrl(i) = float(i-1)*dx
      enddo

      return
      end
```

This first example, *cdumps2dexample* contained in the file ‘‘cdumps2dexample.m4’’ found at the PPMLIB website, is an example of how two dimensional data may be stored, and is intended to be processed by the m4 macro before compiling. It is designed to create source code for either single precision (real*4) or double precision (real*8) through the use of an m4 command line argument. To obtain a single precision source code version this file should be processed thusly:

m4 -Difdouble=0 cdumps2dexample.m4 > cdumps2dexample.f.

Likewise, double precision source code is magically created by

m4 -Difdouble=1 cdumps2dexample.m4 > cdumps2dexample.f.

In the discussion following, attention will be restricted to single precision. The changes produced by double precision are straightforward to ferret out, and are left as an exercise for the unfortunate reader.

The numerical grid is defined by specifying the locations of the lefthand zone interfaces of the grid. The grid is assumed regular so that only two one-dimensional arrays `xintrl` and `yintrl` for the x-axis and y-axis respectively, are required. It should be noted that in order to fully specify the grid one additional array element is required on the right of each of the grid arrays. Also, provision is made for a picture frame of "fake" or boundary zones to be appended to the computational domain in order to permit the imposition of boundary conditions. The exact number of such zones is problem dependent (and may be zero.) The two dimensional data consist of zone averages of density, pressure, x-velocity, y-velocity and a diagnostic (an arbitrary quantity.) These are stored in a three dimensional array `DDD` in interleaved order. That is, the zone averages for zone (i,j) of density, pressure, x-velocity, y-velocity, and the diagnostic are `DDD(1,i,j)`, `DDD(2,i,j)`, `DDD(3,i,j)`, `DDD(4,i,j)`, and `DDD(5,i,j)`, respectively. This interleaving affords greater efficiencies in memory accessing on cache based machines. A commentary array is declared with the dimensions " maximum number of comments" by "maximum comment length." This affords the user (should he wish it) some poetic license in an otherwise blandly prosaic chore. The name of the compressed dump to be created is `q42a0000`. This follows an arbitrary convention of a three character identifier specific to the individual calculation (here "q42"), a four character numeric sequence (here "0000") to identify the specific dump within the calculation, and a one character identifier used to identify the type of file (here "a"). It is recommended that the user follow this convention as the analysis routine **a3d** replaces the one character type identifier in order to create a unique name for the image files it may create. For example, the user may instruct **a3d** to construct a file name "q42density0000" by replacing the "a" with the string "density."

This example program initializes the commentary explicitly. The grid arrays `xintrl` and `yintrl` are initialized by a call to the subroutine `init_grid2d` which creates a uniform grid ranging from zero to one in both directions. The interleaved array `ddd` is initialized by a call to the subroutine `init_ddd`. The initial state consists of a sine wave in both velocities. For the density and pressure values, the computational domain is divided into quadrants, with each quadrant having a unique, constant value. The diagnostic quantity is set to an alternating chess board pattern.

The program then proceeds to open the file "q42a0000" on disk with a call to **ppm98_binopen_trunc** which has two arguments. The first is the character string containing the filename. The second is an integer which specifies the FORTRAN unit number which is to be associated with this open file. Next, the user *must* specify an integer value, and it must not conflict with other open unit numbers. The desired commentary is written to the file by successive calls to **ppm98_write_comment** which requires two arguments, the first of which is the character string to be written and the second the open FORTRAN unit number associated with the diskfile. Commentary is optional, it is purely for the edification and amusement of the user and his progeny. This is followed by a call to **ppm98_timestamp** which includes, in the form of a compressed dump comment, the date and time of file creation (or more exactly, the date and time of the call to this routine). This requires one argument, the value of the open FORTRAN unit number associated with the diskfile. This is also optional and is no more accurate than the system clock of the computer on which the program runs.

Scalar values may be written to the compressed dump using the subroutine **ppm98_var0d**.

```
      call ppm98_var0d('ratio of specific heats',
     ^                'GAMMA',gamma,iunit)
```

```
          subroutine ppm98_var0d( comment, a3dsymbol, value, iunit )

          character(LEN=*)      COMMENT
          character(LEN=*)      A3DSYMBOL
          real*4     VALUE
          integer*4 IUNIT
```

The first argument is a FORTRAN character string in which the user may describe the quantity. The second is a FORTRAN character string which the utility **a3d** will use to reference this value. This must begin with an alphabetic and must be space-free. It is also case sensitive. This is followed by the values of the scaler and the value of the open FORTRAN unit number.

Next, the numerical grid information is included by a call to the subroutine **ppm98_mesho2**.

```
      call ifelse(ifdouble,1,d_,)ppm98_mesho2
     ^                           ('X', 'Y', nbdy, nxzones, nbdy,
     ^                            nbdy, nyzones, nbdy,
     ^                            xintrl, yintrl, iunit )
c
      subroutine ppm98_mesho2( s1, s2, nbdy_1l, n1, nbdy_1r,
     ^                   nbdy_2l, n2, nbdy_2r, sntrl1, sntrl2,
     ^                     iunit )
      character*1 s1, s2
      integer nbdy_1l, n1, nbdy_1r, nbdy_2l, n2, nbdy_2r
      dimension sntrl1(-nbdy_1l+1:n1+nbdy_1r+1)
      dimension sntrl2(-nbdy_2l+1:n2+nbdy_2r+1)
```

The first two arguments are one character alphabetic strings by which **a3d** will reference the individual axes. These must be unique capitol letters. Their lower case equivalents will be used to indicate derivatives with respect to that axis. Here, the two axes are designated ‘‘X’’ and ‘‘Y’’ hence in **a3d** the operation *d/dx* is ‘‘x’’ and the operation *d/dy* is ‘‘y.’’ Next follow the dimensions of the two one dimensional arrays defining the numerical grid. Their dimensions are declared such that there is a provision for an asymmetric number of boundary zones. The last argument is, of course, the value of the open FORTRAN unit number. This call is mandatory if **a3d** is to be used.

The two dimensional arrays are written by successive calls to **ppm98_var2d**. For example,

```
      call var2d ('Density', 'RHO', nbdy, nxzones, nbdy,
     ^                                      nbdy, nyzones, nbdy,
     ^                                 nvar, 1, ddd, 0, iunit)
c
          subroutine ppm98_var2d( comment, a3dsymbol, nbdy_xl, nx,
          nbdy_xr, nbdy_yl, ny,n nbdy_yr, nvar, iovar, array, ifloge,
          iunit )
          character(LEN=*)     COMMENT
          character(LEN=*)     A3DSYMBOL
          dimension     array( 1:NVAR, 1-NBDY_XL:NX+NBDY_XR, 1-NBDY_YL:NY+NBDY_YR )
          integer*4 NBDY_XL, NX, NBDY_XR
          integer*4 NBDY_YL, NY, NBDY_YR
          integer*4 IFLOGE
          integer*4 IUNIT
```

The first argument is a FORTRAN character string in which the user may use to describe the quantity. The second is a FORTRAN character string which the utility **a3d** will use to reference this value. As before, this must begin with an alphabetic character and must not contain spaces and is case sensetive. Next follow closely the dimensions of the interleaved array, where `NVAR` is the total number of individual arrays interleaved within the `array`. The value of `IOVAR` specifies which of the interleaved arrays is to be written to the compressed dump file. If the the array consists of only one two-dimensional array (no interleaving) then both `NVAR` and `IOVAR` are set equal to one. The value of `IFLOGE` serves as a flag to indicate whether the values of the array should be stored or their natural logarithms (`IFLOGE` = 1). The final argument is the value of the open FORTRAN unit number. After all of the desired two-dimensional arrays have been written, the file is closed by a call to **ppm98_binclose**.

The exact ordering of the calls is unimportant. However, it should be noted that since the commentary, mesh information and scalar values are stored as ASCII text, it could prove advantageous to write such quantities first, as in this example, before any two-dimension arrays, which are written as binary data. Then, if the user has been sufficently cleaver with his commentary, the usual UNIX file utilities such as ‘‘more’’ or ‘‘less’’ may be used to examine and document the contents of the file. For example, using ‘‘more’’ on the file ‘‘p14a0000’’ created by running this example produces the following output:

```
COMMENT="  This is the result from the program cdumps2dexample"
COMMENT="  Its purpose is to demonstrate the PPMLIB compressed"
```

```
COMMENT="  dumps routines. The arrays are REAL*4"
COMMENT="File created 13Apr1999 at 13:47:24 Local Time ( -300 minutes from UT)"
NAME0D="ratio of specific heats"   SYMBOL0D="GAMMA"   VALUE0D="   1.400000E+00"
COORD1="X"    COORD2="A"    COORD3="Y"
MESH1="   59"    MESH2="    1"    MESH3="   61"

INTRL3="   993"
  0.0000000E+00
  1.6393442E-02
  3.2786883E-02
  4.9180325E-02
  6.5573767E-02
  8.1967205E-02
  9.8360650E-02
  1.1475410E-01
  1.3114753E-01
  1.4754097E-01
  1.6393441E-01
  1.8032786E-01
  1.9672130E-01
  2.1311474E-01
  2.2950819E-01
  2.4590163E-01
  2.6229507E-01
  2.7868852E-01
  2.9508194E-01
  3.1147540E-01
  3.2786882E-01
  3.4426227E-01
```

The attentive reader will have noted that a third axis (‘‘A’’) has been included which has one zone. This is for the use of **a3d** which assumes three dimensions. This additional axis has no affect in two dimensions. (A unique identifier will be automatically assigned to this axis.)

The next example, *cdumps3dexample* contained in the file ‘‘cdumps3dexample.m4’’ also found at the PPMLIB website is an example of how three dimensional data may be stored. It is intented to be processed by the m4 macro before compiling. Source code for either single precision (real*4) or double precision (real*8) may be created by the use of an m4 command line argument. To obtain a single precision source code version, this file should be processed as:

m4 -Difdouble=0 cdumps3dexample.m4 > cdumps3dexample.f.

Likewise, double precision source code is created by

m4 -Difdouble=1 cdumps3dexample.m4 > cdumps3dexample.f.

The discussion following will be restricted to single precision for clarity. The changes produced by double precision are straightforward.

```
c
c
define(IQX,59)
define(IQY,61)
define(IQZ,67)
define(NBDY,9)
define(NVAR,6)
define(MAXCOMMENTS,6)

      program cdumps3dexample
c
c   WHAT:  This program is designed to test and illustrate the PPMLIB
c          compressed dump routines suitable to three-dimensional data sets.
c          It constructs a data set similar to that found in a typical
c          PPMLIB hydrodynamic calculation.  That is:
c       a)  A skin of fake or boundary zones is appended to all
c           sides of the data.
c       b)  The computational grid is defined by the locations of the lefthand
c           zone interfaces, stored in 1-d arrays XINTRL YINTRL, AND ZINTRL.
c           In order to fully define the computational grid using lefthand zone
c           interfaces, one additional interface is required on the right of
c           these arrays.  They are therefore dimensioned one greater on the
c           right than would otherwise be expected.
c       c)  The three-dimensional arrays are stored in interleaved order in a
c           four-dimensional array DDD.  This interleaved ordering provides
c           for more efficient memory access on cache based machines.
c           It is assumed that the ordering of the values in DDD is: density
c           pressure, x-velocity, y-velocity, z-velocity, diagnostic.  Thus
c           the array elements for zone DDD(1,i,j),   DDD(2,i,j),  DDD(3,i,j),
c           DDD(4,i,j),  DDD(5,i,j), and DDD(6,i,j) refer to the values of
c           density, pressure, x-velocity, y-velocity, z-velocity, and the
c           diagnostic for zone (i,j), respectively.  This specific ordering
c            is arbitrary and determined by the programmer.
c
c   SPECIAL COMMENTS:  It is assumed that this file will be preprocessed using
c          the m4 macro processor with the option -Difdouble=0 for single
c          precision or -Difdouble=1 for double precision.  That is:
c     single precision:  m4 -Difdouble=0 cdumps3dexample.m4 > cdumps3dexample.f
c     double precision:  m4 -Difdouble=1 cdumps3dexample.m4 > cdumps3dexample.f
```

```
      ifelse(ifdouble,1,implicit real*8 (a-h,o-z), )
c     ifelse(ifdouble,1,8,4)
      parameter(nxzones=IQX)
      parameter(nyzones=IQY)
      parameter(nzzones=IQZ)
      parameter(nbdy=NBDY)
      parameter(nvar=NVAR)
      parameter(maxcomments=MAXCOMMENTS)
c
c   These are 1d arrays which contain information defining the grid,
c      that is, the locations of the left hand zone interfaces.  One
c      extra zone is requires on the left to fully specify the grid.

      dimension xintrl(1-nbdy:nxzones+nbdy+1)
      dimension yintrl(1-nbdy:nyzones+nbdy+1)
      dimension zintrl(1-nbdy:nzzones+nbdy+1)

c
      character*20 filename
c
c   This will hold some fascinating commentary describing the problem
c      for inclusion in the compressed dump file.
      character*80 comments(maxcomments)
c
c   This array contains zone averaged information of the state variables
c     in interleaved order, making it a 4d array for three spatial dimensions
c     They are:
c        ddd(1,*,*,*) = density
c        ddd(2,*,*,*) = pressure
c        ddd(3,*,*,*) = x-velocity
c        ddd(4,*,*,*) = y-velocity
c        ddd(5,*,*,*) = z-velocity
c        ddd(6,*,*,*) = diagnostic (an arbitrary quantity)
      dimension ddd(nvar,1-nbdy:nxzones+nbdy,
     ^                   1-nbdy:nyzones+nbdy,
     ^                   1-nbdy:nzzones+nbdy)

c
c   Initialize the disk file name
      filename = 'p14a0000'
c
c   Initialize the commentary
      comments(1)='  This is the result from the program cdumps3dexample'
      comments(2)='  Its purpose is to demonstrate the PPMLIB compressed'
      comments(3)='  dumps routines. The arrays are REAL*ifelse(ifdouble,1,8,4)'
      ncomments = 3

c
c   Initialize the grid
      call init_grid(xintrl, yintrl, zintrl, nxzones, nyzones, nzzones, nbdy)
c
c   Initialize the zone average grid
      call init_ddd(ddd,nvar,nxzones,nyzones,nzzones,nbdy)

c
c   Open the file, write the comments including the date of file
c     creation.
      iunit = 66
      call ppm98_binopen_trunc(filename,iunit)
      do i=1,ncomments
         call ppm98_write_comment(comments(i),iunit)
      enddo
      call ppm98_timestamp(iunit)
c   This illustrates adding a constant value to the compressed dump file
      gamma= 1.4ifelse(ifdouble,1,d,e)+00
      call ifelse(ifdouble,1,d_,)ppm98_var0d('ratio of specific heats',
     ^               'GAMMA',gamma,iunit)

c
c   Now record the grid information in the compressed dump routine.  The
c     three axes will be assigned the names 'X', 'Y', and 'Z'.  These
c     should be capitol letters, as their derivatives will be references
c     with the equivalent lower case letters in a3d.
      call  ifelse(ifdouble,1,d_,)ppm98_mesho3('X', 'Y', 'Z', nbdy, nxzones, nbdy,
     ^                                 nbdy, nyzones, nbdy,
     ^                                 nbdy, nzzones, nbdy,
     ^                                 xintrl, yintrl, zintrl, iunit )

c
c   Now add the individual zone average quantities to the compressed dump
            call  ifelse(ifdouble,1,d_,)ppm98_var3d('Density', 'RHO',
     ^                                        nbdy, nxzones, nbdy,
     ^                                        nbdy, nyzones, nbdy,
     ^                                        nbdy, nzzones, nbdy,
     ^                                   nvar, 1, ddd, 0, iunit)
            call ifelse(ifdouble,1,d_,)ppm98_var3d('Pressure', 'P',
     ^                                       nbdy, nxzones, nbdy,
     ^                                       nbdy, nyzones, nbdy,
     ^                                       nbdy, nzzones, nbdy,
     ^                                   nvar, 2, ddd, 0, iunit)
            call ifelse(ifdouble,1,d_,)ppm98_var3d('X velocity', 'VX',
     ^                                          nbdy, nxzones, nbdy,
     ^                                          nbdy, nyzones, nbdy,
     ^                                          nbdy, nzzones, nbdy,
     ^                                   nvar, 3, ddd, 0, iunit)
            call ifelse(ifdouble,1,d_,)ppm98_var3d('Y velocity', 'VY',
     ^                                          nbdy, nxzones, nbdy,
     ^                                          nbdy, nyzones, nbdy,
```

```
     ^                                             nbdy, nzzones, nbdy,
     ^                                      nvar, 4, ddd, 0, iunit)
            call ifelse(ifdouble,1,d_,)ppm98_var3d('Z velocity', 'VZ',
     ^                                             nbdy, nxzones, nbdy,
     ^                                             nbdy, nyzones, nbdy,
     ^                                             nbdy, nzzones, nbdy,
     ^                                      nvar, 5, ddd, 0, iunit)
            call ifelse(ifdouble,1,d_,)ppm98_var3d(
     ^                           '3D chess board diagnostic', 'DIAG',
     ^                                             nbdy, nxzones, nbdy,
     ^                                             nbdy, nyzones, nbdy,
     ^                                             nbdy, nzzones, nbdy,
     ^                                      nvar, 6, ddd, 0, iunit)

c   Close the file and exit
      call ppm98_binclose(iunit)


      end

      subroutine init_ddd(ddd,nvar,nx,ny,nz,nbdy)
      ifelse(ifdouble,1,implicit real*8 (a-h,o-z), )
      dimension ddd(nvar, 1-nbdy:nx+nbdy,
     ^                    1-nbdy:ny+nbdy,
     ^                    1-nbdy:nz+nbdy)
c
c    ddd(1,*,*,*) = density
c    ddd(2,*,*,*) = pressure
c    ddd(3,*,*,*) = x-velocity
c    ddd(4,*,*,*) = y-velocity
c    ddd(5,*,*,*) = z-velocity
c    ddd(6,*,*,*) = a diagnostic
c
c   The domain will be divided into octants for density and pressure
c
c   The velocities shall consist of a sine wave
      twopi =  2.0*atan2(0.,-1.)
      xfac = twopi/float(nx)
      yfac = twopi/float(ny)
      zfac = twopi/float(nz)
c
c   The diagnostic shall consist of alternating cubes of 0.,1. in
c     a 3d chess board pattern.
      lcube = 16
c
      xcube = 0.0
      do iz=1-nbdy,nz+nbdy
       if (iz.le.nz/2) then
          z = 1.0
       else
          z = 2.0
       endif
       zvel = sin(float(iz)*zfac)
       if (mod(iz,lcube) .eq. 0) xcube = 1.0 - xcube
       ycube = xcube
          do iy=1-nbdy,ny+nbdy
          if (iy.le.ny/2) then
             y = 3.0
          else
             y = 4.0
          endif
          yvel = sin(float(iy)*yfac)
          if (mod(iy,lcube) .eq. 0) ycube = 1.0 - ycube
          zcube = ycube
             do ix=1-nbdy,nx+nbdy
             if (ix.le.nx/2) then
                x = 5.0
             else
                x = 6.0
             endif
             xvel = sin(float(ix)*xfac)
             if (mod(ix,lcube) .eq. 0) zcube = 1.0 - zcube
             ddd(1,ix,iy,iz) = x*y*z
             ddd(2,ix,iy,iz) = x*y*z
             ddd(3,ix,iy,iz) = xvel
             ddd(4,ix,iy,iz) = yvel
             ddd(5,ix,iy,iz) = zvel
             ddd(6,ix,iy,iz) = zcube
             enddo
          enddo
      enddo

       return
      end
      subroutine init_grid(x1intrl,x2intrl,x3intrl,n1zones,n2zones,n3zones,nbdy)
      ifelse(ifdouble,1,implicit real*8 (a-h,o-z), )
      dimension x1intrl(1-nbdy:n1zones+nbdy+1)
      dimension x2intrl(1-nbdy:n2zones+nbdy+1)
      dimension x3intrl(1-nbdy:n3zones+nbdy+1)

c   Set up a uniform 3d grid where the non boundary portion
c      ranges from 0 to 1.

      dx = 1.0 / float(n1zones)
      do i=1-nbdy,n1zones+nbdy+1
        x1intrl(i) = float(i-1)*dx
      enddo
```

```
      dx = 1.0 / float(n2zones)
      do i=1-nbdy,n2zones+nbdy+1
        x2intrl(i) = float(i-1)*dx
      enddo

      dx = 1.0 / float(n3zones)
      do i=1-nbdy,n3zones+nbdy+1
        x3intrl(i) = float(i-1)*dx
      enddo

      return
      end
```

This program is the obvious extension to three dimensions of the previous two dimensional example. The numerical grid is defined by specifying the locations of the lefthand zone interfaces of the grid, with three dimension requiring three one-dimensional arrays `xintrl`, `yintrl`, and `zintrl` for the x-axis, y-axis, and z-axis respectively, are now required for the regular computational grid. It should be noted that in order to fully specify the grid one additional array element is required on the right of each of the grid arrays. Provision is made for a skin of “fake” or boundary zones to be appended to the computational domain in order to allow for the imposition of boundary conditions. The exact number of such zones is problem dependent (and may be zero.) The three dimensional data consist of zone averages of density, pressure, x-velocity, y-velocity, z-velocity, and a diagnostic (an arbitrary quantity.) These are stored in a four dimensional array `DDD` in interleaved order. That is, the zone averages for zone (i,j) of density, pressure, x-velocity, y-velocity, z-velocity, and the diagnostic are `DDD(1,i,j)`, `DDD(2,i,j)`, `DDD(3,i,j)`, `DDD(4,i,j)`, `DDD(5,i,j)`, and `DDD(6,i,j)`, respectively. This interleaving affords greater efficiencies in memory accessing on cache based machines. A commentary array is declared with the dimensions “ maximum number of comments” by “maximum comment length” which affords the user (should he wish it) the ability to provide documentation and extended explanations of hte contents for future reference. The name of the compressed dump to be created is `P14a0000`, following the same arbitrary convention of a three character identifier specific to the individual calculation (here “p14”), a four character numeric sequence (here “0000”) to identify the specific dump within the calculation, and a one character identifier used to identify the type of file (here “a”) introduced in the two dimensional example. This convention is recommended so that the analysis routine **a3d** can construct unique file names by replacing the one character type identifier with a user specified string. For example, the user may instruct **a3d** to construct a file name “p14density0000” by replacing the “a” with the string “density.”

As in the two dimensional example, the commentary is initialized explicitly. The grid arrays `xintrl`, `xintrl`, and `yintrl` are initialized to a uniform grid ranging from zero to one in all three directions. The interleaved array `ddd` is set to an initial state in which each velocity consists of a sine wave. For the density and pressure values, the computational domain is divided into octants, with each octant having a unique, constant value. The diagnostic quantity is set to an alternating three dimensional chess board-like pattern.

The program then proceeds to open the file “p14a0000” on disk with a call to **ppm98_binopen_trunc** described earlier. As in the two dimensional example, the desired commentary is written to the file by successive calls to **ppm98_write_comment**. The inclusion of commentary in the compressed dump file is optional. This is followed by an optional call to **ppm98_timestamp** which includes, in the form of a compressed dump comment, the date and time of file creation (or more exactly, the date and time of the call to this routine).

The inclusion of scalar values using the subroutine **ppm98_var0d** was described above.

In three dimensions, the numerical grid information is included by a call to the subroutine **ppm98_mesho3**.

```
      call  ifelse(ifdouble,1,d_,)ppm98_mesho3('X', 'Y', 'Z', nbdy, nxzones, nbdy,
     ^                                  nbdy, nyzones, nbdy,
     ^                                  nbdy, nzzones, nbdy,
     ^                                  xintrl, yintrl, zintrl, iunit )
```

```
c
          subroutine ppm98_mesho3( sx, sy, sz, nbdy_xl, nx, nbdy_xr,
          nbdy_yl, ny, nbdy_yr, nbdy_zl, nz, nbdy_zr, xintrl, yintrl,
          zintrl, iunit )

          character*1     SX, SY, SZ
          dimension      XINTRL( 1-NBDY_XL:NX+NBDY_XR+1 )
          dimension      YINTRL( 1-NBDY_YL:NY+NBDY_YR+1 )
          dimension      ZINTRL( 1-NBDY_ZL:NZ+NBDY_ZR+1 )
          integer*4 NBDY_XL, NX, NBDY_XR
```

```
          integer*4 NBDY_YL, NY, NBDY_YR
```

The first three arguments are one character alphabetic strings by which **a3d** will reference the individual axes. These are chosen by the user and *must* be unique capitol letters. Their lower case quivalents will be used to indicate derivatives with respect to that axis. Here, the three axes are designated “X,” “Y,” and “Z” hence in **a3d** the operation $\frac{d}{dx}$ is “x” $\frac{d}{dy}$ is “y” and $\frac{d}{dz}$ is “z.” Next follow the dimensions of the three one dimensional arrays defining the numerical grid. Their dimensions are declared such that there is a provision for an asymmetric number of boundary zones. The last argument is, of course, the value of the open FORTRAN unit number. This call is mandatory if **a3d** is to be used.

The three dimensional arrays are written by successive calls to **ppm98_var3d**. For example,

```
            call ppm98_var3d('Density', 'RHO',
     ^                   nbdy, nxzones, nbdy,
     ^                   nbdy, nyzones, nbdy,
     ^                   nbdy, nzzones, nbdy,
     ^                   nvar, 1, ddd, 0, iunit)
```

```
c
          subroutine ppm98_var3d( comment, a3dsymbol, nbdy_xl, nx,
          nbdy_xr, nbdy_yl, ny,n nbdy_yr, nbdy_zl, nz,n nbdy_zr, nvar,
          iovar, array, ifloge, iunit )

          character(LEN=*)      COMMENT
          character(LEN=*)      A3DSYMBOL
          dimension      array( 1:NVAR, 1-NBDY_XL:NX+NBDY_XR, 1-
          NBDY_YL:NY+NBDY_YR, 1-NBDY_ZL:NZ+NBDY_ZR )
          integer*4 NBDY_XL, NX, NBDY_XR
          integer*4 NBDY_YL, NY, NBDY_YR
          integer*4 NBDY_ZL, NZ, NBDY_ZR
          integer*4 IFLOGE
          integer*4 IUNIT
```

The first argument is a FORTRAN character string in which the user may use to describe the quantity. The second is a FORTRAN character string by which the utility **a3d** will reference this array. This must begin with an alphabetic character and must not contain spaces and is case sensetive. Next follow closely the dimensions of the interleaved array, where `NVAR` is the total number of individual arrays interleaved within the `array`. The value of `IOVAR` specifies which of the interleaved arrays is to be written to the compressed dump file. If the the array consists of only one three-dimensional array (no interleaving) then both `NVAR` and `IOVAR` are set equal to one. The value of `IFLOGE` serves as a flag to indicate whether the values of the array should be stored or their natural logarithms (`IFLOGE` = 1). The final argument is the value of the open FORTRAN unit number. After all of the desired two-dimensional arrays have been written, the file is closed by a call to **ppm98_binclose**.

As was indicated with the two dimensional example, the exact ordering of the calls is unimportant. However, it may be advantageous to write the commentary, mesh information and scalar values first as these are stored as ASCII text, before any multi-dimension arrays, which are written as binary data. Then, if the user has been sufficiently cleaver with his commentary, the usual UNIX file utilities such as “more” or “less” may be used to examine and document the contents of the file. For example, using “more” on the file “p14a0000” created by running this example produces the following output:

```
COMMENT="  This is the result from the program cdumps3dexample"
COMMENT="  Its purpose is to demonstrate the PPMLIB compressed"
COMMENT="  dumps routines. The arrays are REAL*4"
COMMENT="File created 13Apr1999 at 13:41:44 Local Time ( -300 minutes from UT)"
NAME0D="ratio of specific heats"   SYMBOL0D="GAMMA"   VALUE0D="   1.400000E+00"
COORD1="X"    COORD2="Y"    COORD3="Z"
MESH1="   59"    MESH2="   61"    MESH3="   67"

INTRL3="  1089"
  0.0000000E+00
  1.4925373E-02
  2.9850746E-02
  4.4776119E-02
  5.9701491E-02
  7.4626863E-02
  8.9552239E-02
  1.0447761E-01
```

```
 1.1940298E-01
 1.3432835E-01
 1.4925373E-01
 1.6417910E-01
 1.7910448E-01
 1.9402985E-01
 2.0895521E-01
 2.2388059E-01
 2.3880596E-01
 2.5373134E-01
 2.6865670E-01
 2.8358209E-01
 2.9850745E-01
 3.1343284E-01
```

---

b. compressed Dump Summary

I. Routines for file handling:

PPM98_BINCLOSE Closes a previously opened file on disk

PPM98_BINOPEN Opens a file on disk for reading or writing

PPM98_BINOPEN_TRUNC Opens a file on disk for writing. If a file of this name already exists, its contents are destroyed.

II. Routines to write:

PPM98_WRITE_COMMENT Writes a line of text to the compressed dump as a comment.

PPM98_MESHO2 Writes the information defining a two dimensional computational grid to the compressed dump assuming that the locations of the left hand zone interfaces defining the grid are stored in REAL*4 one-dimensional arrays. The user assigned a one character designator for each axis. Boundary zones are not written.

D_PPM98_MESHO2 Writes the information defining a two dimensional computational grid to the compressed dump assuming that the locations of the left hand zone interfaces defining the grid are stored in REAL*8 one-dimensional arrays. The user assigned a one character designator for each axis. Boundary zones are not written.

D_PPM98_MESHO3 Writes the information defining a three dimensional computational grid to the compressed dump assuming that the locations of the left hand zone interfaces defining the grid are stored in REAL*8 one-dimensional arrays. The user assigned a one character designator for each axis. Boundary zones are not written.

PPM98_MESHO3 Writes the information defining a three dimensional computational grid to the compressed dump assuming that the locations of the left hand zone interfaces defining the grid are stored in REAL*4 one-dimensional arrays. The user assigned a one character designator for each axis. Boundary zones are not written.

PPM98_WRITE_TIMESTAMP Records in the compressed dump the date and time of file creation as a comment.

D_PPM98_VAR0D Writes a REAL*8 scalar to the compressed dump with its keyword and a comment (if desired.)

PPM98_VAR0D Writes a REAL*4 scalar to the compressed dump with its keyword and a comment (if desired.)

PPM98_VAR2D Writes a REAL*4 two-dimensional array to the compressed dump with its keyword and a comment (if desired.) Boundary zones are not written. Provision is made for interleaved arrays.

D_PPM98_VAR2D Writes a REAL*8 two-dimensional array to the compressed dump with its keyword and a comment (if desired.) Boundary zones are not written. Provision is made for interleaved arrays.

PPM98_VAR3D Writes a REAL*4 three-dimensional array to the compressed dump with its keyword and a comment (if desired.) Boundary zones are not written. Provision is made for interleaved arrays.

D_PPM98_VAR3D Writes a REAL*8 three-dimensional array to the compressed dump with its keyword and a comment (if desired.) Boundary zones are not written. Provision is made for interleaved arrays.