An Introduction to the Piecewise Parabolic Method (PPM) Library I. Interpolation and Hydrodynamics Routines: Part 2

Paul Ralph Woodward,[†] *B Edgar,*
and S. E. Anderson,[†]

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

2. Interpolation

a. Introduction

PPMLIB uses *"piecewise parabolic"* interpolations, which is to say, within any given zone the interpolation function is a continuous parabola, but there may be discontinuities at zone interfaces. Specifically, within zone i, the interpolation has the following form:

$$a_i(\tilde{x}) = a_i^L + \tilde{x}\Big[\Delta a_i + \hat{a}_i(1 - \tilde{x})\Big], \qquad 2.1$$

where

$$\Delta a_i = a_i^R - a_i^L,$$
$$\hat{a}_i = 6\ a_i - 3\ (a_i^L + a_i^R),$$

and $\tilde{x}$, is a normalized coordinate given by

$$\tilde{x} = \frac{x - x_{i-1/2}}{x_{i+1/2} - x_{i-1/2}}\ , \quad x_{i-1/2} \le x \le x_{i+1/2}\,.$$

In the above equations, a_i^L and a_i^R are the (as yet undetermined) point values of the interpolation parabola at the left and right interface of zone i, respectively. To allow for the possibility of discontinuities at zone interfaces, $a_i^L \neq a_{i-1}^R$ in general. With the requirements that the average value of the interpolation for zone i equal the specified zone average, a_i, (provided as the initial data), and that the interpolation pass through these two interface point values, the interpolation is unique, and is defined by whatever procedure is used for determining the left and right interface values, a_i^L and a_i^R for each zone.

e. Interpolating Interface Values

A point interface value a_i^L at the zone interface $x_{i-1/2}$ may be interpolated from zone averages by first finding the unique cubic polynomial which has the same zone averages as the four zones surrounding the zone interface located at $x_{i-1/2}$, namely a_{i-2}, a_{i-1}, a_i, and a_{i+1}. Evaluating this polynomial at the zone interface, $x_{i-1/2}$, produces the desired interface value a_i^L.

Finding this cubic polynomial, $f(x)$, is straightforward, if somewhat tedious. Consider the integral of a function $f(x)$:

$$F(x) = \int_{x_0}^{x} f(x')dx'. \qquad 2.2$$

Here, x_0 is an arbitrary starting point. This results in a quartic polynomial, point values of which are known at zone interfaces from the zone averages. Specifically, with $x_0 = x_{i-5/2}$, then

$$F_{i-5/2} \equiv F(x_{i-5/2}) = 0,$$

$$F_{i-3/2} \equiv F(x_{i-3/2}) = F_{i-5/2} + a_{i-2}(x_{i-3/2} - x_{i-5/2}),$$

$$F_{i-1/2} \equiv F(x_{i-1/2}) = F_{i-3/2} + a_{i-1}(x_{i-1/2} - x_{i-3/2}),$$

$$F_{i+1/2} \equiv F(x_{i+1/2}) = F_{i-1/2} + a_{i}(x_{i+1/2} - x_{i-1/2}),$$

and

$$F_{i+3/2} \equiv F(x_{i+3/2}) = F_{i+1/2} + a_{i+1}(x_{i+3/2} - x_{i+1/2}).$$

The coefficients of the unique quartic polynomial passing through these points may be found by following Newton's interpolation method (cf. Hamming 1962). This polynomial may be written

$$\begin{aligned} y(x) = y_1 + (x - x_1)\Bigg\{ & \mathbf{[}x_2, x_1\mathbf{]} + \\ & (x - x_2)\Bigg[\mathbf{[}x_3, x_2, x_1\mathbf{]} + \\ & \quad (x - x_3)\Bigg(\mathbf{[}x_4, x_3, x_2, x_1\mathbf{]} + \\ & \qquad (x - x_4)(\mathbf{[}x_5, x_4, x_3, x_2, x_1\mathbf{]})\Bigg)\Bigg]\Bigg\}, \end{aligned} \qquad 2.3$$

where the bold square brackets indicate divided differences :

$$\mathbf{[}x_1, x_2\mathbf{]} = \mathbf{[}x_2, x_1\mathbf{]} = \frac{y_2 - y_1}{x_2 - x_1},$$

and

$$\mathbf{[}x_1, x_2, x_3, \cdots, x_n\mathbf{]} = \frac{\mathbf{[}x_1, x_2, x_3, \cdots, x_{n-2}, x_{n-1}\mathbf{]} - \mathbf{[}x_1, x_2, x_3, \cdots, x_{n-2}, x_n\mathbf{]}}{x_{n-1} - x_n}$$

The required divided differences may be easily found by constructing the following divided difference table:

$$\begin{array}{c|ccccc} x_1 & y_1 & & & & \\ & & [x_2, x_1] & & & \\ x_2 & y_2 & & [x_3, x_2, x_1] & & \\ & & [x_3, x_1] & & [x_4, x_3, x_2, x_1] & \\ x_3 & y_3 & & [x_4, x_2, x_1] & & [x_5, \cdots, x_1] \\ & & [x_4, x_1] & & [x_5, x_3, x_2, x_1] & \\ x_4 & y_4 & & [x_5, x_2, x_1] & & \\ & & [x_5, x_1] & & & \\ x_5 & y_5 & & & & \end{array}$$

For this case

$$x_1, y_1 = x_{i-5/2}, F_{i-5/2},$$
$$x_2, y_2 = x_{i-3/2}, F_{i-3/2},$$
$$x_3, y_3 = x_{i-1/2}, F_{i-1/2},$$
$$x_4, y_4 = x_{i+1/2}, F_{i-1/2},$$
and
$$x_5, y_5 = x_{i+3/2}, F_{i-3/2}.$$

The derivative of this quartic polynomial yields the desired cubic interpolation. Evaluation of this cubic at the zone interface $x_{i-1/2}$ provides an interpolated value for $a_{i-1/2}$ at the left hand zone interface. This results in

$$a_{i-1/2} = a_{i-1} + \frac{\Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i}(a_i - a_{i-1}) + \qquad 2.4a$$

$$\frac{1}{\sum_{j=-2}^{1} \Delta x_{i+j}} \left\{ \frac{2\Delta x_i \Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i} \left[\frac{\Delta x_{i-2} + \Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i} - \frac{\Delta x_{i+1} + \Delta x_i}{2\Delta x_i + \Delta x_{i-1}} \right] (a_i - a_{i-1}) \right.$$

$$\left. -\Delta x_{i-1} \frac{\Delta x_{i-2} + \Delta x_{i-1}}{2\Delta x_{i-1} + \Delta x_i} \delta a_i + \Delta x_i \frac{\Delta x_i + \Delta x_{i+1}}{\Delta x_{i-1} + 2\Delta x_i} \delta a_{i-1} \right\}$$

where

$$\delta a_i = \frac{\Delta x_i}{\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1}} \left[\frac{2\Delta x_{i-1} + \Delta x_i}{\Delta x_{i+1} + \Delta x_i}(a_{i+1} - a_i) + \frac{\Delta x_i + 2\Delta x_{i+1}}{\Delta x_{i-1} + \Delta x_i}(a_i - a_{i-1}) \right]$$

is the average slope in zone i of the parabola with zone averages a_{i-1}, a_i, and a_{i+1}. For uniform grids, this reduces to

$$a_{i-1/2} = \frac{7}{12}(a_{i-1} + a_i) - \frac{1}{12}(a_{i+1} + a_{i-2}) \qquad 2.4b$$

This procedure produces values for $a_{i-1/2}$ which are third-order accurate for non-uniform grids. In *smooth, well resolved* regions of the distribution of zone averages, $a^R_{i-1} = a^L_i = a_{i-1/2}$, so that the parabolic interpolation is continuous at $x_{i-1/2}$. However, it will be demonstrated later that these interface values may have to be modified in order to present a better piecewise parabolic interpolation of data which have abrupt changes. It is these modifications which introduce discontinuities at zone interfaces.

The program `INTERPOLATE_1` illustrates the use of one of the PPMLIB interface interpolating routines: **PPM98_INTERP0_INTERFACE**. For computational efficiency, this routine assumes the simplifications available when interpolating on a uniform grid, hence no reference to the grid edge locations is required. (A naming convention used in PPMLIB is to include a ‘‘0’’ in the name of routines specific to uniform grids.) `INTERPOLATE_1` The program is designed to used the data files created by the programs `CREATE_ZONE_AVERAGES_SINE` or `CREATE_ZONE_AVERAGES_TANH` described in the introductory section. Either of these programs will create a data file named ‘‘OUTPUT.AVG.’’ This file must be renamed ‘‘INPUT.AVG’’ that that `INTERPOLATE_1` may access it.

The assumed format for the input data file is for the first line to contain the number of zone averages. *Note, this does not include any boundary zones.* After this number is read, space is allocated for the arrays `XL`, `A_AVG`, and `AL`, the zone interface locations, zone averages, and the yet to be computed interpolated interface values, respectively. This allocation *does* include space for the necessary boundary or fake zones. As shown in the discussion above, this interpolation process requires two zone averages to the left and two to the right of any given zone interface in order to be valid. Hence the correct number of fake zones is 2.

Next, the remaining data are read in from the file. Two numbers per line, one line per zone. The first number gives gives the location of the lefthand zone interface for that zone, `XL`, and the second number is that zone’s average. One additional line with a single value for the final left zone edge location is required, for a total of N+1 lines for N zones. The lines are assumed to be ordered in increasing order of `XL`. The values of XL are not used in the actual interpolation process, but are needed for the plotting routines and to maintain compatibility with the earlier program.

Next, the fake zones are filled with values appropriate to either reflecting or periodic boundary conditions, as selected by the user. After the this call to BOUNDARY, point interface values are interpolated from the supplied zone averages by the call to **PPM98_INTERP0_INTERFACE**. Finally, the coefficients of the PPM parabolae Δa_i and $\hat{a}_i$, are computed and the interpolation is sampled at 11 equally spaced points to provide data for the plotting package MONGO in order to produce a smooth plot of the interpolation. (If the reader is using another type of plotting package, this section may have to be modified.)

```
      PROGRAM INTERPOLATE_1
c
c   This program creates continuous, parabolic interpolations
c   from zone averaged data, assuming a uniform grid.


c
      PARAMETER ( NBDY = 2 )

      ALLOCATABLE :: XL(:), A_AVG(:), AL(:)

      OPEN (unit = 11, file ='INPUT.AVG')

c  The data consists of N+2 lines.  The first line
c  contains the integer N, the number to zone averages in the data set.
c  This is followed by N lines, each line having two numbers, a value for
c  the location of the left edge of the zone XL, and the zone average for
c  that zone A_AVG.  The lines are assumed to be in sequential order according
c  to XL.  An additional line has only one value, the final left zone edge.  A
c  grid of N zones requires N+1 left zone edge locations. An appropriate data
c  set can be created from the program CREATE_ZONE_AVERAGES.  The XL values
c   are used for plotting, not in the interpolation process.



c   Read in the number of zones in the file
           READ (11,*)  N
c
c   Next, allocate the appropriate space
         ALLOCATE    (XL(1-NBDY:N+1+NBDY))
         ALLOCATE (A_AVG(1-NBDY:N+NBDY))
         ALLOCATE    (AL(1-NBDY:N+1+NBDY))
c
c   Read in the rest of the data
c
      DO I=1,N
         READ (11,*) XL(I),A_AVG(I)
      ENDDO
      READ (11,*) XL(N+1)
c
c   Query the user for desired boundary conditions
c
      PRINT *,' Please enter 0 for reflecting boundary conditions'
      PRINT *,'        or    1 for periodic boundary conditions'
      READ *, IBDY_TYPE
      IF ( IBDY_TYPE .EQ. 0 ) THEN
       PRINT *,' Selecting reflecting boundary conditions'
      ELSE
         IF ( IBDY_TYPE .EQ. 1 ) THEN
         PRINT *,' Selecting periodic boundary conditions'
       ELSE
         PRINT *,' Unknown boundary type:',IBDY_TYPE
         call abort()
       ENDIF
      ENDIF
c
c   Initialize the fake zones:
c
          CALL BOUNDARY( XL(1-NBDY), A_AVG(1-NBDY), IBDY_TYPE, N,NBDY )
c
c
c   Interpolate zone interface values
c
      CALL PPM98_INTERP0_INTERFACE (A_AVG(1-NBDY), AL(1-NBDY), N, NBDY)

c   AL(I) are computed on range 1<= I <= N+1

c   Write out the interpolations for plotting [11 samples per zone]

      DO I=1,N
         DX = XL(I+1) - XL(I)
         DA = AL(I+1) - AL(I)
         A6 = 6.0 * (A_AVG(I) - 0.5*(AL(I) + AL(I+1)))
         DO J=0,10
            XX = 0.1*FLOAT(J)
            A_INTERP = AL(I) + XX * (DA + A6 * (1.0 - XX))
            WRITE(6,100) I, XL(I)+XX*DX, A_INTERP
100         FORMAT( I4,1P, 2(1X,e12.5))
         ENDDO
```

```
      ENDDO
      DEALLOCATE (XL, A_AVG, AL)
      END ! PROGRAM INTERPOLATE_1
```

An example is shown in figure 4, using the data generated for figure 1a, a 50 zone sine wave on a uniform grid. The interpolation is shown as a dashed line, and the original function is shown as a dotted line. These lines are virtually coincident.

On non-uniform grids, the various geometrical factors have been collected together into a separate routine, **PPM98_INTERFACE_FACTORS**. This allows for more efficient computation if several quantities must be interpolated on the same grid, as is often the case. These factors are used by the PPMLIB subroutine **PPM98_INTERP_INTERFACE**, the non-uniform grid version of **PPM98_INTERP0_INTERFACE**. Non-uniform grid interpolation is illustrated by the example program `INTERPOLATE_2`.

The data are then read from a file called `INPUT.AVG`, as before. As in the previous example, space must be allocated for the arrays `XL`, the locations of the zone interfaces defining the grid, `A_AVG`, the zone averages, and `AL`, the interpolated interface values. However, several additional arrays are now required to hold the the necessary geometrical factors: `DX, DALFAC, DARFAC, FACDA, FCDAZL`, and `FACDAL`. The only quantity here likely to ever be of use other than in the interpolation routines is `DX`, the zone width. as before the fake zones in the input arrays are filled with values appropriate to either reflecting or periodic boundary conditions, as selected by the user.

Next, the zone widths, `DX`, are computed, and serve as the input data for the call to **PPM98_INTERFACE_FACTORS**. After the necessary factors have been set, the interface values are interpolated by calling **PPM98_INTERP_INTERFACE**. Finally, suitable values for plotting are produced. It should be noted that these two routines, **PPM98_INTERFACE_FACTORS** and **PPM98_INTERP_INTERFACE** have been designed to operate together with the same array offsets (if any) and dimensions.

```
      PROGRAM INTERPOLATE_2
c
c   This program creates continuous, parabolic interpolations
c   from zone averaged data, without assuming a uniform grid.
c
      PARAMETER ( NBDY = 2 )

      ALLOCATABLE :: XL(:), A_AVG(:), AL(:), DX(:), DALFAC(:)
      ALLOCATABLE :: DARFAC(:), FACDA(:), FCDAZL(:), FACDAL(:)

      OPEN (unit = 11, file ='INPUT.AVG')
c  The data consists of N+2 lines.  The first line
c  contains the integer N, the number to zone averages in the data set.
c  This is followed by N lines, each line having two numbers, a value for
c  the location of the left edge of the zone XL, and the zone average for
c  that zone A_AVG.  The lines are assumed to be in sequential order according
c  to XL.  An additional line has only one value, the final left zone edge.  A
c  grid of N zones requires N+1 left zone edge locations. An appropriate data
c  set can be created from the program CREATE_ZONE_AVERAGES.  The XL values
c   are used for plotting, not in the interpolation process.


c   Read in the number of zones in the file
            READ (11,*)  N
c
c   Next, allocate the appropriate space
          ALLOCATE    (XL(1-NBDY:N+1+NBDY))
          ALLOCATE (A_AVG(1-NBDY:N+NBDY))
          ALLOCATE    (AL(1-NBDY:N+1+NBDY))

c   The following arrays will hold  various geometrical factors which can be
c   precomputed for efficiency when several quantities are interpolated on the
c   same grid:
          ALLOCATE    (DX(1-NBDY:N+1+NBDY))
          ALLOCATE    (DALFAC(1-NBDY:N+1+NBDY))
          ALLOCATE    (DARFAC(1-NBDY:N+1+NBDY))
          ALLOCATE     (FACDA(1-NBDY:N+1+NBDY))
          ALLOCATE    (FCDAZL(1-NBDY:N+1+NBDY))
          ALLOCATE    (FACDAL(1-NBDY:N+1+NBDY))
```

```
c
c   Read in the rest of the data
c
      DO I=1,N
         READ (11,*) XL(I),A_AVG(I)
      ENDDO
      READ (11,*) XL(N+1)

c
c   Query the user for desired boundary conditions
c
      PRINT *,' Please enter 0 for reflecting boundary conditions'
      PRINT *,'          or    1 for periodic boundary conditions'
      READ *, IBDY_TYPE
      IF ( IBDY_TYPE .EQ. 0 ) THEN
      PRINT *,' Selecting reflecting boundary conditions'
      ELSE
         IF ( IBDY_TYPE .EQ. 1 ) THEN
         PRINT *,' Selecting periodic boundary conditions'
      ELSE
         PRINT *,' Unknown boundary type:',IBDY_TYPE
         call abort()
      ENDIF
      ENDIF
c
c   Initialize the fake zones:
c
c
          CALL BOUNDARY( XL(1-NBDY), A_AVG(1-NBDY), IBDY_TYPE, N,NBDY )
c
c
c   A uniform grid is NOT assumed.  Therefore, various
c   geometrical factors must be precomputed by a call to
c   _INTERFACE_FACTORS, input to  which consists of DX(I),
c   the individual zone widths.

      DO I=1-NBDY,N+NBDY
         DX(I) = XL(I+1) - XL(I)
      ENDDO

      CALL PPM98_INTERFACE_FACTORS ( DX(1-NBDY), DALFAC(1-NBDY),
     ^             DARFAC(1-NBDY), FACDA(1-NBDY), FCDAZL(1-NBDY),
     ^            FACDAL(1-NBDY), N, NBDY)

c
c   Interpolate zone interface values
c
      CALL PPM98_INTERP_INTERFACE ( DALFAC(1-NBDY), DARFAC(1-NBDY),
     &               FACDA(1-NBDY), FCDAZL(1-NBDY), FACDAL(1-NBDY),
     &               A_AVG(1-NBDY),     AL(1-NBDY), N, NBDY )

c   AL(I) are defined for 1<= I <= N+1

c   Write out the interpolations for plotting [11 samples per zone]

      DO I=1,N
         DA = AL(I+1) - AL(I)
         A6 = 6.0 * (A_AVG(I) - 0.5*(AL(I) + AL(I+1)))
         DO J=0,10
            XX = 0.1*FLOAT(J)
            A_INTERP = AL(I) + XX * (DA + A6 * (1.0 - XX))
            WRITE(6,100) I, XL(I)+XX*DX(I), A_INTERP
100         FORMAT( I4,1P, 2(1X,e12.5))
         ENDDO
      ENDDO

      DEALLOCATE ( XL, A_AVG, AL, DX, DALFAC )
      DEALLOCATE ( DARFAC, FACDA, FCDAZL, FACDAL )

      END ! PROGRAM INTERPOLATE_2
```

Results from program INTERPOLATE_2, using the data sequence illustrated in figure 2 generated from a hyperbolic tangent function on a non-uniform grid produces the sequence shown in in figure 5. As expected, the interpolation (shown as dashed) is virtually identical with the generating function (shown as dots) for the smooth, well resolved distribution with about 8 zone across the jump. However, the small departures just visible in the 4 zone jump become much more pronounced as the width (measured in number of zones) of the jump narrows.

The final two interpolations clearly illustrates "ringing," the spurious over and undershoots shown here. For many applications, this will lead eventually to over- and undershoots in the zone averaged data. Several examples of this effect are shown in Woodward (1986) based on the advection of a square wave. This would be disastrous in many applications such as in hydrodynamics, where such errors can produce unphysical results: negative densities, pressures, and energies. Thus it is necessary to modify the interpolative scheme when the data are poorly resolved.

f. Monotonicity

The interpolative procedure given above can often be improved using the concept of *monotonicity*. Monotonicity constraints ensure that after advecting a distribution which was initially monotone, the resulting distribution will also be monotone, avoiding the introduction of new extrema as well as preventing the amplification of existing extrema during a calculation. This prevents spurious oscillations near sharp features and positive definite quantities remain positive. This interpretation of monotonicity is based on the ideas of van Leer (1977) who developed monotonicity constraints for polynomials in the context of advection. A similar idea of flux limiters for high-order advection schemes was developed earlier by Boris and Book (1973).

The first step is in creating monotonized interpolated interface point values, $a_i^{L_M}$ and $a_i^{R_M}$, is to require that the interpolated interface point value, $a_{i-1/2}$, be bounded by its neighboring zone averages, a_{i-1} and a_i. This can be ensured if δa_i in equation 2.4 above is replaced with

$$\delta a'_i = \begin{cases} -\min\left(|\delta a_i|, 2|a_i - a_{i-1}|, 2|a_{i+1} - a_i| \right) \mathrm{sgn}(\delta a_i) & \text{if } (a_{i+1} - a_i)(a_i - a_{i-1}) > 0, \\ 0 & \text{otherwise}, \end{cases} \qquad 2.5$$

where

$$\delta \mathrm{sgn}(\delta a_i) = \begin{cases} 1 & \text{if } a_i \ge 0, \\ -1 & \text{if } a_i < 0. \end{cases}$$

The interpolated value so found will be denoted $a'_{i-1/2}$.

At this point, the interpolations are still continuous at zone boundaries. That is

$$a'^L_i = a'^R_{i-1} = a'_{i-1/2}. \qquad 2.6$$

However, these interface values must be further modified to ensure that no point on the implied interpolated parabola exceeds the range determined by the two monotonized point values, $a'_{i-1/2}, a'_{i+1/2}$. The check for such a monotone parabola is

$$|\Delta a'_i| \ge |\hat{a}'_i|, \qquad 2.7$$

where

$$\Delta a'_i = a'^R_i - a'^L_i,$$
$$\hat{a}'_i = 6\, a_i - 3\,(a'^L_i + a'^R_i),$$

If this condition is violated, then either a'^L_i or a'^R_i is reset, such that the interpolation parabola is monotone, and that its derivative at the edge of the zone opposite that of the reset value is zero. This may introduce discontinuities at zone interfaces. The final expressions for the monotonized zone interface point values are

$$\left.\begin{aligned} a'^L_i &= a_i \\ a'^R_i &= a_i \end{aligned}\right\} \quad \text{if}(a'_{i+1/2} - a_i)(a_i - a'_{i-1/2}) \le 0$$

$$\left.\begin{aligned} a'^L_i &= 3a_i - 2a'_{i+1/2} \\ a'^R_i &= a'_{i+1/2} \end{aligned}\right\} \quad \text{if } |\Delta a'_i| < \hat{a}'_i \qquad 2.8$$

$$\left.\begin{aligned} a'^L_i &= a'_{i-1/2} \\ a'^R_i &= 3a_i - 2a'_{1/2} \end{aligned}\right\} \quad \text{if } |\Delta a'_i| > \hat{a}'_i$$

Problems could develop if these interface values were to be used exclusively to determine the piecewise parabolic interpolations. Specifically, it should be noted that these clip extrema, including when the function is well resolved. This leads to the distortion of even smooth distributions. Several examples may be seen in Woodward and Colella (1984) and Carpenter, et. al (1990). Therefore, it would be preferable to use these monotonized interface values only where the distribution is poorly resolved. This will require constructing some estimate of the ‘‘smoothness’’ of the distribution, allowing the retention of the excellent properties of the above interpolation scheme when the data are judged to be well resolved.

g. Measuring Function Smoothness

In cases where the function is smooth, that is, it is well resolved on the numerical grid and the resulting zone averages are a good representation of that function, any given zone should be well represented not only by the parabolic interpolation constructed for that zone, but by the interpolations of nearby zones extrapolated into that zone. The differences between the actual zone averages and zone averages computed from these extrapolations will provide the basis for constructing a measurement of "smoothness".

First consider a simpler form for a parabolic interpolation than that given above:

$$\tilde{a}_i(\hat{x}) = \alpha_i + \beta_i \hat{x} + \gamma_i \hat{x}^2 \qquad 2.9$$

where $\hat{x}$ is normalized in a slightly different manner from before:

$$\hat{x} = \frac{x - \frac{1}{2}(x_{i+\frac{1}{2}} + x_{i-\frac{1}{2}})}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}}, \qquad x_{i-\frac{1}{2}} \le x \le x_{i+\frac{1}{2}}$$

Thus

$$-\tfrac{1}{2} \le \hat{x} \le \tfrac{1}{2}.$$

The coefficients may be determined by simply requiring that the parabola have the same zone averages as the zone i and its two neighbors, rather than by finding interface point values through an interpolation. These coefficients may be found following Newton's interpolation method described above. The result of following that procedure is:

$$\gamma_i = \frac{3\Delta x_i^2}{\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1}} \left[\frac{a_{i+1} - a_i}{\Delta x_i + \Delta x_{i+1}} - \frac{a_i - a_{i-1}}{\Delta x_i + \Delta x_{i-1}} \right],$$

$$\beta_i = \frac{\Delta x_i}{\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1}} \left[(2\Delta x_{i-1} + \Delta x_i)(a_{i+1} - a_i) + (\Delta x_i + 2\Delta x_{i+1})(a_i - a_{i-1}) \right], \qquad 2.10a$$

$$\alpha_i = a_i - \frac{\gamma_i}{12}.$$

For uniform grids , these equations reduce to

$$\beta_i = \tfrac{1}{2}\left[(a_{i+1} - a_i) + (a_i - a_{i-1}) \right] = \frac{(a_{i+1} - a_{i-1})}{2}$$

$$\gamma_i = \tfrac{1}{2}\left[(a_{i+1} - a_i) - (a_i - a_{i-1}) \right] = \frac{(a_{i+2} - 2a_i + a_{i+1})}{2} \qquad 2.10b$$

$$\alpha_i = a_i - \frac{\gamma_i}{12}.$$

When the the parabola from zone $i-1$ is averaged over zone $i+1$ and compared to the actual zone average there, a_{i+1}, one indication of the smoothness in the region surrounding zone i can be found from the difference:

$$\sigma_i^a = \int_{x_{i+1/2}}^{x_{i+3/2}} \left(\alpha_{i-1} + \beta_{i-1}\hat{x} + \gamma_{i-1}\hat{x}^2 \right) d\hat{x} \; - \; a_{i+1} \qquad 2.11$$

Similarly when the the parabola from zone $i+1$ is averaged over zone $i-1$ and differenced with the actual zone average there, a_{i-1}, another indication of the smoothness in the region surrounding zone i can be found:

$$\sigma_i^b = \int_{x_{i-3/2}}^{x_{i-1/2}} \left(\alpha_{i+1} + \beta_{i+1}\hat{x} + \gamma_{i+1}\hat{x}^2 \right) d\hat{x} - a_{i-1} \qquad 2.12$$

These two errors, σ_i^a and σ_i^a can be combined into a fractional error estimate, Σ_i, by comparing the two individual errors with with the local absolute variation in the function:

$$\Sigma_i = \frac{1}{2} \frac{|\sigma_i^b - a_{i+1}| + |\sigma_i^a - a_{i-1}|}{|a_i - a_{i-1}| + |a_{i+1} - a_i| + \delta_i} \qquad 2.13$$

where δ_i is a value which is to be considered a trivial difference of the variable a in zone i. Its purpose is to prevent smooth, flat regions of the data from being interpreted as unsmooth merely due to the presence of numerical noise. For example, for positive-definite quantities such as density or pressure, it would be appropriate to set

$$\delta_i = \varepsilon\, a_i \qquad 2.14$$

where ε is a reasonably small value, such as 0.005. For velocity related quantities, δ_i would be set equal to ε times the local sound speed. Generally, in all other cases $\delta_i = \varepsilon$.

On a non-uniform grid, this results in

$$\begin{aligned} \sigma_i^b - a_{i+1} &= \rho_{i-1}^1(a_i - a_{i-1}) + \rho_{i-1}^2(a_i - a_{i-1}) - (a_{i+1} - a_{i-1}) \\ \sigma_i^a - a_{i-1} &= \lambda_{i-1}^1(a_i - a_{i-1}) + \lambda_{i-1}^2(a_i - a_{i-1}) - (a_{i+1} - a_{i-1}) \end{aligned} \qquad 2.15a$$

where

$$\rho_i^1 = \frac{1}{2} \frac{\Delta x_i + 2\Delta x_{i+1} + \Delta x_{i+2}}{\Delta x_i} \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1})} \frac{(\Delta x_i + 2\Delta x_{i+1})}{(\Delta x_{i-1} + \Delta x_i)}$$
$$+ \frac{1}{12}\left\{1 - \left[\frac{\Delta x_i + 2\Delta x_{i+1} + \Delta x_{i+2}}{\Delta x_i}\right]^2 + \left[\Delta x_{i+2}\Delta x_i\right]^2\right\} \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1})} \frac{3\Delta x_i}{\Delta x_{i-1} + \Delta x_i}$$
$$\rho_i^2 = \frac{1}{2} \frac{\Delta x_i + 2\Delta x_{i+1} + \Delta x_{i+2}}{\Delta x_i} \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1})} \frac{(\Delta x_i + 2\Delta x_{i-1})}{(\Delta x_{i+1} + \Delta x_i)}$$
$$+ \frac{1}{12}\left\{1 - \left[\frac{\Delta x_i + 2\Delta x_{i+1} + \Delta x_{i+2}}{\Delta x_i}\right]^2 + \left[\Delta x_{i+2}\Delta x_i\right]^2\right\} \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1})} \frac{3\Delta x_i}{\Delta x_{i+1} + \Delta x_i}$$

and

$$\lambda_i^1 = \frac{1}{2} \frac{-(\Delta x_{i-2} + 2\Delta x_{i-1} + \Delta x_i}{\Delta x_i} \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1})} \frac{(\Delta x_i + 2\Delta x_{i+1})}{(\Delta x_{i-1} + \Delta x_i)}$$
$$+\left\{1 - \frac{1}{12}\left[\frac{-(\Delta x_{i-2} + 2\Delta x_{i-1} + \Delta x_i}{\Delta x_i}\right]^2 + \left[\Delta x_{i-2}\Delta x_i\right]^2\right\} \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1})} \frac{3\Delta x_i}{\Delta x_{i-1} + \Delta x_i}$$
$$\lambda_i^2 = \frac{1}{2} \frac{-(\Delta x_{i-2} + 2\Delta x_{i-1} + \Delta x_i}{\Delta x_i} \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1})} \frac{(\Delta x_i + 2\Delta x_{i-1})}{(\Delta x_{i+1} + \Delta x_i)}$$
$$+\left\{1 - \frac{1}{12}\left[\frac{-(\Delta x_{i-2} + 2\Delta x_{i-1} + \Delta x_i}{\Delta x_i}\right]^2 + \left[\Delta x_{i-2}\Delta x_i\right]^2\right\} \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i + \Delta x_{i+1})} \frac{3\Delta x_i}{\Delta x_{i+1} + \Delta x_i}$$

Mercifully, on a uniform grid these reduce to

$$\begin{aligned} \sigma_i^b - a_{i+1} &= 2\,(\beta_{i-1} - \beta_i) \\ \sigma_i^a - a_{i-1} &= 2\,(\beta_{i+1} - \beta_i) \end{aligned} \qquad 2.15b$$

From this fractional error estimate, a measure of the lack of smoothness of the function a in zone i is constructed:

$$U_i = \min(1, \max[0, \Sigma_1(\Sigma_i - \Sigma_0)], \qquad 2.16$$

where Σ_1 and Σ_0 are dimensionless constants yet to be chosen. Thus the ‘‘unsmoothness’’ measure U_i ranges from 0 for a smooth well, resolved function to 1 for unresolved jumps. The default values for the dimensionless constants are

$$\Sigma_0 = 0.1$$

and

$$\Sigma_1 = 10.0.$$

These values were determined by assuming a 15 zone sine wave is resolved, while a 9 zone sine wave is unresolved. This is discussed in greater detail later in the section *Customizing your PPMLIB Interpolations.*

These factors relating to functional smoothness, U_i, may be combined with the two interface point values computed previously: the unmonotonized values $a_{i-1/2}$, $a_{i+1/2}$ and the monotonized values a'^L_i, a'^R_i:

$$\begin{aligned} a_i^{L_M} &= U_i\ a'^L_i + (1 - U_i)a_i^L \\ a_i^{R_M} &= U_i\ a'^R_i + (1 - U_i)a_i^R, \end{aligned} \qquad 2.17$$

resulting in a smooth application of the monotonicity constraints. Where the distribution is definitely unresolved ($U_i = 1$), the monotonicity constraints are applied fully, preventing spurious oscillations in the interpolations. Where the distribution is well resolved ($U_i = 0$), the unconstrained values are used to retain the accuracy of the cubic interpolation in smooth regions, allowing for the advection of smooth distributions without clipping of extrema. Finally in intermediate cases, $0 < U_i < 1$ the constraints are partially applied, preventing problems which would arise if the interpolation procedure switched abruptly between the two types of interpolating. The use of one of the routines implementing this procedure **PPM98_INTERP_MONO**, is illustrated in the example program `INTERPOLATE_3`, for use on non-uniform grids.

```
      PROGRAM INTERPOLATE_3
c
c   This program creates piecewise parabolic interpolations
c   from zone averaged data, without assuming a uniform grid.
c   Monotonicity constraints are applied.
c
      PARAMETER ( NBDY = 3 )

      ALLOCATABLE :: XL(:), A_AVG(:), AL(:), AR(:), SMALLDA(:),
     ^          UNSMOOTH(:)
c
c   The following arrays will hold  various geometrical factors which can be
c   precomputed for efficiency when several quantities are interpolated on the
c   same grid:

      ALLOCATABLE :: DX(:), DALFAC(:), DARFAC(:),  FACDA(:), FCDAZL(:),
     ^            FACDAL(:), FA6DAL(:), FA6DAR(:), FRRDAL(:), FRRDAR(:),
     ^            FLLDAL(:), FLLDAR(:)

      OPEN (unit = 11, file ='INPUT.AVG')
c  The data consists of N+2 lines.  The first line
c  contains the integer N, the number to zone averages in the data set.
c  This is followed by N lines, each line having two numbers, a value for
c  the location of the left edge of the zone XL, and the zone average for
c  that zone A_AVG.  The lines are assumed to be in sequential order according
c  to XL.  An additional line has only one value, the final left zone edge.  A
c  grid of N zones requires N+1 left zone edge locations. An appropriate data
c  set can be created from the program CREATE_ZONE_AVERAGES.  The XL values
c   are used for plotting, not in the interpolation process.


c   Read in the number of zones in the file
            READ (11,*)  N
c
c   Next, allocate the appropriate space
          ALLOCATE      (XL(1-NBDY:N+1+NBDY))
          ALLOCATE   (A_AVG(1-NBDY:N+NBDY))
          ALLOCATE (SMALLDA(1-NBDY:N+NBDY))
          ALLOCATE      (AL(1-NBDY:N+1+NBDY))
          ALLOCATE      (AR(1-NBDY:N+1+NBDY))

c   The following arrays will hold  various geometrical factors which can be
c   precomputed for efficiency when several quantities are interpolated on the
c   same grid:
          ALLOCATE    (DX(1-NBDY:N+1+NBDY))
          ALLOCATE    (DALFAC(1-NBDY:N+1+NBDY))
          ALLOCATE    (DARFAC(1-NBDY:N+1+NBDY))
          ALLOCATE     (FACDA(1-NBDY:N+1+NBDY))
          ALLOCATE    (FCDAZL(1-NBDY:N+1+NBDY))
          ALLOCATE    (FACDAL(1-NBDY:N+1+NBDY))
          ALLOCATE    (FA6DAL(1-NBDY:N+1+NBDY))
          ALLOCATE    (FA6DAR(1-NBDY:N+1+NBDY))
          ALLOCATE    (FRRDAL(1-NBDY:N+1+NBDY))
          ALLOCATE    (FRRDAR(1-NBDY:N+1+NBDY))
          ALLOCATE    (FLLDAL(1-NBDY:N+1+NBDY))
          ALLOCATE    (FLLDAR(1-NBDY:N+1+NBDY))
```

```
          ALLOCATE  (UNSMOOTH(1-NBDY:N+1+NBDY))



c   First, read in the data, which consists of N+1 lines.  Each
c   line has two numbers, a value for the location of the left edge
c   of the zone ZL, and the zone average for that zone A_AVG.  The
c   lines are assumed to be in sequential order according to XL.  The final
c   line has only one value, the final left zone edge.  A grid of N zones
c   requires N+1 left zone edge locations. An appropriate data set
c   can be created from the program CREATE_ZONE_AVERAGES.

      OPEN (unit = 11, file ='INPUT.AVG')

      DO I=1,N
         READ (11,*) XL(I),A_AVG(I)
      ENDDO
      READ (11,*) XL(N+1)
c
c   Query the user for desired boundary conditions
c
      PRINT *,' Please enter 0 for reflecting boundary conditions'
      PRINT *,'        or    1 for periodic boundary conditions'
      READ *, IBDY_TYPE
      IF ( IBDY_TYPE .EQ. 0 ) THEN
         PRINT *,' Selecting reflecting boundary conditions'
      ELSE
         IF ( IBDY_TYPE .EQ. 1 ) THEN
            PRINT *,' Selecting periodic boundary conditions'
         ELSE
            PRINT *,' Unknown boundary type:',IBDY_TYPE
            call abort()
         ENDIF

      ENDIF
c
c   Initialize the fake zones:

          CALL BOUNDARY( XL(1-NBDY), A_AVG(1-NBDY), IBDY_TYPE, N,NBDY )
c

c
c   A uniform grid is NOT assumed.  Therefore, various
c   geometrical factors must be precomputed by a call to
c   _INTERFACE_FACTORS, input to  which consists of DX(I),
c   the individual zone widths.

      DO I=1-NBDY,N+NBDY
         DX(I) = XL(I+1) - XL(I)
      ENDDO


      CALL PPM98_INTERPOLATION_FACTORS (  DX(1-NBDY), DALFAC(1-NBDY),
     ^                DARFAC(1-NBDY),  FACDA(1-NBDY), FCDAZL(1-NBDY),
     ^                FACDAL(1-NBDY), FA6DAL(1-NBDY), FA6DAR(1-NBDY),
     ^           FRRDAL(1-NBDY), FRRDAR(1-NBDY), FLLDAL(1-NBDY),
     ^           FLLDAR(1-NBDY), N, NBDY)


c
c   Fill an array, SMALLDA, which contains what is to be considered a trivial
c   difference of A_AVG.  In this Program, A_AVG is not positive definite, so
c  a small value is appropriate.

      DO I=1-NBDY,N+NBDY
         SMALLDA(I) = 0.0005
      ENDDO

c   Interpolate zone interface values
c
      CALL PPM98_INTERP_MONO ( DALFAC(1-NBDY), DARFAC(1-NBDY),
     ^        FACDA(1-NBDY),    FCDAZL(1-NBDY), FACDAL(1-NBDY),
     ^       FA6DAL(1-NBDY),    FA6DAR(1-NBDY), FLLDAL(1-NBDY),
     ^       FLLDAR(1-NBDY),    FRRDAL(1-NBDY), FRRDAR(1-NBDY),
     ^      SMALLDA(1-NBDY),     A_AVG(1-NBDY),     AL(1-NBDY),
     ^           AR(1-NBDY), UNSMOOTH(1-NBDY),   N, NBDY)

c   AL(I) is defined for 1<= I <= N
c   AL(I), and AR(I) are defined for 1<= I <= N

c   Write out the interpolations for plotting [11 samples per zone]


      DO I=1,N
         DA = AR(I) - AL(I)
         A6 = 6.0 * (A_AVG(I) - 0.5*(AL(I) + AR(I)))
         DO J=0,10
            XX = 0.1*FLOAT(J)
            A_INTERP = AL(I) + XX * (DA + A6 * (1.0 - XX))
            WRITE(6,100) I, XL(I)+XX*DX(I), A_INTERP, UNSMOOTH(I)
100         FORMAT( I4,1P, 3(1X,e12.5))
         ENDDO
      ENDDO

      DEALLOCATE  (XL, A_AVG, AL, AR, SMALLDA, UNSMOOTH)
      DEALLOCATE (DX, DALFAC, DARFAC,  FACDA, FCDAZL,
     ^        FACDAL, FA6DAL, FA6DAR, FRRDAL, FRRDAR,
```

```
^         FLLDAL, FLLDAR)

      END ! PROGRAM INTERPOLATE_3
```

The structure of `INTERPOLATE_3` is very similar to `INTERPOLATE_2`. As in **PPM98_INTERP_INTERFACE**, the geometrical factors have been collected together into one subroutine for efficiency when several quantities are interpolated on the same grid. Because of the added complexity caused by the unsmoothness calculation, several more factors must be computed than before. The appropriate routine for **PPM98_INTERP_MONO** is **PPM98_INTERPOLATION_FACTORS**.

There are several other important differences. One is the addition of an explicit array for the interpolated right zone interface point values, `AR`. As was noted in the discussion above, the application of monotonicity constraints can introduce discontinuities at the zone interfaces so that `AR(I)` $\neq$ `AL(I+1)`, in general. *It is vitally important that the programmer not equivalence these two arrays.* Additionally, arrays are allocated for `SMALLDA` (δ_i from eqn. 2.14), a value defined to be a trivial difference of `A_AVG`, and `UNSMOOTH`, the zone-by-zone measurement of the "unsmoothness" of the distribution of zone averages. Also, several arrays for geometrical factors in addition to those needed in `INTERPOLATE_2` are allocated: `FA6DAL`, `FA6DAR`, `FRRDAL`, `FRRDAR`, `FLLDAL`, and `FLLDAR`.

The input data are read in as before, and the zone widths `DX` are computed which serve as the input data for the routine which pre-calculates the geometrical factors: **PPM98_INTERPOLATION_FACTORS**. The call to the aforementioned subroutine is followed by the initialization of the array of trivial differences, `SMALLDA`, with the value 0.0005. This array is used to prevent numerical noise from making flat areas in the zone average distribution from appearing unsmooth. This is an appropriately small value for a non-positive-definite distribution like `A_AVG`. The interface point values are then interpolated by a call to **PPM98_INTERP_MONO**. The two routines, **PPM98_INTERPOLATION_FACTORS** and **PPM98_INTERP_MONO** have been designed to operate together with the same array offsets and dimensions. In addition to the monotonized interface values `AL_MONO` and `AR_MONO`, the measure of unsmoothness is returned in the array `UNSMOOTH`. The piecewise parabolic coefficients are computed and the interpolations are sampled for plotting as before, and printed along with the unsmoothness measure for examination.

Results from program `INTERPOLATE_3`, imposing monotonicity constraints on the interpolations where jumps in the distribution of zone averages are poorly resolved are shown in figure 6. The same data sequence as was used for figures 2 and 5, generated from a hyperbolic tangent function on a non-uniform grid. The actual function used to generate the zone averages is shown as a dotted line. The zone averages themselves are not shown. The monotonized interpolation, computed by INTERPOLATE_3, is shown as a dashed line. Comparing figures 5 and 6, it is clear that in addition to preventing spurious oscillations, the application of monotonicity constraints also provides a better representation for unresolved jumps in the data. However, there remains one final modification of the interpolation scheme which will further improve the representation in those zones in which it can be determined that a true discontinuity exists. For such zones, rather than using the cubic interpolation which inherently assumes a smooth distribution, an interpolation suited for discontinuities will be used.

h. Discontinuities and Discontinuity Detection

The measure of how well resolved any jump in a function for poorly resolved data combined with the concept of monotonicity described above improves the interpolation procedure. However, for zones in discontinuities, it is possible to construct an interpolation which generally gives a sharper representation, constructed by using interface values found by extrapolating the *monotonized* zone-averaged differences $\Delta a'_i$ from the *neighboring* zones. Thus the left and right interface point values appropriate for discontinuities are

$$a_i^{L_D} = a_{i-1} + 1/2\,\Delta a'_{i-1}, \qquad 2.18a$$

and

$$a_i^{R_D} = a_{i+1} - 1/2\,\Delta a'_{i+1}. \qquad 2.18b$$

As above with the application of monotonicity constraints, it remains to determine a means of switching between the two possible interpolations.

To discriminate between a discontinuity and merely a poorly resolved function requires further information, and of course, computation. Comparisons among the the difference approximations to the first, second, and third derivatives are required. Small plateaus in an otherwise monotone change in the distribution are avoided by requiring that the first and third derivatives have opposite signs. Thus a discontinuity is signalled by a relatively large third derivative and a second derivative which changes sign across the zone. Only those zones containing jumps greater than some threshold, say 1%, are considered discontinuities, to minimize the effects of numerical noise. As in the application of monotonicity, a parameter, S_i, ranging smoothly from zero for zones requiring no modification to one, for zones considered fully within a discontinuity will be constructed, allowing for a smooth application of the modifications to the interpolations.

However, it is not necessary to go to such an elaborate calculation. By examining the form of equations (2.4), it is apparent that the *unmonotonized* difference

$$\Delta a_i = a_{i+1/2} - a_{i-1/2} \qquad 2.19$$

includes elements of the third derivative of a_i. However, a difference constructed from the simpler interpolation, eqn (2.9)

$$\Delta \tilde{a}_i = \tilde{a}_i(-1/2) - \tilde{a}_i(1/2) = \beta_i \qquad 2.20$$

contains only elements of the second derivative. Structures of the type which benefit from this procedure will have a larger difference when computed from the more elaborate interpolation than when computed from the simpler. Both differences should, however, have the same sign for the desired cases. Thus it would be possible to construct a measure of the steepness of a jump in the following form:

$$\frac{\Delta a_i - \Delta \tilde{a}_i}{\Delta \tilde{a}_i}$$

However this is unsatisfactory as is, because there is nothing to prevent small jumps, such as those due to numerical noise, from being flagged as discontinuities. Thus a more useful formula is

$$J_i = \frac{(\Delta a_i - \Delta \tilde{a}_i)}{\Delta \tilde{a}_i + 10\, \delta_i\, sgn(\Delta \tilde{a}_i)}\,, \qquad 2.21$$

where, as in eqn (2.21), δ_i is a trivial difference in a. The assumption here is that ten times this trivial difference, δ_i, is still small compared to any unresolved jumps in the distribution which should be considered as such.

As constructed, only positive values of J_i correspond to discontinuities, while negative values correspond to plateaus in the distribution. This suggests, analogous to the construction for the "unsmoothness" parameter, that a usable steepness parameter would have the form

$$S'_i = \min\left\{1, \max\left[0, J_1(J_i - J_0)\right]\right\} \qquad 2.22$$

The default values for the dimensionless constants are

$$J_0 = 0.05$$

and

$$J_1 = 25.0.$$

The dimensionless constants in the above equation were chosen to ensure that discontinuities one zone in width would have a steepness parameter of 1, regardless of the phase of the discontinuity with respect to the zone boundaries. They must also be consistent with the dimensionless constants used in the detection of poorly resolved regions of the distribution. With these choices, a 16 zone sound wave will be unaffected, while a 10 zone sine wave will be fully steepened, with a smooth transition between these regions.

The requirement that the second derivative change signs across the jump is enforced using the coefficient γ_i from eq 2.9, which is directly related to the second derivative. This requirement leads to the condition that

$$\gamma_{i-1}\, \gamma_{i+1} < 0$$

for true unresolved discontinuities. The prevention of steepening trivial jumps can be more carefully enforced by examining differences between zone averages and requiring that these differences be greater than the difference considered trivial, δ_i.

$$|a_i - a_{i-1}| + |a_{i+1} - a_i| > \delta_i.$$

Thus the steepness parameter S_i may be written

$$S_i = \begin{cases} 0 & \text{if} \quad \gamma_{i-1}\ \gamma_{i+1} \geq 0 \\ 0 & \text{if} \quad |a_i - a_{i-1}| + |a_{i+1} - a_i| \leq \delta_i \\ \min\left\{1, \max\left[0, J_1(J_i - J_0)\right]\right\} & \text{otherwise.} \end{cases} \qquad 2.23$$

At this point, an optional comparison may be performed which is especially useful in hydrodynamics, to distinguish between the two kinds of discontinuities occur: shocks and contact discontinuities. Contact discontinuities require steepening to preserve their sharp profile as they are advected, but shock waves should not be steepened. The key is to recognize that shocks are associated with pressure jumps, while contact discontinuities are not, while entropy jumps exist across both. In general, a test is made between two additional differences. One which is expected to be small, δp_i, and one which is expected to be large, δq_i. Such discontinuities must pass the additional test $\delta p_i < \kappa \delta q_i$, where the default value for κ is 0.1, although the precise value of this constant is not critical to the proper functioning of the routine. The application of steepening is performed in a similar manner to the application of monotonicity, equation 2.13:

$$a'''^L_i = S_i a_i^{L_D} + (1 - S_i) a''^L_i$$
$$a'''^R_i = S_i a_i^{R_D} + (1 - S_i) a''^R_i \qquad 2.24$$

Finally, the checks given above for monotonicity must again be tested.

```
      PROGRAM INTERPOLATE_4
c
c   This program creates piecewise parabolic interpolations
c   from zone averaged data, without assuming a uniform grid.
c   Monotonicity constraints are applied. For those zones
c   considered to be in a discontinuity, an interpolation more
c   appropriate for discontinuous functions is used, producing
c   a better representation of unresolved shocks.
c
      PARAMETER ( NBDY = 3 )

      ALLOCATABLE :: XL(:),    A_AVG(:),     AL(:), AR(:), DA(:), A6(:),
     ^          SMALLDA(:), UNSMOOTH(:), DISCONT(:)
c
c   The following arrays will hold  various geometrical factors which can be
c   precomputed for efficiency when several quantities are interpolated on the
c   same grid:

      ALLOCATABLE :: DX(:), DALFAC(:), DARFAC(:),  FACDA(:), FCDAZL(:),
     ^           FACDAL(:), FA6DAL(:), FA6DAR(:), FRRDAL(:), FRRDAR(:),
     ^           FLLDAL(:), FLLDAR(:)


      OPEN (unit = 11, file ='INPUT.AVG')
c  The data consists of N+2 lines.  The first line
c  contains the integer N, the number to zone averages in the data set.
c  This is followed by N lines, each line having two numbers, a value for
c  the location of the left edge of the zone XL, and the zone average for
c  that zone A_AVG.  The lines are assumed to be in sequential order according
c  to XL.  An additional line has only one value, the final left zone edge.  A
c  grid of N zones requires N+1 left zone edge locations. An appropriate data
c  set can be created from the program CREATE_ZONE_AVERAGES.  The XL values
c   are used for plotting, not in the interpolation process.


c   Read in the number of zones in the file
            READ (11,*)  N
c
c   Next, allocate the appropriate space
          ALLOCATE        (XL(1-NBDY:N+1+NBDY))
          ALLOCATE        (AL(1-NBDY:N+1+NBDY))
          ALLOCATE        (AR(1-NBDY:N+1+NBDY))
          ALLOCATE        (DA(1-NBDY:N+NBDY))
          ALLOCATE        (A6(1-NBDY:N+NBDY))
          ALLOCATE     (A_AVG(1-NBDY:N+NBDY))
          ALLOCATE   (SMALLDA(1-NBDY:N+NBDY))
```

```
          ALLOCATE  (UNSMOOTH(1-NBDY:N+NBDY))
          ALLOCATE   (DISCONT(1-NBDY:N+NBDY))

c   The following arrays will hold  various geometrical factors which can be
c   precomputed for efficiency when several quantities are interpolated on the
c   same grid:
          ALLOCATE        (DX(1-NBDY:N+NBDY))
          ALLOCATE    (DALFAC(1-NBDY:N+NBDY))
          ALLOCATE    (DARFAC(1-NBDY:N+NBDY))
          ALLOCATE     (FACDA(1-NBDY:N+NBDY))
          ALLOCATE    (FCDAZL(1-NBDY:N+NBDY))
          ALLOCATE    (FACDAL(1-NBDY:N+NBDY))
          ALLOCATE    (FA6DAL(1-NBDY:N+NBDY))
          ALLOCATE    (FA6DAR(1-NBDY:N+NBDY))
          ALLOCATE    (FRRDAL(1-NBDY:N+NBDY))
          ALLOCATE    (FRRDAR(1-NBDY:N+NBDY))
          ALLOCATE    (FLLDAL(1-NBDY:N+NBDY))
          ALLOCATE    (FLLDAR(1-NBDY:N+NBDY))


      DO I=1,N
         READ (11,*) XL(I),A_AVG(I)
      ENDDO
      READ (11,*) XL(N+1)
c
c
c   Query the user for desired boundary conditions
c
      PRINT *,' Please enter 0 for reflecting boundary conditions'
      PRINT *,'        or    1 for periodic boundary conditions'
      READ *, IBDY_TYPE
      IF ( IBDY_TYPE .EQ. 0 ) THEN
         PRINT *,' Selecting reflecting boundary conditions'
      ELSE
         IF ( IBDY_TYPE .EQ. 1 ) THEN
            PRINT *,' Selecting periodic boundary conditions'
         ELSE
            PRINT *,' Unknown boundary type:',IBDY_TYPE
            call abort()
         ENDIF

      ENDIF
c
c   Initialize the fake zones:
c
          IBDY_TYPE = 1
c
          CALL BOUNDARY( XL(1-NBDY), A_AVG(1-NBDY), IBDY_TYPE, N,NBDY )
c

c
c   A uniform grid is NOT assumed.  Therefore, various
c   geometrical factors must be precomputed by a call to
c   _INTERFACE_FACTORS, input to  which consists of DX(I),
c   the individual zone widths.

      DO I=1-NBDY,N+NBDY
         DX(I) = XL(I+1) - XL(I)
      ENDDO


      CALL PPM98_INTERPOLATION_FACTORS (  DX(1-NBDY), DALFAC(1-NBDY),
     ^                DARFAC(1-NBDY),  FACDA(1-NBDY), FCDAZL(1-NBDY),
     ^                FACDAL(1-NBDY), FA6DAL(1-NBDY), FA6DAR(1-NBDY),
     ^           FRRDAL(1-NBDY), FRRDAR(1-NBDY), FLLDAL(1-NBDY),
     ^           FLLDAR(1-NBDY), N, NBDY)



c   Fill an array, SMALLDA, which contains what is to be considered a trivial difference of A_AVG.
c   In this Program, A_AVG is not positive definite, so a small value is appropriate.

      DO I=1-NBDY,N+NBDY
         SMALLDA(I) = 0.0005
      ENDDO

c   Interpolate zone interface values and determine ppm parabola coefficients
c
      CALL PPM98_INTERP_DISCONTINUITY (  DALFAC(1-NBDY), DARFAC(1-NBDY),
     ^                 FACDA(1-NBDY),    FCDAZL(1-NBDY), FACDAL(1-NBDY),
     ^                FA6DAL(1-NBDY),    FA6DAR(1-NBDY), FLLDAL(1-NBDY),
     ^                FLLDAR(1-NBDY),    FRRDAL(1-NBDY), FRRDAR(1-NBDY),
     ^               SMALLDA(1-NBDY),     A_AVG(1-NBDY),     AL(1-NBDY),
     ^                    AR(1-NBDY),        DA(1-NBDY),     A6(1-NBDY),
     ^              UNSMOOTH(1-NBDY),   DISCONT(1-NBDY),      N, NBDY )

c   AL(I), AR(I), DA(I), A6(I), are defined for 1<= I <= N

c   Write out the interpolations for plotting [11 samples per zone]


      DO I=1,N
         DO J=0,10
            XX = 0.1*FLOAT(J)
            A_INTERP = AL(I) + XX * (DA(I) + A6(I) * (1.0 - XX))
            WRITE(6,100) I, XL(I)+XX*DX(I), A_INTERP, UNSMOOTH(I), DISCONT(I)
100         FORMAT( I4,1P, 4(1X,e12.5))
         ENDDO
```

```
      ENDDO

      DEALLOCATE (XL, A_AVG, AL, AR, DA, A6, SMALLDA, UNSMOOTH, DISCONT)
      DEALLOCATE (DX, DALFAC, DARFAC,  FACDA, FCDAZL,
     ^         FACDAL, FA6DAL, FA6DAR, FRRDAL, FRRDAR,
     ^         FLLDAL, FLLDAR)

      END ! PROGRAM INTERPOLATE_4
```

The effects of discontinuity detection and steepening are illustrated by program `INTERPOLATE_4`. By this time the reader should be familiar with the basic structure, which is the same as the previous example. A few additional arrays must be allocated, DISCONT, which returns the results of discontinuity detection, and `DA` and `A6`, the PPMLIB piecewise parabolic coefficients. These coefficients are calculated during the discontinuity detection process, and are therefore passed back to be calling routine in the interests of efficiency.

The zone averaged data are read in, and the necessary geometrical factors are computed, which are the same factors as required in INTERPOLATE_3. Then follows a call to **PPM98_INTERP_DISCONTINUITY**, the PPMLIB routine which performs an interpolation with steepening on a non-uniform grid. This replaces the call to **PPM98_INTERP_MONO**. The same geometrical factors are required by either, so a call to PPM98_INTERFACE_FACTORS is required, as before.

Figure 7 shows results from program INTERPOLATE_4. The same data as for the previous two examples is used, data generated from the series of hyperbolic tangent functions on a non-uniform grid. The actual function used to generate the zone averages is shown as a dotted line. The monotonized, steepened interpolation, computed by INTERPOLATE_4, is shown as a dashed line. The result is an improved interpolation over the previous case. It should be noted that the discontinuities at the edges of the central zones do not violate monotonicity, as used in PPMLIB. It is in the context of advection that monotonicity is imposed, not on a point by point basis.

i. Interpolation of Inexact Differentials

Occasionally, there exist quantities which are not known, or are in fact not knowable, as zone averaged quantities, but are known only as inexact differentials. Therefore, there exist differential forms of all the interpolation routines in PPMLIB. Riemann invariants provide an example from hydrodynamics. For a gamma law gas, the plus Riemann invariant, R^+, is

$$dR^+ = du + \frac{dp}{C}\,, \qquad 2.25$$

where u is the velocity, p the pressure, and C the Lagrangian sound speed, $C = \sqrt{\gamma \rho p}$. When the state quantities are known only as zone averages, dR^+ is most easily expressible as a set of differences centered on a zone interface:

$$dR^+_{i-1/2} = (u_i - u_{i-1}) + \frac{p_i - p_{i-1}}{1/2\,(C_i + C_{i-1})}\,. \qquad 2.26$$

Here, the Riemann invariant has been expressed as *left differences* centered on the left zone interface. Examination of the equations underlying the interpolation routines described in the previous section shows that they have been expressed as left differences whenever possible. Whenever a differential quantity is interpolated, the assumption of zero zone average is made in each zone. Thus if a reference to values from neighboring zones is required, these values may be renormalized using the differences provided as the input data for the interpolation. Further use of this will be made in hydrodynamics.

j. Integration of the Interpolations

A common procedure is to determine the average value of an interpolated quantity over some portion of a zone adjacent to either of its zone interfaces $x_{i-1/2}$ or $x_{i+1/2}$. Thus it will prove advantageous to develop expressions to integrate eqn (2.1) over such portions of zones. Consider first the average of the interpolation over a portion of the zone abutting the lefthand zone interface $x_{i-1/2}$ (figure 8a). This is

$$\frac{1}{\int\limits_{x_{i-1/2}}^{x'} dx} \int\limits_{x_{i-1/2}}^{x'} a(x)dx, \quad x_{i-1/2} \le x' \le x_{i+1/2}\,.$$

In terms of the normalized coordinate of eqn (2.1) this becomes

$$\frac{1}{f}\int\limits_0^f a(\tilde{x})\,d\tilde{x} = a_i^L + \frac{f}{2}\left[(\Delta a_i + (1 - \frac{2}{3}f)\hat{a}_i\right] \qquad 2.27$$

where f is the normalized volume fraction

$$f = \frac{(x' - x_{i-1/2})}{\Delta x_i}, \quad 0 \le f \le 1,$$

over which the integration is performed. Similarly, the average of the interpolation over a portion of the zone adjacent to the righthand zone interface $x_{i+1/2}$ (figure 8b) is

$$\frac{1}{\int\limits_{x'}^{x_{i+1/2}} dx} \int\limits_{x'}^{x_{i+1/2}} a(x)dx, \quad x_{i-i-1/2} \le x' \le x_{i+1/2}\,,$$

or

$$\frac{1}{f}\int\limits_{1-f}^{1} a(\tilde{x})\,d\tilde{x} = a_i^R - \frac{f}{2}\left[(\Delta a_i - (1 - \frac{2}{3}f)\hat{a}_i\right] \qquad 2.28$$

where now and the normalized volume fraction is (defined slightly differently, so that is is not negative)

$$f = \frac{(x_{i+1/2} - x')}{\Delta x_i}, \quad 0 \le f \le 1,$$

One situation requiring the integration of the interpolations arises when one needs to map a zone-averaged quantity on one grid to zone averages on another grid. Advection and grid refinement are two examples of this occurrence.

Figure 9 illustrates one approach to a remap of a distribution from one grid zone (from the ‘‘old’’ grid, defined by the zone interface locations $x_{j\pm 1/2}$) to another zone (from the ‘‘new’’ grid, defined by the interface locations $x'_{j\pm 1/2}$.) A particularly straightforward procedure can be derived when the new zone interfaces locations do not differ by more than one zone width from the old zone interface locations. That is,

$$x_{j-3/2} \le x'_{j-1/2} \le x_{j+1/2}$$

for all j.

First, the interpolations on the old grid are determined. Then the regions of overlap, if any, between the two grid are determined at each zone interface:

$$\delta x_{i-1/2} = x' - x_{i-1/2}\,. \qquad 2.29$$

:These are the shaded portions shown in the top of figure 9. Where the two grids are coincident, there is no overlap region. These are converted to fractional volumes at the lefthand zone interfaces via

$$f_{i-1/2} = \begin{cases} \dfrac{-\delta x_{i-1/2}}{\Delta x_{i-1}} & \text{if} \quad \delta x_{i-1/2} < 0.0, \\[2ex] \dfrac{\delta x_{i-1/2}}{\Delta x_i} & \quad \text{otherwise.} \end{cases} \qquad 2.30$$

Averages at each zone interface of the interpolations $A_{i-1/2}$ over the adjacent overlap region are found next.

$$A_{i-1/2} = \begin{cases} a^L_{i-1} + \dfrac{f_{i-1/2}}{2}\left[(\Delta a_{i-1} + (1 - \dfrac{2}{3} f_{i-1/2})\hat{a}_{i-1}\right] & \text{if} \quad \delta x_{i-1/2} < 0.0, \\ a^R_i - \dfrac{f_{i-1/2}}{2}\left[(\Delta a_i - (1 - \dfrac{2}{3} f_{i-1/2})\hat{a}_i\right] & \text{otherwise.} \end{cases} \qquad 2.31$$

It is evident from figure 9 that the the lefthand zone interface *flux* of a is defined to be

$$\alpha_{i-1/2} = -\delta x_{i-1/2} \; A_{i-1/2}, \qquad 2.32$$

then a conservative remap of zone i could be written then be

$$a'_i = \frac{(a_i \Delta x_i + \alpha_{i-1/2} - \alpha_{i+1/2})}{\Delta x'_i}, \qquad 2.33$$

where

$$\Delta x_i = x_{i+1/2} - x_{i-1/2}$$

and

$$\Delta x'_i = x'_{i+1/2} - x'_{i-1/2}.$$

The following is an example of using piecewise parabolic interpolation for map zone averages from one grid to another.

```
      PROGRAM ADVECT_1
c
c   This program developed from  the  INTERPOLATE_4 example.
c
c   This program creates piecewise parabolic interpolations
c   from zone averaged data, without assuming a uniform grid.
c   Monotonicity constraints are applied. For those zones
c   considered to be in a discontinuity, an interpolation more
c   appropriate for discontinuous functions is used, producing
c   a better representation of unresolved shocks.  A second
c   non-uniform grid (grid B) offset (by less than one zone) from
c   the initial grid (grid A) is created.  Appropriate fluxes
c   are computed which will allow the determination of the zone
c   averages on grid B using the zone averages on the grid A.
c   These fluxes are found by integrating the piecewise
c   parabolic interpolations (determined on grid A)  over the
c   regions of overlap between the two grids.
c
c   The number of boundary zones is: 3 for interpolation and 1 for
c   integration = 4. To determine the lefthand zone interface flux
c   for zone I, the interpolation in zone I-1 may be required,
c   depending upon zone motion.  Likewise, to determine the
c   righthand zone interface flux for zone I, the interpolation in
c   zone I+1 may be required.
c
      PARAMETER ( NBDY = 4 )

      ALLOCATABLE :: XL(:),    A_AVG(:),     AL(:),      AR(:), DA(:), A6(:),
     ^          SMALLDA(:), UNSMOOTH(:), DISCONT(:),     DX(:),     DXINV(:),
     ^             XLNU(:),  A_AVGNU(:),   DVOLL(:), DVOLFL(:),   DVOLFL1(:),
     ^              DAL(:)

      ALLOCATABLE ::        DALFAC(:), DARFAC(:),  FACDA(:), FCDAZL(:),
     ^           FACDAL(:), FA6DAL(:), FA6DAR(:), FRRDAL(:), FRRDAR(:),
     ^           FLLDAL(:), FLLDAR(:)

c

      OPEN (unit = 11, file ='INPUT.AVG')
c  The data consists of N+2 lines.  The first line
c  contains the integer N, the number to zone averages in the data set.
c  This is followed by N lines, each line having two numbers, a value for
c  the location of the left edge of the zone XL, and the zone average for
c  that zone A_AVG.  The lines are assumed to be in sequential order according
c  to XL.  An additional line has only one value, the final left zone edge.  A
c  grid of N zones requires N+1 left zone edge locations. An appropriate data
c  set can be created from the program CREATE_ZONE_AVERAGES.  The XL values
c   are used for plotting, not in the interpolation process.
```

```
c   Read in the number of zones in the file
            READ (11,*)  N
c
c   Next, allocate the appropriate space
          ALLOCATE         (XL(1-NBDY:N+1+NBDY))
          ALLOCATE         (AL(1-NBDY:N+1+NBDY))
          ALLOCATE         (AR(1-NBDY:N+1+NBDY))
          ALLOCATE         (DA(1-NBDY:N+NBDY))
          ALLOCATE         (A6(1-NBDY:N+NBDY))
          ALLOCATE      (A_AVG(1-NBDY:N+NBDY))
          ALLOCATE    (SMALLDA(1-NBDY:N+NBDY))
          ALLOCATE   (UNSMOOTH(1-NBDY:N+NBDY))
          ALLOCATE    (DISCONT(1-NBDY:N+NBDY))
          ALLOCATE         (DX(1-NBDY:N+NBDY))
          ALLOCATE      (DXINV(1-NBDY:N+NBDY))
          ALLOCATE       (XLNU(1-NBDY:N+NBDY))
          ALLOCATE    (A_AVGNU(1-NBDY:N+NBDY))
          ALLOCATE      (DVOLL(1-NBDY:N+NBDY))
          ALLOCATE     (DVOLFL(1-NBDY:N+NBDY))
          ALLOCATE    (DVOLFL1(1-NBDY:N+NBDY))
          ALLOCATE        (DAL(1-NBDY:N+NBDY))
c   The following arrays will hold  various geometrical factors which can be
c   precomputed for efficiency when several quantities are interpolated on the
c   same grid:
          ALLOCATE     (DALFAC(1-NBDY:N+NBDY))
          ALLOCATE     (DARFAC(1-NBDY:N+NBDY))
          ALLOCATE      (FACDA(1-NBDY:N+NBDY))
          ALLOCATE     (FCDAZL(1-NBDY:N+NBDY))
          ALLOCATE     (FACDAL(1-NBDY:N+NBDY))
          ALLOCATE     (FA6DAL(1-NBDY:N+NBDY))
          ALLOCATE     (FA6DAR(1-NBDY:N+NBDY))
          ALLOCATE     (FRRDAL(1-NBDY:N+NBDY))
          ALLOCATE     (FRRDAR(1-NBDY:N+NBDY))
          ALLOCATE     (FLLDAL(1-NBDY:N+NBDY))
          ALLOCATE     (FLLDAR(1-NBDY:N+NBDY))




      DO I=1,N
         READ (11,*) XL(I),A_AVG(I)
      ENDDO
      READ (11,*) XL(N+1)
c
c
c   Query the user for desired boundary conditions
c
      PRINT *,' Please enter 0 for reflecting boundary conditions'
      PRINT *,'        or    1 for periodic boundary conditions'
      READ *, IBDY_TYPE
      IF ( IBDY_TYPE .EQ. 0 ) THEN
         PRINT *,' Selecting reflecting boundary conditions'
      ELSE
         IF ( IBDY_TYPE .EQ. 1 ) THEN
            PRINT *,' Selecting periodic boundary conditions'
         ELSE
            PRINT *,' Unknown boundary type:',IBDY_TYPE
            call abort()
         ENDIF

      ENDIF
c
c   Initialize the fake zones:


          CALL BOUNDARY( XL(1-NBDY), A_AVG(1-NBDY), IBDY_TYPE, N,NBDY )
c
c   A uniform grid is NOT assumed.  Therefore, various
c   geometrical factors must be precomputed by a call to
c   _INTERFACE_FACTORS, input to  which consists of DX(I),
c   the individual zone widths.

      DO I=1-NBDY,N+NBDY
         DX(I) = XL(I+1) - XL(I)
         DXINV(I) = 1.0 / DX(I)
      ENDDO


      CALL PPM98_INTERPOLATION_FACTORS (  DX(1-NBDY), DALFAC(1-NBDY),
     ^                DARFAC(1-NBDY),  FACDA(1-NBDY), FCDAZL(1-NBDY),
     ^                FACDAL(1-NBDY), FA6DAL(1-NBDY), FA6DAR(1-NBDY),
     ^           FRRDAL(1-NBDY), FRRDAR(1-NBDY), FLLDAL(1-NBDY),
     ^           FLLDAR(1-NBDY), N, NBDY)


c
c   Fill an array, SMALLDA, which contains what is to
c   be considered a trivial difference of A_AVG.
c   In this Program, A_AVG is not positive definite,
c   so a small value is appropriate.

      DO I=1-NBDY,N+NBDY
         SMALLDA(I) = 0.0005
      ENDDO

c
c   Interpolate zone interface values and determine ppm parabola coefficients
```

```
c
       CALL PPM98_INTERP_DISCONTINUITY (  DALFAC(1-NBDY), DARFAC(1-NBDY),
     ^                  FACDA(1-NBDY),    FCDAZL(1-NBDY), FACDAL(1-NBDY),
     ^                 FA6DAL(1-NBDY),    FA6DAR(1-NBDY), FLLDAL(1-NBDY),
     ^                 FLLDAR(1-NBDY),    FRRDAL(1-NBDY), FRRDAR(1-NBDY),
     ^                SMALLDA(1-NBDY),     A_AVG(1-NBDY),     AL(1-NBDY),
     ^                     AR(1-NBDY),        DA(1-NBDY),     A6(1-NBDY),
     ^               UNSMOOTH(1-NBDY),   DISCONT(1-NBDY),      N, NBDY )


c   AL(I), AR(I), DA(I), A6(I), are defined for 4-NBDY<= I <= N + NBDY - 3

c
1      PRINT *,' Graciously enter the fractional offset (between -1.0 and 1.0)'
          PRINT *,' between the old and new grids: DXF.  '
          READ *,DXF
          IF( DXF .LT. -1.0  .OR. DXF .GT. 1.0 ) THEN
             PRINT *,' You entered',DXF, ' Do please try again.'
          GOTO 1
          ENDIF
             PRINT *,' The grids will be separated by ',DXF,' times '
             PRINT *,' the smallest zone width.  Thank you.'
c
c   Find the minimum zone width:
c
          DXMIN = DX(1-NBDY)
       DO I=2-NBDY,N+NBDY
          DXMIN = AMIN1( DXMIN, DX(I))
       ENDDO
c
c   Determine the new grid
c
       DO i=2-NBDY,N+NBDY
             XLNU(I) = XL(I) + DXF * DXMIN
       ENDDO

c
c   While the following operations are trivial given the constant
c   offset, they are explicitly computed here to show how the
c   general case would proceed
c
c   Find advected volumes at left hand zone interfaces,
c   also find advected volume fractions at left hand zone interfaces.
       DO i=2-NBDY,N+NBDY
c
c      Advected volume:
          DVOLL(I) = XLNU(I) - XL(I)
c      Advected volume fraction:
          IF ( DVOLL(I) .LE. 0.0 ) THEN
             DVOLFL(I) = - DVOLL(I) * DXINV(I-1)
          ELSE
             DVOLFL(I) =   DVOLL(I) * DXINV(I)
          ENDIF
          DVOLFL1(I) = 1.0 - DVOLFL(I)
       ENDDO

c
c   Find the fluxes of lefthand zone interfaces fluxes of A (DAL) which will
c   map zone averages of A from the grid XL to the grid XLNU:
c
c   Modify offsets and boundary zone count because 3 fake zones have
c   been lost to the interpolation.
c
       NNBDY = NBDY-3
       MBDY = NBDY-3
c
       CALL PPM98_FLUX( AL(1-NNBDY), A_AVG(1-NNBDY), AR(1-NNBDY), DA(1-NNBDY),
     ^     DVOLL(1-NNBDY), DVOLFL(1-NNBDY), DVOLFL1(1-NNBDY), DAL(1-NNBDY),
     ^       N, MBDY )

c
c   Now determine the new zone averages
c
       DO I=1,N
       DXNEW = XLNU(I+1)-XLNU(I)
       print *,' dx,dxnu=',dx(i),dxnew,dxnew*dxinv(i)
       A_AVGNU(I) = ( A_AVG(I) * DX(I) + DAL(I) - DAL(I+1) ) / DXNEW
       ENDDO


c
c   Display the old and new zone averages
       PRINT *,'  I      XMID        A_AVG       XMIDNU       A_AVGNU'
       DO I=1,N
       XMID = 0.5*(XL(I)+XL(I+1))
       XMIDNU = 0.5*(XLNU(I)+XLNU(I+1))
             WRITE(6,100) I, XMID, A_AVG(I), XMIDNU, A_AVGNU(I)
       ENDDO
100          FORMAT( I4,1P, 4(1X,e12.5))


       DEALLOCATE  ( XL,    A_AVG,     AL,      AR, DA, A6,
     ^          SMALLDA, UNSMOOTH, DISCONT,     DX,     DXINV,
     ^             XLNU,  A_AVGNU,   DVOLL, DVOLFL,   DVOLFL1,
     ^              DAL )
```

```
      END ! PROGRAM ADVECT_1
```

The alert reader will find most of this example, ADVECT_1, familiar from the example INTERPOLATE_4. It should be noted however, that the number of fake zones required is now four. The interpolation process consumes its usual three. But now, in order to allow for arbitrary motion of the grid, an additional fake zone is required. Consider that for zone i, reference to the interpolation from zone $i-1$ may be required in order to construct a lefthand zone interface flux (cf. eqn 2.27-2.28), depending upon the relative motion of the two grids. Likewise, in order to construct a *righthand* zone interface flux for zone i, reference to the interpolation in zone $i+1$ may be required. Data are read in in the usual manner, and values in the fake zones are initialized according to the chosen boundary conditions. The interpolations are performed allowing for discontinuities and the application of monotonicity constraints.

The user is carefully queried about the desired fractional grid offset. The remap method followed by this example requires the the offset be no greater than one zone (left or right). Because the user expresses the zone offset as a zone fraction (between -1 and 1) it is first necessary to find the minimum zone width on the entire. The (constant) offset of the lefthand zone interface locations are then the desired fractional zone offset decided upon by the user times this minimum zone width. The operations which then closely follow are trivial in this context of a constant offset of two grids, but are included to familiarize the user with the definitions of various quantities and the general procedure followed when the lefthand zone interface location offsets are not constant.

First the advected volume (here the offset between the two grids) is found at each zone interface, and stored in `DVOLL`. In general, this offset could be spatial, volumetric or inertial, depending upon the requirements of the problem. (Some quantities are more appropriately mass weighted, for example.) This is then converted into a fractional volume, stored in `DVOLFL` (which is always zero or greater). For efficiency when several quantities are mapped at once, the complement of this volume fraction is also found, (`DVOLFL1`). A call to **PPM98_FLUX** then computes the lefthand zone interface fluxes of the interpolated quantity (cf. eqns 2.27-2.28). The reader should note that the array offsets and number of fake zones used in the call to **PPM98_FLUX** *have been adjusted to account for the boundary zones lost due to the interpolation process*. The new zone averages are then be found using eqn 2.33.

Results from this program are shown in figure 10. The initial data consist of zone averages of a single period sine wave on a 20 zone non-uniform grid created by the program `CREATE_ZONE_AVERAGES_SINE` given in the introductory section. This initial zone average data distribution is shown as dotted lines. The plot on the left shows the results of advecting the distribution leftward by an amount equal to the smallest zone width (`DXF` = -1.0) as input to `ADVECT_2`, displayed as a solid line. In plot on the right, a solid line shows the results of advecting the distribution rightward by an amount equal to the smallest zone width (`DXF` = 1.0) as input to `ADVECT_2`. It is difficult to evaluate the results of only one small advection step on the quality of the zone-average distribution. The next example program will develop a program better suited to assessing the quality of the PPMLIB interpolations.

The operations of the interpolation of a quantity, its integration over indicated zone fractions, and completed by updating the zone averages is a frequent occurrence. They have therefore been grouped together into the routine **PPM98_ADVECT_PASSIVE** or (**PPM98_ADVECT_PASSIVE0** for uniform grids). Their use is illustrated in the following example in which a distribution of zone averages is advected through a grid. This involves several iterations of the mapping process described above.

```
      PROGRAM ADVECT_2

c
c   This program advects a given distribution of zone averages through
c   a numerical grid at a fixed Courant number.  Periodic boundary
c   conditions are assumed, as is a uniform grid.
c
c
      PARAMETER ( NBDY = 4 )

      ALLOCATABLE :: XL(:),     A_AVG(:),      AL(:), AR(:), DA(:), A6(:),
     ^          SMALLDA(:), UNSMOOTH(:), DISCONT(:), DX(:),       DAL(:),
     ^          DVOLL(:),     DVOLFL(:),     XLNU(:),       A_AVGNU(:),
     ^          DXINV(:)
c
```

```
      OPEN (unit = 11, file ='INPUT.AVG')
c  The data consists of N+2 lines.  The first line
c  contains the integer N, the number to zone averages in the data set.
c  This is followed by N lines, each line having two numbers, a value for
c  the location of the left edge of the zone XL, and the zone average for
c  that zone A_AVG.  The lines are assumed to be in sequential order according
c  to XL.  An additional line has only one value, the final left zone edge.  A
c  grid of N zones requires N+1 left zone edge locations. An appropriate data
c  set can be created from the program CREATE_ZONE_AVERAGES.  The XL values
c   are used for plotting, not in the interpolation process.


c   Read in the number of zones in the file
            READ (11,*)  N
c
c   Next, allocate the appropriate space
          ALLOCATE        (XL(1-NBDY:N+1+NBDY))
          ALLOCATE        (AL(1-NBDY:N+1+NBDY))
          ALLOCATE        (AR(1-NBDY:N+1+NBDY))
          ALLOCATE        (DA(1-NBDY:N+NBDY))
          ALLOCATE        (A6(1-NBDY:N+NBDY))
          ALLOCATE     (A_AVG(1-NBDY:N+NBDY))
          ALLOCATE       (DAL(1-NBDY:N+NBDY))
          ALLOCATE      (XLNU(1-NBDY:N+1+NBDY))
          ALLOCATE   (A_AVGNU(1-NBDY:N+NBDY))
          ALLOCATE   (SMALLDA(1-NBDY:N+NBDY))
          ALLOCATE  (UNSMOOTH(1-NBDY:N+NBDY))
          ALLOCATE   (DISCONT(1-NBDY:N+NBDY))
          ALLOCATE   (DVOLL(1-NBDY:N+NBDY))
          ALLOCATE     (DVOLFL(1-NBDY:N+NBDY))

c   The following arrays will hold  various geometrical factors
          ALLOCATE        (DX(1-NBDY:N+NBDY))
          ALLOCATE     (DXINV(1-NBDY:N+NBDY))


      DO I=1,N
         READ (11,*) XL(I),A_AVG(I)
      ENDDO
      READ (11,*) XL(N+1)
c
c
1     PRINT *,' Carefully enter the fractional offset (between -1.0 and 1.0)'
         PRINT *,' between the two grids: DXF.  per iteration '
         READ *,DXF
         IF( DXF .LT. -1.0  .OR. DXF .GT. 1.0 ) THEN
            PRINT *,' You entered',DXF, ' Please try again.'
      GOTO 1
         ENDIF
            PRINT *,' The grids will be separated by ',DXF,' zone widths '
            PRINT *,' Thank you.'

c
c
c   Query the user for desired number of revolutions'
c   through the grid'
c
c
2     PRINT *,' Please enter the number of revolutions'
      PRINT *,'    REVS > 0.0 (fractions allowed)'
      READ *, REVS
      IF( REVS .LT. 0.0 ) GOTO 2
c
c   at a Courant number of COURNO, a grid of N zones requires N/|COURNO|
c   iterations for one revolution thus:
c
      NITER = INT ( FLOAT(N) * REVS / ABS(DXF) )
c
c   we will always perform at least 1 iteration
c
      NITER = MAX (1,NITER)

      print *,' NITER = ',niter
c
c   Now perform the required number of iterations.
c
      DO ITER = 1,NITER
c
c   Initialize the fake zones assuming periodic boundary conditions:

             IBDY_TYPE = 1
             CALL BOUNDARY( XL(1-NBDY), A_AVG(1-NBDY), IBDY_TYPE, N,NBDY )
c

c
c   While a uniform grid is assumed, it is convenient to calculate
c   the individual zone widths.

         DO I=1-NBDY,N+NBDY
            DX(I) = XL(I+1) - XL(I)
            DXINV(I) = 1.0 / DX(I)
         ENDDO
c
c
c    Determine the locations of the lefthand zone interface of the new grid
c    (The first and last interface locations will not be needed)
c
         DO I=2-NBDY,N+NBDY
```

```
          IF (DXF .LE. 0.0)   THEN
        XLNU(I) = XL(I) + DXF * DX(I-1)
      ELSE
        XLNU(I) = XL(I) + DXF * DX(I)
      ENDIF
         ENDDO


c   Fill an array, SMALLDA, which contains what is to
c   be considered a trivial difference of A_AVG.
c   In this Program, A_AVG is not positive definite,
c   so a small value is appropriate.

         DO I=1-NBDY,N+NBDY
            SMALLDA(I) = 0.0005
         ENDDO


         DO I=2-NBDY,N+NBDY

c     Advected volume:
          DVOLL(i) = XL(I) - XLNU(I)
c     Advected volume fraction:
          IF (DVOLL(I) .GE. 0.0)   THEN
               DVOLFL(I) = DVOLL(I)*DXINV(i-1)
         ELSE
               DVOLFL(I) = -DVOLL(I)*DXINV(I)
c
         ENDIF
         ENDDO


c
c   Determine new zone-averages
c
c   PPM98_ADVECT_PASSIVE0 interplates A_AVG assuming a uniform grid,
c        with discontinuity detection and monotonicity constraints.
c        It returns the new zone averages A_AVGNU, as well as the
c        interface fluxes DAL and the results of the unsmoothness
c        and discontinuity detection calculations.

         CALL PPM98_ADVECT_PASSIVE0
     ^              ( DVOLFL(1-NBDY),   DVOLL(1-NBDY),       DX(1-NBDY),
     ^                  DXINV(1-NBDY),   A_AVG(1-NBDY),  SMALLDA(1-NBDY),
     ^               UNSMOOTH(1-NBDY), DISCONT(1-NBDY),      DAL(1-NBDY),
     ^                A_AVGNU(1-NBDY), N, NBDY )
c
c   In Because PPM98_ADVECT_PASSIVE0 does not overwrite the input
c   data, it is necessary to replace the values in the arrays for
c   the old data with the new values.  Ordinarily the grid would be
c   replaced as well, but that is not done here to ease comparisons.
c
         DO I=1,N
      A_AVG(I) = A_AVGNU(I)
         ENDDO
      ENDDO
c
c   Display the zone averages
      PRINT *,'  I       XMID        A_AVG'
      DO I=5-NBDY,N+NBDY-4
         XMID = 0.5*(XL(I+1)+XL(I))
         WRITE(6,100) I, XMID, A_AVG(I)
100   FORMAT(3x,i4,1p,4(1x,e12.5))
      ENDDO

      DEALLOCATE (XL, A_AVG, AL, AR, DA, A6, SMALLDA, UNSMOOTH, DISCONT)
      DEALLOCATE (DX, A_AVGNU, XLNU, DAL, DVOLL, DVOLFL, DXINV)

      END ! PROGRAM ADVECT_2
```

The initial setup is similar to the example ADVECT_1. In addition to the fractional offset between grids, the user is also queried for the number of revolutions through the grid. For example, one revolution would mean that the initial distribution is advected through N_{zones} (the total number of zones in the initial distribution) in several iterations of the mapping process used in example ADVECT_1, with each iteration having an offset of

$$f_{grid} \Delta x_{\min}$$

where f_{grid} is the fractional grid offset and $\Delta x_{\min}$ is the the minimum zone width on the initial grid. Thus, on a uniform grid, the total number of iterations $N_{iteration}$ will be

$$N_{iteration} = \frac{N_{zones}}{|f_{grid}|} .$$

In the example, at least one iteration will always be performed. Only periodic boundary conditions are allowed. After reading in the data and performing the usual allocation chores, the number of required iterations is computed. Each iteration consists of initializing the fake zones according to periodic boundary conditions, updating the zone averages by a call to **PPM98_ADVECT_PASSIVE0**, and replacing the initial zone averages with the new zone averages in preparation for the next iteration. The routine **PPM98_ADVECT_PASSIVE0** performs the operations of interpolation, integration, and updating described in the example `ADVECT_1`, but with the simplifying assumptions available with uniform grids. (The non-uniform grid version is **PPM98_ADVECT_PASSIVE**.)

Results from shown in figure 11 for initial data consisting of zone averages of a single period sine wave on a 20 zone uniform grid (unlike before) created by the program `CREATE_ZONE_AVERAGES_SINE`, from the introductory section. The initial zone average distribution is shown as a solid line. The result of 2000 iterations, specifically 10 revolutions at a fractional step size of 0.1, is shown as dotted line. Even after 2000 interpolations and integrations, the distribution is little changed.

k. Customizing your PPMLIB Interpolations

Of course, the inquiring reader noted long ago that the behaviour of the various interpolative routines with regard to monotonicity, discontinuity detection, and steepening of interpolations is controlled by a small number of internal parameters. These parameters have been set to default values determined by decades of experience to be valid in a wide range of circumstances. However, no true scientist was ever able to resist the temptation of take the cover off of anything and poke around. Likewise, it is acknowledged that some users may wish to ‘‘take the cover off’’ of PPMLIB, so to speak, and modify the default behaviours. Therefore, several functions have been created to satisfy this understandable, if mistaken, desire. For those whose childhood memories of prodding the innards of TV sets (or for some, radios) have been lost to the mists of time, they should be reminded that such activities could often produce results both unexpected and unpleasant. Users are therefore extremely cautioned that wide variance from the default parameters may result in unforeseen consequences and will, as always, void the warranty.

These routines are **PPM98_PRINT_CONSTANTS**, which prints out the list of user settable PPMLIB constants with their current values and keywords, **PPM98_SET_CONSTANTS** which allows the user to set a specific PPMLIB internal constant given its keyword and a value, and **PPM98_RESET_CONSTANTS** which reset all PPMLIB internal constants to their default values. It should be noted that once the value of a constant has been changed, it will remain so modified throughout the remaining program execution, unless changed again.

As an example, consider the estimation of how well the data are resolved (data smoothness), sec 2.k. This process is is controlled by two parameters, Σ_0 and Σ_1 in eqn. (2.16), repeated here:

$$U_i = \min(1, \max[0, \Sigma_1(\Sigma_i - \Sigma_0)]. \qquad 2.16$$

The following table gives the keywords controlling the smoothness calculation.

Equation Symbol	PPMLIB Keyword	Default
Σ_0	crterr	0.10
Σ_1	ferrfc	10.0

To understand the source of these constants, consider a sine wave on a uniform mesh:

$$a(x) = \sin(x).$$

The zone averaged values are then

$$a_i = \frac{2}{\Delta x} \sin(\frac{\Delta x}{2}) \sin(x_i)$$

where Δx is the uniform grid spacing, and x_i is the midpoint of the zone. Equation 2.13, the fractional error estimate (the error between the actual zone average and the average found from interpolations) can be approximated with a fractional error estimate centered on the left zone interface:

$$\Sigma_i \approx \frac{1}{2} \frac{|\sigma_i^b - a_{i+1}|}{|a_i - a_{i-1}| + |a_{i+1} - a_i| + \delta_i} \qquad 2.13$$

or

$$\Sigma_i \approx \frac{|\gamma_{i-1} - \gamma_i|}{|a_i - a_{i-1}|} \qquad 2.34$$

For a pure sine wave on a uniform grid

$$\gamma_i = -\frac{2}{\Delta x} \sin(\frac{\Delta x}{2}) \sin(x_i)(1 - \cos(\Delta x)).$$

Therefore, eqn 2.34 becomes

$$\Sigma_i \approx 1 - \cos(\Delta x).$$

The accompanying table shows the dependence for Σ_i on n, the number of zones over which the sine wave is resolved.

Σ_i	n
1.00	4
0.69	5
0.50	6
0.38	7
0.29	8
0.23	9
0.19	10
0.16	11
0.13	12
0.11	13
0.10	14
0.09	15

Thus by examining eqn 2.16 it is clear based on the default parameters, a 15 zone sine wave would be considered completely resolved (completely smooth) while a sine wave resolved over nine zones or fewer are considered completely unresolved for purposed of interpolation. Therefore the results shown in figure 11 are not surprising.

The results of 1200 iterations (60 revolutions at an offset of 0.5 zones per iteration) on a 10 zone sine wave are shown in figure 12. As this is near the interpolation resolution limit based on the default parameters, it is not surprising to see that the waveform has undergone significant modification. The monotonicity constraints have clipped the extrema causing the sine wave to evolve towards a square wave. Numerical dissipation has also reduced the amplitude.

If one wished to attempt to allow a 10 zone sine wave to pass unmolested, the above two parameters can be modified to

$$\Sigma_0 = 0.4$$

and

$$\Sigma_1 = 2.5$$

by inserting the statements

```
      call ppm98_set_constants(0.4,'crterr')
      call ppm98_set_constants(2.5,'ferrfc')
```

just before the beginning of the iteration do loop. (Not in it!) The results also shown in figure 12, for the same initial data and number of iterations as before. While the numerical dissipation is still present, the wave form is relatively unscathed.

The discontinuity detection and steepening process is also controlled by two parameters as shown by eqn (2.22):

$$S'_i = \min\left\{1, \max\left[0, J_1(J_i - J_0)\right]\right\}. \qquad 2.22$$

Their keywords and default values are given in the following table.

Equation Symbol	PPMLIB Keyword	Default
J_0	crtrat	0.05
J_1	stpner	25.0

These values were determined procedurally ensure that discontinuities one zone in width would have a steepness parameter of 1, regardless of the phase of the discontinuity with respect to the zone boundaries.

The following table lists PPMLIB user addressable constants related to interpolation.

Equation Symbol	PPMLIB Keyword	Default	Explanation
Σ_0, eqn 2.16	crterr	0.10	threshold for lack of smoothness This prevents identifying small jumps (such as those due to noise) as indicating that the data are poorly resolved Fractional errors smaller than this will indicated well resolved data
Σ_1, eqn 2.16	ferrfc	10.0	constant controlling the rapidity in which the lack of smoothness grows.
J_0 eqn 2.22	crtrat	0.05	threshold for discontinuity steepening such as those due to noise) as discontinuities
J_1 eqn 2.22	stpner	25.0	constant controlling the rapidity in which discontinuity steepening grows above its threshold
η_0 eqn 2.35	smlrat	0.1	special comparison jump threshold for discontinuity steepening Steepening may occur only when the ratio of smldif to cmpdif is greater than smlrat, where smldif is a small zone centered difference and cmpdif is a zone centered comparison difference

l. Interpolation Summary

This section on interpolation concludes with a listing of the names of available PPMLIB routines related to interpolation, with a brief description of their function.

Set-Up Routines:

PPM98_INTERPOLATION_FACTORS For computational efficiency, pre-computes various geometrical factors used by the PPMLIB interpolation routines. Need be called only once per grid.

PPM98_INTERFACE For computational efficiency, pre-computes various geometrical factors used by the PPMLIB interpolation routines **PPM98_INTERP_INTERFACE** and

PPM98_DIFTERP_INTERFACE. This routine computes a subset of the factors computed by **PPM98_INTERPOLATION_FACTORS**. Is not needed if that program has already been called for this grid.

For Zone-Averaged Data:

PPM98_INTERP0_INTERFACE Returns interpolated interface point values, on uniform grids. No monotonicity or Discontinuity detection are performed, thus is appropriate only for smooth, well resolved data.

PPM98INTERP_INTERFACE Nonuniform grid version of **PPM98INTERP0_interface**. Returns interpolated interface point values, No monotonicity or Discontinuity detection are performed, thus is appropriate only for smooth, well resolved data. Requires a previous call to **PPM98_INTERFACE**.

PPM98INTERP0_MONO Returns interpolated interface point values and a measure of the smoothness or resolution of the data. For use on uniform grids. No Discontinuity detection is performed, but monotonicity constraints applied where the data are judged to be not well resolved. Appropriate when discontinuities are not expected in the data.

PPM98INTERP_MONO Nonuniform grid version of **PPM98INTERP0_MONO**. Returns interpolated interface point values and a measure of the smoothness or resolution of the data. No Discontinuity detection is performed, but monotonicity constraints applied where the data are judged to be not well resolved. Appropriate when discontinuities are not expected in the data. Requires a previous call to **PPM98_INTERPOLATION_FACTORS**.

PPM98INTERP0_DISCONTINUITY Returns interpolated interface point values as well as a a measure of the smoothness or resolution of the data and a measure of how discontinuous the function is. For use on uniform grids. Discontinuity detection is performed and the interpolation modified to better represent jumps in the data. Monotonicity constraints are applied where the data are judged to be not well resolved. Appropriate when discontinuities may be present in the data.

PPM98INTERP_DISCONTINUITY The non-uniform grid version of **PPM98INTERP0_DISCONTINUITY**. Returns interpolated interface point values as well as a a measure of the smoothness or resolution of the data and a measure of how discontinuous the function is. For use on uniform grids. Discontinuity detection is performed and the interpolation modified to better represent jumps in the data. Monotonicity constraints are applied where the data are judged to be not well resolved. Appropriate when discontinuities may be present in the data. Requires a previous call to **PPM98_INTERPOLATION_FACTORS**.

PPM98INTERP0_DISCONT_COMP Same as **PPM98INTERP0_DISCONTINUITY**, but contains a comparison against a change in another quantity. Returns interpolated interface point values as well as a a measure of the smoothness or resolution of the data and a measure of how discontinuous the function is. For use on uniform grids. Discontinuity detection is performed and the interpolation modified to better represent jumps in the data. Monotonicity constraints are applied where the data are judged to be not well resolved. Generally used in hydrodynamics for the detection of contact discontinuities.

PPM98INTERP_DISCONT_COMP The non-uniform grid version of **PPM98INTERP0_DISCONT_COMP**, and is the same as **PPM98INTERP_DISCONTINUITY**, but contains a comparison against a change in another quantity. Returns interpolated interface point values as well as a a measure of the smoothness or resolution of the data and a measure of how discontinuous the function is. Discontinuity detection is performed and the interpolation modified to better represent jumps in the data. Monotonicity constraints are applied where the data are judged to be not well resolved. Generally used in hydrodynamics for the detection of contact discontinuities. Requires a previous call to **PPM98_INTERPOLATION_FACTORS**.

Zone-Averaged Differential Data

(That is, quantities which exist only as differences. The interpolation will assume a zero zone average.)

PPM98_DIFTERP0_INTERFACE Returns interpolated interface point values, assuming an average value of zero in each zone, on uniform grids. The differences are assumed centered on the left zone interface. No monotonicity or Discontinuity detection are performed, thus is appropriate only for smooth, well resolved data.

PPM98_DIFTERP_INTERFACE Nonuniform grid version of **PPM98_DIFTERP0_INTERFACE**. Returns interpolated interface point values assuming an average value of zero in each zone. The differences are assumed centered on the left zone interface. No monotonicity or Discontinuity detection are performed, thus is appropriate only for smooth, well resolved data. Requires a previous call to **PPM98_INTERFACE**.

PPM98_DIFTERP0_MONO Returns interpolated interface point values assuming an average value of zero in each zone, and a measure of the smoothness or resolution of the data. For use on uniform grids. The differences are assumed centered on the left zone interface. No Discontinuity detection is performed, but monotonicity constraints are applied where the data are judged to be not well resolved.

PPM98_DIFTERP_MONO Nonuniform grid version of **PPM98_DIFTERP0_MONO**. Returns interpolated interface point values assuming an average value of zero in each zone, and a a measure of the smoothness or resolution of the data. The differences are assumed centered on the left zone interface. No Discontinuity detection is performed, but monotonicity constraints are applied where the data are judged to be not well resolved. Requires a previous call to **PPM98_INTERPOLATION_FACTORS**.

PPM98_DIFTERP0_DISCONTINUITY Returns interpolated interface point values, assuming an average value of zero in each zone, a measure of the smoothness or resolution of the data, and a measure of how discontinuous the function is. For use on uniform grids. The differences are assumed centered on the left zone interface. Discontinuity detection is performed and the interpolation modified to better represent jumps in the data.

PPM98_DIFTERP_DISCONTINUITY Nonuniform grid version of **PPM98_DIFTERP0_DISCONTINUITY**. Returns interpolated interface point values, assuming an average value of zero in each zone, a measure of the smoothness or resolution of the data, and a measure of how discontinuous the function is. The differences are assumed centered on the left zone interface. Discontinuity detection is performed and the interpolation modified to better represent jumps in the data. Requires a previous call to **PPM98_INTERPOLATION_FACTORS**.

PPM98_DIFTERP0_DISCONT_COMP Is the same as **PPM98_DIFTERP0_DISCONTINUITY** but contains a comparison against a change in another quantity. Discontinuities will be detected and steepened only when

$$\frac{\delta a_i}{\delta c_i} > \eta_0, \qquad 2.35$$

where δa_i is what should be considered to be a smalle change in a, the interpolant, (Any differences below this value would be considered noise.) δc_i is a comparison difference based on some other relevant quantity, and η_0 is a PPMLIB constant, usually about 0.1. Returns interpolated interface point values, assuming an average value of zero in each zone, a measure of the smoothness or resolution of the data, and a measure of how discontinuous the function is. For use on uniform grids. The differences are assumed centered on the left zone interface. Discontinuity detection is performed and the interpolation modified to better represent jumps in the data. Generally used in hydrodynamics when interpolation the adiabatic Riemann invariant.

PPM98_DIFTERP_DISCONT_COMP Nonuniform grid version of **PPM98_DIFTERP0_DISCONT_COMP** and is the same as **PPM98_DIFTERP_DISCONTINUITY** but contains a comparison against a change in another quantity. Returns interpolated interface point values, assuming an average value of zero in each zone, a measure of the smoothness or resolution of the data, and a measure of how discontinuous the function is. The differences are assumed centered on the left zone interface. Discontinuity detection is performed and the interpolation modified to better represent jumps in the data. Generally used in hydrodynamics when interpolation the adiabatic Riemann invariant. Requires a previous call to **PPM98_INTERPOLATION_FACTORS**.

Related Programs

PPM98_MONO_PARABOLA Given zone averages, left and right interface point values for each zone, and a measure of how well the data are resolved per zone, calculated the PPM monotoniced piecewise parabolic coefficients.

PPM98_UNSMOOTH0 Returns the measure of the smoothness or resolution of the data. This is the same smoothness measure constructed in **PPM98INTERP0_MONO** but without performing the interpolation calculation. For use on uniform grids with zone averaged data.

PPM98_UNSMOOTH The non-uniform grid version of **PPM98_UNSMOOTH0**. Returns the measure of the smoothness or resolution of the data. This is the same smoothness measure constructed in **PPM98INTERP_MONO** but without performing the interpolation calculation. For use with zone averaged data.

PPM98_DISCONTINUITY0 Returns the measure of the smoothness or resolution of the data as well as the estimate of how discontinuous the function is. These are the same smoothness and steepness parameters computed in **PPM98_INTERP0_DISCONTINUITY** but without performing the interpolation calculation. For use on uniform grids with zone averaged data.

PPM98_DISCONTINUITY The non-uniform grid version of **PPM98_DISCONTINUITY0**. Returns the measure of the smoothness or resolution of the data as well as the estimate of how discontinuous the function is. These are the same smoothness and steepness parameters computed in **PPM98_INTERP_DISCONTINUITY** but without performing the interpolation calculation. For use with zone averaged data.