3.7
9/20/82
me

I-5429

(1)

dr.827

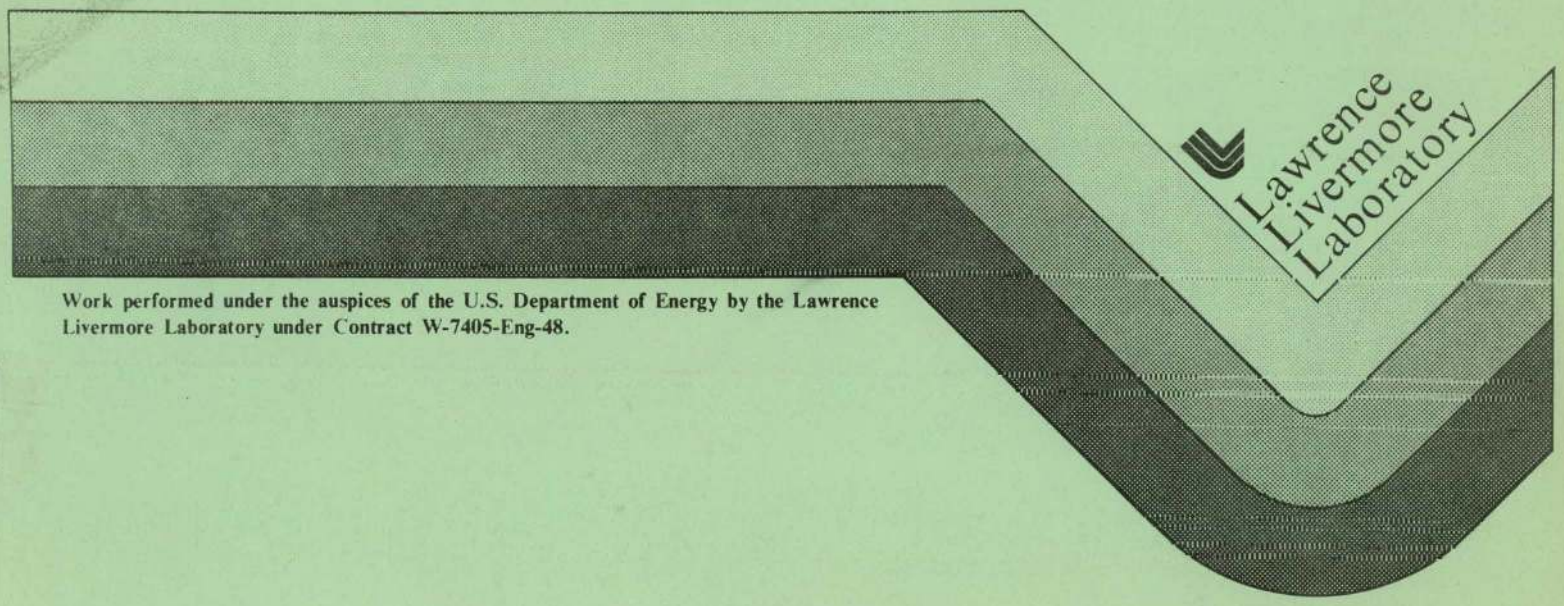
UCID-30194
COMPUTER DOCUMENTATION

PCHIP FINAL SPECIFICATIONS

MASTER

F. N. Fritsch

August 1982



Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-Eng-48.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

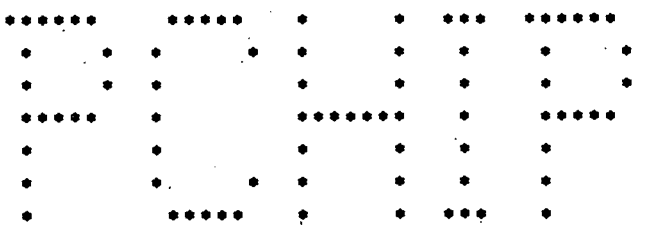
This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

UCID--30194

DE82 021008



Piecewise Cubic Hermite Interpolation Package

(Final Specifications)

Version 8.5 - 28 July 1982

by

Fred N. Fritsch

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Mathematics and Statistics Division

Lawrence Livermore National Laboratory

Livermore, California 94550

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

CONTENTS

INTRODUCTION

This document contains the specifications for PCHIP, a new Fortran package for piecewise cubic Hermite interpolation of data. It features software to produce a monotone and "visually pleasing" interpolant to monotone data. As is demonstrated in Reference 1, such an interpolant may be more reasonable than a cubic spline if the data contains both "steep" and "flat" sections. Interpolation of cumulative probability distribution functions is another application. (See References 1-3 for examples.)

All piecewise cubic functions in PCHIP are represented in cubic Hermite form; that is, f(x) is determined by its values f(i) and derivatives d(i) at the breakpoints x(i), i=1(1)n.

The contents of the package are as follows:

1. Determine derivative values.

   PCHIM — Piecewise Cubic Hermite Interpolation to Monotone data.
   Used if the data are monotonic or if the user wants to guarantee that the interpolant stays within the limits of the data. (See Reference 2.)

   PCHIC — Piecewise Cubic Hermite Interpolation Coefficients.
   Used if neither of the above conditions holds, or if the user wishes control over boundary derivatives. Will generally reproduce monotonicity on subintervals over which the data are monotonic..

   PCHSP — Piecewise Cubic Hermite SPline.
   Produces a cubic spline interpolator in cubic Hermite form. Provided primarily for easy comparison of the spline with other piecewise cubic interpolants. (A modified version of de Boor's CUBSPL, Reference 4.)

2. Evaluate, differentiate, or integrate resulting PCH function.

   NOTE: If derivative values are available from some other source, these routines can be used without calling any of the previous routines.

   CHFEV — Cubic Hermite Function EValuator.
   Evaluates a single cubic Hermite function at an array of points. Used when the interval is known, as in graphing applications. Called by PCHFE.

   PCHFE — Piecewise Cubic Hermite Function Evaluator.
   Used when the interval is unknown or the evaluation array spans more than one data interval.

CHFDV — Cubic Hermite Function and Derivative eValuator. Evaluates a single cubic Hermite function and its first derivative at an array of points. Used when the interval is known, as in graphing applications. Called by PCHFD.

PCHFD — Piecewise Cubic Hermite Function and Derivative evaluator. Used when the interval is unknown or the evaluation array spans more than one data interval.

PCHID — Piecewise Cubic Hermite Integrator, Data limits. Computes the definite integral of a piecewise cubic Hermite function when the integration limits are data points.

PCHIA — Piecewise Cubic Hermite Integrator, Arbitrary limits. Computes the definite integral of a piecewise cubic Hermite function over an arbitrary finite interval.

3. Check for monotonicity.

PCHMC — Piecewise Cubic Hermite Monotonicity Checker.

The calling sequences for these routines are described on the following pages, in the order listed above. (See the Appendix for internal routines .)

To facilitate two-dimensional applications, the representation of a PCH function throughout the package includes incfd, the increment between successive elements in the f- and d-arrays. For "normal" usage incfd=1, and f and d are one-dimensional arrays. one would call PCHxx (where "xx" is "IM", "IC", or "SP") with

```
n, x, f, d, 1  .
```

Suppose, however, that one has data on a rectangular mesh,

```
f2d(i,j) = value at (x(i), y(j)),   i=1(1)nx,
                                    j=1(1)ny.
```

Assume the following dimensions:

```
real  x(nxmax), y(nymax)
real  f2d(nxmax,nymax), fx(nxmax,nymax), fy(nxmax,nymax)
```

where 2.le.nx.le.nxmax and 2.le.ny.le.nymax . To interpolate in x along the line y = y(j), call PCHxx with

```
nx, x, f2d(1,j), fx(1,j), 1  .
```

To interpolate along the line x = x(i), call PCHxx with

```
ny, y, f2d(i,1), fy(i,1), nxmax  .
```

(This example assumes the usual columnwise storage of Fortran.)

subroutine PCHIM (n, x, f, d, incfd, ierr)

PCHIM: Piecewise Cubic Hermite Interpolation to Monotone data.

Sets derivatives needed to determine a monotone piecewise cubic Hermite interpolant to the data given in x and f.

Default boundary conditions are provided which are compatible with monotonicity. (See PCHIC if user control of boundary conditions is desired.)

If the data are only piecewise monotonic, the interpolant will have an extremum at each point where monotonicity switches direction. (See PCHIC if user control is desired in such cases.)

To facilitate two-dimensional applications, includes an increment between successive values of the f- and d-arrays.

The resulting piecewise cubic Hermite function may be evaluated by PCHFE or PCHFD.

---

calling sequence:

```
call PCHIM (n, x, f, d, incfd, ierr)

integer n, incfd, ierr
real x(n), f(incfd,n), d(incfd,n)
```

parameters:

n - (input) number of data points. (Error return if n.lt.2 .) If n=2, simply does linear interpolation.

x - (input) real array of independent variable values. The elements of x must be strictly increasing:
x(i-1) .lt. x(i), i = 2(1)n.
(Error return if not.)

f - (input) real array of dependent variable values to be interpolated. f(1+(i-1)*incfd) is value corresponding to x(i). PCHIM is designed for monotonic data, but it will work for any f-array. It will force extrema at points where monotonicity switches direction. If some other treatment of switch points is desired, PCHIC should be used instead.

d - (output) real array of derivative values at the data points. If the data are monotonic, these values will determine a a monotone cubic Hermite function.
The value corresponding to x(i) is stored in
d(1+(i-1)*incfd), i=1(1)n.
No other entries in d are changed.

incfd - (input) increment between successive values in f and d.
This argument is provided primarily for 2-D applications.
(Error return if incfd.lt.1 .)

ierr - (output) error flag.
normal return:
ierr = 0 (no errors).
warning error:
ierr.gt.0 means that ierr switches in the direction
of monotonicity were detected.
"recoverable" errors:
ierr = -1 if n.lt.2 .
ierr = -2 if incfd.lt.1 .
ierr = -3 if the x-array is not strictly increasing.
(The d-array has not been changed in any of these cases.)
NOTE: The above errors are checked in the order listed,
and following arguments have **not** been validated.

subroutine PCHIC (ic, vc, switch, n, x, f, d, incfd, wk, nwk, ierr)

PCHIC: Piecewise Cubic Hermite Interpolation Coefficients.

Sets derivatives needed to determine a piecewise monotone piecewise cubic interpolant to the data given in x and f satisfying the boundary conditions specified by ic and vc.

The treatment of points where monotonicity switches direction is controlled by argument switch.

To facilitate two-dimensional applications, includes an increment between successive values of the f- and d-arrays.

The resulting piecewise cubic Hermite function may be evaluated by PCHFE or PCHFD.

---

calling sequence:

```
call PCHIC (ic, vc, switch, n, x, f, d, incfd, wk, nwk, ierr)

integer  ic(2), n, incfd, nwk, ierr
real  vc(2), switch, x(n), f(incfd,n), d(incfd,n), wk(nwk)
```

parameters:

ic - (input) integer array of length 2 specifying desired boundary conditions:
ic(1) = ibeg, desired condition at beginning of data.
ic(2) = iend, desired condition at end of data.

ibeg = 0 for the default boundary condition (the same as used by PCHIM).
if ibeg.ne.0, then its sign indicates whether the boundary derivative is to be adjusted, if necessary, to be compatible with monotonicity:
ibeg.gt.0 if no adjustment is to be performed.
ibeg.lt.0 if the derivative is to be adjusted for monotonicity.

Allowable values for the magnitude of ibeg are:
ibeg = 1 if first derivative at x(1) is given in vc(1).
ibeg = 2 if second derivative at x(1) is given in vc(1).
ibeg = 3 to use the 3-point difference formula for d(1). (Reverts to the default b.c. if n.lt.3 .)
ibeg = 4 to use the 4-point difference formula for d(1). (Reverts to the default b.c. if n.lt.4 .)
ibeg = 5 to set d(1) so that the second derivative is continuous at x(2). (Reverts to the default b.c. if n.lt.4.) This option is somewhat analogous to the "not a knot" boundary condition provided by PCHSP.

NOTES (ibeg):
1. An error return is taken if abs(ibeg).gt.5 .
2. Only in case ibeg.le.0 is it guaranteed that the interpolant will be monotonic in the first interval. If the returned value of d(1) lies between zero and 3*slope(1), the interpolant will be monotonic. This is **not** checked if ibeg.gt.0 .
3. If ibeg.lt.0 and d(1) had to be changed to achieve monotonicity, a warning error is returned.

iend may take on the same values as ibeg, but applied to derivative at x(n). In case iend = 1 or 2, the value is given in vc(2).

NOTES (iend):
1. An error return is taken if abs(iend).gt.5 .
2. Only in case iend.le.0 is it guaranteed that the interpolant will be monotonic in the last interval. If the returned value of d(1+(n-1)*incfd) lies between zero and 3*slope(n-1), the interpolant will be monotonic. This is **not** checked if iend.gt.0 .
3. If iend.lt.0 and d(1+(n-1)*incfd) had to be changed to achieve monotonicity, a warning error is returned.

vc – (input) real array of length 2 specifying desired boundary values, as indicated above.
vc(1) need be set only if ic(1) = 1 or 2 .
vc(2) need be set only if ic(2) = 1 or 2 .

switch – (input) indicates desired treatment of points where direction of monotonicity switches:
Set switch to zero if interpolant is required to be monotonic in each interval, regardless of monotonicity of data.
NOTES:
1. This will cause d to be set to zero at all switch points, thus forcing extrema there.
2. The result of using this option with the default boundary conditions will be identical to using PCHIM, but will generally cost more compute time. This option is provided only to facilitate comparison of different switch and/or boundary conditions.

Set switch nonzero to use a formula based on the 3-point difference formula in the vicinity of switch points.
If switch is positive, the interpolant on each interval containing an extremum is controlled to not deviate from the data by more than switch*dfloc, where dfloc is the maximum of the change of f on this interval and its two immediate neighbors.
If switch is negative, no such control is to be imposed.

n – (input) number of data points. (Error return if n.lt.2 .)

x – (input) real array of independent variable values. The elements of x must be strictly increasing:
x(i-1) .lt. x(i), i = 2(1)n.
(Error return if not.)

f - (input) real array of dependent variable values to be interpolated. f(1+(i-1)*incfd) is value corresponding to x(i).

d - (output) real array of derivative values at the data points. These values will determine a monotone cubic Hermite function on each subinterval on which the data are monotonic, except possibly adjacent to switches in monotonicity. The value corresponding to x(i) is stored in
d(1+(i-1)*incfd), i=1(1)n.
No other entries in d are changed.

incfd - (input) increment between successive values in f and d. This argument is provided primarily for 2-D applications. (Error return if incfd.lt.1 .)

wk - (scratch) real array of working storage. The user may wish to know that the returned values are:
wk(i) = h(i) = x(i+1) - x(i) ;
wk(n-1+i) = slope(i) = (f(1,i+1) - f(1,i)) / h(i)
for i = 1(1)n-1.

nwk - (input) length of work array.
(Error return if nwk.lt.2*(n-1) .)

ierr - (output) error flag.
normal return:
ierr = 0 (no errors).
warning errors:
ierr = 1 if ibeg.lt.0 and d(1) had to be adjusted for monotonicity.
ierr = 2 if iend.lt.0 and d(1+(n-1)*incfd) had to be adjusted for monotonicity.
ierr = 3 if both of the above are true.
"recoverable" errors:
ierr = -1 if n.lt.2 .
ierr = -2 if incfd.lt.1 .
ierr = -3 if the x-array is not strictly increasing.
ierr = -4 if abs(ibeg).gt.5 .
ierr = -5 if abs(iend).gt.5 .
ierr = -6 if both of the above are true.
ierr = -7 if nwk.lt.2*(n-1) .
(The d-array has not been changed in any of these cases.)
NOTE: The above errors are checked in the order listed, and following arguments have **not** been validated.

subroutine PCHSP (ic, vc, n, x, f, d, incfd, wk, nwk, ierr)

PCHSP: Piecewise Cubic Hermite SPline

Computes the Hermite representation of the cubic spline interpolant to the data given in x and f satisfying the boundary conditions specified by ic and vc.

To facilitate two-dimensional applications, includes an increment between successive values of the f- and d-arrays.

The resulting piecewise cubic Hermite function may be evaluated by PCHFE or PCHFD.

NOTE: This is a modified version of C. de Boor's cubic spline routine CUBSPL.

---

calling sequence:

```
call PCHSP (ic, vc, n, x, f, d, incfd, wk, nwk, ierr)

integer ic(2), n, incfd, nwk, ierr
real vc(2), x(n), f(incfd,n), d(incfd,n), wk(nwk)
```

parameters:

ic - (input) integer array of length 2 specifying desired boundary conditions:
ic(1) = ibeg, desired condition at beginning of data.
ic(2) = iend, desired condition at end of data.

ibeg = 0 to set d(1) so that the third derivative is continuous at x(2). This is the "not a knot" condition provided by de Boor's cubic spline routine CUBSPL.
< This is the default boundary condition. >
ibeg = 1 if first derivative at x(1) is given in vc(1).
ibeg = 2 if second derivative at x(1) is given in vc(1).
ibeg = 3 to use the 3-point difference formula for d(1).
(Reverts to the default b.c. if n.lt.3 .)
ibeg = 4 to use the 4-point difference formula for d(1).
(Reverts to the default b.c. if n.lt.4 .)

NOTES:
1. An error return is taken if ibeg is out of range.
2. For the "natural" boundary condition, use ibeg=2 and vc(1)=0.

iend may take on the same values as ibeg, but applied to derivative at x(n). In case iend = 1 or 2, the value is given in vc(2).

NOTES:
1. An error return is taken if iend is out of range.

2. For the "natural" boundary condition, use iend=2 and vc(2)=0.

vc - (input) real array of length 2 specifying desired boundary values, as indicated above.
vc(1) need be set only if ic(1) = 1 or 2 .
vc(2) need be set only if ic(2) = 1 or 2 .

n - (input) number of data points. (Error return if n.lt.2 .)

x - (input) real array of independent variable values. The elements of x must be strictly increasing:
x(i-1) .lt. x(i), i = 2(1)n.
(Error return if not.)

f - (input) real array of dependent variable values to be interpolated. f(1+(i-1)*incfd) is value corresponding to x(i).

d - (output) real array of derivative values at the data points. These values will determine the cubic spline interpolant with the requested boundary conditions.
The value corresponding to x(i) is stored in
d(1+(i-1)*incfd), i=1(1)n.
No other entries in d are changed.

incfd - (input) increment between successive values in f and d.
This argument is provided primarily for 2-D applications.
(Error return if incfd.lt.1 .)

wk - (scratch) real array of working storage.

nwk - (input) length of work array.
(Error return if nwk.lt.2*n .)

ierr - (output) error flag.
normal return:
ierr = 0 (no errors).
"recoverable" errors:
ierr = -1 if n.lt.2 .
ierr = -2 if incfd.lt.1 .
ierr = -3 if the x-array is not strictly increasing.
ierr = -4 if ibeg.lt.0 or ibeg.gt.4 .
ierr = -5 if iend.lt.0 of iend.gt.4 .
ierr = -6 if both of the above are true.
ierr = -7 if nwk is too small.
NOTE: The above errors are checked in the order listed, and following arguments have **not** been validated.
(The d-array has not been changed in any of these cases.)
ierr = -8 in case of trouble solving the linear system for the interior derivative values.
(The d-array may have been changed in this case.)
(Do **not** use it!)

subroutine CHFEV (x1,x2, f1,f2, d1,d2, ne, xe, fe, next, ierr)

CHFEV: Cubic Hermite Function EValuator

Evaluates the cubic polynomial determined by function values f1,f2 and derivatives d1,d2 on interval (x1,x2) at the points xe(j), j=1(1)ne.

---

calling sequence:

```
call CHFEV (x1,x2, f1,f2, d1,d2, ne, xe, fe, next, ierr)

integer ne, next(2), ierr
real x1, x2, f1, f2, d1, d2, xe(ne), fe(ne)
```

parameters:

x1,x2 - (input) endpoints of interval of definition of cubic. (Error return if x1.eq.x2 .)

f1,f2 - (input) values of function at x1 and x2, respectively.

d1,d2 - (input) values of derivative at x1 and x2, respectively.

ne - (input) number of evaluation points. (Error return if ne.lt.1 .)

xe - (input) real array of points at which the function is to be evaluated. If any of the xe are outside the interval [x1,x2], a warning error is returned in next.

fe - (output) real array of values of the cubic function defined by x1,x2, f1,f2, d1,d2 at the points xe.

next - (output) integer array indicating number of extrapolation points:
next(1) = number of evaluation points to left of interval.
next(2) = number of evaluation points to right of interval.

ierr - (output) error flag.
normal return:
ierr = 0 (no errors).
"recoverable" errors:
ierr = -1 if ne.lt.1 .
ierr = -2 if x1.eq.x2 .
(The fe-array has not been changed in either case.)

subroutine PCHFE (n, x, f, d, incfd, skip, ne, xe, fe, ierr)

PCHFE: Piecewise Cubic Hermite Function eValuator

Evaluates the cubic Hermite function defined by n, x, f, d at the points xe(j), j=1(1)ne.

To provide compatibility with PCHIM and PCHIC, includes an increment between successive values of the f- and d-arrays.

---

calling sequence:

```
call PCHFE (n, x, f, d, incfd, skip, ne, xe, fe, ierr)

integer n, incfd, ne, ierr
real x(n), f(incfd,n), d(incfd,n), xe(ne), fe(ne)
logical skip
```

parameters:

n - (input) number of data points. (Error return if n.lt.2 .)

x - (input) real array of independent variable values. The elements of x must be strictly increasing:
x(i-1) .lt. x(i), i = 2(1)n.
(Error return if not.)

f - (input) real array of function values. f(1+(i-1)*incfd) is the value corresponding to x(i).

d - (input) real array of derivative values. d(1+(i-1)*incfd) is the value corresponding to x(i).

incfd - (input) increment between successive values in f and d.
(Error return if incfd.lt.1 .)

skip - (input/output) logical variable which should be set to .true. if the user wishes to skip checks for validity of preceding parameters, or to .false. otherwise.
This will save time in case these checks have already been performed (say, in PCHIM or PCHIC).
skip will be set to .true. on normal return.

ne - (input) number of evaluation points. (Error return if ne.lt.1 .)

xe - (input) real array of points at which the function is to be evaluated.

NOTES:
1. The evaluation will be most efficient if the elements of xe are increasing relative to x;

```
         that is,   xe(j) .ge. x(i)
         implies    xe(k) .ge. x(i),  all k.ge.j .
      2. If any of the xe are outside the interval [x(1),x(n)],
         values are extrapolated from the nearest extreme cubic,
         and a warning error is returned.

fe  -  (output) real array of values of the cubic Hermite function
       defined by  n, x, f, d  at the points  xe.

ierr -  (output) error flag.
       normal return:
          ierr = 0  (no errors).
       warning error:
          ierr.gt.0  means that extrapolation was performed at
             ierr points.
       "recoverable" errors:
          ierr = -1  if n.lt.2 .
          ierr = -2  if incfd.lt.1 .
          ierr = -3  if the x-array is not strictly increasing.
          ierr = -4  if ne.lt.1 .
         (The fe-array has not been changed in any of these cases.)
           NOTE:  The above errors are checked in the order listed,
               and following arguments have **not** been validated.
```

```
subroutine CHFDV (x1,x2, f1,f2, d1,d2, ne, xe, fe, de, next, ierr)
```

CHFDV: Cubic Hermite Function and Derivative eValuator

Evaluates the cubic polynomial determined by function values f1,f2 and derivatives d1,d2 on interval (x1,x2), together with its first derivative, at the points xe(j), j=1(1)ne.

If only function values are required, use CHFEV, instead.

---

calling sequence:

```
      call  CHFDV (x1,x2, f1,f2, d1,d2, ne, xe, fe, de, next, ierr)

   integer  ne, next(2), ierr
   real  x1, x2, f1, f2, d1, d2, xe(ne), fe(ne), de(ne)
```

parameters:

x1,x2 – (input) endpoints of interval of definition of cubic. (Error return if x1.eq.x2 .)

f1,f2 – (input) values of function at x1 and x2, respectively.

d1,d2 – (input) values of derivative at x1 and x2, respectively.

ne – (input) number of evaluation points. (Error return if ne.lt.1 .)

xe – (input) real array of points at which the functions are to be evaluated. If any of the xe are outside the interval [x1,x2], a warning error is returned in next.

fe – (output) real array of values of the cubic function defined by x1,x2, f1,f2, d1,d2 at the points xe.

de – (output) real array of values of the first derivative of the same function at the points xe.

next – (output) integer array indicating number of extrapolation points:
next(1) = number of evaluation points to left of interval.
next(2) = number of evaluation points to right of interval.

ierr – (output) error flag.
normal return:
ierr = 0 (no errors).
"recoverable" errors:
ierr = -1 if ne.lt.1 .
ierr = -2 if x1.eq.x2 .
(Output arrays have not been changed in either case.)

subroutine PCHFD (n, x, f, d, incfd, skip, ne, xe, fe, de, ierr)

PCHFD: Piecewise Cubic Hermite Function and Derivative evaluator

Evaluates the cubic Hermite function defined by n, x, f, d, together with its first derivative, at the points xe(j), j=1(1)ne.

If only function values are required, use PCHFE, instead.

To provide compatibility with PCHIM and PCHIC, includes an increment between successive values of the f- and d-arrays.

---

calling sequence:

```
call PCHFD (n, x, f, d, incfd, skip, ne, xe, fe, de, ierr)

integer n, incfd, ne, ierr
real x(n), f(incfd,n), d(incfd,n), xe(ne), fe(ne), de(ne)
logical skip
```

parameters:

n - (input) number of data points. (Error return if n.lt.2 .)

x - (input) real array of independent variable values. The elements of x must be strictly increasing:
x(i-1) .lt. x(i), i = 2(1)n.
(Error return if not.)

f - (input) real array of function values. f(1+(i-1)*incfd) is the value corresponding to x(i).

d - (input) real array of derivative values. d(1+(i-1)*incfd) is the value corresponding to x(i).

incfd - (input) increment between successive values in f and d. (Error return if incfd.lt.1 .)

skip - (input/output) logical variable which should be set to .true. if the user wishes to skip checks for validity of preceding parameters, or to .false. otherwise.
This will save time in case these checks have already been performed (say, in PCHIM or PCHIC).
skip will be set to .true. on normal return.

ne - (input) number of evaluation points. (Error return if ne.lt.1 .)

xe - (input) real array of points at which the functions are to be evaluated.

NOTES:
1. The evaluation will be most efficient if the elements of xe are increasing relative to x;
   that is, xe(j) .ge. x(i)
   implies xe(k) .ge. x(i), all k.ge.j .
2. If any of the xe are outside the interval [x(1),x(n)], values are extrapolated from the nearest extreme cubic, and a warning error is returned.

fe – (output) real array of values of the cubic Hermite function defined by n, x, f, d at the points xe.

de – (output) real array of values of the first derivative of the same function at the points xe.

ierr – (output) error flag.
normal return:
   ierr = 0 (no errors).
warning error:
   ierr.gt.0 means that extrapolation was performed at
      ierr points.
"recoverable" errors:
   ierr = -1 if n.lt.2 .
   ierr = -2 if incfd.lt.1 .
   ierr = -3 if the x-array is not strictly increasing.
   ierr = -4 if ne.lt.1 .
(Output arrays have not been changed in any of these cases.)
   NOTE: The above errors are checked in the order listed, and following arguments have **not** been validated.

real function PCHID (n, x, f, d, incfd, skip, ia, ib, ierr)

PCHID: Piecewise Cubic Hermite Integrator, Data limits

Evaluates the definite integral of the cubic Hermite function defined by n, x, f, d over the interval [x(ia), x(ib)].

To provide compatibility with PCHIM and PCHIC, includes an increment between successive values of the f- and d-arrays.

---

calling sequence:

```
value = PCHID (n, x, f, d, incfd, skip, ia, ib, ierr)

integer  n, incfd, ia, ib, ierr
real  x(n), f(incfd,n), d(incfd,n)
logical  skip
```

parameters:

value - (output) value of the requested integral.

n - (input) number of data points. (Error return if n.lt.2 .)

x - (input) real array of independent variable values. The elements of x must be strictly increasing:
x(i-1) .lt. x(i), i = 2(1)n.
(Error return if not.)

f - (input) real array of function values. f(1+(i-1)*incfd) is the value corresponding to x(i).

d - (input) real array of derivative values. d(1+(i-1)*incfd) is the value corresponding to x(i).

incfd - (input) increment between successive values in f and d.
(Error return if incfd.lt.1 .)

skip - (input/output) logical variable which should be set to .true. if the user wishes to skip checks for validity of preceding parameters, or to .false. otherwise.
This will save time in case these checks have already been performed (say, in PCHIM or PCHIC).
skip will be set to .true. on return with ierr = 0 or -4.

ia,ib - (input) indices in x-array for the limits of integration.
both must be in the range [1,n]. (Error return if not.)
No restrictions on their relative values.

ierr - (output) error flag.
normal return:
ierr = 0 (no errors).

"recoverable" errors:
   ierr = -1 if n.lt.2 .
   ierr = -2 if incfd.lt.1 .
   ierr = -3 if the x-array is not strictly increasing.
   ierr = -4 if ia or ib is out of range.
     (Value has not been computed in any of these cases.)
    NOTE: The above errors are checked in the order listed,
        and following arguments have **not** been validated.

real function PCHIA (n, x, f, d, incfd, skip, a, b, ierr)

PCHIA: Piecewise Cubic Hermite Integrator, Arbitrary limits

Evaluates the definite integral of the cubic Hermite function defined by n, x, f, d over the interval [a, b].

To provide compatibility with PCHIM and PCHIC, includes an increment between successive values of the f- and d-arrays.

---

calling sequence:

```
value = PCHIA (n, x, f, d, incfd, skip, a, b, ierr)

integer  n, incfd, ierr
real  x(n), f(incfd,n), d(incfd,n), a, b
logical  skip
```

parameters:

value - (output) value of the requested integral.

n - (input) number of data points. (Error return if n.lt.2 .)

x - (input) real array of independent variable values. The elements of x must be strictly increasing:
x(i-1) .lt. x(i), i = 2(1)n.
(Error return if not.)

f - (input) real array of function values. f(1+(i-1)*incfd) is the value corresponding to x(i).

d - (input) real array of derivative values. d(1+(i-1)*incfd) is the value corresponding to x(i).

incfd - (input) increment between successive values in f and d.
(Error return if incfd.lt.1 .)

skip - (input/output) logical variable which should be set to .true. if the user wishes to skip checks for validity of preceding parameters, or to .false. otherwise.
This will save time in case these checks have already been performed (say, in PCHIM or PCHIC).
skip will be set to .true. on return with ierr.ge.0 .

a,b - (input) the limits of integration.
NOTE: There is no requirement that [a,b] be contained in [x(1),x(n)]. However, the resulting integral value will be highly suspect, if not.

ierr - (output) error flag.
normal return:

```
      ierr = 0  (no errors).
   warning errors:
      ierr = 1  if  a  is outside the interval [x(1),x(n)].
      ierr = 2  if  b  is outside the interval [x(1),x(n)].
      ierr = 3  if both of the above are true.  (Note that this
                means that either [a,b] contains data interval
                or the intervals do not intersect at all.)
   "recoverable" errors:
      ierr = -1  if n.lt.2 .
      ierr = -2  if incfd.lt.1 .
      ierr = -3  if the x-array is not strictly increasing.
        (Value has not been computed in any of these cases.)
       NOTE:  The above errors are checked in the order listed,
           and following arguments have **not** been validated.
```

subroutine PCHMC (n, x, f, d, incfd, skip, ismon, ierr)

PCHMC: Piecewise Cubic Hermite Monotonicity Checker.

Checks the cubic Hermite function defined by n, x, f, d for monotonicity.

To provide compatibility with PCHIM and PCHIC, includes an increment between successive values of the f- and d-arrays.

---

calling sequence:

```
call PCHMC (n, x, f, d, incfd, skip, ismon, ierr)

integer  n, incfd, ismon(n), ierr
real  x(n), f(incfd,n), d(incfd,n)
logical  skip
```

parameters:

n - (input) number of data points. (Error return if n.lt.2 .)

x - (input) real array of independent variable values. The elements of x must be strictly increasing:
x(i-1) .lt. x(i), i = 2(1)n.
(Error return if not.)

f - (input) real array of function values. f(1+(i-1)*incfd) is the value corresponding to x(i).

d - (input) real array of derivative values. d(1+(i-1)*incfd) is the value corresponding to x(i).

incfd - (input) increment between successive values in f and d.
(Error return if incfd.lt.1 .)

skip - (input/output) logical variable which should be set to .true. if the user wishes to skip checks for validity of preceding parameters, or to .false. otherwise.
This will save time in case these checks have already been performed.
skip will be set to .true. on normal return.

ismon - (output) integer array indicating on which intervals the PCH function defined by n, x, f, d is monotonic.
For data interval [x(i),x(i+1)],
ismon(i) = -1 if function is strictly decreasing;
ismon(i) = 0 if function is constant;
ismon(i) = 1 if function is strictly increasing;
ismon(i) = 2 if function is non-monotonic;
ismon(i) = 3 if unable to determine. (This means that the d-values are near the boundary of the

monotonicity region. A small increase produces non-monotonicity; decrease, strict monotonicity.)

The above applies to i=1(1)n-1. ismon(n) indicates whether the entire function is monotonic on [x(1),x(n)].

ierr - (output) error flag.

normal return:

ierr = 0 (no errors).

"recoverable" errors:

ierr = -1 if n.lt.2 .

ierr = -2 if incfd.lt.1 .

ierr = -3 if the x-array is not strictly increasing.

(The ismon-array has not been changed in any of these cases.)

NOTE: The above errors are checked in the order listed, and following arguments have **not** been validated.

REFERENCES

[1] F.N.Fritsch and R.E.Carlson, "Monotone piecewise cubic interpolation," *SIAM J. Numer. Anal.* 17, 2 (April 1980), 238-246.

[2] F.N.Fritsch and J.Butland, "A method for constructing local monotone piecewise cubic interpolants," LLNL report UCRL-87559 (April 1982). [Submitted to *Mathematics of Computation.*]

[3] F.N.Fritsch, "Piecewise Cubic Hermite Interpolation Package," LLNL report UCRL-87285 (July 1982). [Poster presented at the SIAM 30th Anniversary Meeting, 19-23 July 1982.]

[4] Carl de Boor, *A Practical Guide to Splines*, Springer-Verlag (New York, 1978). [esp. Chapter iv, pp.49-62.]

APPENDIX - Complete Structure of PCHIP.

The following lower-level routines are also included in PCHIP. They are listed here in alphabetical order.

CHFIV - Cubic Hermite Function Integral eValuator.
(Real function called by PCHIA.)

CHFMC - Cubic Hermite Function Monotonicity Checker.
(Integer function called by PCHMC.)

PCHCE - PCHIC End derivative setter.
(Called by PCHIC.)

PCHCI - PCHIC Initial derivative setter.
(Called by PCHIC.)

PCHCS - PCHIC monotonicity Switch derivative setter.
(Called by PCHIC.)

PCHDF - PCHIP finite Difference Formula.
(Real function called by PCHCE and PCHSP.)

PCHST - PCHIP Sign Testing routine.
(Real function called by various PCHIP routines.)

PCHSW - PCHCS SWitch excursion adjuster.
(Called by PCHCS.)

The logical structure of PCHIP is indicated in the following figures.

Figure 1.
PCHIP Structure -- Interpolation Routines

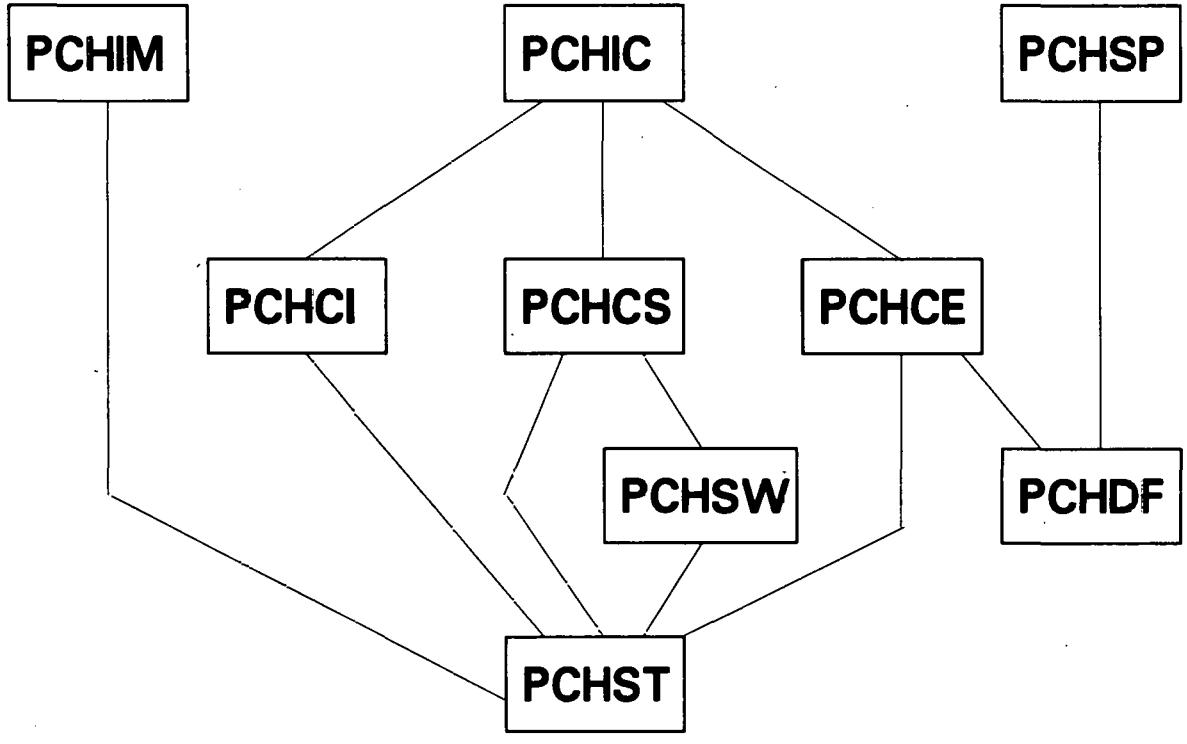
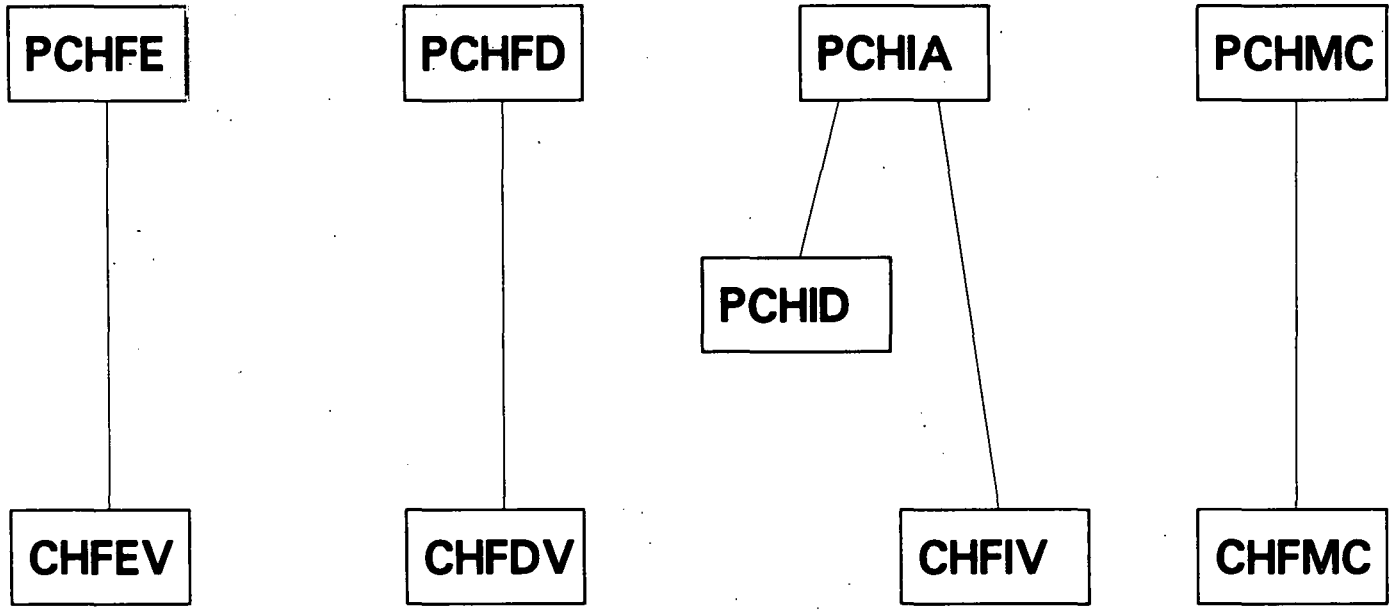


Figure 2.
PCHIP Structure -- Evaluators & Auxiliary Routines



DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government thereof, and shall not be used for advertising or product endorsement purposes.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $3.50

Page Range	Domestic Price	Page Range	Domestic Price
001-025	$ 5.00	326-350	$ 18.00
026-050	6.00	351-375	19.00
051-075	7.00	376-400	20.00
076-100	8.00	401-425	21.00
101-125	9.00	426-450	22.00
126-150	10.00	451-475	23.00
151-175	11.00	476-500	24.00
176-200	12.00	501-525	25.00
201-225	13.00	526-550	26.00
226-250	14.00	551-525	27.00
251-275	15.00	526-550	28.00
276-300	16.00	601-up [1]	
301-325	17.00		

[1] Add 2.00 for each additional 25 page increment from 601 pages up.

Technical Information Department · Lawrence Livermore Laboratory
University of California · Livermore, California 94550