Module 8.1: nag_pch_interp
Piecewise Cubic Hermite Interpolation

nag_pch_interp provides procedures for computing and evaluating piecewise cubic Hermite interpolants to arbitrary data sets in one dimension. In particular, the module contains a procedure for generating interpolating functions which preserve monotonicity in the data set.

Contents

Introduction

This module is concerned with interpolating a set of data using an interpolating function known as a *piecewise cubic Hermite polynomial*. Such an interpolant, denoted $h(x)$, is fully defined by its values, and the values of its first derivative $h'(x)$, at a given set of points. More precisely, let $x_1 < x_2 < \cdots < x_m$, then $h(x)$ is defined by the relations $h(x_i) = f_i$, $h'(x_i) = d_i$, for $i = 1, 2, \ldots, m$. It is a C^1 interpolant; that is, it has a continuous first derivative.

If derivative values d_i are not given, then approximations to their values can be *constructed*; a particularly useful property is that in the case of monotonic or piecewise monotonic data this may be done in such a way that the resulting interpolant *preserves monotonicity* in the data. As a result, the interpolant does not exhibit unwanted oscillations, and is often considered more 'visually pleasing' than other types of interpolant such as cubic splines.

Constructing a monotonicity-preserving interpolant from a given set of data values is the principal function of the procedure `nag_pch_monot_interp`. This procedure stores the values x_i, f_i and d_i in a structure of the derived type `nag_pch_comm_wp`, which can then be passed to `nag_pch_eval` (to evaluate the interpolant or its derivative) or to `nag_pch_intg` (to compute its definite integral). `nag_pch_monot_interp` can also create a structure of type `nag_pch_comm_wp` from given derivative values d_i, although the interpolant will then not necessarily preserve monotonicity.

Procedure: nag_pch_monot_interp

1 Description

nag_pch_monot_interp determines a piecewise cubic Hermite function $h(x)$, defined in the range $[a, b] = [x_1, x_m]$, which interpolates the set of data points (x_i, f_i) for $i = 1, 2, \ldots, m$, where $x_1 < x_2 < \cdots < x_m$ and $m \geq 2$.

The principal function of the procedure is to produce an interpolant $h(x)$ which preserves monotonicity over ranges where the data points are monotonic. This means that if $f_i < f_{i+1}$, then $h(x)$ is strictly increasing over the interval $[x_i, x_{i+1}]$; if $f_i > f_{i+1}$, it is strictly decreasing, and if $f_i = f_{i+1}$, it is constant. The monotonicity-preserving property is achieved by approximating values d_i of the derivative $h'(x_i)$, for $i = 1, 2, \ldots, m$.

Alternatively, by supplying the optional input argument d you can use this procedure to initialize the structure interp with chosen derivative values d_i. The interpolant will then satisfy $h'(x_i) = d_i$ in addition to $h(x_i) = f_i$, but will not generally preserve monotonicity. This facility can be useful for initializing a structure with data read from file.

2 Usage

```
USE nag_pch_interp
CALL nag_pch_monot_interp(x, f, interp  [, optional arguments])
```

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the value of the following problem parameter from the shape of the supplied arrays.

$m \geq 2$ — the number of data points

3.1 Mandatory Arguments

x(m) — real(kind=*wp*), intent(in)

Input: the data points x_i for $i = 1, 2, \ldots, m$.

Constraints: x(1) $<$ x(2) $< \cdots <$ x(m).

f(m) — real(kind=*wp*), intent(in)

Input: the values f_i for $i = 1, 2, \ldots, m$.

interp — type(nag_pch_comm_*wp*), intent(out)

Output: a structure containing details of the interpolant $h(x)$ generated. This structure may be passed to nag_pch_eval to evaluate $h(x)$ at specified points, or to nag_pch_intg to compute its definite integral.

Note: to reduce the risk of corrupting the data accidentally, the components of this structure are private; details of the interpolant may be extracted by calling nag_pch_extract.

The procedure allocates $3m$ real(kind=*wp*) elements of storage to the structure. If you wish to deallocate this storage when the structure is no longer required, you must call the procedure nag_deallocate, as illustrated in Example 1 of this module document.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

d(m) — real(kind=*wp*), intent(in), optional

Input: the values d_i of $h'(x_i)$ for $i = 1, 2, \ldots, m$.

Note: if this argument is present the monotonicity-preserving property of $h(x)$ will not hold in general.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
302	An array argument has an invalid shape.
303	Array arguments have inconsistent shapes.
320	The procedure was unable to allocate enough memory.

5 Examples of Usage

Complete examples of the use of this procedure appear in Examples 1 and 2 of this module document.

6 Further Comments

6.1 Algorithmic Detail

If the optional argument `d` is not present the procedure initializes the structure `interp` with the given values of `x`, `f`, and estimated first derivatives at the points x_i. These are approximated using a formula due to Brodlie, which is described in Fritsch and Butland [2], with suitable changes at the boundary points. If `d` is present this procedure simply initializes the structure `interp` with the given values of `x`, `f` and `d`.

This procedure is derived from PCHIM in Fritsch [1].

6.2 Accuracy

If $f \in C^4[a, b]$ is a function such that $f(x_i) = f_i$, $f'(x_i) = d_i$, for $i = 1, 2, \ldots, m$, then the piecewise cubic Hermite interpolant $h(x)$ satisfies the error bound

$$| f(x) - h(x) | \leq \frac{H^4}{384} \sup_{\eta \in (a,b)} | f^{(4)}(\eta) |,$$

where $H = \max_i (x_{i+1} - x_i)$, for $i = 1, 2, \ldots, m - 1$.

6.3 Timing

The time taken by the procedure is approximately proportional to m.

Procedure: nag_pch_eval

1 Description

nag_pch_eval computes values of a piecewise cubic Hermite interpolant $h(x)$, as determined by nag_pch_monot_interp, at a set of points u_i, for $i = 1, 2, \ldots, n$. If the optional argument hd is supplied the procedure also computes values of the first derivative of $h(x)$ at the same points.

If an evaluation point lies outside the region of definition $[a, b]$ of the interpolant, as described in Section 1 of the procedure document for nag_pch_monot_interp, the value is extrapolated and a warning issued.

2 Usage

```
USE nag_pch_interp
CALL nag_pch_eval(interp, u, h  [, optional arguments])
```

2.1 Interfaces

Distinct interfaces are provided for the following cases.

Evaluation at an array of points / at a single point

Array of points: u, h and the optional argument hd are all rank-1 arrays.

Single point: u, h and the optional argument hd are all scalar.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the value of the following problem parameter from the shape of the supplied arrays.

$n \geq 1$ — the number of evaluation points

3.1 Mandatory Arguments

interp — type(nag_pch_comm_*wp*), intent(in)

Input: a structure containing details of the interpolant $h(x)$ to be evaluated.

Constraints: interp must be as output from a previous call to nag_pch_monot_interp.

u(n) / u — real(kind=*wp*), intent(in)

Input: the point(s) u_i, for $i = 1, 2, \ldots, n$, at which $h(x)$ is to be evaluated.

Note: if $n = 1$, u may be declared as a scalar.

h(n) / h — real(kind=*wp*), intent(out)

Output: the value(s) of the interpolant $h(u_i)$, for $i = 1, 2, \ldots, n$.

Note: h must have the same rank as u.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

hd(n) **/ hd** — real(kind=*wp*), intent(out), optional

Output: the value(s) of the first derivative of the interpolant $h'(u_i)$, for $i = 1, 2, \ldots, n$.

Note: `hd` must have the same rank as `u`.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
302	An array argument has an invalid shape.
303	Array arguments have inconsistent shapes.

Warnings (error%level = 1):

error%code	Description
101	The computation involved extrapolation. Results may be unreliable.
	Not all evaluation points lie within the interval of definition of $h(x)$.

5 Examples of Usage

A complete example of the use of this procedure appears in Example 2 of this module document.

6 Further Comments

6.1 Algorithmic Detail

This procedure is derived from PCHFD in Fritsch [1].

6.2 Timing

The time taken by the procedure is approximately proportional to n, although a single call to the procedure with $n > 1$ is more efficient than several calls with $n = 1$. The evaluation will be more efficient if the elements of `u` are in non-decreasing order.

Procedure: nag_pch_intg

1 Description

The function nag_pch_intg evaluates the definite integral

$$I = \int_{\alpha}^{\beta} h(x)\, dx$$

of a piecewise cubic Hermite interpolant $h(x)$.

By default the region of integration $[\alpha,\beta]$ is taken as the region of definition $[a,b]$ of $h(x)$, as described in Section 1 of the procedure document for nag_pch_monot_interp.

Integration may be performed over a non-default region by supplying one or both of the optional arguments alpha and beta. If either α or β lies outside $[a,b]$, computation of the integral involves extrapolation and a warning is issued.

2 Usage

```
USE nag_pch_interp
[value =] nag_pch_intg(interp  [, optional arguments])
```

The function result value is a scalar of type real(kind=*wp*).

3 Arguments

3.1 Mandatory Argument

interp — type(nag_pch_comm_*wp*), intent(in)

Input: a structure containing details of the interpolant $h(x)$ to be integrated.

Constraints: interp must be output from a previous call to nag_pch_monot_interp.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

alpha — real(kind=*wp*), intent(in), optional

beta — real(kind=*wp*), intent(in), optional

Input: the lower and upper limits α and β of the integral.

Note: it is *not* required that alpha < beta.

Default: alpha=a, beta=b, where $[a,b]$ is the region of definition of $h(x)$, see Section 1.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document nag_error_handling (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to nag_set_error before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.

Warnings (error%level = 1):

error%code	Description
101	The computation involved extrapolation. Results may be unreliable.
	The integration limits do not lie within the interval of definition of $h(x)$.

5 Examples of Usage

A complete example of the use of this procedure appears in Example 1 of this module document.

6 Further Comments

6.1 Algorithmic Detail

This procedure is derived from PCHIA in Fritsch [1].

6.2 Timing

The time taken by this function is approximately proportional to the number of data points included within the interval $[\alpha, \beta]$.

Procedure: nag_pch_extract

1 Description

Given a structure of type nag_pch_comm_*wp* representing a piecewise cubic Hermite interpolant $h(x)$, defined for $x \in [a, b]$, this procedure optionally returns the region of definition, the data points x_i, the function values $f_i = h(x_i)$, and derivative values $d_i = h'(x_i)$, for $i = 1, 2, \ldots, m$.

The arguments x, f and d, which return the data points, function values and derivative values, are pointers which are allocated internally. It is your responsibility to deallocate this storage when the results are no longer required.

2 Usage

```
USE nag_pch_interp
CALL nag_pch_extract(interp  [, optional arguments])
```

3 Arguments

3.1 Mandatory Argument

interp — type(nag_pch_comm_*wp*), intent(in)

Input: a structure containing details of the interpolant $h(x)$.

Constraints: interp must be output from a previous call to nag_pch_monot_interp.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

a — real(kind=*wp*), intent(out), optional

b — real(kind=*wp*), intent(out), optional

Output: the lower and upper limits a and b of the region of definition of $h(x)$ respectively.

x(:) — real(kind=*wp*), pointer, optional

Output: x(i) holds the data point x_i, for $i = 1, 2, \ldots, m$.

Note: this array is allocated by this procedure. It should be deallocated when no longer required.

f(:) — real(kind=*wp*), pointer, optional

Output: f(i) holds the function value f_i, for $i = 1, 2, \ldots, m$.

Note: this array is allocated by this procedure. It should be deallocated when no longer required.

d(:) — real(kind=*wp*), pointer, optional

Output: d(i) holds the derivative value d_i, for $i = 1, 2, \ldots, m$.

Note: this array is allocated by this procedure. It should be deallocated when no longer required.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document nag_error_handling (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to nag_set_error before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.

5 Examples of Usage

A complete example of the use of this procedure appears in Example 1 of this module document.

6 Further Comments

6.1 Algorithmic Detail

This procedure may be used for example to extract data from the structure `interp` in order to write to file. The procedure `nag_pch_monot_interp` may then be used to initialize a new structure with this data.

If called with no optional arguments this procedure merely checks that the structure `interp` has been created by the generation procedure `nag_pch_monot_interp`.

Derived Type: nag_pch_comm_*wp*

Note. The names of derived types containing real/complex components are precision dependent. For double precision the name of this type is `nag_pch_comm_dp`. For single precision the name is `nag_pch_comm_sp`. Please read the Users' Note for your implementation to check which precisions are available.

1 Description

The derived type `nag_pch_comm_wp` is used to represent the piecewise cubic Hermite interpolant $h(x)$ as described in the Module Introduction. The procedure `nag_pch_monot_interp`, returns a structure of this type suitable for passing to the evaluation procedures `nag_pch_eval`, `nag_pch_intg` or `nag_pch_extract`.

The generation procedure `nag_pch_monot_interp` allocates storage to the pointer components of the structure. For details of the amount of storage allocated see the description of the argument `interp` in the procedure document for `nag_pch_monot_interp`.

If you wish to deallocate the storage when the structure is no longer required, you must call the generic deallocation procedure `nag_deallocate`, which is described in the module document `nag_lib_support` (1.1).

`nag_pch_monot_interp` checks whether the structure has already had storage allocated to it in a previous call; if it has, that storage is deallocated before allocating the storage required for the new call.

The components of this type are private.

2 Type Definition

```
type nag_pch_comm_wp
  private
  .
  .
  .
end type nag_pch_comm_wp
```

3 Components

In order to reduce the risk of accidental data corruption the components of this type are private and may not be accessed directly.

The procedures `nag_pch_monot_interp` and `nag_pch_extract` may be used to initialize and extract data from structures of the type.

Example 1: Monotonicity-preserving interpolation

Generate a piecewise cubic monotonicity-preserving Hermite interpolant $h(x)$ to a set of data points (x_i, f_i), for $i = 1, 2, \ldots, m$. Output full details of the interpolant. Calculate the definite integral over the interval (x_1, x_m) on which $h(x)$ is defined.

1 Program Text

Note. The listing of the example program presented below is double precision. Single precision users are referred to Section 5.2 of the Essential Introduction for further information.

```
PROGRAM nag_pch_interp_ex01

  ! Example Program Text for nag_pch_interp
  ! NAG fl90, Release 3. NAG Copyright 1997.

  ! .. Use Statements ..
  USE nag_examples_io, ONLY : nag_std_in, nag_std_out
  USE nag_pch_interp, ONLY : nag_pch_comm_wp => nag_pch_comm_dp, &
   nag_pch_monot_interp, nag_pch_extract, nag_pch_intg, nag_deallocate
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND
  ! .. Parameters ..
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: i, m
  REAL (wp) :: integral
  TYPE (nag_pch_comm_wp) :: interp
  ! .. Local Arrays ..
  REAL (wp), POINTER :: comp_d(:), comp_f(:), comp_x(:)
  REAL (wp), ALLOCATABLE :: f(:), x(:)
  ! .. Executable Statements ..

  WRITE (nag_std_out,*) 'Example Program Results for nag_pch_interp_ex01'

  READ (nag_std_in,*)          ! Skip heading in data file
  READ (nag_std_in,*) m

  ALLOCATE (x(m),f(m))         ! Allocate storage

  READ (nag_std_in,*) x
  READ (nag_std_in,*) f

  ! Generate monotonicity preserving interpolant.

  CALL nag_pch_monot_interp(x,f,interp)

  ! Extract and output data points, data values and computed
  ! derivatives.

  CALL nag_pch_extract(interp,x=comp_x,f=comp_f,d=comp_d)

  WRITE (nag_std_out,'(/,''Details of interpolant'',//, 7X,''i'',8X, &
   & ''x(i)'',11X,''f(i)'',11X,''d(i)'')')
  WRITE (nag_std_out,'(I8,3ES15.4)') (i,comp_x(i),comp_f(i),comp_d(i),i=1, &
   m)

  ! Compute integral over interval of definition.

  integral = nag_pch_intg(interp)
```

```
      WRITE (nag_std_out,'(/,''Integral ='',ES15.4)') integral

      DEALLOCATE (x,f,comp_x,comp_f,comp_d) ! Deallocate storage

      NULLIFY (comp_x,comp_f,comp_d)

      CALL nag_deallocate(interp)  ! Free structure allocated by NAG fl90

    END PROGRAM nag_pch_interp_ex01
```

2 Program Data

```
Example Program Data for nag_pch_interp_ex01
 9                                                          : m
 7.99        8.09        8.19        8.70        9.20
 10.00       12.00       15.00       20.00                  : x(1) ... x(m)
 0.00000E+0  0.27643E-4  0.43750E-1  0.16918E+0  0.46943E+0
 0.94374E+0  0.99864E+0  0.99992E+0  0.99999E+0             : f(1) ... f(m)
```

3 Program Results

```
 Example Program Results for nag_pch_interp_ex01

Details of interpolant

       i        x(i)           f(i)           d(i)
       1     7.9900E+00     0.0000E+00     0.0000E+00
       2     8.0900E+00     2.7643E-05     5.5251E-04
       3     8.1900E+00     4.3750E-02     3.3587E-01
       4     8.7000E+00     1.6918E-01     3.4944E-01
       5     9.2000E+00     4.6943E-01     5.9696E-01
       6     1.0000E+01     9.4374E-01     6.0326E-02
       7     1.2000E+01     9.9864E-01     8.9833E-04
       8     1.5000E+01     9.9992E-01     2.9405E-05
       9     2.0000E+01     9.9999E-01     0.0000E+00

Integral =     1.0765E+01
```

Example 2: Evaluation of an interpolant read from file

Read details of the interpolant $h(x)$ generated in Example 1 from file. Evaluate $h(x)$ at equally spaced points within its region of definition.

1 Program Text

Note. The listing of the example program presented below is double precision. Single precision users are referred to Section 5.2 of the Essential Introduction for further information.

```
PROGRAM nag_pch_interp_ex02

  ! Example Program Text for nag_pch_interp
  ! NAG fl90, Release 3. NAG Copyright 1997.

  ! .. Use Statements ..
  USE nag_examples_io, ONLY : nag_std_in, nag_std_out
  USE nag_pch_interp, ONLY : nag_pch_comm_wp => nag_pch_comm_dp, &
   nag_pch_monot_interp, nag_pch_eval, nag_deallocate
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND, REAL
  ! .. Parameters ..
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: i, m, n
  REAL (wp) :: dx
  TYPE (nag_pch_comm_wp) :: interp
  ! .. Local Arrays ..
  REAL (wp), ALLOCATABLE :: d(:), f(:), h(:), hd(:), u(:), x(:)
  ! .. Executable Statements ..

  WRITE (nag_std_out,*) 'Example Program Results for nag_pch_interp_ex02'

  READ (nag_std_in,*)          ! Skip heading in data file
  READ (nag_std_in,*) m, n

  ALLOCATE (x(m),f(m),d(m),u(n),h(n),hd(n)) ! Allocate storage

  READ (nag_std_in,*) x
  READ (nag_std_in,*) f
  READ (nag_std_in,*) d

  ! Initialize structure interp.

  CALL nag_pch_monot_interp(x,f,interp,d=d)

  ! Evaluate interp at n equally spaced points in [x(1), x(m)].
  IF (n<=1) THEN
    WRITE (nag_std_out,*) 'n <= 1.'
  ELSE
    dx = (x(m)-x(1))/REAL(n-1,kind=wp)
    u = (/ (x(1)+(i-1)*dx,i=1,n-1), x(m) /)

    CALL nag_pch_eval(interp,u,h,hd=hd)

    WRITE (nag_std_out,'(/,9X,''u'',11X,''h'',11X,''hd'')')
    WRITE (nag_std_out,'(1X,2F12.4,ES15.4)') (u(i),h(i),hd(i),i=1,n)
  END IF

  DEALLOCATE (x,f,d,u,h,hd)    ! Deallocate storage
```

```
      CALL nag_deallocate(interp)  ! Free structure allocated by NAG fl90

    END PROGRAM nag_pch_interp_ex02
```

2 Program Data

```
Example Program Data for nag_pch_interp_ex02
 9 11                                                        : m, n
 7.9900E+00  8.0900E+00  8.1900E+00  8.7000E+00  9.2000E+00
 1.0000E+01  1.2000E+01  1.5000E+01  2.0000E+01              : x(1) ... x(m)
 0.0000E+00  2.7643E-05  4.3750E-02  1.6918E-01  4.6943E-01
 9.4374E-01  9.9864E-01  9.9992E-01  9.9999E-01              : f(1) ... f(m)
 0.0000E+00  5.5251E-04  3.3587E-01  3.4944E-01  5.9696E-01
 6.0326E-02  8.9833E-04  2.9405E-05  0.0000E+00              : d(1) ... d(m)
```

3 Program Results

```
 Example Program Results for nag_pch_interp_ex02

        u           h           hd
      7.9900      0.0000     0.0000E+00
      9.1910      0.4640     6.0601E-01
     10.3920      0.9645     4.5688E-02
     11.5930      0.9965     9.9166E-03
     12.7940      0.9992     6.2491E-04
     13.9950      0.9998     2.7077E-04
     15.1960      0.9999     2.8094E-05
     16.3970      1.0000     2.0341E-05
     17.5980      1.0000     1.3074E-05
     18.7990      1.0000     6.2938E-06
     20.0000      1.0000     0.0000E+00
```

References

[1] Fritsch F N (1982) PCHIP final specifications *Report UCID-30194* Lawrence Livermore National Laboratory

[2] Fritsch F N and Butland J (1984) A Method for constructing local monotone piecewise cubic interpolants *SIAM J. Sci. Statist. Comput.* **5** 300–304