1 Purpose

nag_pde_interp_1d_coll (d03pyc) may be used in conjunction with either nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc). It computes the solution and its first derivative at user-specified points in the spatial co-ordinate.

2 Specification

void nag_pde_interp_1d_coll (Integer npde, const double u[], Integer nbkpts, const double xbkpts[], Integer npoly, Integer npts, const double xp[], Integer intpts, Integer itype, double up[], double rsave[], Integer lrsave, NagError *fail)

3 Description

nag_pde_interp_1d_coll (d03pyc) is an interpolation function for evaluating the solution of a system of partial differential equations (PDEs), or the PDE components of a system of PDEs with coupled ordinary differential equations (ODEs), at a set of user-specified points. The solution of a system of equations can be computed using nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc) on a set of mesh points; nag_pde_interp_1d_coll (d03pyc) can then be employed to compute the solution at a set of points other than those originally used in nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc). It can also evaluate the first derivative of the solution. Polynomial interpolation is used between each of the break-points $x_{bkpts}[i-1]$, for $i = 1, 2, \ldots, nbkpts$. When the derivative is needed ($itype = 2$), the array $xp[intpts - 1]$ must not contain any of the break-points, as the method, and consequently the interpolation scheme, assumes that only the solution is continuous at these points.

4 References

None.

5 Parameters

Note: the parameters $u$, $npts$, $npde$, $xbkpts$, $nbkpts$, $rsave$ and $lrsave$ must be supplied unchanged from either nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc).

1: npde – Integer

   On entry: the number of PDEs.

   Constraint: $npde \geq 1$.

2: $u[npde \times npts]$ – const double

   Note: where $U(i, j)$ appears in this document it refers to the array element $u[npde \times (j - 1) + i - 1]$. We recommend using a #define to make the same definition in your calling program.

   On entry: the PDE part of the original solution returned in the parameter $u$ by the function nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc).

3: nbkpts – Integer

   On entry: the number of break-points.

   Constraint: $nbkpts \geq 2$. 
4:  xbkpts[nbkpts] – const double  
   *Input*
   
   *On entry:* xbkpts[i − 1], for i = 1, 2, . . . , nbkpts, must contain the break-points as used by
   nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc).
   
   *Constraint:* xbkpts[0] < xbkpts[1] < . . . < xbkpts[nbkpts − 1].

5:  npoly – Integer  
   *Input*
   
   *On entry:* the degree of the Chebyshev polynomial used for approximation as used by
   nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc).
   
   *Constraint:* 1 ≤ npoly ≤ 49.

6:  npts – Integer  
   *Input*
   
   *On entry:* the number of mesh points as used by nag_pde_parab_1d_coll (d03pdc) or
   nag_pde_parab_1d_coll_ode (d03pjc).
   
   *Constraint:* npts = (nbkpts − 1) × npoly + 1.

7:  xp[intpts] – const double  
   *Input*
   
   *On entry:* xp[i − 1], for i = 1, 2, . . . , intpts, must contain the spatial interpolation points.
   
   
   When itype = 2, xp[i − 1] ≠ xbkpts[j − 1], for i = 1, 2, . . . , intpts; j = 2, 3, . . . , nbkpts − 1.

8:  intpts – Integer  
   *Input*
   
   *On entry:* the number of interpolation points.
   
   *Constraint:* intpts ≥ 1.

9:  itype – Integer  
   *Input*
   
   *On entry:* specifies the interpolation to be performed.
   
   If itype = 1, the solution at the interpolation points are computed. If itype = 2, both the solution
   and the first derivative at the interpolation points are computed.
   
   *Constraint:* itype = 1 or 2.

10:  up[npde × intpts × itype] – double  
    *Output*
    
    *Note:* where UP(i, j, k) appears in this document it refers to the array element
    up[npde × (intpts × (k − 1) + j − 1) + i − 1]. We recommend using a #define to make the same
    definition in your calling program.
    
    *On exit:* if itype = 1, UP(i, j, 1), contains the value of the solution U_j(x_j,t_out), at the interpolation
    points x_j = xp[j − 1], for j = 1, 2, . . . , intpts; i = 1, 2, . . . , npde.
    
    If itype = 2, UP(i, j, 1) contains U_j(x_j,t_out) and UP(i, j, 2) contains \( \frac{\partial U_j}{\partial x} \) at these points.

11:  rsave[lrsave] – double  
    *Input/Output*
    
    *On entry:* the array rsave as returned by nag_pde_parab_1d_coll (d03pdc) or
    nag_pde_parab_1d_coll_ode (d03pjc). The contents of rsave must not be changed from the call
    to nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc).

12:  lrsave – Integer  
    *Input*
    
    *On entry:* the size of the workspace rsave, as in nag_pde_parab_1d_coll (d03pdc) or
    nag_pde_parab_1d_coll_ode (d03pjc).
6 Error Indicators and Warnings

**NE_INT**

On entry, \( \text{itype} \) is not equal to 1 or 2, \( \text{itype} = \langle \text{value} \rangle. \)

On entry, \( \text{intpts} \leq 0: \text{intpts} = \langle \text{value} \rangle. \)

On entry, \( \text{npoly} = \langle \text{value} \rangle. \)
Constraint: \( \text{npoly} > 0. \)

On entry, \( \text{nbkpts} = \langle \text{value} \rangle. \)
Constraint: \( \text{nbkpts} > 2. \)

On entry, \( \text{npde} = \langle \text{value} \rangle. \)
Constraint: \( \text{npde} > 0. \)

**NE_INT_3**

On entry, \( \text{npts} \) is not equal to \( (\text{nbkpts} - 1) \times \text{npoly} + 1: \text{npts} = \langle \text{value} \rangle, \text{nbkpts} = \langle \text{value} \rangle, \text{npoly} = \langle \text{value} \rangle. \)

**NE_EXTRAPOLATION**

Extrapolation is not allowed.

**NE_INCOMPAT_PARAM**

On entry, \( \text{itype} = 2 \) and at least one interpolation point coincides with a break-point i.e., interpolation point no \( \langle \text{value} \rangle \) with value \( \langle \text{value} \rangle \) is close to break-point \( \langle \text{value} \rangle \) with value \( \langle \text{value} \rangle \).

**NE_NOT STRICTLY INCREASING**

On entry, interpolation points \( \text{xp} \) badly ordered: \( i = \langle \text{value} \rangle, \text{xp}[i - 1] = \langle \text{value} \rangle \), \( j = \langle \text{value} \rangle, \text{xp}[j - 1] = \langle \text{value} \rangle. \)

On entry, break points \( \text{xbkpts} \) badly ordered: \( i = \langle \text{value} \rangle, \text{xbkpts}[i - 1] = \langle \text{value} \rangle \), \( j = \langle \text{value} \rangle, \text{xbkpts}[j - 1] = \langle \text{value} \rangle. \)

**NE_BAD_PARAM**

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.

**NE_INTERNAL_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

7 Accuracy

See the documents for nag_pde_parab_1d_coll (d03pdc) or nag_pde_parab_1d_coll_ode (d03pjc).

8 Further Comments

None.

9 Example

See Section 9 of the document for nag_pde_parab_1d_coll (d03pdc).