NAG Library Function Document

nag_pde_interp_1d_fd (d03pzc)

1 Purpose

nag_pde_interp_1d_fd (d03pzc) interpolates in the spatial coordinate the solution and derivative of a system of partial differential equations (PDEs). The solution must first be computed using one of the finite difference schemes nag_pde_parab_1d_fd (d03pcc), nag_pde_parab_1d_fd_ode (d03phc) or nag_pde_parab_1d_fd_ode_remesh (d03ppc), or one of the Keller box schemes nag_pde_parab_1d_keller (d03pec), nag_pde_parab_1d_keller_ode (d03pkc) or nag_pde_parab_1d_keller_ode_remesh (d03prc).

2 Specification

```
#include <nag.h>
#include <nagd03.h>
```

```
void nag_pde_interp_1d_fd (Integer npde, Integer m, const double u[],
     Integer npts, const double x[], const double xp[], Integer intpts,
     Integer itype, double up[], NagError *fail)
```

3 Description

nag_pde_interp_1d_fd (d03pzc) is an interpolation function for evaluating the solution of a system of partial differential equations (PDEs), at a set of user-specified points. The solution of the system of equations (possibly with coupled ordinary differential equations) must be computed using a finite difference scheme or a Keller box scheme on a set of mesh points. nag_pde_interp_1d_fd (d03pzc) can then be employed to compute the solution at a set of points anywhere in the range of the mesh. It can also evaluate the first spatial derivative of the solution. It uses linear interpolation for approximating the solution.

4 References

None.

5 Arguments

Note: the arguments **x**, **m**, **u**, **npts** and **npde** must be supplied unchanged from the PDE function.

1: **npde** – Integer *Input*

On entry: the number of PDEs.

Constraint: **npde** ≥ 1.

2: **m** – Integer *Input*

On entry: the coordinate system used. If the call to nag_pde_interp_1d_fd (d03pzc) follows one of the finite difference functions then **m** must be the same argument **m** as used in that call. For the Keller box scheme only Cartesian coordinate systems are valid and so **m must** be set to zero. No check will be made by nag_pde_interp_1d_fd (d03pzc) in this case.

m $= 0$
Indicates Cartesian coordinates.

m $= 1$
Indicates cylindrical polar coordinates.

m $= 2$
Indicates spherical polar coordinates.

Constraints:

$0 \leq$ **m** ≤ 2 following a finite difference function;
m $= 0$ following a Keller box scheme function.

3: **u**[**npde** × **npts**] – const double *Input*

On entry: the PDE part of the original solution returned in the argument **u** by the PDE function.

Constraint: **npde** ≥ 1.

4: **npts** – Integer *Input*

On entry: the number of mesh points.

Constraint: **npts** ≥ 3.

5: **x**[**npts**] – const double *Input*

On entry: **x**$[i-1]$, for $i = 1, 2, \ldots,$ **npts**, must contain the mesh points as used by the PDE function.

6: **xp**[**intpts**] – const double *Input*

On entry: **xp**$[i-1]$, for $i = 1, 2, \ldots,$ **intpts**, must contain the spatial interpolation points.

Constraint: **x**$[0] \leq$ **xp**$[0] <$ **xp**$[1] < \cdots <$ **xp**[**intpts** $- 1] \leq$ **x**[**npts** $- 1]$.

7: **intpts** – Integer *Input*

On entry: the number of interpolation points.

Constraint: **intpts** ≥ 1.

8: **itype** – Integer *Input*

On entry: specifies the interpolation to be performed.

itype $= 1$
The solutions at the interpolation points are computed.

itype $= 2$
Both the solutions and their first derivatives at the interpolation points are computed.

Constraint: **itype** $= 1$ or 2.

9: **up**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **up** must be at least **npde** × **intpts** × **itype**.

The element **UP**(i, j, k) is stored in the array element **up**$[(k-1) \times$ **npde** $\times$ **intpts** $+ (j-1) \times$ **npde** $+ i - 1]$.

On exit: if **itype** $= 1$, **UP**$(i, j, 1)$, contains the value of the solution $U_i(x_j, t_{\mathrm{out}})$, at the interpolation points $x_j =$ **xp**$[j-1]$, for $j = 1, 2, \ldots,$ **intpts** and $i = 1, 2, \ldots,$ **npde**.

If **itype** $= 2$, **UP**$(i, j, 1)$ contains $U_i(x_j, t_{\mathrm{out}})$ and **UP**$(i, j, 2)$ contains $\frac{\partial U_i}{\partial x}$ at these points.

10: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_EXTRAPOLATION

On entry, interpolating point ⟨*value*⟩ with the value ⟨*value*⟩ is outside the **x** range.

NE_INT

On entry, **intpts** ≤ 0: **intpts** $=$ ⟨*value*⟩.

On entry, **itype** $=$ ⟨*value*⟩.
Constraint: **itype** $= 1$ or 2.

On entry, **m** $=$ ⟨*value*⟩.
Constraint: **m** $= 0$, 1 or 2.

On entry, **npde** $=$ ⟨*value*⟩.
Constraint: **npde** > 0.

On entry, **npts** $=$ ⟨*value*⟩.
Constraint: **npts** > 2.

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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_NOT_STRICTLY_INCREASING

On entry, interpolation points **xp** badly ordered: $I =$ ⟨*value*⟩, **xp**$[I-1] =$ ⟨*value*⟩, $J =$ ⟨*value*⟩ and **xp**$[J-1] =$ ⟨*value*⟩.

On entry, mesh points **x** badly ordered: $I =$ ⟨*value*⟩, **x**$[I-1] =$ ⟨*value*⟩, $J =$ ⟨*value*⟩ and **x**$[J-1] =$ ⟨*value*⟩.

7 Accuracy

See the PDE function documents.

8 Parallelism and Performance

nag_pde_interp_1d_fd (d03pzc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

See Section 10 in nag_pde_parab_1d_fd (d03pcc), nag_pde_parab_1d_fd_ode_remesh (d03ppc) and nag_pde_parab_1d_keller_ode_remesh (d03prc).