NAG Library Function Document

nag_ode_ivp_rk_interp_setup (d02phc)

1 Purpose

nag_ode_ivp_rk_interp_setup (d02phc) is a reverse communication function that computes the interpolant for evaluation by nag_ode_ivp_rk_interp_eval (d02pjc) anywhere on an integration step taken by nag_ode_ivp_rk_step_revcomm (d02pgc). The direct communication version of the nag_ode_ivp_rk_interp_setup (d02phc) and nag_ode_ivp_rk_interp_eval (d02pjc) pair is nag_ode_ivp_rkts_interp (d02psc). A significant difference in functionality between the forward and reverse communication versions is that nag_ode_ivp_rk_interp_setup (d02phc) and nag_ode_ivp_rk_interp_eval (d02pjc) can interpolate for the high-order Runge–Kutta method.

2 Specification

```
#include <nag.h>
#include <nagd02.h>
void nag_ode_ivp_rk_interp_setup (Integer *irevcm, Integer n, Integer nwant,
     double *t, double y[], const double yp[], double wcomm[],
     Integer lwcomm, Integer iwsav[], double rwsav[], NagError *fail)
```

3 Description

nag_ode_ivp_rk_interp_setup (d02phc) and its associated functions (nag_ode_ivp_rk_step_revcomm (d02pgc), nag_ode_ivp_rk_interp_eval (d02pjc), nag_ode_ivp_rkts_setup (d02pqc), nag_ode_ivp_rkts_reset_tend (d02prc), nag_ode_ivp_rkts_diag (d02ptc) and nag_ode_ivp_rkts_errass (d02puc)) solve the initial value problem for a first-order system of ordinary differential equations. The functions, based on Runge–Kutta methods and derived from RKSUITE (see Brankin *et al.* (1991)), integrate

$$y' = f(t, y) \quad \text{given} \quad y(t_0) = y_0$$

where y is the vector of n solution components and t is the independent variable.

nag_ode_ivp_rk_step_revcomm (d02pgc) computes the solution at the end of an integration step. Using the information computed on that step nag_ode_ivp_rk_interp_setup (d02phc) computes the interpolant which can be evaluated at any point on that step by nag_ode_ivp_rk_interp_eval (d02pjc). If **method** = Nag_RK_2_3 then there is enough information available from the stages of the last step to provide an interpolant of sufficient order of accuracy; no further derivative evaluations will therefore be requested. If **method** = Nag_RK_4_5 then the interpolant is an order 8 continuous Runge–Kutta process that requires a further 3 stages of derivative evaluations that will be requested in turn before a final exit. If **method** = Nag_RK_7_8 was specified in the call to setup function nag_ode_ivp_rkts_setup (d02pqc) then the interpolant is a continuous Runge–Kutta process requiring a further 7 stages of derivative evaluations that will be requested in turn.

4 References

Brankin R W, Gladwell I and Shampine L F (1991) RKSUITE: A suite of Runge–Kutta codes for the initial value problems for ODEs *SoftReport 91-S1* Southern Methodist University

5 Arguments

Note: this function uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **irevcm**. Between intermediate exits and re-entries, **all arguments other than those specified by the value of irevcm must remain unchanged**.

1: **irevcm** – Integer * *Input/Output*

On initial entry: **irevcm** must be set to zero to indicate that the interpolant for a new step is being taken.

On intermediate re-entry: **irevcm** should remain unchanged.

On intermediate exit: **irevcm** returns a value 1 to indicate that a function evaluation is required prior to re-entry; the value of the derivatives must be returned in **yp** where the value of t is supplied in **t** and the values $y(t)$ are supplied in the array **y**.

On final exit:

irevcm $= -1$
Successful exit; **rwsav** and **wcomm** contain details of the interpolant.

irevcm $= -2$
Error exit; **fail** should be interrogated to determine the nature of the error.

2: **n** – Integer *Input*

On entry: n, the number of ordinary differential equations in the system to be solved by the integration function.

Constraint: **n** ≥ 1. This must be the same value as supplied in a previous call to nag_ode_ivp_rkts_setup (d02pqc).

3: **nwant** – Integer *Input*

On entry: the number of components of the solution to be computed. The first **nwant** components are evaluated.

Constraint: $1 \leq$ **nwant** $\leq$ **n**.

4: **t** – double * *Output*

On intermediate exit: **t** contains the value of the independent variable t at which the derivatives y' are to be evaluated.

On final exit: contains no useful information.

5: **y**[**n**] – double *Output*

On intermediate exit: **y** contains the value of the solution y at which the derivatives y' are to be evaluated.

On final exit: contains no useful information.

6: **yp**[**n**] – const double *Input*

On initial entry: need not be set.

On intermediate re-entry: **yp** must contain the values of the derivatives y'_i for the given values of the arguments t, y_i.

7: **wcomm**[**lwcomm**] – double *Communication Array*

On entry: **wcomm** need not be set.

On intermediate re-entry: **wcomm** contains the partial computation of the polynomial coefficients corresponding to a continuous Runge–Kutta process for interpolating medium and high order Runge–Kutta methods.

On final exit: if **method** = Nag_RK_4_5 or Nag_RK_7_8, **wcomm** contains details of the interpolant which must be passed unchanged to nag_ode_ivp_rk_interp_eval (d02pjc) for evaluation of the interpolant.

8: **lwcomm** – Integer *Input*

On entry: length of **wcomm**.

If in a previous call to nag_ode_ivp_rkts_setup (d02pqc):

method = Nag_RK_2_3, **lwcomm** must be at least 1.

method = Nag_RK_4_5, **lwcomm** must be at least $\mathbf{n} + \max(\mathbf{n}, 5 \times \mathbf{nwant})$.

method = Nag_RK_7_8, **lwcomm** must be at least $8 \times \mathbf{nwant}$.

9: **iwsav**[**130**] – Integer *Communication Array*

10: **rwsav**[$\mathbf{32} \times \mathbf{n} + \mathbf{350}$] – double *Communication Array*

On entry: these must be the same arrays supplied in a previous call nag_ode_ivp_rk_step_rev comm (d02pgc). They must remain unchanged between calls.

On exit: information about the integration for use on subsequent calls to nag_ode_ivp_rk_ step_revcomm (d02pgc), nag_ode_ivp_rk_interp_eval (d02pjc) or other associated functions.

11: **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_INT

On entry, **lwcomm** = ⟨*value*⟩.
Constraint: for **method** = Nag_RK_2_3, **lwcomm** ≥ 1.

NE_INT_2

On entry, **lwcomm** = ⟨*value*⟩ and **nwant** = ⟨*value*⟩.
Constraint: for **method** = Nag_RK_7_8, **lwcomm** $\geq 8 \times$ **nwant**.

On entry, **nwant** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: $1 \leq$ **nwant** $\leq$ **n**.

NE_INT_3

On entry, **lwcomm** = ⟨*value*⟩, **n** = ⟨*value*⟩ and **nwant** = ⟨*value*⟩.
Constraint: for **method** = Nag_RK_4_5, **lwcomm** $\geq \mathbf{n} + \max(\mathbf{n}, 5 \times \mathbf{nwant})$.

NE_INT_CHANGED

On entry, **n** = ⟨*value*⟩, but the value passed to the setup function was **n** = ⟨*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 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_MISSING_CALL

On entry, a previous call to the setup function has not been made or the communication arrays have become corrupted, or a catastrophic error has already been detected elsewhere.
You cannot continue integrating the problem.

You cannot call this function before you have called the step integrator.

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_PREV_CALL_INI

You cannot call this function after the integrator has returned an error.

NE_RK_INVALID_CALL

You cannot call this function after the range integrator has been called.

7 Accuracy

The computed values will be of a similar accuracy to that computed by nag_ode_ivp_rk_step_revcomm (d02pgc).

8 Parallelism and Performance

nag_ode_ivp_rk_interp_setup (d02phc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

See Section 10 in nag_ode_ivp_rk_step_revcomm (d02pgc).