NAG Library Function Document

nag_ode_ivp_rk_reset_tend (d02pwc)

1 Purpose

nag_ode_ivp_rk_reset_tend (d02pwc) is a function to reset the end-point in an integration performed by nag_ode_ivp_rk_onestep (d02pdc).

2 Specification

```
#include <nag.h>
#include <nagd02.h>
void nag_ode_ivp_rk_reset_tend (double tend_new, Nag_ODE_RK *opt,
     NagError *fail)
```

3 Description

nag_ode_ivp_rk_reset_tend (d02pwc) and its associated functions (nag_ode_ivp_rk_setup (d02pvc), nag_ode_ivp_rk_onestep (d02pdc), nag_ode_ivp_rk_interp (d02pxc), nag_ode_ivp_rk_errass (d02pzc)) 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 (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.

This function is used to reset the final value of the independent variable, t_f when the integration is already underway. It can be used to extend or reduce the range of integration. The new value must be beyond the current value of the independent variable (as returned in **tnow** by nag_ode_ivp_rk_onestep (d02pdc)) in the current direction of integration. It is much more efficient to use nag_ode_ivp_rk_reset_tend (d02pwc) for this purpose than to use nag_ode_ivp_rk_setup (d02pvc) which involves the overhead of a complete restart of the integration.

If you want to change the direction of integration then you must restart by a call to nag_ode_ivp_rk_setup (d02pvc).

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

1: **tend_new** – double *Input*

On entry: the new value for t_f

Constraint: sign(**tend_new** − **tnow**) = sign(**tend** − **tstart**), where **tstart** and **tend** are as supplied in the previous call to nag_ode_ivp_rk_setup (d02pvc) and **tnow** is returned by the preceding call to nag_ode_ivp_rk_onestep (d02pdc). **tend** must be distinguishable from **tnow** for the method and the precision of the machine being used.

2: **opt** – Nag_ODE_RK * *Input/Output*

On entry: the structure of type Nag_ODE_RK as output from nag_ode_ivp_rk_onestep (d02pdc). You must not change this structure.

On exit: **opt** is suitably modified to reset the end-point.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_MEMORY_FREED

Internally allocated memory has been freed by a call to nag_ode_ivp_rk_free (d02ppc) without a subsequent call to the setup function nag_ode_ivp_rk_setup (d02pvc).

NE_MISSING_CALL

Previous call to nag_ode_ivp_rk_onestep (d02pdc) has not been made, hence nag_ode_ivp_rk_reset_tend (d02pwc) must not be called.

NE_PREV_CALL

The previous call to a function had resulted in a severe error. You must call nag_ode_ivp_rk_setup (d02pvc) to start another problem.

NE_PREV_CALL_INI

The previous call to the function nag_ode_ivp_rk_onestep (d02pdc) had resulted in a severe error. You must call nag_ode_ivp_rk_setup (d02pvc) to start another problem.

NE_RK_DIRECTION_NEG

Integration is proceeding in the negative direction with the current value for the independent variable **t** being $\langle value \rangle$. However **tend_new** has been set to $\langle value \rangle$. **tend_new** must be less than **t**.

NE_RK_DIRECTION_POS

Integration is proceeding in the positive direction with the current value for the independent variable **t** being $\langle value \rangle$. However **tend_new** has been set to $\langle value \rangle$. **tend_new** must be greater than **t**.

NE_RK_INVALID_CALL

The function to be called as specified in the setup function nag_ode_ivp_rk_setup (d02pvc) was nag_ode_ivp_rk_range (d02pcc). However the actual call was made to nag_ode_ivp_rk_reset_tend (d02pwc). This is not permitted.

NE_RK_STEP

The current value of the independent variable t is $\langle value \rangle$. The **tend_new** that is supplied has $\text{abs}(\textbf{tend\_new} - t) = \langle value \rangle$. For the method and the precision of the computer being used, this difference must be at least $\langle value \rangle$.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

We integrate a two body problem. The equations for the coordinates $(x(t), y(t))$ of one body as functions of time t in a suitable frame of reference are

$$x'' = \frac{-x}{r^3} \quad y'' = \frac{-y}{r^3}, r = \sqrt{(x^2 + y^2)}.$$

The intial conditions

$$x(0) = 1 - \epsilon, x'(0) = 0 \quad y(0) = 0, y'(0) = \sqrt{\frac{1+\epsilon}{1-\epsilon}}$$

lead to elliptic motion with $0 < \epsilon < 1$. We select $\epsilon = 0.7$ and repose as

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_4 \\ y_3' &= \tfrac{-y_1}{r^3} \\ y_4' &= \tfrac{-y_1}{r^3} \end{aligned}$$

over the range $[0, 6\pi]$. We use relative error control with threshold values of 1.0e−10 for each solution component and compute the solution at intervals of length π across the range using nag_ode_ivp_rk_reset_tend (d02pwc) to reset the end of the integration range. We use a high order Runge–Kutta method (**method** = Nag_RK_7_8) with tolerances **tol** = 1.0e−4 and **tol** = 1.0e−5 in turn so that we may compare the solutions. The value of π is obtained by using nag_pi (X01AAC).

10.1 Program Text

```
/* nag_ode_ivp_rk_reset_tend (d02pwc) Example Program.
 *
 * Copyright 1992 Numerical Algorithms Group.
 *
 * Mark 3, 1992.
 * Mark 7 revised, 2001.
 * Mark 8 revised, 2004.
 *
 */

#include <nag.h>
#include <math.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagd02.h>
#include <nagx01.h>

#ifdef __cplusplus
extern "C" {
#endif
static void NAG_CALL f(Integer neq, double t1, const double y[], double yp[],
                       Nag_User *comm);
#ifdef __cplusplus
}
#endif

#define NEQ  4
#define ZERO 0.0
#define ONE  1.0
#define SIX  6.0
#define ECC  0.7

int main(void)
{
  static Integer use_comm[1] = {1};
  Integer         exit_status = 0, i, j, neq, nout;
  NagError        fail;
  Nag_ErrorAssess errass;
  Nag_ODE_RK      opt;
  Nag_RK_method   method;
```

```
  Nag_User        comm;
  double          hstart, pi, tend, tfinal, *thres = 0, tinc, tnow, tol,
                  tstart, *ynow = 0;
  double          *ypnow = 0, *ystart = 0;

  INIT_FAIL(fail);

  printf(
          "nag_ode_ivp_rk_reset_tend (d02pwc) Example Program Results\n");

  /* For communication with user-supplied functions: */
  comm.p = (Pointer)

  /* Set initial conditions and input for nag_ode_ivp_rk_setup (d02pvc) */
  neq = NEQ;
  if (neq >= 1)
    {
      if (!(thres = NAG_ALLOC(neq, double)) ||
          !(ynow = NAG_ALLOC(neq, double)) ||
          !(ypnow = NAG_ALLOC(neq, double)) ||
          !(ystart = NAG_ALLOC(neq, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
    }
  else
    {
      exit_status = 1;
      return exit_status;
    }

  /* nag_pi (x01aac).
   * pi
   */
  pi = nag_pi;
  tstart = ZERO;
  ystart[0] = ONE - ECC;
  ystart[1] = ZERO;
  ystart[2] = ZERO;
  ystart[3] = sqrt((ONE+ECC)/(ONE-ECC));
  tfinal = SIX*pi;
  for (i = 0; i < neq; i++)
    thres[i] = 1.0e-10;
  errass = Nag_ErrorAssess_off;
  hstart = ZERO;
  method = Nag_RK_7_8;
  /*
   * Set control for output
   */
  nout = 6;
  tinc = tfinal/nout;
  for (i = 1; i <= 2; i++)
    {
      if (i == 1)
        tol = 1.0e-4;
      else
        tol = 1.0e-5;
      j = nout - 1;
      tend = tfinal - j*tinc;
      /* nag_ode_ivp_rk_setup (d02pvc).
       * Setup function for use with nag_ode_ivp_rk_range (d02pcc)
       * and/or nag_ode_ivp_rk_onestep (d02pdc)
       */
      nag_ode_ivp_rk_setup(neq, tstart, ystart, tend, tol, thres, method,
                           Nag_RK_onestep, errass, hstart, &opt, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_ode_ivp_rk_setup (d02pvc).\n%s\n",
                  fail.message);
```

```
          exit_status = 1;
          goto END;
        }

      printf("\nCalculation with tol = %10.1e\n\n", tol);
      printf("   t        y1        y2        y3        y4\n\n");
      printf("%7.3f   %7.4f   %7.4f   %7.4f   %7.4f\n",
              tstart, ystart[0], ystart[1], ystart[2], ystart[3]);
      do
        {
          do
            {
              /* nag_ode_ivp_rk_onestep (d02pdc).
               * Ordinary differential equations solver, initial value
               * problems, one time step using Runge-Kutta methods
               */
              nag_ode_ivp_rk_onestep(neq, f, &tnow, ynow, ypnow, &opt, &comm,
                                     &fail);
              if (fail.code != NE_NOERROR)
                {
                  printf(
                          "Error from nag_ode_ivp_rk_onestep (d02pdc).\n%s\n",
                          fail.message);
                  exit_status = 1;
                  goto END;
                }
            } while (tnow < tend);
          printf("%7.3f   %7.4f   %7.4f   %7.4f   %7.4f\n", tnow,
                  ynow[0],
                  ynow[1], ynow[2], ynow[3]);
          j = j - 1;
          tend = tfinal - j*tinc;
          /* nag_ode_ivp_rk_reset_tend (d02pwc).
           * A function to re-set the end point following a call to
           * nag_ode_ivp_rk_onestep (d02pdc)
           */
          nag_ode_ivp_rk_reset_tend(tend, &opt, &fail);
          if (fail.code != NE_NOERROR)
            {
              printf(
                      "Error from nag_ode_ivp_rk_reset_tend (d02pwc).\n%s\n",
                      fail.message);
              exit_status = 1;
              goto END;
            }
        } while (tnow < tfinal);
      printf("\nCost of the integration in evaluations of f is"
              " %ld\n\n", opt.totfcn);
      /* nag_ode_ivp_rk_free (d02ppc).
       * Freeing function for use with the Runge-Kutta suite (d02p
       * functions)
       */
      nag_ode_ivp_rk_free(&opt);
    }
 END:
  NAG_FREE(thres);
  NAG_FREE(ynow);
  NAG_FREE(ypnow);
  NAG_FREE(ystart);
  return exit_status;
}
static void NAG_CALL f(Integer neq, double t, const double y[], double yp[],
                       Nag_User *comm)

{
  double r, rp3;
  Integer *use_comm = (Integer *)comm->p;

  if (use_comm[0])
    {
      printf("(User-supplied callback f, first invocation.)\n");
```

```
      use_comm[0] = 0;
    }

  r = sqrt(y[0]*y[0] + y[1]*y[1]);
  rp3 = pow(r, 3.0);
  yp[0] = y[2];
  yp[1] = y[3];
  yp[2] = -y[0]/rp3;
  yp[3] = -y[1]/rp3;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_ode_ivp_rk_reset_tend (d02pwc) Example Program Results

Calculation with tol =    1.0e-04

   t        y1        y2        y3        y4

  0.000    0.3000    0.0000    0.0000    2.3805
(User-supplied callback f, first invocation.)
  3.142   -1.7000    0.0000   -0.0000   -0.4201
  6.283    0.3000   -0.0000    0.0001    2.3805
  9.425   -1.7000    0.0000   -0.0000   -0.4201
 12.566    0.3000   -0.0003    0.0016    2.3805
 15.708   -1.7001    0.0001   -0.0001   -0.4201
 18.850    0.3000   -0.0010    0.0045    2.3805

Cost of the integration in evaluations of f is 571


Calculation with tol =    1.0e-05

   t        y1        y2        y3        y4

  0.000    0.3000    0.0000    0.0000    2.3805
  3.142   -1.7000   -0.0000    0.0000   -0.4201
  6.283    0.3000    0.0000   -0.0000    2.3805
  9.425   -1.7000    0.0000   -0.0000   -0.4201
 12.566    0.3000   -0.0001    0.0004    2.3805
 15.708   -1.7000    0.0000   -0.0000   -0.4201
 18.850    0.3000   -0.0003    0.0012    2.3805

Cost of the integration in evaluations of f is 748
```