# 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

This function 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

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

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

This function is used to reset the 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 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. Parameters

**tend_new**

Input: the new value for $t_f$

Constraints: 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.

**opt**

Input: the structure of type **Nag_ODE_RK** as output from nag_ode_ivp_rk_onestep (d02pdc). This structure must not be changed by the user.

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

**fail**

The NAG error parameter, see the Essential Introduction to the NAG C Library.

## 5. Error Indications and Warnings

**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_RK_INVALID_CALL**

The function to be called as specified in the setup routine 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_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_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_POS**
Integration is proceeding in the positive direction with the current value for the independent variable **t** being ⟨*value*⟩. However **tend_new** has been set to ⟨*value*⟩. **tend_new** must be greater than **t**.

**NE_RK_DIRECTION_NEG**
Integration is proceeding in the negative direction with the current value for the independent variable **t** being ⟨*value*⟩. However **tend_new** has been set to ⟨*value*⟩. **tend_new** must be less than **t**.

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

**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 set up function nag_ode_ivp_rk_setup (d02pvc).

## 6. Further Comments

None.

### 6.1. Accuracy

Not applicable.

### 6.2. References

Brankin R W, Gladwell I and Shampine L F (1991) *RKSUITE: a suite of Runge–Kutta codes for the initial value problem for ODEs* SoftReport 91-S1, Department of Mathematics, Southern Methodist University, Dallas, TX 75275, U.S.A.

## 7. See Also

nag_ode_ivp_rk_setup (d02pvc)
nag_ode_ivp_rk_onestep (d02pdc)
nag_ode_ivp_rk_interp (d02pxc)
nag_ode_ivp_rk_errass (d02pzc)

## 8. 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

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

The intial conditions

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

lead to elliptic motion with $0 < \varepsilon < 1$. We select $\varepsilon = 0.7$ 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 $\pi$ 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 $\pi$ is obtained by using X01AAC.

## 8.1. Program Text

```
/* nag_ode_ivp_rk_reset_tend(d02pwc) Example Program
 *
 * Copyright 1994 Numerical Algorithms Group.
 *
 * Mark 3, 1994.
 *
 */

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

#ifdef NAG_PROTO
static void f(Integer neq, double t1, double y[], double yp[], Nag_User *comm);
#else
static void f();
#endif

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

main()
{
  Integer neq;
  double hstart, pi, tnow, tend;
  double tol, tstart, tinc, tfinal;
  Integer i, j, nout;
  Nag_RK_method method;
  Nag_ErrorAssess errass;
  Nag_ODE_RK opt;
  Nag_User comm;
  double  thres[NEQ], ynow[NEQ], ypnow[NEQ], ystart[NEQ];

  Vprintf("d02pwc Example Program Results\n");

  /*  Set initial conditions and input for d02pvc */
  neq = NEQ;
  pi = X01AAC;
  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;
      if (i==2) tol = 1.0e-5;
      j = nout - 1;
      tend = tfinal - j*tinc;
      d02pvc(neq, tstart, ystart, tend, tol, thres, method,
             Nag_RK_onestep, errass, hstart, &opt, NAGERR_DEFAULT);
      Vprintf("\nCalculation with tol = %8.1e\n\n",tol);
      Vprintf ("   t         y1         y2         y3         y4\n\n");
      Vprintf("%7.3f   %7.4f   %7.4f   %7.4f   %7.4f\n",
              tstart, ystart[0], ystart[1], ystart[2], ystart[3]);
      do{
        do
          {

            d02pdc(neq, f, &tnow, ynow, ypnow, &opt, &comm,
                   NAGERR_DEFAULT);
          } while (tnow<tend);
        Vprintf("%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;
        d02pwc(tend, &opt, NAGERR_DEFAULT);
      } while (tnow<tfinal);
      Vprintf("\nCost of the integration in evaluations of f is %ld\n\n",
              opt.totfcn);
      d02ppc(&opt);
    }
  exit(EXIT_SUCCESS);
}
#ifdef NAG_PROTO
static void f(Integer neq, double t, double y[], double yp[], Nag_User *comm)
#else
     static void f(neq, t, y, yp, comm)
     Integer neq;
     double t;
     double y[], yp[];
     Nag_User *comm;
#endif

{
  double  r, rp3;
  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;
}
```

## 8.2. Program Data

None.

## 8.3. Program Results

```
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
  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
```