NAG Library Function Document

nag_ode_ivp_rkts_reset_tend (d02prc)

1 Purpose

nag_ode_ivp_rkts_reset_tend (d02prc) resets the end point in an integration performed by
nag_ode_ivp_rkts_onestep (d02pfc).

2 Specification

```c
#include <nag.h>
#include <nagd02.h>
void nag_ode_ivp_rkts_reset_tend (double tendnu, Integer iwsav[],
                                 double rwsav[], NagError *fail)
```

3 Description

nag_ode_ivp_rkts_reset_tend (d02prc) and its associated functions (nag_ode_ivp_rkts_onestep (d02pfc),
nag_ode_ivp_rkts_setup (d02pqc), nag_ode_ivp_rkts_interp (d02psc), 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_rkts_reset_tend (d02prc) 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 \( t_{\text{now}} \) by
nag_ode_ivp_rkts_onestep (d02pfc)) in the current direction of integration. It is much more efficient to
use nag_ode_ivp_rkts_reset_tend (d02prc) for this purpose than to use nag_ode_ivp_rkts_setup (d02pqc)
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_rkts_setup (d02pqc).

4 References

initial value problems for ODEs SoftReport 91-S1 Southern Methodist University

5 Arguments

1:  **tendnu** – double

   *Input*

   On entry: the new value for \( t_f \).

   *Constraint: sign(tendnu – tnow) = sign(tend - tstart),* where \( tstart \) and \( tend \) are as supplied in
   the previous call to nag_ode_ivp_rkts_setup (d02pqc) and \( t_{\text{now}} \) is returned by the preceding call
to nag_ode_ivp_rkts_onestep (d02pfc) (i.e., integration must proceed in the same direction as
before). \( tendnu \) must be distinguishable from \( t_{\text{now}} \) for the method and the *machine precision*
   being used.
2: \textbf{iwsav[130]} – Integer 
\textit{Communication Array}

3: \textbf{rwsav[350]} – double 
\textit{Communication Array}

\textbf{Note:} the communication array \texttt{rwsav} used by the other functions in the suite must be used here however, only the first 350 elements will be referenced.

\textit{On entry:} these must be the same arrays supplied in a previous call to \texttt{nag_ode_ivp_rkts_onestep (d02pfc)}. They must remain unchanged between calls.

\textit{On exit:} information about the integration for use on subsequent calls to \texttt{nag_ode_ivp_rkts_onestep (d02pfc)} or other associated functions.

4: \textbf{fail} – \texttt{NagError *} 
\textit{Input/Output}

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

\section{Error Indicators and Warnings}

\textbf{NEALLOC_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NEBAD_PARAM}

On entry, argument \texttt{〈value〉} had an illegal value.

\textbf{NEINTERNAL_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 3.6.6 in the Essential Introduction for further information.

\textbf{NEMISSING_CALL}

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

\textbf{NE_NO_LICENCE}

Your licence key may have expired or may not have been installed correctly.

See Section 3.6.5 in the Essential Introduction for further information.

\textbf{NEPREV_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.

\textbf{NEPREV_CALL_INI}

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

\textbf{NERK_DIRECTION_NEG}

On entry, \texttt{tendnu} is not beyond \texttt{tnow} (step integrator) in the direction of integration. The direction is negative, \texttt{tendnu = 〈value〉} and \texttt{tnow = 〈value〉}.

\textbf{NERK_DIRECTION_POS}

On entry, \texttt{tendnu} is not beyond \texttt{tnow} (step integrator) in the direction of integration. The direction is positive, \texttt{tendnu = 〈value〉} and \texttt{tnow = 〈value〉}.
NE_RK_INVALID_CALL
You cannot call this function when the range integrator has been used.

NE_RK_STEP
On entry, tendnu is too close to tnow (step integrator). Their difference is \( \text{value} \), but this quantity must be at least \( \text{value} \).

7 Accuracy
Not applicable.

8 Parallelism and Performance
Not applicable.

9 Further Comments
None.

10 Example
This example integrates 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

\[
x'' = -\frac{x}{r^3}
\]

\[
y'' = -\frac{y}{r^3}, \quad r = \sqrt{x^2 + y^2}.
\]

The initial conditions
\[
x(0) = 1 - \epsilon, \quad x'(0) = 0
\]
\[
y(0) = 0, \quad y'(0) = \sqrt{\frac{1 + \epsilon}{1 - \epsilon}}
\]
lead to elliptic motion with \( 0 < \epsilon < 1 \). \( \epsilon = 0.7 \) is selected and the system of ODEs is reposed as

\[
y'_1 = y_3
\]
\[
y'_2 = y_4
\]
\[
y'_3 = -\frac{y_1}{r^3}
\]
\[
y'_4 = -\frac{y_2}{r^3}
\]
over the range \([0, 6\pi]\). Relative error control is used with threshold values of \( 1.0 \times 10^{-10} \) for each solution component and compute the solution at intervals of length \( \pi \) across the range using nag_ode_ivp_rkts_reset_tend (d02prc) to reset the end of the integration range. A high-order Runge–Kutta method (method = Nag_RK_7_8) is also used with tolerances \( \text{tol} = 1.0 \times 10^{-4} \) and \( \text{tol} = 1.0 \times 10^{-5} \) in turn so that the solutions may be compared.
10.1 Program Text

/* nag_ode_ivp_rkts_reset_tend (d02prc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 24, 2013. */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd02.h>

#ifdef __cplusplus
extern "C" {
#endif
static void NAG_CALL f(double t, Integer n, const double *y,
double *yp, Nag_Comm *comm);
#ifdef __cplusplus
}
#endif
#define N 4

int main(void)
{
  /* Scalars */
double tol0 = 1.0e-3;
Integer npts = 6, exit_status = 0;
Integer liwsav, lrwsav, n;
double hnext, hstart, tendnu, tfinal, tinc, tgot, tol, tstart,
waste;
Integer fevals, i, j, k, stepcost, stepsok;
  /* Arrays */
static double ruser[1] = {-1.0};
double *rwsav = 0, *thresh = 0, *ygot = 0, *yinit = 0, *ypgot = 0;
Integer *iwsav = 0;
char nag_enum_arg[40];
  /* NAG types */
NagError fail;
Nag_RK_method method;
Nag_ErrorAssess errass;
Nag_Comm comm;
INIT_FAIL(fail);
printf("nag_ode_ivp_rkts_reset_tend (d02prc) Example Program Results\n\n");
/* For communication with user-supplied functions: */
comm.user = ruser;
n = N;
liwsav = 130;
lrwsav = 350 + 32 * n;
if (! (thresh = NAG_ALLOC(n, double)) ||
! (ygot = NAG_ALLOC(n, double)) ||
! (yinit = NAG_ALLOC(n, double)) ||
! (ypgot = NAG_ALLOC(n, double)) ||
! (iwsav = NAG_ALLOC(liwsav, Integer)) ||
! (rwsav = NAG_ALLOC(lrwsav, double)))
{
printf("Allocation failure\n");
exit_status = -1;
goto END;
}
/* Skip heading in data file*/
#ifdef _WIN32
scanf_s("%*
");
#endif
}
#else
scanf("%*[\n] ");
#endif

/* Set initial conditions for ODE and parameters for the integrator. */
#ifdef _WIN32
    scanf_s(" %39s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s%*[\n] ", nag_enum_arg);
#endif

/* nag_enum_name_to_value (x04nac) Converts NAG enum member name to value. */
method = (Nag_RK_method) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
    scanf_s(" %39s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s%*[\n] ", nag_enum_arg);
#endif
errass = (Nag_ErrorAssess) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
    scanf_s("%lf%lf%*[\n] ", &tstart, &tfinal);
#else
    scanf("%lf%lf%*[\n] ", &tstart, &tfinal);
#endif
    for (j = 0; j < n; j++)
    #ifdef _WIN32
        scanf_s("%lf", &yinit[j]);
    #else
        scanf("%lf", &yinit[j]);
    #endif
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
    #ifdef _WIN32
        scanf_s("%lf%*[\n] ", &hstart);
    #else
        scanf("%lf%*[\n] ", &hstart);
    #endif
    for (j = 0; j < n; j++)
    #ifdef _WIN32
        scanf_s("%lf", &thresh[j]);
    #else
        scanf("%lf", &thresh[j]);
    #endif
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif

tinc = (tfinal - tstart)/(double) (npts);
tol = tol0;
    for (i = 1; i <= 2; i++)
    {
        tol = tol * 0.1;
tendnu = tstart + tinc;
        /* Initialize Runge-Kutta method for integrating ODE using
        * nag_ode_ivp_rkts_setup (d02pqc).
        */
        nag_ode_ivp_rkts_setup(n, tstart, tendnu, yinit, tol, thresh, method,
        errass, hstart, iwsav, rwsav, &fail);
        if (fail.code != NE_NOERROR)
        {
            printf("Error from nag_ode_ivp_rkts_setup (d02pqc).\n%5s
", fail.message);
        }
exit_status = 1;
goto END;
}

printf(" Calculation with tol = %8.1e\n", tol);
printf(" t y1 y2 y3 y4\n");
printf("%6.3f", tstart);
for (k = 0; k < n; k++)
    printf(" %7.3f", yinit[k]);
printf("\n");
tgot = tstart;
while (tgot < tfinal)
{
    /* Solve ODE by Runge-Kutta method by a sequence of single steps using
     * nag_ode_ivp_rkts_onestep (d02pfc).
     */
    nag_ode_ivp_rkts_onestep(f, n, &tgot, ygot, ypgot, &comm, 
iwsav, rwsav, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_ode_ivp_rkts_onestep (d02pfc).\n", fail.message);
        exit_status = 2;
        goto END;
    }

    /* When incremental stage in t has been reached:
     * print solution and reset end time for next stage.
     */
    if (tgot == tendnu)
    {
        printf("%6.3f", tgot);
        for (k = 0; k < n; k++)
            printf(" %7.3f", ygot[k]);
        printf("\n");

        /* Reset end-time for integration by adding increment tinc using
         * nag_ode_ivp_rkts_reset_tend (d02prc): resets end of range for
         * nag_ode_ivp_rkts_onestep (d02pfc).
         */
        tendnu = tendnu + tinc;
        nag_ode_ivp_rkts_reset_tend(tendnu, iwsav, rwsav, &fail);
        if (fail.code != NE_NOERROR)
        {
            printf("Error from nag_ode_ivp_rkts_reset_tend (d02prc).\n", fail.message);
            exit_status = 3;
            goto END;
        }
    }

    /* Get diagnostics on whole integration using
     * nag_ode_ivp_rkts_diag (d02ptc).
     */
    nag_ode_ivp_rkts_diag(&fevals, &stepcost, &waste, &stepsok, &hnext, 
iwsav, rwsav, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_ode_ivp_rkts_diag (d02ptc).\n", fail.message);
        exit_status = 4;
        goto END;
    }
    printf("Cost of the integration in evaluations of f is %6"NAG_IFMT"\n", 
fevals);
}

END:
NAG_FREE(thresh);
NAG_FREE(yinit);
NAG_FREE(ygot);
NAG_FREE(ypgot);
NAG_FREE(rwsav);
NAG_FREE(iwsav);
return exit_status;
}

static void NAG_CALL f(double t, Integer n, const double *y, double *yp,
   Nag_Comm *comm)
{
    /* Scalars */
    double r;
    if (comm->user[0] == -1.0)
    {
        printf("(User-supplied callback f, first invocation.)\n");
        comm->user[0] = 0.0;
    }
    r = sqrt(y[0]*y[0] + y[1]*y[1]);
    yp[0] = y[2];
    yp[1] = y[3];
    yp[2] = -y[0]/ pow(r, 3);
    yp[3] = -y[1]/ pow(r, 3);
}

10.2 Program Data

nag_ode_ivp_rkts_reset_tend (d02prc) Example Program Data
Nag_RK_7_8 : method
Nag_ErrorAssess_off : errass
0.0 18.849559215387594307 : tstart, tfinal
0.3 0.0 0.0 2.38047614284761666599 : yinit
0.0 : hstart
1.0E-10 1.0E-10 1.0E-10 1.0E-10 : thresh

10.3 Program Results

nag_ode_ivp_rkts_reset_tend (d02prc) Example Program Results
Calculation with tol = 1.0e-04
   t   y1   y2   y3   y4
0.000  0.300  0.000  0.000  2.380
(User-supplied callback f, first invocation.)
   t   y1   y2   y3   y4
  3.142 -1.700  0.000 -0.000 -0.420
  6.283  0.300 -0.000  0.000  2.380
  9.425 -1.700  0.000 -0.000 -0.420
 12.566  0.300 -0.000  0.002  2.380
 15.708 -1.700  0.000 -0.000 -0.420
 18.850  0.300 -0.001  0.004  2.380
Cost of the integration in evaluations of f is 571

Calculation with tol = 1.0e-05
   t   y1   y2   y3   y4
0.000  0.300  0.000  0.000  2.380
  3.142 -1.700 -0.000  0.000 -0.420
  6.283  0.300  0.000 -0.000  2.380
  9.425 -1.700  0.000 -0.000 -0.420
 12.566  0.300 -0.000  0.002  2.380
 15.708 -1.700  0.000 -0.000 -0.420
 18.850  0.300 -0.000  0.001  2.380
Cost of the integration in evaluations of f is 748
Example Program
Solution with TOL = 0.1e-04

$x$ Deviation from True Ellipse

Orbit - $y$

deviation

Orbit - $x$