NAG Library Function Document

nag_ode_ivp_rkts_errass (d02puc)

1 Purpose

nag_ode_ivp_rkts_errass (d02puc) provides details about global error assessment computed during an integration with either nag_ode_ivp_rkts_range (d02pec) or nag_ode_ivp_rkts_onestep (d02pfc).

2 Specification

```c
#include <nag.h>
#include <nagd02.h>

void nag_ode_ivp_rkts_errass (Integer n, double rmserr[], double *errmax,
   double *termx, Integer iwsav[], double rwsav[], NagError *fail)
```

3 Description

nag_ode_ivp_rkts_errass (d02puc) and its associated functions (nag_ode_ivp_rkts_range (d02pec), nag_ode_ivp_rkts_onestep (d02pfc), nag_ode_ivp_rkts_setup (d02pqc), nag_ode_ivp_rkts_reset_tend (d02prc), nag_ode_ivp_rkts_interp (d02psc) and nag_ode_ivp_rkts_diag (d02ptc)) 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

\[ \frac{dy}{dt} = f(t, y) \quad \text{given} \quad y(t_0) = y_0 \]

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

After a call to nag_ode_ivp_rkts_range (d02pec) or nag_ode_ivp_rkts_onestep (d02pfc), nag_ode_ivp_rkts_errass (d02puc) can be called for information about error assessment, if this assessment was specified in the setup function nag_ode_ivp_rkts_setup (d02pqc). A more accurate ‘true’ solution \( \hat{y} \) is computed in a secondary integration. The error is measured as specified in nag_ode_ivp_rkts_setup (d02pqc) for local error control. At each step in the primary integration, an average magnitude \( \mu_i \) of component \( y_i \) is computed, and the error in the component is

\[ \frac{|y_i - \hat{y}_i|}{\max(\mu_i, \text{thresh}[i-1])} \]

It is difficult to estimate reliably the true error at a single point. For this reason the RMS (root-mean-square) average of the estimated global error in each solution component is computed. This average is taken over all steps from the beginning of the integration through to the current integration point. If all has gone well, the average errors reported will be comparable to \( \text{tol} \) (see nag_ode_ivp_rkts_setup (d02pqc)). The maximum error seen in any component in the integration so far and the point where the maximum error first occurred are also reported.

4 References


5 Arguments

1: \( n \) – Integer 

\( Input \)

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

Constraint: \( n \geq 1 \).
On exit: \( \text{rmserr}[i - 1] \) approximates the RMS average of the true error of the numerical solution for the \( i \)th solution component, for \( i = 1, 2, \ldots, n \). The average is taken over all steps from the beginning of the integration to the current integration point.

On exit: the maximum weighted approximate true error taken over all solution components and all steps.

On exit: the first value of the independent variable where an approximate true error attains the maximum value, \( \text{errmax} \).

On entry: these must be the same arrays supplied in a previous call to \text{nag_ode_ivp_rkts_range} (d02pec) or \text{nag_ode_ivp_rkts_onestep} (d02pfc). They must remain unchanged between calls.

On exit: information about the integration for use on subsequent calls to \text{nag_ode_ivp_rkts_range} (d02pec) or \text{nag_ode_ivp_rkts_onestep} (d02pfc) or other associated functions.

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

6 Error Indicators and Warnings

\text{NE_ALLOC_FAIL}
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\text{NE_BAD_PARAM}
On entry, argument \textit{value} had an illegal value.

\text{NE_ERRASS_REQ}
No error assessment is available since you did not ask for it in your call to the setup function.

\text{NE_INT_2}
On entry, \( n = \langle \text{value} \rangle \), but the value passed to the setup function was \( n = \langle \text{value} \rangle \).

\text{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 3.6.6 in the Essential Introduction for further information.

\text{NE_MISSING_CALL}
You cannot call this function before you have called the integrator.

\text{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.
NE_PREV_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.

NE_RK_INVALID_CALL
You have already made one call to this function after the integrator could not achieve specified accuracy. You cannot call this function again.

NE_RK_NOSTEP
No error assessment is available since the integrator has not actually taken any successful steps.

7 Accuracy
Not applicable.

8 Parallelism and Performance
Not applicable.

9 Further Comments
If the integration has proceeded ‘well’ and the problem is smooth enough, stable and not too difficult then the values returned in the arguments rmserr and errmax should be comparable to the value of tol specified in the prior call to nag_ode_ivp_rkts_setup (d02pqc).

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 \([0, 3\pi]\). Relative error control is used with threshold values of 1.0e–10 for each solution component and a high-order Runge–Kutta method (method = Nag_RK_7_8) with tolerance tol = 1.0e–6.
Note that for illustration purposes since it is not necessary for this problem, this example integrates to the end of the range regardless of efficiency concerns (i.e., returns from nag_ode_ivp_rkts_range (d02pec) with fail.code = NE_RK_POINTS, NE_STIFF_PROBLEM or NW_RK_TOO_MANY).

10.1 Program Text

/* nag_ode_ivp_rkts_errass (d02puc) 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 */
    Integer exit_status = 0;
    Integer liwsav, lrwsav, lwcomm, n;
    double errmax, hnext, hstart, tend, terrmx, tgot, tol, tstart,
    twant, waste;
    Integer fevals, j, k, stepcost, stepsok;
    /* Arrays */
    static double ruser[1] = {-1.0};
    double *rmserr = 0, *rwsav = 0, *lwcomm = 0, n;
    double *ygot = 0, *yinit = 0, *ymax = 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);

    n = N;
    liwsav = 130;
    lrwsav = 350 + 32 * n;
    lwcomm = 6*n;

    printf("nag_ode_ivp_rkts_errass (d02puc) Example Program Results\n\n");

    /* For communication with user-supplied functions: */
    comm.user = ruser;

    if (!rmserr || !rwsav || !lwcomm || !ygot || !yinit || !ymax || !ypgot || !wcomm)
        !wcomm = NAG_ALLOC(lwcomm, double))
    
}
!(rw sav = N AG_ALLOC(lrwsav, double)) ||
!(iw sav = N AG_ALLOC(liwsav, integer))
)
{
 printf("Allocation failure\n");
 exit_status = -1;
 goto END;
}

/* Skip heading in data file*/
#endif
scanf_s("%*\[\n"]);
#else
 scanf("%*\[\n"]);
#endif

/* Set initial conditions for ODE and parameters for the integrator. */
#endif
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);
#endif
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);
#endif
scanf_s("%lf%lf%*\[\n"] , &tstart, &tend);
#else
 scanf("%lf%lf%*\[\n"] , &tstart, &tend);
#endif

for (j = 0; j < n; j++)
#endif
scanf_s("%lf", &yinit[j]);
#else
 scanf("%lf", &yinit[j]);
#endif

for (j = 0; j < n; j++)
#endif
scanf_s("%lf%lf%*\[\n"] , &hstart, &tol);
#else
 scanf("%lf%lf%*\[\n"] , &hstart, &tol);
#endif

for (j = 0; j < n; j++)
#endif
scanf("%lf", &thresh[j]);
#else
 scanf("%lf", &thresh[j]);
#endif

/* Initialize Runge-Kutta method for integrating ODE using
 * nag_ode_ivp_rkts_setup (d02pqc).
 */
nag_ode_ivp_rkts_setup(n, tstart, tend, yinit, tol, thresh, method, errass, hstart, iwsav, rwsav, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ode_ivp_rkts_setup (d02pqc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

printf(" Calculation with tol = %8.1e\n", tol);
printf(" t y1 y1'\n");
for (k = 0; k < n; k++)
    printf(" %8.4f", yinit[k]);
printf("\n");
twant = tend;
tgot = tstart;
while (tgot<twant)
{
    /**< Solve ODE by Runge-Kutta method up to next time increment using
       * nag_ode_ivp_rkts_range (d02pec).
       */
    nag_ode_ivp_rkts_range(f, n, twant, &tgot, ygot, ypgot, ymax, &comm, iwsav, rwsav, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_ode_ivp_rkts_range (d02pec).\n%s\n", fail.message);
        exit_status = 2;
        goto END;
    }
    printf(" %8.4f", ygot[k]);
    printf("\n");
}

 /**< Compute and print error estimates using
    * nag_ode_ivp_rkts_errass (d02puc).
    */
    nag_ode_ivp_rkts_errass(n, rmserr, &errmax, &terrmx, iwsav, rwsav, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_ode_ivp_rkts_errass (d02puc).\n%s\n", fail.message);
        exit_status = 3;
        goto END;
    }
    printf("\n Componentwise error assessment\n");
    printf("\n");
    for (j=0; j<n; j++)
        printf("%11.2e", rmserr[j]);
    printf("\n\n Worst global error observed was %9.2e", errmax);
    printf(" - occuring at t = %6.3f\n", terrmx);

 /**< 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%s\n", fail.message);
        exit_status = 4;
        goto END;
    }
    printf(" Cost of the integration in evaluations of f is %6"NAG_IFMT":\n", d02puc}
END:
NAG_FREE(rmserr);
NAG_FREE(thresh);
NAG_FREE(ygot);
NAG_FREE(yinit);
NAG_FREE(ymax);
NAG_FREE(ypgot);
NAG_FREE(rwsav);
NAG_FREE(iwsav);
NAG_FREE(wcomm);
return exit_status;
}

static void NAG_CALL f(double t, Integer n, const double *y, double *yp, Nag_Comm *comm)
{
    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]);
    r = r*r*r;
    yp[0] = y[2];
    yp[1] = y[3];
    yp[2] = -y[0]/r;
    yp[3] = -y[1]/r;
}

10.2 Program Data

nag_ode_ivp_rkts_errass (d02puc) Example Program Data
    Nag_RK_7_8 : method
    Nag_ErrorAssess_on : errass
    0.0 9.4247796076937971538 : tstart, tend
    0.3 0.0 0.0 2.38047614284761666599 : yinit(1:n)
    0.0 1.0E-6 : hstart, tol
    1.0E-10 1.0E-10 1.0E-10 1.0E-10 : thresh(1:n)

10.3 Program Results

nag_ode_ivp_rkts_errass (d02puc) Example Program Results

Calculation with tol = 1.0e-06
    t    y1     y1'
    0.000  0.3000  0.0000  0.0000  2.3805
(User-supplied callback f, first invocation.)
    9.425 -1.7000  0.0000 -0.0000 -0.4201
Componentwise error assessment
    3.81e-06  7.10e-06  6.92e-06  2.10e-06
Worst global error observed was 3.43e-05 - occurring at t = 6.302
Cost of the integration in evaluations of f is 1361
Example Program
Solution to a Two-body Problem using High-order Runge-Kutta

Orbit - y

Deviation from True Ellipse

Orbit - x

d02puc
NAG Library Manual

Example Program
Solution to a Two-body Problem using High-order Runge-Kutta

Orbit - y

Deviation from True Ellipse

Orbit - x