nag_ode_ivp_rk_errass (d02pzc)

1. Purpose

nag_ode_ivp_rk_errass (d02pzc) provides details about global error assessment computed during an integration with either nag_ode_ivp_rk_range (d02pcc) or nag_ode_ivp_rk_onestep (d02pdc).

2. Specification

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

void nag_ode_ivp_rk_errass(Integer neq, double rmserr[], double *errmax, double *ter-
rmx,
            Nag_ODE_RK *opt, NagError *fail)
```

3. Description

This function and its associated functions (nag_ode_ivp_rk_range (d02pcc), nag_ode_ivp_rk_onestep (d02pdc), nag_ode_ivp_rk_setup (d02pvc), nag_ode_ivp_rk_reset_tend (d02pwc), nag_ode_ivp_rk_interp (d02pxc)) 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 **neq** solution components and t is the independent variable.

After a call to nag_ode_ivp_rk_range (d02pcc) or nag_ode_ivp_rk_onestep (d02pdc), nag_ode_ivp_rk_errass can be called for information about error assessment, if this assessment was specified in the setup routine nag_ode_ivp_rk_setup (d02pvc). A more accurate "true" solution $\hat{y}$ is computed in a secondary integration. The error is measured as specified in nag_ode_ivp_rk_setup (d02pvc) for local error control. At each step in the primary integration, an average magnitude σ_i of component y_i is computed, and the error in the component is

$$\frac{| y_i - \hat{y}_i |}{\max(\sigma_i, thres(i))}.$$

where $thres(i)$ denotes the threshold value used in the error requirement, see nag_ode_ivp_rk_setup (d02pvc).
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 **tol** (see nag_ode_ivp_rk_setup (d02pvc)). 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. Parameters

neq
Input: the number of ordinary differential equations in the system.
Constraint: **neq** ≥ 1.

rmserr[neq]
Output: **rmserr**$[i-1]$ approximates the RMS average of the true error of the numerical solution for the ith solution component y_i, for $i = 1, 2, \ldots,$**neq**. The average is taken over all steps from the beginning of the integration to the current integration point.

errmax
Output: the maximum weighted approximate true error taken over all solution components and all steps.

terrmx
Output: the first value of the independent variable where an approximate true error attains the maximum value, **errmax**.

opt
Input: the structure of type **Nag_ODE_RK** as output from nag_ode_ivp_rk_range (d02pcc) or nag_ode_ivp_rk_onestep (d02pdc). This structure must not be changed by the user.
Output: some members of **opt** are changed internally.

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

NE_NEQ
The value of **neq** supplied is not the same as that given to the set up function.

NE_MISSING_CALL
Previous call to nag_ode_ivp_rk_onestep (d02pdc) has not been made, hence nag_ode_ivp_rk_errass must not be called.
Previous call to nag_ode_ivp_rk_range (d02pcc) has not been made, hence nag_ode_ivp_rk_errass must not be called.

NE_ERRASS_REQ
No error assessment is available as it was not requested in the call to nag_ode_ivp_rk_setup (d02pvc).

NE_RK_NOSTEP
The integrator has not actually taken any successful steps. This function must not be called in this circumstance.

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

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_rk_setup (d02pvc).

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_range (d02pcc)

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 $[0, 3\pi]$. We use relative error control 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. The value of π is obtained by using X01AAC.

Note, for illustration purposes we select to integrate to the end of the range regardless of efficiency concerns.

8.1. Program Text

```
/* nag_ode_ivp_rk_errass(d02pzc) 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 THREE 3.0
#define ECC 0.7

main()
{
  double hstart, pi, tgot, tend,tol, tstart, twant;
  double errmax, terrmx;
  Integer neq, i;
  Nag_ErrorAssess errass;
  Nag_ODE_RK opt;
  Nag_User comm;
  Nag_RK_method method;
  static NagError fail;

  double thres[NEQ], ygot[NEQ], ypgot[NEQ], ystart[NEQ];
  double ymax[NEQ], rmserr[NEQ];
```

```
  Vprintf("d02pzc 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));
  tend = THREE*pi;
  for (i=0; i<neq; i++)
    thres[i] = 1.0e-10;
  errass = Nag_ErrorAssess_on;
  hstart = ZERO;
  tol = 1.0e-6;
  method  = Nag_RK_7_8;
  d02pvc(neq, tstart, ystart, tend, tol, thres, method,
         Nag_RK_range, errass, hstart, &opt, NAGERR_DEFAULT);


  Vprintf("\nCalculation with tol = %8.1e\n\n",tol);
  Vprintf("    t          y1          y2          y3          y4\n\n");
  Vprintf("%8.3f    %8.4f    %8.4f    %8.4f    %8.4f\n", tstart, ystart[0],
          ystart[1], ystart[2], ystart[3]);

  twant = tend;
  do
    d02pcc(neq, f, twant, &tgot, ygot, ypgot, ymax, &opt, &comm,
           &fail);
  while (fail.code==NE_RK_PDC_POINTS || fail.code==NE_STIFF_PROBLEM);

  if (fail.code != NE_NOERROR)
    {
      Vprintf("%s\n", fail.message);
      exit(EXIT_FAILURE);
    }
  else
    {
      Vprintf("%8.3f    %8.4f    %8.4f    %8.4f    %8.4f\n\n", tgot, ygot[0],
              ygot[1], ygot[2], ygot[3]);
      d02pzc(neq, rmserr, &errmax, &terrmx, &opt, NAGERR_DEFAULT);
      Vprintf ("Componentwise error assessment\n          ");
      for (i=0; i<neq; i++)
        Vprintf("%9.2e ", rmserr[i]);
      Vprintf("\n\n");
      Vprintf("Worst global error observed was %9.2e - \
it occurred at t = %6.3f\n\n", errmax, terrmx);
      Vprintf("Cost of the integration in evaluations of f is %ld\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

```
d02pzc Example Program Results

Calculation with tol =  1.0e-06

    t          y1          y2          y3          y4

   0.000      0.3000      0.0000      0.0000      2.3805
   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 - it occurred at t =  6.302

Cost of the integration in evaluations of f is 1361
```