NAG Toolbox for MATLAB

d02pz

1 Purpose

d02pz provides details about global error assessment computed during an integration with either d02pc or d02pd.

2 Syntax

```
[rmserr, errmax, terrmx, ifail] = d02pz(neq, work)
```

3 Description

d02pz and its associated functions (d02pc, d02pd, d02pv, d02pw, d02px and d02py) 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

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

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

After a call to d02pc or d02pd, d02pz can be called for information about error assessment, if this assessment was specified in the setup function d02pv. A more accurate 'true' solution $\hat{y}$ is computed in a secondary integration. The error is measured as specified in d02pv for local error control. At each step in the primary integration, an average magnitude μ_i of component y_i is computed, and the error in the component is

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

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 d02pv). 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

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 Parameters

5.1 Compulsory Input Parameters

1: **neq – int32 scalar**

n, the number of ordinary differential equations in the system to be solved by the integration function.

Constraint: **neq** ≥ 1.

2: **work**$(*)$ **– double array**

Note: the dimension of the array **work** must be at least **lenwrk** (see d02pv).

This **must** be the same array as supplied to d02pc or d02pd and **must** remain unchanged between calls.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **rmserr**($*$) – **double array**

Note: the dimension of the array **rmserr** must be at least n.

rmserr(i) approximates the RMS average of the true error of the numerical solution for the ith 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.

2: **errmax – double scalar**

The maximum weighted approximate true error taken over all solution components and all steps.

3: **terrmx – double scalar**

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

4: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

An invalid call to d02pz has been made, for example without a previous call to d02pc or d02pd, or without error assessment having been specified in a call to d02pv. You cannot continue integrating the problem.

7 Accuracy

Not applicable.

8 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 d02pv.

9 Example

d02pz_f.m

```
function [yp] = f(t, y)
  yp = zeros(4, 1);

  r = sqrt(y(1)^2 + y(2)^2);
  yp(1) = y(3);
  yp(2) = y(4);
```

```
   yp(3) = -y(1)/rˆ3;
   yp(4) = -y(2)/rˆ3;
```

```
tstart = 0;
ystart = [0.3;
     0;
     0;
     2.380476142847617];
tend = 9.424777960769379;
tol = 1e-06;
thres = [1e-10;
     1e-10;
     1e-10;
     1e-10];
method = int32(3);
task = 'Usual Task';
errass = true;
lenwrk = int32(128);
neq = int32(4);
twant = 9.424777960769379;
ygot = [0; 0; 0; 0];
ymax = [0; 0; 0; 0];
[work, ifail] = ...
       d02pv(tstart, ystart, tend, tol, thres, method, task, errass,
lenwrk);
[tgot, ygot, ypgot, ymax, work, ifail] = d02pc('d02pz_f', neq, twant,
ygot, ymax, work);
[rmserr, errmax, terrmx, ifail] = d02pz(neq, work)
```

```
rmserr =
   1.0e-05 *
    0.3808
    0.7105
    0.6925
    0.2102
errmax =
   3.4348e-05
terrmx =
    6.3024
ifail =
           0
```