NAG Toolbox for MATLAB

d02py

1 Purpose

d02py provides details about an integration performed by either d02pc or d02pd.

2 Syntax

```
[totfcn, stpcst, waste, stpsok, hnext, ifail] = d02py()
```

3 Description

d02py and its associated functions (d02pc, d02pd, d02pv, d02pw, d02px, d02pz) 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, d02py can be called to obtain information about the cost of the integration and the size of the next step.

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

None.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **totfcn – int32 scalar**

The total number of evaluations of f used in the primary integration so far; this does not include evaluations of f for the secondary integration specified by a prior call to d02pv with **errass** = **true**.

2: **stpcst – int32 scalar**

The cost in terms of number of evaluations of f of a typical step with the method being used for the integration. The method is specified by the parameter **method** in a prior call to d02pv.

3: **waste – double scalar**

The number of attempted steps that failed to meet the local error requirement divided by the total number of steps attempted so far in the integration. A 'large' fraction indicates that the integrator is having trouble with the problem being solved. This can happen when the problem is ‘stiff’ and also when the solution has discontinuities in a low-order derivative.

4: **stpsok – int32 scalar**

The number of accepted steps.

5: **hnext – double scalar**

The step size the integrator will attempt to use for the next step.

6: **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 d02py has been made, for example without a previous call to d02pc or d02pd. You cannot continue integrating the problem.

7 Accuracy

Not applicable.

8 Further Comments

When a secondary integration has taken place, that is when global error assessment has been specified using **errass** = **true** in a prior call to d02pv, then the approximate extra number of evaluations of f used is given by 2 × **stpsok** × **stpcst** for **method** = 2 or 3 and 3 × **stpsok** × **stpcst** for **method** = 1.

9 Example

```
d02pd_f.m
```

```
function [yp] = f(t, y)
  yp = zeros(2, 1);
  yp(1) = y(2);
  yp(2) = -y(1);
```

```
d02px_f.m
```

```
function [yp] = f(t, y)
  yp = zeros(2, 1);
  yp(1) = y(2);
  yp(2) = -y(1);
```

```
tstart = 0;
ystart = [0; 1];
tend = 6.283185307179586;
tol = 0.001;
thres = [1e-08; 1e-08];
method = int32(2);
task = 'Complex Task';
errass = false;
```

```
lenwrk = int32(64);
neq = int32(2);
twant = 0.3926990816987241;
reqest = 'Both';
nwant = int32(1);
wrkint = zeros(7, 1);
[work, ifail] = ...
       d02pv(tstart, ystart, tend, tol, thres, method, task, errass,
lenwrk);
npts = 16;
tnow = tend-1;
while (tnow < tend)
  [tnow, ynow, ypnow, work, ifail] = d02pd('d02pd_f', neq, work);
  j = npts -1;
  tinc = tend/npts;
  while (twant <= tnow)
    [ywant, ypwant, work, wrkint, ifail] = ...
    d02px(neq, twant, reqest, nwant, 'd02px_f', work, wrkint);
    j = j-1;
    twant = tend -j*tinc;
  end
end
[totfcn, stpcst, waste, stpsok, hnext, ifail] = d02py()
```

```
totfcn =
          68
stpcst =
           7
waste =
    0.1429
stpsok =
           6
hnext =
    1.4387
ifail =
           0
```