NAG Toolbox for MATLAB

d02ny

1 Purpose

d02ny is a diagnostic function which you may call either after any user-specified exit or after a mid-integration error exit from any of the integrators in sub-chapter D02M/N.

2 Syntax

```
[hu, h, tcur, tolsf, nst, nre, nje, nqu, nq, niter, imxer, algequ,
ifail] = d02ny(neq, ldysav, rwork, inform)
```

3 Description

d02ny permits you to inspect statistics produced by any integrator in this sub-chapter. These statistics concern the integration only.

4 References

See the D02M/N Sub-chapter Introduction.

5 Parameters

5.1 Compulsory Input Parameters

1: **neq – int32 scalar**

the value used for the parameter **neq** when calling the integrator.

Constraint: **neq** ≥ 1.

2: **ldysav – int32 scalar**

The value used for the parameter **ldysav** when calling the integrator.

Constraint: **ldysav** $\geq$ **neq**.

3: **rwork**$(\mathbf{50} + \mathbf{4} \times \mathbf{ldysav})$ **– double array**

Contains information supplied by the integrator.

4: **inform**$(\mathbf{23})$ **– int32 array**

Contains information supplied by the integrator.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **hu – double scalar**

The last successful step size.

2: **h – double scalar**

The proposed next step size for continuing the integration.

3: **tcur – double scalar**

The value of the independent variable, t, which the integrator has actually reached. **tcur** will always be at least as far as the output value of the argument t in the direction of integration, but may be further (if overshooting and interpolation at **tout** was specified, e.g., see d02nb).

4: **tolsf – double scalar**

A tolerance scale factor, **tolsf** ≥ 1.0, which is computed when a request for too much accuracy is detected by the integrator (indicated by a return with **ifail** $= 3$ or 14). If **itol** is left unaltered but **rtol** and **atol** are uniformly scaled up by a factor of **tolsf** the next call to the integrator is deemed likely to succeed.

5: **nst – int32 scalar**

The number of steps taken in the integration so far.

6: **nre – int32 scalar**

The number of function or residual evaluations (user-supplied (sub)program **fcn** (e.g., see d02nb) or user-supplied (sub)program **resid** (e.g., see d02ng) calls) used in the integration so far.

7: **nje – int32 scalar**

The number of Jacobian evaluations used in the integration so far. This equals the number of matrix LU decompositions.

8: **nqu – int32 scalar**

The order of the method last used (successfully) in the integration.

9: **nq – int32 scalar**

The proposed order of the method for continuing the integration.

10: **niter – int32 scalar**

The number of iterations performed in the integration so far by the nonlinear equation solver.

11: **imxer – int32 scalar**

The index of the component of largest magnitude in the weighted local error vector (e_i/w_i), for $i = 1, 2, \ldots,$ **neq**.

12: **algequ(neq) – logical array**

algequ$(i) =$ **true** if the ith equation integrated was detected to be algebraic, otherwise **algequ**$(i) =$ **false**. Note that when the integrators for explicit equations are being used, then **algequ**$(i) =$ **false**, for $i = 1, 2, \ldots,$ **neq**.

13: **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

On entry, **neq** < 1,
or **ldysav** < 1,
or **neq** > **ldysav**.

7 Accuracy

Not applicable.

8 Further Comments

Statistics for sparse matrix linear algebra calls (if appropriate) may be determined by a call to d02nx.

9 Example

d02nb_fcn.m

```
function [f, ires] = fcn(neq, t, y, ires)
      f = zeros(3,1);
      f(1) = -0.04d0*y(1) + 1.0d4*y(2)*y(3);
      f(2) = 0.04d0*y(1) - 1.0d4*y(2)*y(3) - 3.0d7*y(2)*y(2);
      f(3) = 3.0d7*y(2)*y(2);
```

d02nb_jac.m

```
function p = jac(neq, t, y, h, d)
  p = zeros(neq, neq);
  hxd = h*d;
  p(1,1) = 1.0d0 - hxd*(-0.04d0);
  p(1,2) = -hxd*(1.0d4*y(3));
  p(1,3) = -hxd*(1.0d4*y(2));
  p(2,1) = -hxd*(0.04d0);
  p(2,2) = 1.0d0 - hxd*(-1.0d4*y(3)-6.0d7*y(2));
  p(2,3) = -hxd*(-1.0d4*y(2));
  p(3,2) = -hxd*(6.0d7*y(2));
  p(3,3) = 1.0d0 - hxd*(0.0d0);
```

d02nb_monitr.m

```
function [hnext, y, imon, inln, hmin, hmax] = ...
    monitr(neq, neqmax, t, hlast, hnext, y, ydot, ysave, r, acor, imon,
hmin, hmax, ngu)
  inln=int32(0);
```

```
neq = int32(3);
neqmax = int32(3);
t = 0;
tout = 10;
y = [1; 0; 0];
rwork = zeros(62,1);
rtol = [0.0001];
atol = [1e-07];
itol = int32(1);
inform = zeros(23, 1, 'int32');
```

```
ysave = zeros(3, 6);
wkjac = zeros(12, 1);
itask = int32(4);
itrace = int32(0);
[const, rwork, ifail] = ...
    d02nv(int32(3), int32(6), int32(5), 'Newton', false, zeros(6), ...
    10, 1e-10, 10, 0, int32(200), int32(5), 'Average-L2', rwork);
[rwork, ifail] = d02ns(int32(3), int32(3), 'Numerical', int32(12),
rwork);
[t, y, ydot, rwork, inform, ysave, wkjac, ifail] = ...
    d02nb(neq, t, tout, y, rwork, rtol, atol, itol, inform, ...
       'd02nb_fcn', ysave, 'd02nb_jac', wkjac, 'd02nb_monitr', itask,
itrace);
[hu, h, tcur, tolsf, nst, nre, nje, nqu, nq, niter, imxer, algequ, ifail]
= ...
    d02ny(neq, neqmax, rwork, inform)
```

```
hu =
    0.5187
h =
    0.5187
tcur =
    10
tolsf =
   9.0399e-13
nst =
          55
nre =
         132
nje =
          17
nqu =
           3
nq =
           3
niter =
          79
imxer =
           3
algequ =
     0
     0
     0
ifail =
           0
```