NAG Toolbox

nag_ode_ivp_stiff_integ_diag (d02ny)

1 Purpose

nag_ode_ivp_stiff_integ_diag (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 those integrators in Sub-chapter D02M–N that use methods set up by calls to nag_ode_ivp_stiff_dassl (d02mv), nag_ode_ivp_stiff_bdf (d02nv) or nag_ode_ivp_stiff_blend (d02nw).

2 Syntax

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

3 Description

nag_ode_ivp_stiff_integ_diag (d02ny) permits you to inspect statistics produced by any integrator in this sub-chapter that has been set up a call to one of nag_ode_ivp_stiff_dassl (d02mv), nag_ode_ivp_stiff_bdf (d02nv) or nag_ode_ivp_stiff_blend (d02nw). These statistics concern the integration only.

4 References

See the D02M–N Sub-chapter Introduction.

5 Parameters

5.1 Compulsory Input Parameters

1: **neq** – INTEGER

The value used for the argument **neq** when calling the integrator.

Constraint: **neq** ≥ 1.

2: **neqmax** – INTEGER

The value used for the argument **neqmax** when calling the integrator.

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

3: **rwork**$(\mathbf{50} + \mathbf{4} \times \mathbf{neq})$ – REAL (KIND=nag_wp) array

Contains information supplied by the integrator.

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

Contains information supplied by the integrator.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **hu** – REAL (KIND=nag_wp)

The last successful step size.

2: **h** – REAL (KIND=nag_wp)

The proposed next step size for continuing the integration.

3: **tcur** – REAL (KIND=nag_wp)

t, the value of the independent variable 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 nag_ode_ivp_stiff_exp_fulljac (d02nb)).

4: **tolsf** – REAL (KIND=nag_wp)

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** – INTEGER

The number of steps taken in the integration so far.

6: **nre** – INTEGER

The number of function or residual evaluations (**fcn** (e.g., see nag_ode_ivp_stiff_exp_fulljac (d02nb)) or **resid** (e.g., see nag_ode_ivp_stiff_imp_fulljac (d02ng)) calls) used in the integration so far.

7: **nje** – INTEGER

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

8: **nqu** – INTEGER

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

9: **nq** – INTEGER

The proposed order of the method for continuing the integration.

10: **niter** – INTEGER

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

11: **imxer** – INTEGER

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** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

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

ifail = −99

An unexpected error has been triggered by this routine. Please contact NAG.

ifail = −399

Your licence key may have expired or may not have been installed correctly.

ifail = −999

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

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

9 Example

See Section 10 in nag_ode_ivp_stiff_exp_fulljac (d02nb).

9.1 Program Text

```
function d02ny_example

fprintf('d02ny example results\n\n');

% Initialize integration method setup variables and arrays.
neq    = nag_int(3);
neqmax = nag_int(neq);
nwkjac = nag_int(neqmax*(neqmax + 1));
maxord = nag_int(5);
sdysav = nag_int(maxord+1);
maxstp = nag_int(200);
mxhnil = nag_int(5);

h0    = 0;
hmax  = 10;
hmin  = 1.0e-10;
tcrit = 20;
petzld = false;

const  = zeros(6, 1);
rwork  = zeros(50+4*neqmax, 1);

[const, rwork, ifail] = d02nv(neqmax, sdysav, maxord, 'Newton', petzld, ...
```

```
                              const, tcrit, hmin, hmax, h0, maxstp, ...
                              mxhnil, 'Average-L2', rwork);

% Setup for numerical Jacbian using d02ns
nwkjac = nag_int(neq*(neq+1));
[rwork, ifail] = d02ns( ...
 neq, neq, 'Numerical', nwkjac, rwork);

% Initialize variables and arrays for integration
t      = 0;
tout   = 10;
y      = [1; 0; 0];
rtol   = [0.0001];
atol   = [1e-07];
itol   = nag_int(1);
inform = zeros(23, 1, nag_int_name);
ysave  = zeros(neq, sdysav);
wkjac  = zeros(nwkjac, 1);
itask  = nag_int(4);
itrace = nag_int(0);

% Integrate ODE from t=0 to t=tout, no monitoring, using d02nb
[t, y, ydot, rwork, inform, ysave, wkjac, ifail] = ...
  d02nb( ...
  t, tout, y, rwork, rtol, atol, itol, inform, @fcn, ysave, ...
  @jac, wkjac, 'd02nby', itask, itrace);

% Get diagnostics
[hu, h, tcur, tolsf, nst, nre, nje, nqu, nq, nit, imxer, algequ, ifail] = ...
  d02ny( ...
         neq, neqmax, rwork, inform);

% Get solution at tout
[y, ifail] = d02mz( ...
                    tout, neq, neq, ysave, rwork);

% Print solution and diagnostics
fprintf(' At t = %8.3f, y(1:3) = %5.1f, %5.1f, %5.1f\n', t, y);
fprintf('\nDiagnostic information\n integration status:\n');
fprintf('   last and next step sizes = %8.5f, %8.5f\n',hu, h);
fprintf('   integration stopped at x = %8.5f\n',tcur);
fprintf(' algorithm statistics:\n');
fprintf('   number of time-steps and Newton iterations  = %5d %5d\n',nst,nit);
fprintf('   number of residual and jacobian evaluations = %5d %5d\n',nre,nje);
fprintf('   order of method last used and next to use   = %5d %5d\n',nqu,nq);
fprintf('   component with largest error                = %5d\n',imxer);

function [f, ires] = fcn(neq, t, y, ires)
  % Evaluate derivative vector.
  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);

function p = jac(neq, t, y, h, d, p)
  % Evaluate the Jacobian.
  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);
```

9.2 Program Results

```
     d02ny example results

 At t =   10.000, y(1:3) =   0.8,   0.0,   0.2

Diagnostic information
 integration status:
   last and next step sizes =  0.90178,  0.90178
   integration stopped at x = 10.76621
 algorithm statistics:
   number of time-steps and Newton iterations  =    55    78
   number of residual and jacobian evaluations =   128    16
   order of method last used and next to use   =     4     4
   component with largest error                =     3
```