NAG Toolbox

nag_ode_ivp_2nd_rkn_diag (d02ly)

1 Purpose

nag_ode_ivp_2nd_rkn_diag (d02ly) is a diagnostic function which may be called after a call of the integrator nag_ode_ivp_2nd_rkn (d02la).

2 Syntax

```
[hnext, hused, hstart, nsucc, nfail, natt, thres, thresp, ifail] =
nag_ode_ivp_2nd_rkn_diag(neq, rwork)
[hnext, hused, hstart, nsucc, nfail, natt, thres, thresp, ifail] = d02ly(neq,
rwork)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 22: *lrwork* was removed from the interface.

3 Description

nag_ode_ivp_2nd_rkn_diag (d02ly) permits you to extract information about the performance of nag_ode_ivp_2nd_rkn (d02la) and the setting of some optional parameters. It may be called only after a call of nag_ode_ivp_2nd_rkn (d02la).

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **neq** – INTEGER

The number of second-order ordinary differential equations solved by nag_ode_ivp_2nd_rkn (d02la). It must be the same as the argument **neq** supplied to nag_ode_ivp_2nd_rkn (d02la) and nag_ode_ivp_2nd_rkn_setup (d02lx).

2: **rwork**(*lrwork*) – REAL (KIND=nag_wp) array

This **must** be the same argument **rwork** as supplied to nag_ode_ivp_2nd_rkn (d02la). It is used to pass information from nag_ode_ivp_2nd_rkn (d02la) to nag_ode_ivp_2nd_rkn_diag (d02ly) and therefore the contents of this array **must not** be changed before calling nag_ode_ivp_2nd_rkn_diag (d02ly).

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **hnext** – REAL (KIND=nag_wp)

The next step size which nag_ode_ivp_2nd_rkn (d02la), if called, would attempt.

2: **hused** – REAL (KIND=nag_wp)

The last successful step size used by nag_ode_ivp_2nd_rkn (d02la).

3: **hstart** – REAL (KIND=nag_wp)

The initial step size used on the current integration problem by nag_ode_ivp_2nd_rkn (d02la).

4: **nsucc** – INTEGER

The number of steps attempted by nag_ode_ivp_2nd_rkn (d02la) that have been successful since the start of the current problem.

5: **nfail** – INTEGER

The number of steps attempted by nag_ode_ivp_2nd_rkn (d02la) that have failed since the start of the current problem.

6: **natt** – INTEGER

The number of steps attempted before the initial step was successful. Over a large number of problems the cost of an attempted step of this type is approximately half that of a normal attempted step.

7: **thres**(**neq**) – REAL (KIND=nag_wp) array

The ith solution threshold value used in the error control strategy. (See nag_ode_ivp_2nd_rkn_setup (d02lx).)

8: **thresp**(**neq**) – REAL (KIND=nag_wp) array

The ith derivative threshold value used in the error control strategy. (See nag_ode_ivp_2nd_rkn_setup (d02lx).)

9: **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 $= 1$

nag_ode_ivp_2nd_rkn (d02la) has not been called, or one or both of the arguments **neq** and $lrwork$ does not match the corresponding argument supplied to nag_ode_ivp_2nd_rkn_setup (d02lx).

This error exit can be caused if elements of **rwork** have been overwritten.

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

None.

9 Example

See Section 10 in nag_ode_ivp_2nd_rkn (d02la).

9.1 Program Text

```
     function d02ly_example

fprintf('d02ly example results\n\n');

% Variables and arrays for integration setup
neq    = 2;
h      = 0;
tol    = 1e-4;
thres  = zeros(neq, 1);
thresp = zeros(neq, 1);
maxstp = nag_int(1000);
start  = true;
onestp = true;
high   = false;
rwork  = zeros(16+20*neq,1);

% Integration setup
[startOut, rwork, ifail] = d02lx...
   (h, tol, thres, thresp, maxstp, start, onestp, high, rwork);

% Integration variables
t      = 0;
tend   = 20;
y      = [0.5; 0];
yp     = [0;   sqrt(3)];
ydp    = zeros(neq, 1);
tnext  = 2;
nwant  = nag_int(2);

fprintf('\n  T       Y(1)       Y(2)\n');
fprintf('%4.1f %10.5f %10.5f\n', t, y(1), y(2));

% Integrate by steps and interpolate onto selected values
while (t < tend && ifail == 0)
  [t, y, yp, ydp, rwork, ifail] = d02la(@fcn, t, tend, y, yp, ydp, rwork);
  while (tnext <= t && ifail == 0)
    [ywant, ypwant, ifail] = d02lz(t, y, yp, nwant, tnext, rwork);
    fprintf('%4.1f %10.5f %10.5f\n', tnext, ywant(1:2));
    tnext = tnext + 2;
  end
end

if (ifail == 0)
  [hnext, hused, hstart, nsucc, nfail, natt, thres, thresp, ifail] = ...
    d02ly(nwant, rwork);

  fprintf('\n Number of successful steps = %d\n', nsucc);
  fprintf(' Number of failed steps     = %d\n', nfail);
end
```

```
function ydp = fcn(neq, t, y)
% Evaluate second derivatives.
r      = sqrt(y(1)^2+y(2)^2)^3;
ydp(1) = -y(1)/r;
ydp(2) = -y(2)/r;
```

9.2 Program Results

```
     d02ly example results

  T       Y(1)       Y(2)
 0.0    0.50000    0.00000
 2.0   -1.20573    0.61357
 4.0   -1.33476   -0.47685
 6.0    0.35748   -0.44558
 8.0   -1.03762    0.73022
10.0   -1.42617   -0.32658
12.0    0.05515   -0.72032
14.0   -0.82880    0.81788
16.0   -1.48103   -0.16788
18.0   -0.26719   -0.84223
20.0   -0.57803    0.86339

 Number of successful steps = 108
 Number of failed steps     = 16
```