NAG Toolbox

nag_ode_ivp_stiff_contin (d02nz)

1 Purpose

nag_ode_ivp_stiff_contin (d02nz) is a setup function which must be called, if optional inputs need resetting, prior to a continuation call to 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

```
[rwork, ifail] = nag_ode_ivp_stiff_contin(neqmax, tcrit, h, hmin, hmax, maxstp, mxhnil, rwork)
[rwork, ifail] = d02nz(neqmax, tcrit, h, hmin, hmax, maxstp, mxhnil, rwork)
```

3 Description

nag_ode_ivp_stiff_contin (d02nz) is provided to permit you to reset many of the arguments which control the integration 'on the fly', that is in conjunction with the interrupt facility permitted through the argument **itask** of the integrator (e.g., see nag_ode_ivp_stiff_exp_fulljac (d02nb)). In addition to a number of arguments which you can set initially through one of the integrator setup functions, the step size to be attempted on the next step may be changed.

4 References

See the D02M–N Sub-chapter Introduction.

5 Parameters

5.1 Compulsory Input Parameters

1: **neqmax** – INTEGER

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

Constraint: **neqmax** ≥ 1.

2: **tcrit** – REAL (KIND=nag_wp)

A point beyond which integration must not be attempted. The use of **tcrit** is described under the argument **itask** in the specification for the integrator (e.g., see nag_ode_ivp_stiff_exp_fulljac (d02nb)). A value, 0.0 say, must be specified even if **itask** subsequently specifies that **tcrit** will not be used.

3: **h** – REAL (KIND=nag_wp)

The next step size to be attempted. Set **h** $= 0.0$ if the current value of **h** is not to be changed.

4: **hmin** – REAL (KIND=nag_wp)

The minimum absolute step size to be allowed. Set **hmin** $= 0.0$ if this option is not required. Set **hmin** < 0.0 if the current value of **hmin** is not to be changed.

5: **hmax** – REAL (KIND=nag_wp)

The maximum absolute step size to be allowed. Set **hmax** $= 0.0$ if this option is not required. Set **hmax** < 0.0 if the current value of **hmax** is not to be changed.

6: **maxstp** – INTEGER

The maximum number of steps to be attempted during one call to the integrator after which it will return with **ifail** $= 2$ (see nag_ode_ivp_stiff_exp_bandjac (d02nc)). Set **maxstp** $= 0$ if this option is not required. Set **maxstp** < 0 if the current value of **maxstp** is not to be changed.

7: **mxhnil** – INTEGER

The maximum number of warnings printed (if **itrace** ≥ 0, e.g., see nag_ode_ivp_stiff_exp_fulljac (d02nb)) per problem when $t + h = t$ on a step ($h =$ current step size). If **mxhnil** ≤ 0, a default value of 10 is assumed.

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

This must be the same workspace array as the array **rwork** supplied to the integrator. It is used to pass information from the integrator to nag_ode_ivp_stiff_contin (d02nz) and therefore its contents must not be changed before calling nag_ode_ivp_stiff_contin (d02nz).

5.2 Optional Input Parameters

None.

5.3 Output Parameters

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

2: **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 $= 1$

neqmax < 1.

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_stiff_exp_bandjac (d02nc).

9.1 Program Text

```
     function d02nz_example

fprintf('d02nz example results\n\n');

% Initialize variables and arrays for setup routine
n      = nag_int(3);
ord    = nag_int(11);
sdy    = nag_int(ord + 3);
petzld = false;
con    = zeros(6);
tcrit  = 0;
hmin   = 1.0e-10;
hmax   = 10;
h0     = 0;
maxstp = nag_int(200);
mxhnil = nag_int(5);
lrwork = 50 + 4*n;
rwork  = zeros(lrwork);
[const, rwork, ifail] = d02nw(n, sdy, ord, con, tcrit, hmin, hmax, h0, ...
                              maxstp, mxhnil, 'Average-L2', rwork);

% Setup for banded Jacbian using d02nt
ml     = nag_int(1);
mu     = nag_int(2);
nwkjac = nag_int(15);
[rwork, ifail] = d02nt(n, n, 'Analytical', ml, mu, nwkjac, n, rwork);

% Initialize variables and arrays for integration
t      = 0;
tout   = 5;
y      = [1; 0; 0];
rtol   = [0.0001];
atol   = [1e-07; 1e-08; 1e-07];
itol   = nag_int(2);
inform = zeros(23, 1, nag_int_name);
ysave  = zeros(n, sdy);
wkjac  = zeros(nwkjac, 1);
jacpvt = zeros(n, 1, nag_int_name);
itask  = nag_int(1);
itrace = nag_int(0);

% Integrate ODE from t=0 to t=tout, no monitoring, using d02nc
[t, y, ydot, rwork, inform, ysave, wkjac, jacpvt, ifail] = ...
    d02nc(t, tout, y, rwork, rtol, atol, itol, inform, @fcn, ysave, @jac, ...
          wkjac, jacpvt, 'd02nby', itask, itrace);

fprintf('Solution y and derivative y'' at t = %7.4f is:\n',t);
fprintf('\n %10s %10s\n','y','y''');
for i=1:n
  fprintf(' %10.4f %10.4f\n',y(i),ydot(i));
end

% Call to d02nz to alow continuation of integration up to t=10.
[rwork, ifail] = d02nz(n, tcrit, h0, hmin, hmax, maxstp, mxhnil, rwork);

tout = 10;
% Integrate ODE from t to t=tout, no monitoring, using d02nc
[t, y, ydot, rwork, inform, ysave, wkjac, jacpvt, ifail] = ...
    d02nc(t, tout, y, rwork, rtol, atol, itol, inform, @fcn, ysave, @jac, ...
          wkjac, jacpvt, 'd02nby', itask, itrace);

fprintf('Solution y and derivative y'' at t = %7.4f is:\n',t);
fprintf('\n %10s %10s\n','y','y''');
for i=1:n
```

```
  fprintf(' %10.4f %10.4f\n',y(i),ydot(i));
end

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, ml, mu, pIn)
% Evaluate the Jacobian.
p = zeros(ml+mu+1, neq);
hxd = h*d;
p(1,1) = 1.0d0 - hxd*(-0.04d0);
p(2,1) = -hxd*(1.0d4*y(3));
p(3,1) = -hxd*(1.0d4*y(2));
p(1,2) = -hxd*(0.04d0);
p(2,2) = 1.0d0 - hxd*(-1.0d4*y(3)-6.0d7*y(2));
p(3,2) = -hxd*(-1.0d4*y(2));
p(1,3) = -hxd*(6.0d7*y(2));
p(2,3) = 1.0d0 - hxd*(0.0d0);
```

9.2 Program Results

```
     d02nz example results

Solution y and derivative y' at t =  5.0000 is:

          y         y'
     0.8915    -0.0124
     0.0000    -0.0000
     0.1085     0.0124
Solution y and derivative y' at t = 10.0000 is:

          y         y'
     0.8414    -0.0078
     0.0000    -0.0000
     0.1586     0.0078
```