NAG Toolbox

nag_ode_ivp_stiff_bandjac_setup (d02nt)

1 Purpose

nag_ode_ivp_stiff_bandjac_setup (d02nt) is a setup function which you must call prior to an integrator in Sub-chapter D02M–N, if banded matrix linear algebra is required.

2 Syntax

```
[rwork, ifail] = nag_ode_ivp_stiff_bandjac_setup(neq, neqmax, jceval, ml, mu, nwkjac, njcpvt, rwork)
[rwork, ifail] = d02nt(neq, neqmax, jceval, ml, mu, nwkjac, njcpvt, rwork)
```

3 Description

nag_ode_ivp_stiff_bandjac_setup (d02nt) defines the linear algebra to be used as banded matrix linear algebra, permits you to specify the method for calculating the Jacobian and checks the validity of certain input values.

4 References

See the D02M–N Sub-chapter Introduction.

5 Parameters

5.1 Compulsory Input Parameters

1: **neq** – INTEGER

The number of differential equations.

Constraint: $1 \leq$ **neq** $\leq$ **neqmax**.

2: **neqmax** – INTEGER

A bound on the maximum number of differential equations to be solved during the integration.

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

3: **jceval** – CHARACTER(1)

Specifies the technique to be used to compute the Jacobian as follows:

jceval = 'N'
: The Jacobian is to be evaluated numerically by the integrator. If this option is used, then the actual argument corresponding to **jac** in the call to nag_ode_ivp_stiff_exp_bandjac (d02nc) or nag_ode_ivp_stiff_imp_bandjac (d02nh) must be either nag_ode_ivp_stiff_exp_bandjac_dummy_jac (d02ncz) or nag_ode_ivp_stiff_imp_bandjac_dummy_jac (d02nhz) respectively.

jceval = 'A'
: You must supply a (sub)program to evaluate the Jacobian on a call to the integrator.

jceval = 'D'
: The default choice is to be made. In this case `D' is interpreted as `N'.

Only the first character of the actual argument **jceval** is passed to nag_ode_ivp_stiff_bandjac_setup (d02nt); hence it is permissible for the actual argument to be more descriptive, e.g., 'Numerical', 'Analytical' or 'Default', on a call to nag_ode_ivp_stiff_bandjac_setup (d02nt).

Constraint: **jceval** = 'N', 'A' or 'D'.

4: **ml** – INTEGER

m_L, the number of subdiagonals in the band.

Constraint: $0 \leq \mathbf{ml} \leq \mathbf{neq} - 1$.

5: **mu** – INTEGER

m_U, the number of superdiagonals in the band.

Constraint: $0 \leq \mathbf{mu} \leq \mathbf{neq} - 1$.

6: **nwkjac** – INTEGER

The size of the workspace array **wkjac**, which you are supplying to the integrator, as declared in the (sub)program from which nag_ode_ivp_stiff_bandjac_setup (d02nt) is called.

Constraint: $\mathbf{nwkjac} \geq (2 \times \mathbf{ml} + \mathbf{mu} + 1) \times \mathbf{neqmax}$.

7: **njcpvt** – INTEGER

The size of the workspace array **jacpvt**, which you are supplying to the integrator, as declared in the (sub)program from which nag_ode_ivp_stiff_bandjac_setup (d02nt) is called.

Constraint: $\mathbf{njcpvt} \geq \mathbf{neqmax}$.

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 setup function to the integrator and therefore the contents of this array must not be changed before calling the integrator.

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 $= 1$

On entry, **jceval** $\neq$ 'N', 'A' or 'D',
or **neq** < 1,
or **ml** < 0 or **ml** $>$ **neq** $- 1$,
or **mu** < 0 or **mu** $>$ **neq** $- 1$,
or **neq** $>$ **neqmax**,
or **njcpvt** $<$ **neqmax**,
or **nwkjac** $< (2 \times$ **ml** $+$ **mu** $+ 1) \times$ **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

nag_ode_ivp_stiff_bandjac_setup (d02nt) must be called as a setup function before a call to either nag_ode_ivp_stiff_exp_bandjac (d02nc) or nag_ode_ivp_stiff_imp_bandjac (d02nh) and may be called as the linear algebra setup function before a call to either nag_ode_ivp_stiff_exp_revcom (d02nm) or nag_ode_ivp_stiff_imp_revcom (d02nn).

9 Example

See Section 10 in nag_ode_ivp_stiff_exp_bandjac (d02nc) and nag_ode_ivp_stiff_imp_bandjac (d02nh).

9.1 Program Text

```
function d02nt_example

fprintf('d02nt 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

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

9.2 Program Results

```
     d02nt 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
```