NAG Toolbox

nag_ode_dae_dassl_linalg (d02np)

1 Purpose

nag_ode_dae_dassl_linalg (d02np) is a setup function which you must call prior to nag_ode_dae_dassl_gen (d02ne) and after a call to nag_ode_dae_dassl_setup (d02mw), if the Jacobian is to be considered as having a banded structure.

2 Syntax

```
[icom, ifail] = nag_ode_dae_dassl_linalg(neq, ml, mu, icom, 'licom', licom)
[icom, ifail] = d02np(neq, ml, mu, icom, 'licom', licom)
```

3 Description

A call to nag_ode_dae_dassl_linalg (d02np) specifies that the Jacobian to be used is banded in structure. If nag_ode_dae_dassl_linalg (d02np) is not called before a call to nag_ode_dae_dassl_gen (d02ne) then the Jacobian is assumed to be full.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **neq** – INTEGER

The number of differential-algebraic equations to be solved.

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

2: **ml** – INTEGER

m_L, the number of subdiagonals in the band.

Constraint: $0 \leq$ **ml** $\leq$ **neq** $- 1$.

3: **mu** – INTEGER

m_U, the number of superdiagonals in the band.

Constraint: $0 \leq$ **mu** $\leq$ **neq** $- 1$.

4: **icom**(**licom**) – INTEGER array

icom is used to communicate details of the integration from nag_ode_dae_dassl_setup (d02mw) and details of the banded structure of the Jacobian to nag_ode_dae_dassl_gen (d02ne).

5.2 Optional Input Parameters

1: **licom** – INTEGER

Default: the dimension of the array **icom**.

The dimension of the array **icom**.

Constraint: **licom** $\geq 50 +$ **neq**.

5.3 Output Parameters

1: **icom**(**licom**) – INTEGER 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 $= 1$

Constraint: **neq** ≥ 1.

ifail $= 2$

Constraint: **ml** $\leq$ **neq** $- 1$.

Constraint: **ml** ≥ 0.

ifail $= 3$

Constraint: **mu** $\leq$ **neq** $- 1$.

Constraint: **mu** ≥ 0.

ifail $= 4$

Either the initialization function has not been called prior to the first call of this function or the communication array has become corrupted.

ifail $= 5$

On entry, **licom** is too small.

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_dae_dassl_gen (d02ne) and nag_ode_dae_dassl_setup (d02mw).

9.1 Program Text

```
     function d02np_example

fprintf('d02np example results\n\n');

% Initialize the problem, specifying that the Jacobian is to be
% evaluated analytically using the provided routine jac.

neq    = nag_int(3);
maxord = nag_int(5);
jceval = 'Analytic';
hmax   = 0;
h0     = 0;
itol   = nag_int(1);
lcom   = nag_int(200);
[icom, com, ifail] = d02mw(neq, maxord, jceval, hmax, h0, itol, lcom);

% Specify that the Jacobian is banded
mu = nag_int(2);
ml = nag_int(1);
[icom, ifail] = d02np(neq, ml, mu, icom);

% Set initial values
rtol = [1e-3; 1e-3; 1e-3];
atol = [1e-6; 1e-6; 1e-6];
y    = [1; 0; 0];
ydot = zeros(neq,1);
t    = 0;
tout = 0.02;

% Use the user parameter to pass the band dimensions through to jac.
% An alternative would be to hard code values for ml and mu in jac.
user = {ml, mu};

fprintf('\n    t            y(1)        y(2)        y(3)   \n');
fprintf(' %8.4f   %12.6f %12.6f %12.6f\n', t, y);

itask = nag_int(0);
% Obtain the solution at 5 equally spaced values of T.
for j = 1:5
  if ifail == 0
    [t, y, ydot, rtol, atol, itask, icom, com, user, ifail] = ...
      d02ne(t, tout, ...
            y, ydot, rtol, atol, itask, @res, @jac, icom, com, 'user', user);
    fprintf(' %8.4f   %12.6f %12.6f %12.6f\n', t, y);
    tout = tout + 0.02;
    icom = d02mc(icom);
  end
end

fprintf('\nThe integrator completed task, ITASK = %d\n', itask);

function [pd, user] = jac(neq, t, y, ydot, pd, cj, user)
  ml = user{1};
  mu = user{2};

  stride = 2*ml+mu+1;
  % Main diagonal pdfull(i,i), i=1,neq
  md = mu + ml + 1;
  pd(md) = -0.04 - cj;
  pd(md+stride) = -1.0e4*y(3) - 6.0e7*y(2) - cj;
  pd(md+2*stride) = -cj;
  % 1 sub-diagonal pdfull(i+1:i), i=1,neq-1
  ms = md + 1;
  pd(ms) = 0.04;
  pd(ms+stride) = 6.0e7*y(2);
  % First super-diagonal pdfull(i-1,i), i=2, neq
  ms = md - 1;
  pd(ms+stride) = 1.0e4*y(3);
  pd(ms+2*stride) = -1.0e4*y(2);
```

```
  % Second super-diagonal pdfull(i-2,i), i=3, neq
  ms = md - 2;
  pd(ms+2*stride) = 1.0e4*y(2);

function [r, ires, user] = res(neq, t, y, ydot, ires, user)
  r = zeros(neq, 1);
  r(1) = -0.04*y(1) + 1.0e4*y(2)*y(3)                   - ydot(1);
  r(2) =  0.04*y(1) - 1.0e4*y(2)*y(3) - 3.0e7*y(2)*y(2) - ydot(2);
  r(3) =                                3.0e7*y(2)*y(2) - ydot(3);
```

9.2 Program Results

```
     d02np example results

    t            y(1)        y(2)        y(3)
   0.0000       1.000000    0.000000     0.000000
   0.0200       0.999204    0.000036     0.000760
   0.0400       0.998415    0.000036     0.001549
   0.0600       0.997631    0.000036     0.002333
   0.0800       0.996852    0.000036     0.003112
   0.1000       0.996080    0.000036     0.003884

The integrator completed task, ITASK = 3
```