NAG Toolbox

nag_ode_ivp_stiff_sparjac_enq (d02nr)

1 Purpose

nag_ode_ivp_stiff_sparjac_enq (d02nr) is an enquiry function for communicating with nag_ode_ivp_stiff_exp_revcom (d02nm) or nag_ode_ivp_stiff_imp_revcom (d02nn) when supplying columns of a sparse Jacobian matrix.

2 Syntax

```
[j, iplace] = nag_ode_ivp_stiff_sparjac_enq(inform)
[j, iplace] = d02nr(inform)
```

3 Description

nag_ode_ivp_stiff_sparjac_enq (d02nr) is required when nag_ode_ivp_stiff_exp_revcom (d02nm) or nag_ode_ivp_stiff_imp_revcom (d02nn) is being used with sparse matrix linear algebra. After an exit from nag_ode_ivp_stiff_exp_revcom (d02nm) or nag_ode_ivp_stiff_imp_revcom (d02nn) with **irevcm** $= 8$, nag_ode_ivp_stiff_sparjac_enq (d02nr) must be called to determine which column of the Jacobian is required and where it is to be placed in the array **rwork** (a argument of nag_ode_ivp_stiff_exp_revcom (d02nm) or nag_ode_ivp_stiff_imp_revcom (d02nn)).

4 References

See the D02M–N Sub-chapter Introduction.

5 Parameters

5.1 Compulsory Input Parameters

1: **inform**(**23**) – INTEGER array

Contains information supplied by the integrator.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **j** – INTEGER

The index j of the column of the Jacobian which is required.

2: **iplace** – INTEGER

Indicates which locations in the array **rwork** to fill with the jth column.

If **iplace** $= 1$, the (i,j)th element of the Jacobian must be placed in **rwork**$(50 + 2 \times$ **ldysav** $+ i)$, otherwise the (i,j)th element must be placed in **rwork**$(50 +$ **ldysav** $+ i)$.

If **jceval** $=$ 'F', in the previous call to nag_ode_ivp_stiff_sparjac_setup (d02nu), then **iplace** $= 2$ always, hence the jth column of the Jacobian must be placed in **rwork**$(50 +$ **ldysav** $+ i)$, for $i = 1, 2, \ldots,$ **neq**.

rwork, **neq** and **ldysav** are arguments of nag_ode_ivp_stiff_exp_revcom (d02nm) and nag_ode_ivp_stiff_imp_revcom (d02nn).

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

See Section 10 in nag_ode_ivp_stiff_imp_revcom (d02nn).

9.1 Program Text

```
function d02nr_example

fprintf('d02nr 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 = 0;
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);

% Sparse Jacobian supplied, setup
ia     = nag_int(0);
ja     = nag_int(0);
njcpvt = nag_int(150);
nwkjac = nag_int(100);
eta    = 1.0e-4;
u      = 0.1;
sens   = 0.0;
lblock = true;

[jacpvt, rwork, ifail] = d02nu(...
    neq, neqmax, 'Analytical', nwkjac, ia, ja, ...
    njcpvt, sens, u, eta, lblock, rwork);

% Integration input variable intialization
t      = 0;
tout   = 10;
y      = [1; 0; 0];
```

```
ydot   = [0; 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);
imon   = nag_int(0);
inln   = nag_int(0);
ires   = nag_int(1);
irevcm = nag_int(0);
lderiv = [false; false];
itask  = nag_int(3);
itrace = nag_int(0);

nfails = 0;

% pointers into rwork locations
lacorb = neqmax + 50;
lsavrb = lacorb + neqmax;
l1 = lsavrb+1; l2 = l1+1; l3 = l2+1;
m1 = lacorb+1; m2 = m1+1; m3 = m2+1;

fprintf('  Analytic Jacobian\n\n');
fprintf('     x           y_1          y_2           y_3\n');
fprintf(' %8.3f      %5.1f        %5.1f         %5.1f\n', t, y);
first_time = true;

% Main reverse communication loop controlled by irevcm

while (irevcm ~= 0 || first_time)
  first_time = false;

  [t, tout, y, ydot, rwork, inform, ysave, wkjac, ...
   jacpvt, imon, inln, ires, irevcm, lderiv, ifail] = ...
     d02nn(t, tout, y, ydot, rwork, rtol, atol, itol, inform, ...
     ysave, wkjac, jacpvt, imon, inln, ires, irevcm, lderiv, itask, itrace);

  if irevcm == 1 || irevcm == 3 || irevcm == 6 || irevcm == 11
    % Equivalent to resid evaluation in forward communication routines
    % ydot stored in ydot, resid returned in rwork(l1)
    rwork(l1) = -ydot(1) - ydot(2) - ydot(3);
    rwork(l2) = -ydot(2);
    rwork(l3) = -ydot(3);
    if (ires == 1)
      rwork(l1) =                                                  rwork(l1);
      rwork(l2) = 0.04*y(1) - 1e4*y(2)*y(3) - 3e7*y(2)*y(2) + rwork(l2);
      rwork(l3) =                             3e7*y(2)*y(2) + rwork(l3);
    end
  elseif (irevcm == 2)
    % Equivalent to resid evaluation in forward communication routines
    % ydot stored in rwork(l1:), resid returned in rwork(l1)
    rwork(l1) = -rwork(l1) - rwork(l2) - rwork(l3);
    rwork(l2) = -rwork(l2);
    rwork(l3) = -rwork(l3);
  elseif (irevcm == 4 || irevcm == 7)
    % Equivalent to resid evaluation in forward communication routines
    % ydot stored in ydot, resid returned in rwork(m1)
    rwork(m1) = -ydot(1) - ydot(2) - ydot(3);
    rwork(m2) = -ydot(2);
    rwork(m3) = -ydot(3);
    if (ires == 1)
      rwork(m1) =                                                  rwork(m1);
      rwork(m2) = 0.04*y(1) - 1e4*y(2)*y(3) - 3e7*y(2)*y(2) + rwork(m2);
      rwork(m3) =                             3e7*y(2)*y(2) + rwork(m3);
    end
  elseif (irevcm == 5)
    % Equivalent to resid evaluation in forward communication routines
    % ydot stored in rwork(l1:), resid returned in ydot
    ydot(1) = -rwork(l1) - rwork(l2) - rwork(l1);
    ydot(2) = 0.04*y(1) - 1e4*y(2)*y(3) - 3e7*y(2)*y(2) - rwork(l2);
```

```
    ydot(3) =                              3e7*y(2)*y(2) - rwork(l3);
  elseif (irevcm == 8)
    % Equivalent to jac evaluation in forward communication routines
    [j, iplace] = d02nr(inform);
    hxd = rwork(16)*rwork(20);
    if (iplace < 2)
      if (j < 2)
        rwork(l1) = 1 - hxd*(0);
        rwork(l2) = 0 - hxd*(0.04);
        rwork(l3) = 0 - hxd*(0);
      elseif (j == 2)
        rwork(l1) = 1 - hxd*(0);
        rwork(l2) = 1 - hxd*(-1e4*y(3)-6e7*y(2));
        rwork(l3) = 0 - hxd*(6e7*y(2));
      elseif (j > 2)
        rwork(l1) = 1 - hxd*(0);
        rwork(l2) = 0 - hxd*(-1e4*y(2));
        rwork(l3) = 1 - hxd*(0);
      end
    else
      if (j < 2)
        rwork(m1) = 1 - hxd*(0);
        rwork(m2) = 0 - hxd*(0.04);
        rwork(m3) = 0 - hxd*(0);
      elseif (j == 2)
        rwork(m1) = 1 - hxd*(0);
        rwork(m2) = 1 - hxd*(-1e4*y(3)-6e7*y(2));
        rwork(m3) = 0 - hxd*(6e7*y(2));
      elseif (j > 2)
        rwork(m1) = 1 - hxd*(0);
        rwork(m2) = 0 - hxd*(-1e4*y(2));
        rwork(m3) = 1 - hxd*(0);
      end
    end
  elseif (irevcm == 10)
    % Step failure
    nfails = nfails + 1;
  end
end

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

[y, ifail] = d02mz(tout, neq, neq, ysave, rwork);

% Print solution and diagnostics
fprintf(' %8.3f      %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);
fprintf('   number of failed steps                      = %5d\n\n',nfails);

icall = nag_int(0);
[liwreq, liwusd, lrwreq, lrwusd, nlu, nz, ngp, isplit, igrow, nblock] = ...
    d02nx(icall, true, inform);

fprintf(' sparse jacobian statistics:\n');
fprintf('   njcpvt - required = %3d, used = %3d\n',liwreq,liwusd);
fprintf('   nwkjac - required = %3d, used = %3d\n',lrwreq,lrwusd);
fprintf('   No. of LU-decomps = %3d, No. of nonzeros = %3d\n',nlu,nz);
fprintf(['   No. of function calls to form Jacobian = %3d, try ', ...
    'isplit = %3d\n'], ngp, isplit);
fprintf([' Growth estimate = %d, No. of blocks on diagonal %3d\n\n'], ...
    igrow, nblock);
```

9.2 Program Results

```
     d02nr example results

  Analytic Jacobian

     x           y_1          y_2          y_3
    0.000        1.0          0.0          0.0
 Warning: Equation(=i1) and possibly other equations are
 implicit and in calculating the initial values the
 equations will be treated as implicit.
 In above message i1 =         1
   10.767        0.8          0.0          0.2

Diagnostic information
 integration status:
   last and next step sizes =  0.90181,  0.90181
   integration stopped at x = 10.76669
 algorithm statistics:
   number of time-steps and Newton iterations  =    55    80
   number of residual and jacobian evaluations =    89    17
   order of method last used and next to use   =     4     4
   component with largest error                =     3
   number of failed steps                      =     4

 sparse jacobian statistics:
   njcpvt - required =  99, used = 150
   nwkjac - required =  31, used =  75
   No. of LU-decomps =  17, No. of nonzeros =   8
   No. of function calls to form Jacobian =   0, try isplit =  73
 Growth estimate = 1531, No. of blocks on diagonal   1
```