NAG Toolbox

nag_ode_ivp_stiff_sparjac_diag (d02nx)

1 Purpose

nag_ode_ivp_stiff_sparjac_diag (d02nx) is an optional output function which you may call, on exit from an integrator in Sub-chapter D02M–N, if sparse matrix linear algebra has been selected.

2 Syntax

```
[liwreq, liwusd, lrwreq, lrwusd, nlu, nnz, ngp, isplit, igrow, nblock] = nag_ode_ivp_stiff_sparjac_diag(icall, lblock, inform)
[liwreq, liwusd, lrwreq, lrwusd, nlu, nnz, ngp, isplit, igrow, nblock] = d02nx(icall, lblock, inform)
```

3 Description

nag_ode_ivp_stiff_sparjac_diag (d02nx) permits you to examine the various outputs from the sparse linear algebra functions called by the integrator.

4 References

See the D02M–N Sub-chapter Introduction.

5 Parameters

5.1 Compulsory Input Parameters

1: **icall** – INTEGER

Indicates whether or not all output arguments have been set during the call to the integrator. If so, that is, if the integrator returned with **ifail** $= 0$ or 12, then **icall** must be set to 0. Otherwise **icall** must be set to 1, indicating that integration did not take place due to lack of space in arrays **wkjac** and **jacpvt**, and only **liwreq**, **liwusd**, **lrwreq**, **lrwusd** have been set.

2: **lblock** – LOGICAL

The value used for the argument **lblock** when calling nag_ode_ivp_stiff_sparjac_setup (d02nu).

3: **inform**(**23**) – INTEGER array

Contains information supplied by the integrator.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **liwreq** – INTEGER

The length of the integer workspace **jacpvt** reserved for the sparse matrix functions.

2: **liwusd** – INTEGER

The length of the integer workspace **jacpvt** actually used by the sparse matrix functions.

3: **lrwreq** – INTEGER

The length of the double workspace **wkjac** reserved for the sparse matrix functions.

4: **lrwusd** – INTEGER

The length of the double workspace **wkjac** actually used by the sparse matrix functions.

5: **nlu** – INTEGER

The number of LU decompositions done during the integration.

6: **nz** – INTEGER

The number of nonzeros in the Jacobian.

7: **ngp** – INTEGER

The number of **fcn** or **resid** calls needed to form the Jacobian.

8: **isplit** – INTEGER

An appropriate value for the argument **isplit** when calling nag_ode_ivp_stiff_sparjac_setup (d02nu) for subsequent runs of similar problems.

9: **igrow** – INTEGER

An estimate of the growth of the elements encountered during the last LU decomposition performed. If the actual estimate exceeds the largest possible integer value for the machine being used (see nag_machine_integer_max (x02bb)) **igrow** is set to the value returned by nag_machine_integer_max (x02bb).

10: **nblock** – INTEGER

If **lblock** $= true$, **nblock** contains the number of diagonal blocks in the Jacobian matrix permuted to block lower triangular form. If **nblock** $= 1$ then on subsequent runs of a similar problem **lblock** should be set to *false* in the call to nag_ode_ivp_stiff_sparjac_setup (d02nu).

If **lblock** $= false$, **nblock** $= 1$.

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Further Comments

The output from nag_ode_ivp_stiff_sparjac_diag (d02nx), in particular the values of **liwreq**, **liwusd**, **lrwreq**, **lrwusd**, **isplit** and **igrow**, should be used to determine appropriate values for the arguments of the setup function nag_ode_ivp_stiff_sparjac_setup (d02nu) on further calls to the integrator for the same or similar problems.

9 Example

See Section 10 in nag_ode_ivp_stiff_exp_sparjac (d02nd), nag_ode_ivp_stiff_imp_sparjac (d02nj) and nag_ode_ivp_stiff_imp_revcom (d02nn).

9.1 Program Text

```
     function d02nx_example

fprintf('d02nx 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(l);
    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

```
     d02nx 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
```