NAG Toolbox

nag_ode_ivp_stiff_nat_interp (d02xj)

1 Purpose

nag_ode_ivp_stiff_nat_interp (d02xj) interpolates components of the solution of a system of first-order ordinary differential equations from information provided by the integrators in Sub-chapter D02M–N.

2 Syntax

```
[sol, ifail] = nag_ode_ivp_stiff_nat_interp(xsol, m, ysav, neq, x, nqu, hu, h, 'sdysav', sdysav)
[sol, ifail] = d02xj(xsol, m, ysav, neq, x, nqu, hu, h, 'sdysav', sdysav)
```

3 Description

nag_ode_ivp_stiff_nat_interp (d02xj) evaluates the first m components of the solution of a system of ordinary differential equations at any point using natural polynomial interpolation based on information generated by the integrator. This information must be passed unchanged to nag_ode_ivp_stiff_nat_interp (d02xj). nag_ode_ivp_stiff_nat_interp (d02xj) should not normally be used to extrapolate outside the range of values obtained from the above functions.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **xsol** – REAL (KIND=nag_wp)

The point at which the first m components of the solution are to be evaluated. **xsol** should not be an extrapolation point, that is **xsol** should satisfy $(\mathbf{xsol} - \mathbf{x}) \times \mathbf{hu} \leq 0.0$. Extrapolation is permitted but not recommended.

2: **m** – INTEGER

m, the number of components of the solution whose values at **xsol** are required. The first **m** components are evaluated.

Constraint: $1 \leq \mathbf{m} \leq \mathbf{neq}$.

3: **ysav**$(ldysav, \mathbf{sdysav})$ – REAL (KIND=nag_wp) array

$ldysav$, the first dimension of the array, must satisfy the constraint $ldysav \geq 1$.

The values provided in the argument **ysav** on return from the integrator.

4: **neq** – INTEGER

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

Constraint: $1 \leq \mathbf{neq} \leq ldysav$.

5: **x** – REAL (KIND=nag_wp)

The latest value at which the solution has been computed, as provided in the argument **tcur** on return from the optional output nag_ode_ivp_stiff_integ_diag (d02ny).

6: **nqu** – INTEGER

The order of the method used up to the latest value at which the solution has been computed, as provided in the argument **nqu** on return from the optional output nag_ode_ivp_stiff_integ_diag (d02ny).

Constraint: **nqu** ≥ 1.

7: **hu** – REAL (KIND=nag_wp)

The last successful step used, that is the step used in the integration to get to **x**, as provided in the argument **hu** on return from the optional output nag_ode_ivp_stiff_integ_diag (d02ny).

8: **h** – REAL (KIND=nag_wp)

The next step size to be attempted in the integration, as provided in the argument **h** on return from the optional output nag_ode_ivp_stiff_integ_diag (d02ny).

5.2 Optional Input Parameters

1: **sdysav** – INTEGER

Default: the second dimension of the array **ysav**.

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

Constraint: **sdysav** $\geq$ **nqu** $+ 1$.

5.3 Output Parameters

1: **sol**(**m**) – REAL (KIND=nag_wp) array

The calculated value of the ith component of the solution at **xsol**, for $i = 1, 2, \ldots, m$.

2: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

If nag_ode_ivp_stiff_nat_interp (d02xj) is to be used for extrapolation, **ifail** must be set to 1 before entry. It is then essential to test the value of **ifail** on exit for **ifail** $= 1$ or 2.

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **m** < 1,
or **neq** < 1,
or $ldysav < 1$,
or **neq** $> ldysav$,
or **m** $>$ **neq**,
or **nqu** < 1,
or **sdysav** $<$ **nqu** $+ 1$.

ifail $= 2$

On entry, **hu** $= 0.0$ or **h** $= 0.0$. This error can only occur if **h** and **hu** have been changed by you or possibly if the integrator has failed before calling nag_ode_ivp_stiff_nat_interp (d02xj).

ifail $= 3$ (*warning*)

nag_ode_ivp_stiff_nat_interp (d02xj) has been called for extrapolation. Before returning with this error exit, the value of the solution at **xsol** is calculated and placed in **sol**.

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

The solution values returned will be of a similar accuracy to those computed by the integrator.

8 Further Comments

nag_ode_ivp_stiff_nat_interp (d02xj) is that employed for prediction purposes internally by the integrator. It is supplied for purposes of consistency only. You are recommended to employ the C^1 interpolant provided by nag_ode_ivp_stiff_c1_interp (d02xk) wherever possible.

9 Example

See Section 10 in nag_ode_ivp_stiff_imp_fulljac (d02ng).

9.1 Program Text

```
function d02xj_example

fprintf('d02xj example results\n\n');

% Initialize setup variables and arrays.
neq    = nag_int(3);
neqmax = neq;
maxord = nag_int(5);
sdysav = maxord+1;
petzld = false;
tcrit  = 0;
hmin   = 1.0e-10;
hmax   = 10;
h0     = 0;
maxstp = nag_int(200);
mxhnil = nag_int(5);

const = zeros(6, 1);
rwork = zeros(50+4*neq, 1);

% d02nv is a setup routine to be called prior to d02ng.
[const, rwork, ifail] = ...
d02nv( ...
       neqmax, sdysav, maxord, 'Functional', petzld, ...
       const, tcrit, hmin, hmax, h0, maxstp, mxhnil, 'Average-L2', rwork);

nwkjac = nag_int(neq*(neq+1));
% Numerical Jacobian.
[rwork, ifail] = d02ns( ...
                        neq, neqmax, 'Numerical', nwkjac, rwork);

% Initialize integration variables and arrays
```

```
sol   = zeros(neq, 1);
wkjac = zeros(nwkjac, 1);
ysave = zeros(neq, sdysav);
ydot  = zeros(neq, 1);
inform(1:23) = nag_int(0);
algequ = false(neq, 1);

% Integrate to tout by overshooting tout in one step mode (itask=2);
% use BDF with functional iteration; use vector tolerances (itol=4);
% dummy d02nbz and d02nby are used for Jacobian and monitor functions.
t    = 0;
tout = 0.1;
itask  = nag_int(2);
itrace = nag_int(0);
y      = [1; 0; 0];
lderiv = [false; false];
itol = nag_int(4);
rtol = [0.0001; 0.001; 0.0001];
atol = [1e-07; 1e-08; 1e-07];

% Output header and initial results.
fprintf('    x             y_1            y_2            y_3\n');
fprintf('%8.3f       %8.4f       %8.6f       %8.4f\n',t,y);

% Prepare to store results for plotting.
ncall = 1;
tkeep = t;
ykeep = y;
tstep = 0.02;

% xout is intermediate output points for printing solution.
iout = 1;
xout = iout*tstep;

while t < tout
  % Calculate solution at this point.
  [t, tout, y, ydot, rwork, inform, ysave, wkjac, lderiv, ifail] = ...
  d02ng( ...
         t, tout, y, ydot, rwork, rtol, atol, itol, ...
         inform, @resid, ysave, 'd02ngz', wkjac, 'd02nby', ...
         lderiv, itask, itrace);
  if (t<tstep)
    ncall = ncall+1;
    ykeep(:,ncall) = y;
    tkeep(ncall) = t;
  elseif (xout <= t)
    % Passed over intermediate output point (xout)
    % Interpolate the solution to get its value at xout.
    [hu, h, tcur, tolsf, nst, nre, nje, nqu, nq, niter, imxer, algequ, ...
     ifail] = d02ny(...
                    neq, neqmax, rwork, inform);
    [sol, ifail] = d02xj( ...
                          xout, neq, ysave, neq, tcur, ...
                          nqu, hu, h, 'sdysav', sdysav);

    % Accumulate results for plotting.
    ncall = ncall + 1;
    ykeep(:,ncall) = sol;
    tkeep(ncall) = xout;
    % output intermediate results
    fprintf('%8.3f       %8.4f       %8.6f       %8.4f\n',xout,sol);

    % Bump the counter for the next output point.
    iout = iout + 1;
    xout = iout*tstep;
  end
end

function [r, ires] = resid(neq, t, y, ydot, ires)
% Evaluate the residual.
```

```
r(1:3) = -ydot(1:3);
if ires == 1
  r(1) = -0.04*y(1) + 1.0e4*y(2)*y(3) + r(1);
  r(2) =  0.04*y(1) - 1.0e4*y(2)*y(3) - 3.0e7*y(2)*y(2) + r(2);
  r(3) =  3.0e7*y(2)*y(2) + r(3);
end
```

9.2 Program Results

```
     d02xj example results

    x             y_1           y_2           y_3
   0.000         1.0000       0.000000         0.0000
   0.020         0.9992       0.000036         0.0008
   0.040         0.9984       0.000036         0.0016
   0.060         0.9976       0.000036         0.0023
   0.080         0.9969       0.000036         0.0031
   0.100         0.9961       0.000036         0.0039
```