NAG Toolbox

nag_ode_ivp_stiff_interp (d02mz)

1 Purpose

nag_ode_ivp_stiff_interp (d02mz) interpolates components of the solution of a system of first-order differential equations from information provided by those integrators in Sub-chapter D02M–N using methods set up by calls to nag_ode_ivp_stiff_dassl (d02mv), nag_ode_ivp_stiff_bdf (d02nv) or nag_ode_ivp_stiff_blend (d02nw).

2 Syntax

```
[sol, ifail] = nag_ode_ivp_stiff_interp(tsol, m, neq, ysav, rwork, 'sdysav', sdysav)
[sol, ifail] = d02mz(tsol, m, neq, ysav, rwork, 'sdysav', sdysav)
```

3 Description

nag_ode_ivp_stiff_interp (d02mz) 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_interp (d02mz). nag_ode_ivp_stiff_interp (d02mz) should not normally be used to extrapolate outside the range of values obtained from the above function.

4 References

See the D02M–N Sub-chapter Introduction.

5 Parameters

5.1 Compulsory Input Parameters

1: **tsol** – REAL (KIND=nag_wp)

The point at which the first **m** components of the solution are to be evaluated. **tsol** should not normally be an extrapolation point. Extrapolation is permitted but not recommended.

2: **m** – INTEGER

The number of components of the solution whose values are required.

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

3: **neq** – INTEGER

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

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

4: **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 array **ysav** on return from the integrator.

5: **rwork**$(\mathbf{50} + \mathbf{4} \times \mathbf{neq})$ – REAL (KIND=nag_wp) array

The values provided in the array **rwork** on return from the integrator.

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.

5.3 Output Parameters

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

The calculated value of the solution at **tsol**.

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, **m** < 1,
or $ldysav < 1$,
or **neq** < 1,
or **m** $>$ **neq**,
or **neq** $> ldysav$.

ifail $= 2$

On entry, when accessing an element of the array **rwork** an unexpected quantity was found. You have not passed the correct array to nag_ode_ivp_stiff_interp (d02mz) or has overwritten elements of this array.

ifail $= 3$ (*warning*)

On entry, nag_ode_ivp_stiff_interp (d02mz) has been called for extrapolation. Before returning with this error exit, the value of the solution at **tsol** 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

None.

9 Example

This example solves the well-known stiff Robertson problem written in implicit form

$$\begin{array}{rcrcrcrcl} r_1 & = & -0.04a & + & 1.0\text{E}4bc & & & - & a' \\ r_2 & = & 0.04a & - & 1.0\text{E}4bc & - & 3.0\text{E}7b^2 & - & b' \\ r_3 & = & & & & & 3.0\text{E}7b^2 & - & c' \end{array}$$

with initial conditions $a = 1.0$ and $b = c = 0.0$ over the range $[0, 0.1]$ with vector error control (**itol** = 4), the BDF method (setup function nag_ode_ivp_stiff_bdf (d02nv)) and functional iteration. The Jacobian is calculated numerically if the functional iteration encounters difficulty and the integration is in one-step mode (**itask** = 2), with natural interpolation to calculate the solution at intervals of 0.02 using nag_ode_ivp_stiff_interp (d02mz) externally. nag_ode_ivp_stiff_exp_fulljac_-dummy_monit (d02nby) is used for **monitr**.

9.1 Program Text

```
     function d02mz_example

fprintf('d02mz example results\n\n');

% Integrate to using B.D.F formulae with a functional iteration method
neq    = nag_int(3);
maxord = nag_int(5);
sdysav = nag_int(maxord+1);
petzld = false;
const  = zeros(6,1);
tcrit  = 0;
hmin   = 1e-10;
hmax =   10;
h0     = 0;
maxstp = nag_int(200);
mxhnil = nag_int(5);
rwork  = zeros(16+20*neq,1);
[const, rwork, ifail] = d02nv(...
                              neq, sdysav, maxord, 'functional-iteration', ...
                              petzld, const, tcrit, hmin, hmax, h0, maxstp, ...
                              mxhnil, 'average-l2', rwork);

% Use numerical Jacobian if functional iteration encounters any difficulty.
nwkjac = nag_int(neq * (neq + 1));
[rwork, ifail] = d02ns(neq, neq, 'numerical', nwkjac, rwork);

algequ = zeros(1, neq);
lderiv = zeros(1, 2);

% Variable and array initializations for integrator
% Initial conditions
t      = 0;
tout   = 0.1;
y      = [1; 0; 0];
ydot   = zeros(1, neq);
% vector tolerances
itol   = nag_int(4);
rtol   = [1.0e-4; 1.0e-3; 1.0e-4];
atol   = [1.0e-7; 1.0e-8; 1.0e-7];
inform = zeros(23, 1, nag_int_name);
ysav   = zeros(neq, sdysav);
wkjac  = zeros(1, nwkjac);
lderiv = [false; false];
itask  = nag_int(2);
itrace = nag_int(0);

% Intialize for solution output
xout = 0.02e0;
fprintf('\n     x           y(1)          y(2)          y(3)\n');
fprintf('%8.3f %13.5f %13.5f %13.5f\n', t, y);

while (t < tout)
```

```
   [t, tout, y, ydot, rwork, inform, ysav, wkjac, lderiv, ifail] = ...
       d02ng(...
             t, tout, y, ydot, rwork, rtol, atol, itol, inform, ...
             @resid, ysav, 'd02ngz', wkjac, 'd02nby', lderiv, itask, ...
             itrace);
   while (xout <= t)
      [sol, ifail] = d02mz(xout, neq, neq, ysav, rwork);
      fprintf('%8.3f %13.5f %13.5f %13.5f\n', xout, sol);
      xout = xout + 0.02e0;
   end
end

function [r, ires] = resid(neq, t, y, ydot, ires)
  r = -ydot;
  if ires == nag_int(1)
    r(1) = -0.04e0*y(1) + 1.0e4*y(2)*y(3)                   + r(1);
    r(2) =  0.04e0*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

```
     d02mz example results

     x           y(1)          y(2)          y(3)
   0.000       1.00000       0.00000       0.00000
   0.020       0.99920       0.00004       0.00076
   0.040       0.99841       0.00004       0.00155
   0.060       0.99763       0.00004       0.00234
   0.080       0.99685       0.00004       0.00311
   0.100       0.99608       0.00004       0.00389
```