NAG Toolbox

nag_ode_ivp_2nd_rkn_interp (d02lz)

1 Purpose

nag_ode_ivp_2nd_rkn_interp (d02lz) interpolates components of the solution of a non-stiff system of second-order differential equations from information provided by the integrator nag_ode_ivp_2nd_rkn (d02la), when the low-order method has been used.

2 Syntax

```
[ywant, ypwant, ifail] = nag_ode_ivp_2nd_rkn_interp(t, y, yp, nwant, twant, rwork, 'neq', neq)
[ywant, ypwant, ifail] = d02lz(t, y, yp, nwant, twant, rwork, 'neq', neq)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 22: *lrwork* was removed from the interface.

3 Description

nag_ode_ivp_2nd_rkn_interp (d02lz) evaluates the first **nwant** ($\leq$ **neq**) components of the solution of a non-stiff system of second-order ordinary differential equations at any point using a special Runge–Kutta–Nystrom formula (see Dormand and Prince (1986)) and information generated by nag_ode_ivp_2nd_rkn (d02la) when the low-order method has been used. This information must be presented unchanged to nag_ode_ivp_2nd_rkn_interp (d02lz). nag_ode_ivp_2nd_rkn_interp (d02lz) should not normally be used to extrapolate outside the range of the values from nag_ode_ivp_2nd_rkn (d02la).

4 References

Dormand J R and Prince P J (1986) Runge–Kutta–Nystrom triples *Mathematical Report TP-CS-86-05* Teesside Polytechnic

5 Parameters

5.1 Compulsory Input Parameters

1: **t** – REAL (KIND=nag_wp)

t, the current value at which the solution and its derivative have been computed (as returned in argument **t** on output from nag_ode_ivp_2nd_rkn (d02la)).

2: **y**(**neq**) – REAL (KIND=nag_wp) array

The ith component of the solution at t, for $i = 1, 2, \ldots,$ **neq**, as returned from nag_ode_ivp_2nd_rkn (d02la).

3: **yp**(**neq**) – REAL (KIND=nag_wp) array

The ith component of the derivative at t, for $i = 1, 2, \ldots,$ **neq**, as returned from nag_ode_ivp_2nd_rkn (d02la).

4: **nwant** – INTEGER

The number of components of the solution and derivative whose values at **twant** are required. The first **nwant** components are evaluated.

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

5: **twant** – REAL (KIND=nag_wp)

The point at which components of the solution and derivative are to be evaluated. **twant** should not normally be an extrapolation point, that is **twant** should satisfy

$$told \leq \mathbf{twant} \leq \mathbf{t},$$

or if integration is proceeding in the negative direction

$$told \geq \mathbf{twant} \geq \mathbf{t},$$

where $told$ is the previous integration point which is held in an element of the array **rwork** and is, to within rounding, $\mathbf{t} - \mathbf{hused}$. (**hused** is given by nag_ode_ivp_2nd_rkn_diag (d02ly).) Extrapolation is permitted but not recommended, and $\mathbf{ifail} = 2$ is returned whenever extrapolation is attempted.

6: **rwork**$(lrwork)$ – REAL (KIND=nag_wp) array

This **must** be the same argument **rwork** as supplied to nag_ode_ivp_2nd_rkn (d02la). It is used to pass information from nag_ode_ivp_2nd_rkn (d02la) to nag_ode_ivp_2nd_rkn_interp (d02lz) and therefore the contents of this array **must not** be changed before calling nag_ode_ivp_2nd_rkn_interp (d02lz).

5.2 Optional Input Parameters

1: **neq** – INTEGER

Default: the dimension of the arrays **y**, **yp**. (An error is raised if these dimensions are not equal.)

The number of second-order ordinary differential equations being solved by nag_ode_ivp_2nd_rkn (d02la). It must contain the same value as the argument **neq** in a prior call to nag_ode_ivp_2nd_rkn (d02la).

5.3 Output Parameters

1: **ywant**(**nwant**) – REAL (KIND=nag_wp) array

The calculated value of the ith component of the solution at $t = \mathbf{twant}$, for $i = 1, 2, \ldots, \mathbf{nwant}$.

2: **ypwant**(**nwant**) – REAL (KIND=nag_wp) array

The calculated value of the ith component of the derivative at $t = \mathbf{twant}$, for $i = 1, 2, \ldots, \mathbf{nwant}$.

3: **ifail** – INTEGER

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

If nag_ode_ivp_2nd_rkn_interp (d02lz) is to be used for extrapolation at **twant**, **ifail** should be set to 1 before entry. It is then essential to test the value of **ifail** on exit.

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

Illegal input detected, i.e., one of the following conditions:

– nag_ode_ivp_2nd_rkn (d02la) has not been called;

– one or both of the arguments **neq** and *lrwork* does not match the corresponding argument supplied to the setup function nag_ode_ivp_2nd_rkn_setup (d02lx);

– no integration steps have been taken since the last call to nag_ode_ivp_2nd_rkn_setup (d02lx) with **start** = *true*;

– **nwant** < 1 or **nwant** $>$ **neq**.

This error exit can be caused if elements of **rwork** have been overwritten.

ifail $= 2$ (*warning*)

nag_ode_ivp_2nd_rkn_interp (d02lz) has been called for extrapolation. The values of the solution and its derivative at **twant** have been calculated and placed in **ywant** and **ypwant** before returning with this error number (see Section 7).

ifail $= 3$

nag_ode_ivp_2nd_rkn (d02la) last used the high order method to integrate the system of differential equations. Interpolation is not permitted with this method.

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 error in interpolation is of a similar order to the error arising from the integration using nag_ode_ivp_2nd_rkn (d02la) with the lower order method.

The same order of accuracy can be expected when extrapolating using nag_ode_ivp_2nd_rkn_interp (d02lz). However, the actual error in extrapolation will, in general, be much larger than for interpolation.

8 Further Comments

When interpolation for only a few components is required then it is more efficient to order the components of interest so that they are numbered first.

9 Example

See Section 10 in nag_ode_ivp_2nd_rkn (d02la).

9.1 Program Text

```
function d02lz_example

fprintf('d02lz example results\n\n');

% Variables and arrays for integration setup
neq    = 2;
h      = 0;
tol    = 1e-4;
thres  = zeros(neq, 1);
thresp = zeros(neq, 1);
```

```
maxstp = nag_int(1000);
start  = true;
onestp = true;
high   = false;
rwork  = zeros(16+20*neq,1);

% Integration setup
[startOut, rwork, ifail] = d02lx...
   (h, tol, thres, thresp, maxstp, start, onestp, high, rwork);

% Integration variables
t      = 0;
tend   = 20;
y      = [0.5; 0];
yp     = [0;   sqrt(3)];
ydp    = zeros(neq, 1);
tnext  = 2;
nwant  = nag_int(2);

fprintf('\n  T       Y(1)       Y(2)\n');
fprintf('%4.1f %10.5f %10.5f\n', t, y(1), y(2));

% Integrate by steps and interpolate onto selected values
while (t < tend && ifail == 0)
  [t, y, yp, ydp, rwork, ifail] = d02la(@fcn, t, tend, y, yp, ydp, rwork);
  while (tnext <= t && ifail == 0)
    [ywant, ypwant, ifail] = d02lz(t, y, yp, nwant, tnext, rwork);
    fprintf('%4.1f %10.5f %10.5f\n', tnext, ywant(1:2));
    tnext = tnext + 2;
  end
end

if (ifail == 0)
  [hnext, hused, hstart, nsucc, nfail, natt, thres, thresp, ifail] = ...
    d02ly(nwant, rwork);

  fprintf('\n Number of successful steps = %d\n', nsucc);
  fprintf(' Number of failed steps     = %d\n', nfail);
end

function ydp = fcn(neq, t, y)
% Evaluate second derivatives.
r      = sqrt(y(1)^2+y(2)^2)^3;
ydp(1) = -y(1)/r;
ydp(2) = -y(2)/r;
```

9.2 Program Results

```
     d02lz example results

  T       Y(1)       Y(2)
 0.0    0.50000    0.00000
 2.0   -1.20573    0.61357
 4.0   -1.33476   -0.47685
 6.0    0.35748   -0.44558
 8.0   -1.03762    0.73022
10.0   -1.42617   -0.32658
12.0    0.05515   -0.72032
14.0   -0.82880    0.81788
16.0   -1.48103   -0.16788
18.0   -0.26719   -0.84223
20.0   -0.57803    0.86339

 Number of successful steps = 108
 Number of failed steps     = 16
```