NAG Toolbox

nag_ode_ivp_rkts_range (d02pe)

1 Purpose

nag_ode_ivp_rkts_range (d02pe) solves an initial value problem for a first-order system of ordinary differential equations using Runge–Kutta methods.

2 Syntax

```
[tgot, ygot, ypgot, ymax, user, iwsav, rwsav, ifail] = nag_ode_ivp_rkts_range(f, twant, ygot, ymax, iwsav, rwsav, 'n', n, 'user', user)
[tgot, ygot, ypgot, ymax, user, iwsav, rwsav, ifail] = d02pe(f, twant, ygot, ymax, iwsav, rwsav, 'n', n, 'user', user)
```

3 Description

nag_ode_ivp_rkts_range (d02pe) and its associated functions (nag_ode_ivp_rkts_setup (d02pq), nag_ode_ivp_rkts_diag (d02pt) and nag_ode_ivp_rkts_errass (d02pu)) solve an initial value problem for a first-order system of ordinary differential equations. The functions, based on Runge–Kutta methods and derived from RKSUITE (see Brankin *et al.* (1991)), integrate

$$y' = f(t, y) \quad \text{given} \quad y(t_0) = y_0$$

where y is the vector of n solution components and t is the independent variable.

nag_ode_ivp_rkts_range (d02pe) is designed for the usual task, namely to compute an approximate solution at a sequence of points. You must first call nag_ode_ivp_rkts_setup (d02pq) to specify the problem and how it is to be solved. Thereafter you call nag_ode_ivp_rkts_range (d02pe) repeatedly with successive values of **twant**, the points at which you require the solution, in the range from **tstart** to **tend** (as specified in nag_ode_ivp_rkts_setup (d02pq)). In this manner nag_ode_ivp_rkts_range (d02pe) returns the point at which it has computed a solution **tgot** (usually **twant**), the solution there (**ygot**) and its derivative (**ypgot**). If nag_ode_ivp_rkts_range (d02pe) encounters some difficulty in taking a step toward **twant**, then it returns the point of difficulty (**tgot**) and the solution and derivative computed there (**ygot** and **ypgot**, respectively).

In the call to nag_ode_ivp_rkts_setup (d02pq) you can specify either the first step size for nag_ode_ivp_rkts_range (d02pe) to attempt or that it computes automatically an appropriate value. Thereafter nag_ode_ivp_rkts_range (d02pe) estimates an appropriate step size for its next step. This value and other details of the integration can be obtained after any call to nag_ode_ivp_rkts_range (d02pe) by a call to nag_ode_ivp_rkts_diag (d02pt). The local error is controlled at every step as specified in nag_ode_ivp_rkts_setup (d02pq). If you wish to assess the true error, you must set **method** to a positive value in the call to nag_ode_ivp_rkts_setup (d02pq). This assessment can be obtained after any call to nag_ode_ivp_rkts_range (d02pe) by a call to nag_ode_ivp_rkts_errass (d02pu).

For more complicated tasks, you are referred to functions nag_ode_ivp_rkts_onestep (d02pf), nag_ode_ivp_rkts_reset_tend (d02pr) and nag_ode_ivp_rkts_interp (d02ps), all of which are used by nag_ode_ivp_rkts_range (d02pe).

4 References

Brankin R W, Gladwell I and Shampine L F (1991) RKSUITE: A suite of Runge–Kutta codes for the initial value problems for ODEs *SoftReport 91-S1* Southern Methodist University

5 Parameters

5.1 Compulsory Input Parameters

1: **f** – SUBROUTINE, supplied by the user.

f must evaluate the functions f_i (that is the first derivatives y_i') for given values of the arguments t, y_i.

```
[yp, user] = f(t, n, y, user)
```

Input Parameters

1: **t** – REAL (KIND=nag_wp)

t, the current value of the independent variable.

2: **n** – INTEGER

n, the number of ordinary differential equations in the system to be solved.

3: **y**(**n**) – REAL (KIND=nag_wp) array

The current values of the dependent variables, y_i, for $i = 1, 2, \ldots, n$.

4: **user** – INTEGER array

f is called from nag_ode_ivp_rkts_range (d02pe) with the object supplied to nag_ode_ivp_rkts_range (d02pe).

Output Parameters

1: **yp**(**n**) – REAL (KIND=nag_wp) array

The values of f_i, for $i = 1, 2, \ldots, n$.

2: **user** – INTEGER array

2: **twant** – REAL (KIND=nag_wp)

t, the next value of the independent variable where a solution is desired.

Constraint: **twant** must be closer to **tend** than the previous value of **tgot** (or **tstart** on the first call to nag_ode_ivp_rkts_range (d02pe)); see nag_ode_ivp_rkts_setup (d02pq) for a description of **tstart** and **tend**. **twant** must not lie beyond **tend** in the direction of integration.

3: **ygot**(**n**) – REAL (KIND=nag_wp) array

On the first call to nag_ode_ivp_rkts_range (d02pe), **ygot** need not be set. On all subsequent calls **ygot** must remain unchanged.

4: **ymax**(**n**) – REAL (KIND=nag_wp) array

On the first call to nag_ode_ivp_rkts_range (d02pe), **ymax** need not be set. On all subsequent calls **ymax** must remain unchanged.

5: **iwsav**(**130**) – INTEGER array

6: **rwsav**(**32** × **n** + **350**) – REAL (KIND=nag_wp) array

These must be the same arrays supplied in a previous call to nag_ode_ivp_rkts_setup (d02pq). They must remain unchanged between calls.

5.2 Optional Input Parameters

1: **n** – INTEGER

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

n, the number of ordinary differential equations in the system to be solved.

Constraint: $\mathbf{n} \geq 1$.

2: **user** – INTEGER array

user is not used by nag_ode_ivp_rkts_range (d02pe), but is passed to **f**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

5.3 Output Parameters

1: **tgot** – REAL (KIND=nag_wp)

t, the value of the independent variable at which a solution has been computed. On successful exit with $\mathbf{ifail} = 0$, **tgot** will equal **twant**. On exit with $\mathbf{ifail} > 1$, a solution has still been computed at the value of **tgot** but in general **tgot** will not equal **twant**.

2: **ygot**(**n**) – REAL (KIND=nag_wp) array

An approximation to the true solution at the value of **tgot**. At each step of the integration to **tgot**, the local error has been controlled as specified in nag_ode_ivp_rkts_setup (d02pq). The local error has still been controlled even when $\mathbf{tgot} \neq \mathbf{twant}$, that is after a return with $\mathbf{ifail} > 1$.

3: **ypgot**(**n**) – REAL (KIND=nag_wp) array

An approximation to the first derivative of the true solution at **tgot**.

4: **ymax**(**n**) – REAL (KIND=nag_wp) array

ymax(i) contains the largest value of $|y_i|$ computed at any step in the integration so far.

5: **user** – INTEGER array

6: **iwsav**(**130**) – INTEGER array

7: **rwsav**($\mathbf{32} \times \mathbf{n} + \mathbf{350}$) – REAL (KIND=nag_wp) array

Information about the integration for use on subsequent calls to nag_ode_ivp_rkts_range (d02pe) or other associated functions.

8: **ifail** – INTEGER

$\mathbf{ifail} = 0$ 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, a previous call to the setup function has not been made or the communication arrays have become corrupted.

On entry, $\mathbf{n} = \langle value \rangle$, but the value passed to the setup function was .

On entry, the communication arrays have become corrupted, or a catastrophic error has already been detected elsewhere. You cannot continue integrating the problem.

tend (setup) had already been reached in a previous call.
To start a new problem, you will need to call the setup function.

twant does not lie in the direction of integration.

twant is too close to the last value of **tgot** (**tstart** on setup).

twant lies beyond **tend** (setup) in the direction of integration, but is very close to **tend**.

twant lies beyond **tend** (setup) in the direction of integration.

You cannot call this function after it has returned an error.
You must call the setup function to start another problem.

You cannot call this function when you have specified, in the setup function, that the step integrator will be used.

ifail $= 2$ (*warning*)

This function is being used inefficiently because the step size has been reduced drastically many times to obtain answers at many points. Using the order 4 and 5 pair method at setup is more appropriate here. You can continue integrating this problem.

ifail $= 3$ (*warning*)

Approximately $\langle value \rangle$ function evaluations have been used to compute the solution since the integration started or since this message was last printed. However, you can continue integrating the problem.

ifail $= 4$ (*warning*)

Approximately $\langle value \rangle$ function evaluations have been used to compute the solution since the integration started or since this message was last printed. Your problem has been diagnosed as stiff. If the situation persists, it will cost roughly $\langle value \rangle$ times as much to reach **tend** (setup) as it has cost to reach the current time. You should probably call functions intended for stiff problems. However, you can continue integrating the problem.

ifail $= 5$ (*warning*)

In order to satisfy your error requirements the solver has to use a step size of $\langle value \rangle$ at the current time, $\langle value \rangle$. This step size is too small for the ***machine precision***, and is smaller than $\langle value \rangle$.

ifail $= 6$ (*warning*)

The global error assessment algorithm failed at start of integration.
The integration is being terminated.

The global error assessment may not be reliable for times beyond $\langle value \rangle$.
The integration is being terminated.

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 accuracy of integration is determined by the arguments **tol** and **thresh** in a prior call to nag_ode_ivp_rkts_setup (d02pq) (see the function document for nag_ode_ivp_rkts_setup (d02pq) for further details and advice). Note that only the local error at each step is controlled by these arguments. The error estimates obtained are not strict bounds but are usually reliable over one step. Over a number of steps the overall error may accumulate in various ways, depending on the properties of the differential system.

8 Further Comments

If nag_ode_ivp_rkts_range (d02pe) returns with **ifail** $= 5$ and the accuracy specified by **tol** and **thresh** is really required then you should consider whether there is a more fundamental difficulty. For example, the solution may contain a singularity. In such a region the solution components will usually be large in magnitude. Successive output values of **ygot** and **ymax** should be monitored (or nag_ode_ivp_rkts_onestep (d02pf) should be used since this takes one integration step at a time) with the aim of trapping the solution before the singularity. In any case numerical integration cannot be continued through a singularity, and analytical treatment may be necessary.

Performance statistics are available after any return from nag_ode_ivp_rkts_range (d02pe) by a call to nag_ode_ivp_rkts_diag (d02pt). If **method** > 0 in the call to nag_ode_ivp_rkts_setup (d02pq), global error assessment is available after a return from nag_ode_ivp_rkts_range (d02pe) with **ifail** $= 0$, 2, 3, 4, 5 or 6 by a call to nag_ode_ivp_rkts_errass (d02pu).

After a failure with **ifail** $= 5$ or 6 each of the diagnostic functions nag_ode_ivp_rkts_diag (d02pt) and nag_ode_ivp_rkts_errass (d02pu) may be called only once.

If nag_ode_ivp_rkts_range (d02pe) returns with **ifail** $= 4$ then it is advisable to change to another code more suited to the solution of stiff problems. nag_ode_ivp_rkts_range (d02pe) will not return with **ifail** $= 4$ if the problem is actually stiff but it is estimated that integration can be completed using less function evaluations than already computed.

9 Example

This example solves the equation

$$y'' = -y, \quad y(0) = 0, \quad y'(0) = 1$$

reposed as

$$y_1' = y_2$$
$$y_2' = -y_1$$

over the range $[0, 2\pi]$ with initial conditions $y_1 = 0.0$ and $y_2 = 1.0$. Relative error control is used with threshold values of 1.0e−8 for each solution component and compute the solution at intervals of length $\pi/4$ across the range. A low-order Runge–Kutta method (see nag_ode_ivp_rkts_setup (d02pq)) is also used with tolerances **tol** $=$ 1.0e−3 and **tol** $=$ 1.0e−4 in turn so that the solutions can be compared.

See also Section 10 in nag_ode_ivp_rkts_errass (d02pu).

9.1 Program Text

```
function d02pe_example

fprintf('d02pe example results\n\n');

% Set initial conditions and input
method = nag_int(1);
tstart = 0;
tend = 2*pi;
yinit = [0;1];
hstart = 0;
thresh = [1e-08; 1e-08];
```

```
npts = 40;
tol0 =  1.0E-3;
ygot = zeros(npts+1, 2);
tgot = zeros(npts+1, 1);
err1 = zeros(npts+1, 2);
err2 = zeros(npts+1, 2);
ymax = zeros(1, 2);

% Set control for output
tinc = (tend-tstart)/npts;
tol = 10.0*tol0;

% We run through the calculation twice with two tolerance values
for i = 1:2

  tol = tol*0.1;

  % Call setup function
  [iwsav, rwsav, ifail] = d02pq(tstart, tend, yinit, tol, thresh, method);

  tgot(1) = tstart;
  ygot(1,:) = yinit;
  twant = tstart;
  for j=1:npts
    twant = twant + tinc;
    [tgot(j+1), ygot(j+1,:), ypgot, ymax, user, iwsav, rwsav, ifail] = ...
      d02pe(@f, twant, ygot(j, :), ymax, iwsav, rwsav);
    err1(j+1, i) =  ygot(j+1, 1)-sin(tgot(j+1));
    err2(j+1, i) =  ygot(j+1, 2)-cos(tgot(j+1));
  end

  fprintf('\nCalculation with TOL = %8.1e:\n\n', tol);
  [fevals, stepcost, waste, stepsok, hnext, iwsav, ifail] = d02pt(iwsav, rwsav);
  fprintf('  Number of evaluations of f = %d\n', fevals);

end

% Plot results
fig1 = figure;
title('First-order ODEs using Runge-Kutta Low-order Method, Two Tolerances');
hold on;
axis([0 10 -1.2 1.2]);
xlabel('t');
ylabel('Solution (y, y'')');
plot(tgot, ygot(:, 1), '-xr');
text(ceil(tgot(npts+1)), ygot(npts+1, 1)-0.2, 'y', 'Color', 'r');
plot(tgot, ygot(:, 2), '-xg');
text(ceil(tgot(npts+1)), ygot(npts+1, 2), 'y''', 'Color', 'g');
% Plot errors with a different (log) scale
ax1 = gca;
ax2 = axes('Position',get(ax1,'Position'),...
           'XAxisLocation','bottom','YAxisLocation','right',...
           'YScale','log','Color','none','XColor','k','YColor','k');
hold on;
axis([0 10 1e-7 0.01]);
ylabel('abs(Error)');
plot(ax2, tgot, abs(err1(:, 1)), '-*b');
text(ceil(tgot(npts+1)), err1(npts+1, 1), 'y-error (tol=0.001)', 'Color', 'b');
plot(ax2, tgot, abs(err1(:, 2)), '-sm');
text(ceil(tgot(npts+1)), err1(npts+1, 2), 'y-error (tol=0.0001)', 'Color', 'm');

hold off

function [yp, user] = f(t, n, y, user)
  yp = [y(2); -y(1)];
```

9.2 Program Results

```
     d02pe example results

Calculation with TOL =  1.0e-03:

  Number of evaluations of f = 421

Calculation with TOL =  1.0e-04:

  Number of evaluations of f = 871
```