NAG Toolbox

nag_ode_ivp_rkts_diag (d02pt)

1 Purpose

nag_ode_ivp_rkts_diag (d02pt) provides details about an integration performed by either nag_ode_ivp_rkts_range (d02pe) or nag_ode_ivp_rkts_onestep (d02pf).

2 Syntax

```
[fevals, stepcost, waste, stepsok, hnext, iwsav, ifail] = nag_ode_ivp_rkts_diag(iwsav, rwsav)
[fevals, stepcost, waste, stepsok, hnext, iwsav, ifail] = d02pt(iwsav, rwsav)
```

3 Description

nag_ode_ivp_rkts_diag (d02pt) and its associated functions (nag_ode_ivp_rkts_range (d02pe), nag_ode_ivp_rkts_onestep (d02pf), nag_ode_ivp_rkts_setup (d02pq), nag_ode_ivp_rkts_reset_tend (d02pr), nag_ode_ivp_rkts_interp (d02ps) and nag_ode_ivp_rkts_errass (d02pu)) solve the 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.

After a call to nag_ode_ivp_rkts_range (d02pe) or nag_ode_ivp_rkts_onestep (d02pf), nag_ode_ivp_rkts_diag (d02pt) can be called to obtain information about the cost of the integration and the size of the next step.

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: **iwsav**(**130**) – INTEGER array

2: **rwsav**(**350**) – REAL (KIND=nag_wp) array

Note: the communication **rwsav** used by the other functions in the suite must be used here however, only the first 350 elements will be referenced.

These must be the same arrays supplied in a previous call to nag_ode_ivp_rkts_range (d02pe) or nag_ode_ivp_rkts_onestep (d02pf). They must remain unchanged between calls.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **fevals** – INTEGER

The total number of evaluations of f used in the integration so far; this includes evaluations of f required for the secondary integration necessary if nag_ode_ivp_rkts_setup (d02pq) had previously been called with **method** > 0.

2: **stepcost** – INTEGER

The cost in terms of number of evaluations of f of a typical step with the method being used for the integration. The method is specified by the argument **method** in a prior call to nag_ode_ivp_rkts_setup (d02pq).

3: **waste** – REAL (KIND=nag_wp)

The number of attempted steps that failed to meet the local error requirement divided by the total number of steps attempted so far in the integration. A 'large' fraction indicates that the integrator is having trouble with the problem being solved. This can happen when the problem is 'stiff' and also when the solution has discontinuities in a low-order derivative.

4: **stepsok** – INTEGER

The number of accepted steps.

5: **hnext** – REAL (KIND=nag_wp)

The step size the integrator will attempt to use for the next step.

6: **iwsav**(**130**) – INTEGER array

Note: the communication **rwsav** used by the other functions in the suite must be used here however, only the first 350 elements will be referenced.

Information about the integration for use on subsequent calls to nag_ode_ivp_rkts_range (d02pe) or nag_ode_ivp_rkts_onestep (d02pf) or other associated functions.

7: **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, a previous call to the setup function has not been made or the communication arrays have become corrupted, or a catastrophic error has already been detected elsewhere.
You cannot continue integrating the problem.

You cannot call this function before you have called the integrator.

You have already made one call to this function after the integrator could not achieve specified accuracy.
You cannot call this function again.

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

Not applicable.

8 Further Comments

When a secondary integration has taken place, that is when global error assessment has been specified using **method** > 0 in a prior call to nag_ode_ivp_rkts_setup (d02pq), then the approximate number of evaluations of f used in this secondary integration is given by $2 \times$ **stepsok** $\times$ **stepcost** for **method** $= 2$ or 3 and $3 \times$ **stepsok** $\times$ **stepcost** for **method** $= 1$.

9 Example

See Section 10 in nag_ode_ivp_rkts_range (d02pe), nag_ode_ivp_rkts_onestep (d02pf), nag_ode_ivp_rkts_reset_tend (d02pr), nag_ode_ivp_rkts_interp (d02ps) and nag_ode_ivp_rkts_errass (d02pu).

9.1 Program Text

```
function d02pt_example

fprintf('d02pt 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);

  fprintf('\nCalculation with TOL = %8.1e\n\n', tol);
  fprintf('    t         y1        y2       err1     err2\n');
  fprintf(' %6.3f   %7.3f   %7.3f   %7.3f   %7.3f\n', tstart, yinit, 0, 0);

  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));

    if rem(j, 5) == 0
      fprintf(' %6.3f   %7.3f   %7.3f   %7.3f   %7.3f\n', ...
              tgot(j+1), ygot(j+1, :), err1(j+1, i), err2(j+1));
    end
  end

  [fevals, stepcost, waste, stepsok, hnext, iwsav, ifail] = d02pt(iwsav, rwsav);
  fprintf('Cost of the integration in evaluations of f is %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)');
n1 = npts + 1;
plot(ax2, tgot, abs(err1(:, 1)), '-*b');
text(ceil(tgot(n1)), err1(n1, 1), 'y-error (tol=0.001)', 'Color', 'b');
plot(ax2, tgot, abs(err1(:, 2)), '-sm');
text(ceil(tgot(n1)), err1(n1, 2), 'y-error (tol=0.0001)', 'Color', 'm');

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

9.2 Program Results

```
     d02pt example results

Calculation with TOL =  1.0e-03

    t         y1        y2       err1     err2
  0.000     0.000     1.000     0.000     0.000
  0.785     0.707     0.707    -0.000    -0.000
  1.571     0.999    -0.000    -0.001    -0.000
  2.356     0.706    -0.706    -0.001     0.001
  3.142    -0.000    -0.998    -0.000     0.002
  3.927    -0.706    -0.705     0.001     0.002
  4.712    -0.998     0.001     0.002     0.001
  5.498    -0.705     0.706     0.002    -0.002
  6.283     0.001     0.997     0.001    -0.003
Cost of the integration in evaluations of f is 421

Calculation with TOL =  1.0e-04

    t         y1        y2       err1     err2
  0.000     0.000     1.000     0.000     0.000
  0.785     0.707     0.707    -0.000    -0.000
  1.571     1.000    -0.000    -0.000    -0.000
  2.356     0.707    -0.707    -0.000     0.001
  3.142    -0.000    -1.000    -0.000     0.002
```

```
  3.927    -0.707    -0.707     0.000     0.002
  4.712    -1.000     0.000     0.000     0.001
  5.498    -0.707     0.707     0.000    -0.002
  6.283     0.000     1.000     0.000    -0.003
Cost of the integration in evaluations of f is 871
```