nag_ode_ivp_adams_interp (d02qzc)

1. Purpose

nag_ode_ivp_adams_interp (d02qzc) interpolates components of the solution of a non-stiff system of first order ordinary differential equations from information provided by nag_ode_ivp_adams_roots (d02qfc). Normally this function will be used in conjunction with the integration function, nag_ode_ivp_adams_roots (d02qfc), operating in one-step mode.

2. Specification

```
#include <nag.h>
#include <nagd02.h>

void nag_ode_ivp_adams_interp(Integer neqf, double twant, Integer nwant,
            double ywant[], double ypwant[], Nag_ODE_Adams *opt, NagError *fail)
```

3. Description

nag_ode_ivp_adams_interp evaluates the first **nwant** components of the solution of a non-stiff system of first order ordinary differential equations at any point using the method of Watts and Shampine (1986) and information generated by nag_ode_ivp_adams_roots (d02qfc). nag_ode_ivp_adams_interp should not normally be used to extrapolate outside the current range of the values produced by the integration routine.

4. Parameters

neqf

Input: the number of differential equations.
Constraint: **neqf** ≥ 1.

twant

Input: 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

opt.tcurr $-$ **opt.hlast** $\leq$ **twant** $\leq$ **opt.tcurr**.

or if integration is proceeding in the negative direction

opt.tcurr $-$ **opt.hlast** $\geq$ **twant** $\geq$ **opt.tcurr**.

Extrapolation is permitted but not recommended and a **fail.code** value of **NW_EXTRAPOLATION** is returned whenever extrapolation is attempted.

nwant

Input: the number of components of the solution and derivative whose values, at **twant**, are required. The first **nwant** components are evaluated.
Constraint: $1 \leq$ **nwant** $\leq$ **neqf**.

ywant[nwant]

Output: **ywant**$[i-1]$ contains the calculated value of the ith component of the solution at **twant**, for $i = 1, 2, \ldots,$**nwant**.

ypwant[nwant]

Output: **ypwant**$[i-1]$ contains the calculated value of the ith component of the derivative at **twant**, for $i = 1, 2, \ldots,$**nwant**.

opt

Input: the structure of type Nag_ODE_Adams as output from the integration function nag_ode_ivp_adams_roots (d02qfc). The structure **must** be passed unchanged. (See Section 6 for comments about deallocation of memory from **opt**.)

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.
It is recommended that **fail.print** be set to **TRUE**.

5. Error Indications and Warnings

NE_NO_INTEGRATE

The integrator function nag_ode_ivp_adams_roots (d02qfc) has not been called.

NE_NEQF

The value of **neqf** supplied is not the same as that given to the setup function nag_ode_ivp_adams_setup (d02qwc). **neqf** = ⟨*value*⟩ but the value given to nag_ode_ivp_adams_setup (d02qwc) was ⟨*value*⟩.

NE_NWANT_GT

nwant is greater than the value of **neqf** given to the setup function nag_ode_ivp_adams_setup (d02qwc). **nwant** = ⟨*value*⟩, **neqf** = ⟨*value*⟩.

NE_INT_ARG_LT

On entry, **nwant** must not be less than 1: **nwant** = ⟨*value*⟩.

NE_NO_STEPS

No successful integration steps were taken in the call(s) to the integration function nag_ode_ivp_adams_roots (d02qfc).

NW_EXTRAPOLATION

Extrapolation requested, **twant** = ⟨*value*⟩.

6. 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.

The structure **opt** will contain pointers which have been allocated memory during a call to nag_ode_ivp_adams_setup (d02qwc). This allocated memory is used by nag_ode_ivp_adams_roots (d02qfc) and nag_ode_ivp_adams_interp. When all calls to these functions have been completed the function nag_ode_ivp_adams_free (d02qyc) may be called to free the allocated memory from the structure.

6.1. Accuracy

The error in interpolation is of a similar order to the error arising from the integration. The same order of accuracy can be expected when extrapolating using nag_ode_ivp_adams_interp. However, the actual error in extrapolation will, in general, be much larger than for interpolation.

6.2. References

Watts H A and Shampine L F (1986) Smoother Interpolants for Adams Codes *SIAM J. Sci. Statist. Comput.* **7** 334–345.

7. See Also

nag_ode_ivp_adams_roots (d02qfc)
nag_ode_ivp_adams_setup (d02qwc)
nag_ode_ivp_adams_free (d02qyc)

8. Example

We solve the equation

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

reposed as

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -y_1 \end{aligned}$$

over the range $[0, \pi/2]$ with initial conditions $y_1 = 0$ and $y_2 = 1$ using vector error control (**vectol** = **TRUE**) and nag_ode_ivp_adams_roots (d02qfc) in one-step mode (**one_step** = **TRUE**). nag_ode_ivp_adams_interp is used to provide solution values at intervals of $\pi/16$.

8.1. Program Text

```
/* nag_ode_ivp_adams_interp(d02qzc) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagd02.h>
#include <nagx01.h>

#ifdef NAG_PROTO
static void ftry03(Integer neqf, double x, double y[], double yp[],
                   Nag_User *comm);
#else
static void ftry03();
#endif

#define NEQF 2
#define TSTART 0.0

main()
{
  double atol[NEQF], rtol[NEQF], y[NEQF], ywant[NEQF], ypwant[NEQF];
  Boolean crit, alter_g, vectol, one_step, sophist;
  double t, tstart, tinc, tout, tcrit, twant, hmax, pi;
  Integer max_step, i, j, neqf, neqg, nwant;
  Nag_Start state;
  Nag_ODE_Adams opt;
  static NagError fail;

  fail.print = TRUE;

  Vprintf("d02qzc Example Program Results\n");
  pi = X01AAC;
  state = Nag_NewStart;
  neqf = NEQF;
  neqg = 0;
  vectol = TRUE;
  for (i = 0; i < 2; ++i)
    {
      atol[i] = 1e-08;
      rtol[i] = 0.0001;
    }
  one_step = TRUE;
  crit = TRUE;
  tinc = pi * 0.0625;
  tcrit = tinc * 8.0;
  tout = tcrit;
  max_step = 500;
  hmax = 2.0;
  t = TSTART;
  twant = TSTART + tinc;
  nwant = 2;
  y[0] = 0.0;
  y[1] = 1.0;
  Vprintf("\n   T        Y(1)    Y(2)\n");
  Vprintf(" %6.4f   %7.4f %7.4f  \n",t, y[0], y[1]);

  d02qwc(&state, neqf, vectol, atol, rtol, one_step, crit,
         tcrit, hmax, max_step, neqg, &alter_g, sophist, &opt, &fail);

  j = 1;

  while (t < tout && fail.code == NE_NOERROR)
    {
```

```
      d02qfc(neqf, ftry03, &t, y, tout, NULLDFN, NAGUSER_DEFAULT, &opt, &fail);

      while (twant <= t && fail.code == NE_NOERROR)
        {
          d02qzc(neqf, twant, nwant, ywant, ypwant, &opt, &fail);

          Vprintf(" %6.4f   %7.4f %7.4f  \n",twant, ywant[0], ywant[1]);
          ++j;
          twant = (double)j*tinc + 0.0;
        }
    }
  /* Free the memory which was allocated by
   * d02qwc to the pointers inside opt.
   */
  d02qyc(&opt);

  if (fail.code == NE_NOERROR) exit(EXIT_SUCCESS);
  else exit(EXIT_FAILURE);
}                               /* main */


#ifdef NAG_PROTO
static void ftry03(Integer neqf, double x, double y[], double yp[],
                   Nag_User *comm)
#else
     static void ftry03(neqf, x, y, yp, comm)
     Integer neqf;
     double x;
     double y[], yp[];
     Nag_User *comm;
#endif
{
  yp[0] = y[1];
  yp[1] = -y[0];
}                               /* ftry03 */
```

8.2. Program Data

None.

8.3. Program Results

```
d02qzc Example Program Results

   T        Y(1)    Y(2)
 0.0000    0.0000  1.0000
 0.1963    0.1951  0.9808
 0.3927    0.3827  0.9239
 0.5890    0.5556  0.8315
 0.7854    0.7071  0.7071
 0.9817    0.8315  0.5556
 1.1781    0.9239  0.3827
 1.3744    0.9808  0.1951
 1.5708    1.0000  0.0000
```