NAG Library Function Document

nag_ode_ivp_rk_interp (d02pxc)

1 Purpose

nag_ode_ivp_rk_interp (d02pxc) is a function to compute the solution of a system of ordinary
differential equations using interpolation anywhere on an integration step taken by
nag_ode_ivp_rk_onestep (d02pdc).

2 Specification

```c
#include <nag.h>
#include <nagd02.h>
void nag_ode_ivp_rk_interp (Integer neq, double twant, Nag_SolDeriv request,
   Integer nwant, double ywant[], double ypwant[],
   void (*f)(Integer neq, double t, const double y[], double yp[],
             Nag_User *comm),
   Nag_ODE_RK *opt, Nag_User *comm, NagError *fail)
```

3 Description

nag_ode_ivp_rk_interp (d02pxc) and its associated functions (nag_ode_ivp_rk_setup (d02pvc),
nag_ode_ivp_rk_onestep (d02pdc), nag_ode_ivp_rk_reset_tend (d02pwc), nag_ode_ivp_rk_errass
(d02pzc)) 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 (Brankin
et al. (1991)) integrate

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

where \( y \) is the vector of \( neq \) solution components and \( t \) is the independent variable.

nag_ode_ivp_rk_onestep (d02pdc) computes the solution at the end of an integration step. Using the
information computed on that step nag_ode_ivp_rk_interp (d02pxc) computes the solution by
interpolation at any point on that step. It cannot be used if method = Nag_RK_7_8 was specified in
the call to setup function nag_ode_ivp_rk_setup (d02pvc).

4 References

initial value problems for ODEs SoftReport 91-S1 Southern Methodist University

5 Arguments

1: \( neq \) – Integer \( \quad \text{Input} \)

\( On \ entry: \) the number of ordinary differential equations in the system.

\( Constraint: \ neq \geq 1. \)

2: \( twant \) – double \( \quad \text{Input} \)

\( On \ entry: \) the value of the independent variable, \( t \), where a solution is desired.

3: \( request \) – Nag_SolDeriv \( \quad \text{Input} \)

\( On \ entry: \) determines whether the solution and/or its first derivative are computed as follows:
request = Nag_Sol – compute approximate solution only;
request = Nag_Der – compute approximate first derivative of the solution only;
request = Nag_SolDer – compute both approximate solution and first derivative.

Constraint: request = Nag_Sol, Nag_Der or Nag_SolDer.

4: nwant – Integer
   Input
   On entry: the number of components of the solution to be computed. The first nwant components are evaluated.
   Constraint: 1 ≤ nwant ≤ neq.

5: ywant[nwant] – double
   Output
   On exit: an approximation to the first nwant components of the solution at twant when specified by request.

6: ypwant[nwant] – double
   Output
   On exit: an approximation to the first nwant components of the first derivative of the solution at twant when specified by request.

7: f – function, supplied by the user
   External Function
   f must evaluate the functions \( f_i \) (that is the first derivatives \( y'_i \)) for given values of the arguments \( t, y_i \). It must be the same procedure as supplied to nag_ode_ivp_rk_onestep (d02pdc).

The specification of f is:

```c
void f (Integer neq, double t, const double y[], double yp[],
        Nag_User *comm)
```

1: neq – Integer
   Input
   On entry: the number of differential equations.

2: t – double
   Input
   On entry: the current value of the independent variable, t.

3: y[neq] – const double
   Input
   On entry: the current values of the dependent variables, \( y_i \) for \( i = 1, 2, \ldots, \text{neq} \).

4: yp[neq] – double
   Output
   On exit: the values of \( f_i \) for \( i = 1, 2, \ldots, \text{neq} \).

5: comm – Nag_User *
   Pointer to a structure of type Nag_User with the following member:
   p – Pointer
   On entry/exit: the pointer comm->p should be cast to the required type, e.g.,
   struct user *s = (struct user *)comm -> p, to obtain the original object’s address with appropriate type. (See the argument comm below.)

8: opt – Nag_ODE_RK *
   Input/Output
   On entry: the structure of type Nag_ODE_RK as output from nag_ode_ivp_rk_onestep (d02pdc).
   You must not change this structure.
On exit: some members of opt are changed internally.

9: comm – Nag_User *

Pointer to a structure of type Nag_User with the following member:

p – Pointer

On entry/exit: the pointer comm—p, of type Pointer, allows you to communicate information to and from f. An object of the required type should be declared, e.g., a structure, and its address assigned to the pointer comm—p by means of a cast to Pointer in the calling program, e.g., comm.p = (Pointer)&s. The type pointer will be void * with a C compiler that defines void * and char * otherwise.

10: fail – NagError *

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_GT

On entry, nwant = (value) while neq = (value). These arguments must satisfy neq ≤ nwant.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument request had an illegal value.

NE_INT_ARG_LT

On entry, nwant = (value).
Constraint: nwant ≥ 1.

NE_MEMORY_FREED

Internally allocated memory has been freed by a call to nag_ode_ivp_rk_free (d02ppc) without a subsequent call to the setup function nag_ode_ivp_rk_setup (d02pvc).

NE_MISSING_CALL

Previous call to nag_ode_ivp_rk_onestep (d02pdc) has not been made, hence nag_ode_ivp_rk_interp (d02pxc) must not be called.

NE_NEQ

The value of neq supplied is not the same as that given to the setup function nag_ode_ivp_rk_setup (d02pvc). neq = (value) but the value given to nag_ode_ivp_rk_setup (d02pvc) was (value).

NE_PREV_CALL

The previous call to a function had resulted in a severe error. You must call nag_ode_ivp_rk_setup (d02pvc) to start another problem.

NE_PREV_CALL_INI

The previous call to the function nag_ode_ivp_rk_onestep (d02pdc) resulted in a severe error. You must call nag_ode_ivp_rk_setup (d02pvc) to start another problem.
The function to be called as specified in the setup function nag_ode_ivp_rk_setup (d02pvc) was nag_ode_ivp_rk_range (d02pcc). However the actual call was made to nag_ode_ivp_rk_interp (d02pxc). This is not permitted.

Interpolation is not available with method = Nag_RK_7_8. Either use method = Nag_RK_2_3 or Nag_RK_4_5 for which interpolation is available. Alternatively use nag_ode_ivp_rk_reset_tend (d02pwc) to make nag_ode_ivp_rk_onestep (d02pdc) step exactly to the points where you want output.

The computed values will be of a similar accuracy to that computed by nag_ode_ivp_rk_onestep (d02pdc).

Not applicable.

None.

We solve the equation

\[ y'' = -y, \quad y(0) = 0, y'(0) = 1 \]

reposed as

\[ y_1' = y_2, \quad y_2' = -y_1 \]

over the range [0, 2\pi] with initial conditions \( y_1 = 0.0 \) and \( y_2 = 1.0 \). We use relative error control with threshold values of 1.0e-8 for each solution component. nag_ode_ivp_rk_onestep (d02pdc) is used to integrate the problem one step at a time and nag_ode_ivp_rk_interp (d02pxc) is used to compute the first component of the solution and its derivative at intervals of length \pi/8 across the range whenever these points lie in one of those integration steps. We use a moderate order Runge–Kutta method (method = Nag_RK_4_5) with tolerances tol = 1.0e−3 and tol = 1.0e−4 in turn so that we may compare the solutions. The value of \pi\) is obtained by using nag_pi (X01AAC).

/* nag_ode_ivp_rk_interp (d02pxc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group. *
 * Mark 3, 1992. *
 * Mark 7 revised, 2001. *
 * Mark 8 revised, 2004. *
 */
#include <nag.h>
#include <stdio.h>
#include <math.h>
#include <nagd02.h>
#include <nagx01.h>
```c
#define NEQ 2
#define NWANT 1
#define ZERO 0.0
#define ONE 1.0
#define TWO 2.0
#define FOUR 4.0

int main(void)
{
    static Integer use_comm[1] = {1};
    Integer    exit_status = 0, i, j, neq, nout, nwant;
    NagError   fail;
    Nag_ErrorAssess errass;
    Nag_ODE_RK opt;
    Nag_RK_method method;
    Nag_User   comm;
    double     hstart, pi, tend, *thres = 0, tinc, tnow, tol, tstart, twant,
               *ynow = 0;
    double     *ypnow = 0, *ypwant = 0, *ystart = 0, *ywant = 0;

    INIT_FAIL(fail);
    printf("nag_ode_ivp_rk_interp (d02pxc) Example Program Results\n"));
    /* For communication with user-supplied functions: */
    comm.p = (Pointer)&use_comm;
    /* Set initial conditions and input for nag_ode_ivp_rk_setup (d02pvc) */
    neq = NEQ;
    nwant = NWANT;
    if (neq >= 1)
    {
        if (!(thres = NAG_ALLOC(neq, double)) ||
             !(ynow = NAG_ALLOC(neq, double)) ||
             !(ypnow = NAG_ALLOC(neq, double)) ||
             !(ystart = NAG_ALLOC(neq, double)) ||
             !(ywant = NAG_ALLOC(nwant, double)) ||
             !(ypwant = NAG_ALLOC(nwant, double)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    else
    {
      exit_status = 1;
      return exit_status;
    }

    method = Nag_RK_4_5;
    /* nag_pi (x01aac). */
    pi = nag_pi;
    tstart = ZERO;
    ystart[0] = ZERO;
    ystart[1] = ONE;
    tend = TWO*pi;
    for (i = 0; i < neq; i++)
        thres[i] = 1.0e-8;
```
errass = Nag_ErrorAssess_off;
hstart = ZERO;

/*
 * Set control for output
 */
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
ypwant[0]);
    j = j - 1;
    twant = tend - j*tinc;
  }
} while (tnow < tend);

printf("\nCost of the integration in evaluations of f is"
  " %"NAG_IFMT"
  ", opt.totfcn);
/* nag_ode_ivp_rk_free (d02ppc).
  * Freeing function for use with the Runge-Kutta suite (d02p
  * functions)
  */
  nag_ode_ivp_rk_free(&opt);
}
END:
NAG_FREE(thres);
NAG_FREE(ynow);
NAG_FREE(ypnow);
NAG_FREE(ystart);
NAG_FREE(ywant);
NAG_FREE(ypwant);
return exit_status;
}

static void NAG_CALL f(Integer neq, double t, const double y[],
    double yp[],
    Nag_User *comm)
{
  Integer *use_comm = (Integer *)comm->p;
  if (use_comm[0])
    { printf("(User-supplied callback f, first invocation.\\n"));
      use_comm[0] = 0;
    }
  yp[0] = y[1];
  yp[1] = -y[0];
}

10.2 Program Data

None.

10.3 Program Results

nag_ode_ivp_rk_interp (d02pxc) Example Program Results

Calculation with tol = 1.0e-03

<table>
<thead>
<tr>
<th>t</th>
<th>y1</th>
<th>y2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.393</td>
<td>0.3827</td>
<td>0.9239</td>
</tr>
<tr>
<td>0.785</td>
<td>0.7071</td>
<td>0.7071</td>
</tr>
<tr>
<td>1.178</td>
<td>0.9239</td>
<td>0.3826</td>
</tr>
<tr>
<td>1.571</td>
<td>1.0000</td>
<td>-0.0001</td>
</tr>
<tr>
<td>1.963</td>
<td>0.9238</td>
<td>-0.3828</td>
</tr>
<tr>
<td>2.356</td>
<td>0.7070</td>
<td>-0.7073</td>
</tr>
<tr>
<td>2.749</td>
<td>0.3825</td>
<td>-0.9240</td>
</tr>
<tr>
<td>3.142</td>
<td>-0.0002</td>
<td>-0.9999</td>
</tr>
<tr>
<td>3.534</td>
<td>-0.3829</td>
<td>-0.9238</td>
</tr>
<tr>
<td>3.927</td>
<td>-0.7072</td>
<td>-0.7069</td>
</tr>
<tr>
<td>4.320</td>
<td>-0.9239</td>
<td>-0.3823</td>
</tr>
<tr>
<td>4.712</td>
<td>-0.9999</td>
<td>0.0004</td>
</tr>
<tr>
<td>5.105</td>
<td>-0.9236</td>
<td>0.3830</td>
</tr>
<tr>
<td>5.498</td>
<td>-0.7068</td>
<td>0.7073</td>
</tr>
<tr>
<td>5.890</td>
<td>-0.3823</td>
<td>0.9239</td>
</tr>
<tr>
<td>6.283</td>
<td>0.0004</td>
<td>0.9998</td>
</tr>
</tbody>
</table>
Cost of the integration in evaluations of $f$ is 68

Calculation with $\text{tol} = 1.0\times10^{-04}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$y_1$</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.393</td>
<td>0.3827</td>
<td>0.9239</td>
</tr>
<tr>
<td>0.785</td>
<td>0.7071</td>
<td>0.7071</td>
</tr>
<tr>
<td>1.178</td>
<td>0.9239</td>
<td>0.3827</td>
</tr>
<tr>
<td>1.571</td>
<td>1.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>1.963</td>
<td>0.9239</td>
<td>-0.3827</td>
</tr>
<tr>
<td>2.356</td>
<td>0.7071</td>
<td>-0.7071</td>
</tr>
<tr>
<td>2.749</td>
<td>0.3827</td>
<td>-0.9239</td>
</tr>
<tr>
<td>3.142</td>
<td>-0.0000</td>
<td>-1.0000</td>
</tr>
<tr>
<td>3.534</td>
<td>-0.3827</td>
<td>-0.9239</td>
</tr>
<tr>
<td>3.927</td>
<td>-0.7071</td>
<td>-0.7071</td>
</tr>
<tr>
<td>4.320</td>
<td>-0.9239</td>
<td>-0.3827</td>
</tr>
<tr>
<td>4.712</td>
<td>-1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>5.105</td>
<td>-0.9238</td>
<td>0.3827</td>
</tr>
<tr>
<td>5.498</td>
<td>-0.7071</td>
<td>0.7071</td>
</tr>
<tr>
<td>5.890</td>
<td>-0.3826</td>
<td>0.9239</td>
</tr>
<tr>
<td>6.283</td>
<td>0.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Cost of the integration in evaluations of $f$ is 105