Chapter 11

Ordinary Differential Equations

This chapter describes the CMSSL routine that solves the initial value problem for a system of first-order ordinary differential equations (ODEs) using a Runge-Kutta method. Section 11.2 provides references.

11.1 Explicit Integration of Ordinary Differential Equations Using a Runge-Kutta Method

The initial value problem for a system of N coupled first-order ODEs,

$$dy_i(x)/dx = f_i(x, y_1, \ldots, y_N) \quad i=1, \ldots, N \qquad (1)$$

consists of finding the values $y_i(x_1)$ at some value x_1 of the independent variable x, given the values $y_i(x_0)$ of the dependent variables at x_0. The **ode_rkf** routine solves the initial value problem by integrating explicitly the set of equations (1) using a fifth-order Runge-Kutta-Fehlberg formula. Control of the step size during integration is automatic. The evaluation of the right-hand side and possibly the scaling array for accuracy control are provided by the user through a reverse communication interface.

For detailed information about the **ode_rkf** routine, refer to the man page at the end of this section. The examples below use the syntax and reverse communication interface described in the man page.

Copyright © 1994 Thinking Machines Corporation

11.1.1 Examples

Provided below are schematic examples of how to use the **ode_rkf** reverse communication interface. For actual codes, refer to the on-line examples (the pathname is included in the man page). The following examples assume the variables are stored in a two-dimensional array.

Example 1

In this example, we want to integrate an autonomous system of coupled ODEs using the default error control. The routine `deriv(y,dy)` evaluates the derivatives of *y*. Since the system is autonomous, the derivatives are independent of *x*. We check for possible zero values of *yscal* and set them to a small value, *epsilon*. The reverse communication proceeds as follows:

```
      info=0
      ido=0

10    continue
      call ode_rkf(ido, y, src, dst, w, x, xc, xf, dx, rtol,
     &         atol, ipntr, info, setup)

      if(ido.eq.1) then

          call deriv(src,dst)

      else if (ido.eq.3)

          dst=max(dst,epsilon)

      else
          stop
      endif
      goto 10
```

Example 2

Suppose we want to integrate a time-dependent system of ODEs and we choose our own error control. The routine `deriv(xc,y,dy)` evaluates the derivatives at *xc* while the routine `scal(dx,y,dy,yscal)` evaluates *yscal* = $tol(|y| + dx * |dy/dx|)$. We do not check for zero values in *yscal* (*ido* = 3), since we assume this is done in the routine `scal`.

Copyright © 1994 Thinking Machines Corporation

```
      info=1
      ido=0

10    continue

      call ode_rkf(ido, y, src, dst, w, x, xc, xf, dx, rtol,
     &       atol, ipntr, info, setup)

      if(ido.eq.1) then

          call deriv(xc,src,dst)

      else if (ido.eq.2) then

          call scal(dx,y,src,dst)

      else
          stop
      endif

      goto 10
```


Copyright © 1994 Thinking Machines Corporation

Explicit Integration of Ordinary Differential Equations Using a Runge-Kutta Method

The routines described below integrate ordinary differential equations (ODEs) explicitly using a fifth-order Runge-Kutta-Fehlberg formula. Control of the step size during integration is automatic.

SYNTAX

ode_rkf_setup (***y, w,*** *setup, ier*)

ode_rkf (*ido,* ***y, src, dst, w,*** *x, xc, xf, dx, rtol, atol, ipntr, info, setup*)

deallocate_ode_rkf_setup (*setup*)

ARGUMENTS

ido Scalar integer variable. Reverse communication flag. *ido* must be zero on the first call to **ode_rkf**. The **ode_rkf** routine sets *ido* to indicate the type of operation to be performed by the calling program. The calling program has the responsibility of carrying out the requested operation and calling **ode_rkf** again. The values of *ido* have the meanings listed below. All values except 0 are returned to the calling program.

- 0 The calling program must supply this value on the first call to **ode_rkf**.
- 1 The calling program must compute the derivatives of the dependent variables $dy/dx=f(x,y)$ at $x=xc$. The source array, ***y*** at $x = xc$, is contained in ***src***; the result must be placed in ***dst***.
- 2 (Returned only if *info* =1.) The calling program must compute the scaling array for accuracy control. The values of the dependent variables $y(x)$ are in array ***y***, while ***src*** contains dy/dx at x. The scaling array must be placed in ***dst***.

Version 3.2, April 1994
Copyright © 1994 Thinking Machines Corporation

3 At least one component of the scaling array stored in ***dst*** is zero. Reset the zero values to avoid overflow in the stepsize control (equation (2) in the Description section).

99 The integration is completed.

y Real CM array of arbitrary dimension and shape. Represents the collection of dependent variables. The values of ***y*** are always the values of the variables at *x*. Input values are taken as initial values for the integration.

src Real CM array with the same rank, shape, and layout as ***y***. The source array used by the reverse communication interface. When *ido* = 1, ***src*** contains the values of the dependent variables at *xc*. When *ido* = 2, ***src*** contains the values of the derivatives at *x*.

dst Real CM array with the same rank, shape, and layout as ***y***. The destination array used by the reverse communication interface. When *ido* = 1, ***dst*** must contain the values of the derivatives at *xc* after a user-supplied routine is called using the reverse communication interface. When *ido* = 2, ***dst*** must contain the values of the scaling array used for accuracy control after a user-supplied routine is called using the reverse communication interface.

w Real CM work array of rank one greater than that of ***y***. The first axis must have extent at least 6 and must be declared **:serial**. The remaining axes must match the axes of ***y*** in order of declaration, extents, and layout. This array is used internally.

x Real scalar variable. Independent variable. The value of *x* on input (*ido*=0) corresponds to the initial values of the dependent variables ***y***. During the integration, *x* always has the last value of the independent variable at which the variables ***y*** have been computed successfully.

xc Real scalar variable. Intermediate value of the independent variable to be used for the evaluation of the derivatives. When the derivatives depend explicitly on *x*, it is the variable *xc*, and not *x*, that must be passed to the user-supplied subroutine that evaluates the derivatives when *ido*=1.

Copyright © 1994 Thinking Machines Corporation

xf Real scalar variable. Value of the independent variable x at which the integration is terminated. Input only.

dx Real scalar variable. Step size. On input ($ido=0$), *dx* is the estimated first increment of the independent variable to be attempted by the integrator. Since the integrator is adaptive, a step size that is too large (small) will be reduced (increased) automatically. However, the input value of *dx* cannot be 0. The sign of the input value of *dx* is set internally to the sign of $xf-x$. During the integration, *dx* is the value of the step size currently being attempted by the integrator.

rtol Relative tolerance for the local error control. Must have the same precision (single or double) as ***y***. Ignored if *info* = 1. See the Description section for details.

atol Absolute tolerance for the local error control. Must have the same precision (single or double) as ***y***. Ignored if *info* = 1. See the Description section for details.

ipntr One-dimensional front-end integer array of length at least 2. Reserved for internal use.

info Scalar integer variable. On input, set *info* to 1 if you want to provide your own scaling array for accuracy control using the reverse communication interface. Any other integer value causes **ode_rfk** to use the default scaling array (see the Description section).

setup One-dimensional front-end integer array of length 2.

ier Scalar integer variable. Set to 0 upon successful return. Upon return from **ode_rkf_setup**, may contain the following error codes:

−1	The first dimension of ***w*** is not declared **:serial**.
−2	The serial dimension of ***w*** has extent less than 6.
−3	The rank of ***w*** is not one greater than the rank of ***y***.
−4	The sections of ***w*** containing the vectors and indexed by the first dimension do not have the same shape as ***y***.

Version 3.2, April 1994
Copyright © 1994 Thinking Machines Corporation

DESCRIPTION

The initial value problem for a system of N coupled first-order ordinary differential equations (ODEs),

$$dy_i(x)/dx = f_i(x, y_1, \ldots, y_N) \quad i=1, \ldots, N \qquad (1)$$

consists of finding the values $y_i(x_1)$ at some value x_1 of the independent variable x, given the values $y_i(x_0)$ of the dependent variables at x_0. The **ode_rkf** routine solves the initial value problem by integrating explicitly the set of equations (1) using a fifth-order Runge-Kutta-Fehlberg formula. Control of the step size during integration is automatic. The evaluation of the right-hand side and possibly the scaling array for accuracy control are provided by the user through a reverse communication interface.

Setup and Deallocation. To use **ode_rkf**, follow these steps:

1. Call **ode_rkf_setup**.

   This routine generates a setup ID and returns it in the front-end array *setup*. You must supply this *setup* array in all subsequent **ode_rkf** and **deallocate_ode_rkf** calls associated with this setup call.

2. Call **ode_rkf** iteratively, as described under "Reverse Communication Interface," below.

   You can use the same *setup* array to solve more than one initial value problem sequentially, as long as the array geometries are the same. You can also have more than one setup active at a time.

3. Call **deallocate_ode_rkf**.

   This routine deallocates the memory associated with the setup ID.

Step Size Control and Accuracy. For a trial integration step dx, **ode_rkf** estimates the errors $yerr_i(x + dx)$ of the computed solution at $x + dx$. This allows for automatic control of the step size if one is able to accept or reject the step based on a prescribed accuracy requirement. The test implemented in **ode_rkf** is to accept the step if $errmax < 1$, where

$$errmax = \max_i \frac{|yerr_i(x + dx)|}{yscal_i(x)} \quad i = 1, \ldots, N \quad (2)$$

and the scaling array *yscal* is defined by

Copyright © 1994 Thinking Machines Corporation

$$yscal_i(x) = rtol * |y_i(x)| + atol \qquad i = 1, \ldots, N$$

The input arguments *rtol* and *atol* are the relative and absolute precision, respectively. This combination covers the range between relative (*atol* = 0) and absolute (*rtol* = 0) accuracy. If any component of *yscal* is zero, the reverse communication interface returns the code *ido* = 3, allowing you to reset the zero components in order to avoid overflow in the evaluation of (2).

You may wish to monitor the accuracy of the integration in a way that is more appropriate to your problem. For instance, it may be desirable to scale the error with the time step *dx* and to include the value of the derivatives in *yscal*, as in Example 2 in Section 11.1.1. This combination enforces accuracy over the whole integration range, whereas the default formula merely ensures local accuracy. It is possible to set your own array *yscal* by seting *info* = 1 and using the reverse communication interface (*ido* = 2).

Reverse Communication Interface. The aim of the reverse communication interface is to isolate from the **ode_rkf** code the evaluation of the derivatives and possibly the construction of the scaling array. Such operations are performed by user-supplied routines, on data structures that are the most natural to the problem at hand. To this end, **ode_rkf** is called iteratively; **ode_rkf** gives control back to the calling routine whenever the evaluation of the derivatives or the scaling array is required. The reverse communication interface also allows you to reset possible zero values of *yscal* in order to avoid overflow in (2).

The reverse communication flag, *ido*, which must be 0 for the first call to **ode_rkf**, dictates which operation is to be performed. When *ido* = 1, the derivatives must be evaluated at the intermediate value *xc*; that is, you must evaluate the right-hand side of (1)

$$f_i(xc, y_1, \ldots, y_N) \qquad i = 1, \ldots, N$$

at *xc*. The source array, which contains the values of the dependent variables at *xc*, is the array ***src***. The user-supplied routine that evaluates the derivatives at *xc* must place the derivatives into the destination array, ***dst***.

When *ido* = 2, you must set up your own scaling array to be used for accuracy control. You are prompted to do so only if the input parameter *info* has been set to 1. When **ode_rkf** returns *ido* = 2, the values of the dependent variables and their derivatives to be used in the construction of the scaling array are in the arrays ***y*** and ***src***, respectively. The time step currently attempted by the integrator is *dx*. The user-supplied routine that creates the scaling array must place the scaling array into ***dst***.

Version 3.2, April 1994
Copyright © 1994 Thinking Machines Corporation

A fifth-order Runge-Kutta step requires six right-hand-side evaluations at different values of the independent variable *xc*. Note that it is the array ***src***, and not ***y***, that contains the values of the dependent variables at *xc*, to be used in the evaluation of the derivatives. The array ***y*** contains the values of the dependent variables at *x*, and is only updated once the currently attempted step *dx* is successful.

Finally, *ido* = 3 indicates that at least one component of the scaling array is zero. To avoid overflow in the error computation (2), you need only replace the zero components with small values, as in Example 1 in Section 11.1.1.

NOTES

Use of Array *w*. Do not use the CM array ***w*** as temporary workspace.

Data Layout. The **ode_rkf** routine imposes several constraints on the way the CM arrays ***y*** and ***w*** are laid out on the machine. The array ***y*** contains the dependent variables. The arrays ***src*** and ***dst*** must have the same rank, shape, and layout as ***y***. The number of dimensions of ***y*** and its layout on the machine can (and should) be chosen in such a way as to optimize the evaluation of the right-hand side of (1). In particular, ***y*** can be a multidimensional array. The product of the dimensions must be equal to the size of the problem (the number of dependent variables). Once the layout of ***y*** (and hence that of ***src*** and ***dst***) is chosen, ***w*** must adhere to the same layout directives as ***y*** except for its first extra dimension, which must be local to the processing elements and of size at least 6. Thus, ***w*** contains at least 6 arrays identical to ***y***, "stacked up" in memory. The axis indexing these arrays in ***w*** must be the first axis; to ensure that it is local, the calling program must use a layout directive to declare it **:serial**.

For example, in the one-dimensional case, the array declarations should be as follows:

```
      real y(n), w(6,n), src(n), dst(n)
CMF$LAYOUT w(:serial,),y(),src(),dst()
```

In the two-dimensional case, the declarations would be

```
      real y(n1,n2), w(6,n1,n2),src(n1,n2),dst(n1,n2)
CMF$LAYOUT w(:serial,,),y(,)src(,),dst(,)
```

In this second case, there are *n1* * *n2* dependent variables.

Copyright © 1994 Thinking Machines Corporation

EXAMPLES

Sample CM Fortran code that uses the **ode_rkf** routine can be found on-line in the subdirectory

```
ode/cmf
```

of a CMSSL examples directory whose location is site-specific.

Version 3.2, April 1994
Copyright © 1994 Thinking Machines Corporation

11.2 References

1. Press, W. H. and S. A. Teukolsky. Adaptive stepsize Runge-Kutta integration. *Computers in Physics* **6** (1992): 188−91.

2. Cash, J. R. and A. H. Karp. A variable order Runge-Kutta method for initial value problem with rapidly varying right hand sides. *ACM Trans. Math. Software* **16** (1990): 201−22.

Copyright © 1994 Thinking Machines Corporation