NAG Library Chapter Introduction

D02 – Ordinary Differential Equations

Contents

1 Scope of the Chapter

This chapter is concerned with the numerical solution of ordinary differential equations. There are two main types of problem: those in which all boundary conditions are specified at one point (initial value problems), and those in which the boundary conditions are distributed between two or more points (boundary value problems and eigenvalue problems). Routines are available for initial value problems, two-point boundary value problems and Sturm–Liouville eigenvalue problems.

2 Background to the Problems

For most of the routines in this chapter a system of ordinary differential equations must be written in the form

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, \ldots, y_n), \\ y_2' &= f_2(x, y_1, y_2, \ldots, y_n), \\ &\vdots \\ y_n' &= f_n(x, y_1, y_2, \ldots, y_n), \end{aligned}$$

that is the system must be given in first-order form. The n dependent variables (also, the solution) $y_1, y_2, \ldots, y_n$ are functions of the independent variable x, and the differential equations give expressions for the first derivatives $y_i' = \frac{dy_i}{dx}$ in terms of x and $y_1, y_2, \ldots, y_n$. For a system of n first-order equations, n associated boundary conditions are usually required to define the solution.

A more general system may contain derivatives of higher order, but such systems can almost always be reduced to the first-order form by introducing new variables. For example, suppose we have the third-order equation

$$z''' + zz'' + k\left(l - z'^2\right) = 0.$$

We write $y_1 = z$, $y_2 = z'$, $y_3 = z''$, and the third-order equation may then be written as the system of first-order equations

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ y_3' &= -y_1 y_3 - k\left(l - y_2^2\right). \end{aligned}$$

For this system $n = 3$ and we require 3 boundary conditions in order to define the solution. These conditions must specify values of the dependent variables at certain points. For example, we have an **initial value problem** if the conditions are

$$\begin{array}{ll} y_1 = 0 & \text{at } x = 0 \\ y_2 = 0 & \text{at } x = 0 \\ y_3 = 0.1 & \text{at } x = 0. \end{array}$$

These conditions would enable us to integrate the equations numerically from the point $x = 0$ to some specified end point. We have a **boundary value problem** if the conditions are

$$\begin{array}{ll} y_1 = 0 & \text{at } x = 0 \\ y_2 = 0 & \text{at } x = 0 \\ y_2 = 1 & \text{at } x = 10. \end{array}$$

These conditions would be sufficient to define a solution in the range $0 \leq x \leq 10$, but the problem could not be solved by direct integration (see Section 2.2). More general boundary conditions are permitted in the boundary value case.

It is sometimes advantageous to solve higher-order systems directly. In particular, there is an initial value routine to solve a system of second-order ordinary differential equations of the special form

$$y_1'' = f_1(x, y_1, y_2, \ldots, y_n),$$
$$y_2'' = f_2(x, y_1, y_2, \ldots, y_n),$$
$$\vdots$$
$$y_n'' = f_n(x, y_1, y_2, \ldots, y_n).$$

For this second-order system initial values of the derivatives of the dependent variables, y_i', for $i = 1, 2, \ldots, n$, are required.

There is also a boundary value routine that can treat directly a mixed order system of ordinary differential equations.

There is a broader class of initial value problems known as differential algebraic systems which can be treated. Such a system may be defined as

$$\begin{aligned} y' &= f(x, y, z) \\ 0 &= g(x, y, z) \end{aligned}$$

where y and f are vectors of length n and g and z are vectors of length m. The functions g represent the algebraic part of the system.

In addition implicit systems can also be solved, that is systems of the form

$$A(x, y)y' = f(x, y)$$

where A is a matrix of functions; such a definition can also incorporate algebraic equations. Note that general systems of this form may contain higher-order derivatives and that they can usually be transformed to first-order form, as above.

2.1 Initial Value Problems

To solve first-order systems, initial values of the dependent variables y_i, for $i = 1, 2, \ldots, n$, must be supplied at a given point, a. Also a point, b, at which the values of the dependent variables are required, must be specified. The numerical solution is then obtained by a step-by-step calculation which approximates values of the variables y_i, for $i = 1, 2, \ldots, n$, at finite intervals over the required range $[a, b]$. The routines in this chapter adjust the step length automatically to meet specified accuracy tolerances. Although the accuracy tests used are reliable over each step individually, in general an accuracy requirement cannot be guaranteed over a long range. For many problems there may be no serious accumulation of error, but for unstable systems small perturbations of the solution will often lead to rapid divergence of the calculated values from the true values. A simple check for stability is to carry out trial calculations with different tolerances; if the results differ appreciably the system is probably unstable. Over a short range, the difficulty may possibly be overcome by taking sufficiently small tolerances, but over a long range it may be better to try to reformulate the problem.

A special class of initial value problems are those for which the solutions contain rapidly decaying transient terms. Such problems are called **stiff**; an alternative way of describing them is to say that certain eigenvalues of the Jacobian matrix $\left(\dfrac{\partial f_i}{\partial y_j}\right)$ have large negative real parts when compared to others. These problems require special methods for efficient numerical solution; the methods designed for non-stiff problems when applied to stiff problems tend to be very slow, because they need small step lengths to avoid numerical instability. A full discussion is given in Hall and Watt (1976) and a discussion of the methods for stiff problems is given in Berzins *et al.* (1988).

2.2 Boundary Value Problems

In general, a system of nonlinear differential equations with boundary conditions at two or more points cannot be guaranteed to have a solution. The solution, if it exists, has to be determined iteratively. A comprehensive treatment of the numerical solution of boundary value problems can be found in Ascher *et al.* (1988) and Keller (1992). The methods for this chapter are discussed in Ascher *et al.* (1979), Ascher and Bader (1987) and Gladwell (1987).

2.2.1 Collocation methods

In the collocation method, the solution components are approximated by piecewise polynomials on a mesh. The coefficients of the polynomials form the unknowns to be computed. The approximation to the solution must satisfy the boundary conditions and the differential equations at collocation points in each mesh sub-interval. A modified Newton method is used to solve the nonlinear equations. The mesh is refined by trying to equidistribute the estimated error over the whole interval. An initial estimate of the solution across the mesh is required.

2.2.2 Shooting methods

In the shooting method, the unknown boundary values at the initial point are estimated to form an initial value problem, and the equations are then integrated to the final point. At the final point the computed solution and the known boundary conditions should be equal. The condition for equality gives a set of nonlinear equations for the estimated values, which can be solved by Newton's method or one of its variants. The iteration cannot be guaranteed to converge, but it is usually successful if

the system has a solution,

the system is not seriously unstable or very stiff for step-by-step solution, and

good initial estimates can be found for the unknown boundary conditions.

It may be necessary to simplify the problem and carry out some preliminary calculations, in order to obtain suitable starting values. A fuller discussion is given in Chapters 16, 17 and 18 of Hall and Watt (1976), Chapter 11 of Gladwell and Sayers (1980) and Chapter 8 of Gladwell (1979a).

2.2.3 Finite-difference methods

If a boundary value problem seems insoluble by the above methods and a good estimate for the solution of the problem is known at all points of the range then a finite-difference method may be used. Finite-difference equations are set up on a mesh of points and estimated values for the solution at the grid points are chosen. Using these estimated values as starting values a Newton iteration is used to solve the finite-difference equations. The accuracy of the solution is then improved by deferred corrections or the addition of points to the mesh or a combination of both. The method does not suffer from the difficulties associated with the shooting method but good initial estimates of the solution may be required in some cases and the method is unlikely to be successful when the solution varies very rapidly over short ranges. A discussion is given in Chapters 9 and 11 of Gladwell and Sayers (1980) and Chapter 4 of Gladwell (1979a).

2.3 Chebyshev Collocation for Linear Differential Equations

The collocation method gives a different approach to the solution of ordinary differential equations. It can be applied to problems of either initial value or boundary value type. Suppose the approximate solution is represented in polynomial form, say as a series of Chebyshev polynomials. The coefficients may be determined by matching the series to the boundary conditions, and making it satisfy the differential equation at a number of selected points in the range. The calculation is straightforward for linear differential equations (nonlinear equations may also be solved by an iterative technique based on linearization). The result is a set of Chebyshev coefficients, from which the solution may be evaluated at any point using E02AKF. A fuller discussion is given in Chapter 24 of Gladwell (1979a) and Chapter 11 of Gladwell and Sayers (1980).

This method can be useful for obtaining approximations to standard mathematical functions. For example, suppose we require values of the Bessel function $J_{\frac{1}{3}}(x)$ over the range $(0,5)$, for use in another calculation. We solve the Bessel differential equation by collocation and obtain the Chebyshev coefficients of the solution, which we can use to construct a function for $J_{\frac{1}{3}}(x)$. (Note that routines for many common standard functions are already available in Chapter S.)

2.4 Eigenvalue Problems

Sturm–Liouville problems of the form

$$\left(p(x)y'\right)' + q(x,\lambda)y = 0$$

with appropriate boundary conditions given at two points, can be solved by a Scaled Prüfer method. In this method the differential equation is transformed to another which can be solved for a specified eigenvalue by a shooting method. A discussion is given in Chapter 11 of Gladwell and Sayers (1980) and a complete description is given in Pryce (1986). Some more general eigenvalue problems can be solved by the methods described in Section 2.2.

3 Recommendations on Choice and Use of Available Routines

There are no routines which deal directly with COMPLEX equations. These may however be transformed to larger systems of real equations of the required form. Split each equation into its real and imaginary parts and solve for the real and imaginary parts of each component of the solution. Whilst this process doubles the size of the system and may not always be appropriate it does make available for use the full range of routines provided presently.

3.1 Initial Value Problems

In general, for non-stiff first-order systems, Runge–Kutta (RK) routines should be used. For the usual requirement of integrating across a range the appropriate routines are D02PCF and D02PVF; D02PVF is a setup routine for D02PCF. For more complex tasks there are a further five related routines: D02PDF, D02PWF, D02PXF, D02PYF and D02PZF. When a system is to be integrated over a long range or with relatively high accuracy requirements the variable-order, variable-step Adams codes may be more efficient. The appropriate routine in this case is D02CJF. For more complex tasks using an Adams code there are a further six related routines: D02QFF, D02QGF, D02QWF, D02QXF, D02QYF and D02QZF.

For stiff systems, that is those which usually contain rapidly decaying transient components, the Backward Differentiation Formula (BDF) variable-order, variable-step codes should be used. The appropriate routine in this case is D02EJF. For more complex tasks where the system residual is difficult to evaluate in forward communication, or is coupled with alebraic equations, there are a collection of routines in sub-chapter D02M–N. These routines can treat implicit differential algebraic systems and contain methods alternative to BDF techniques which may be appropriate in some circumstances.

If you are not sure how to classify a problem, you are advised to perform some preliminary calculations with D02PCF, which can indicate whether the system is stiff. We also advise performing some trial calculations with D02PCF (RK), D02CJF (Adams) and D02EJF (BDF) so as to determine which type of routine is best applied to the problem. The conclusions should be based on the computer time used and the number of evaluations of the derivative function f_i. See Gladwell (1979b) for more details.

For second-order systems of the special form described in Section 2 the Runge–Kutta–Nystrom (RKN) routine D02LAF should be used.

3.1.1 Runge–Kutta routines

The basic RK routine is D02PDF which takes one integration step at a time. An alternative is D02PCF, which provides output at user-specified points. The initialization of either D02PCF or D02PDF and the setting of optional inputs, including choice of method, is made by a call to the setup routine D02PVF. Optional output information about the integration and about error assessment, if selected, can be obtained by calls to the diagnostic routines D02PYF and D02PZF respectively. D02PXF may be used to interpolate on information produced by D02PDF to give solution and derivative values between the integration points. D02PWF may be used to reset the end of the integration range whilst integrating using D02PDF.

There is a simple driving routine D02BJF, which integrates a system over a range and, optionally, computes intermediate output and/or determines the position where a specified function of the solution is zero.

3.1.2 Adams routines

The general Adams variable-order variable-step routine is D02QFF, which provides a choice of automatic error control and the option of a sophisticated root-finding technique. Reverse communication for both the differential equation and root definition function is provided in D02QGF, which otherwise has the same facilities as D02QFF. A reverse communication routine makes a return to the calling subroutine for evaluations of equations rather than calling a user-supplied procedure. The initialization of either of

D02QFF and D02QGF and the setting of optional inputs is made by a call to the setup routine D02QWF. Optional output information about the integration and any roots detected can be obtained by calls to the diagnostic routines D02QXF and D02QYF respectively. D02QZF may be used to interpolate on information produced by D02QFF or D02QGF to give solution and derivative values between the integration points.

There is a simple driving routine D02CJF, which integrates a system over a range and, optionally, computes intermediate output and/or determines the position where a specified function of the solution is zero.

3.1.3 BDF routines

General routines for explicit and implicit ordinary differential equations with a wide range of options for integrator choice and special forms of numerical linear algebra are provided in sub-chapter D02M–N. A separate document describing the use of this sub-chapter is given immediately before the routines of the sub-chapter.

There is a simple driving routine D02EJF, which integrates a system over a range and, optionally, computes intermediate output and/or determines the position where a specified function of the solution is zero. It has a specification similar to the Adams routine D02CJF except that to solve the equations arising in the BDF method an approximation to the Jacobian $\left(\frac{\partial f_i}{\partial y_j}\right)$ is required. This approximation can be calculated internally but you may supply an analytic expression. In most cases supplying a correct analytic expression will reduce the amount of computer time used.

3.1.4 Runge–Kutta–Nystrom routines

The Runge–Kutta–Nystrom routine D02LAF uses either a low- or high-order method (chosen by you). The choice of method and error control and the setting of optional inputs is made by a call to the setup routine D02LXF. Optional output information about the integration can be obtained by a call to the diagnostic routine D02LYF. When the low-order method has been employed D02LZF may be used to interpolate on information produced by D02LAF to give the solution and derivative values between the integration points.

3.2 Boundary Value Problems

In general, for a nonlinear system of mixed order with separated boundary conditions, the collocation method (D02TKF and its associated routines) can be used. Problems of a more general nature can often be transformed into a suitable form for treatment by D02TKF, for example nonseparated boundary conditions or problems with unknown parameters (see Section 8 in D02TVF for details).

For simple boundary value problems with assigned boundary values you may prefer to use a code based on the shooting method or finite difference method for which there are routines with simple calling sequences (D02GAF and D02HAF).

For difficult boundary value problems, where you need to exercise some control over the calculation, and where the collocation method proves unsuccessful, you may wish to try the alternative methods of shooting (D02SAF) or finite-differences (D02RAF).

Note that it is not possible to make a fully automatic boundary value routine, and you should be prepared to experiment with different starting values or a different routine if the problem is at all difficult.

3.2.1 Collocation methods

The collocation routine D02TKF solves a nonlinear system of mixed order boundary value problems with separated boundary conditions. The initial mesh and accuracy requirements must be specified by a call to the setup routine D02TVF. Optional output information about the final mesh and estimated maximum error can be obtained by a call to the diagnostic routine D02TZF. The solution anywhere on the mesh can be computed by a call to the interpolation routine D02TYF. If D02TKF is being used to solve a sequence of related problems then the continuation routine D02TXF should also be used.

3.2.2 Shooting methods

D02HAF may be used for simple boundary value problems, where the unknown parameters are the missing boundary conditions. More general boundary value problems may be handled by using D02HBF. This routine allows for a generalized parameter structure, and is fairly complicated. The older routine D02AGF has been retained for use when an interior matching-point is essential; otherwise the newer routine D02HBF should be preferred.

For particularly complicated problems where, for example, the parameters must be constrained or the range of integration must be split to enable the shooting method to succeed, the recommended routine is D02SAF, which extends the facilities provided by D02HBF. If you are a sophisticated user D02SAF permits you much more control over the calculation than does D02HBF; in particular you are permitted precise control of solution output and intermediate monitoring information.

3.2.3 Finite-difference methods

D02GAF may be used for simple boundary value problems with assigned boundary values. The calling sequence of D02GAF is very similar to that for D02HAF discussed above.

You may find that convergence is difficult to achieve using D02GAF since only specifying the unknown boundary values and the position of the finite-difference mesh is permitted. In such cases you may use D02RAF, which permits specification of an initial estimate for the solution at all mesh points and allows the calculation to be influenced in other ways too. D02RAF is designed to solve a general nonlinear two-point boundary value problem with nonlinear boundary conditions.

A routine, D02GBF, is also supplied specifically for the general linear two-point boundary value problem written in a standard ‘textbook’ form.

You are advised to use interpolation routines from Chapter E01 to obtain solution values at points not on the final mesh.

3.3 Chebyshev Collocation Method

D02TGF may be used to obtain the approximate solution of a system of differential equations in the form of a Chebyshev series. The routine treats linear differential equations directly, and makes no distinction between initial value and boundary value problems. This routine is appropriate for problems where it is known that the solution is smooth and well-behaved over the range, so that each component can be represented by a single polynomial. Singular problems can be solved using D02TGF as long as their polynomial-like solutions are required.

D02TGF permits the differential equations to be specified in higher order form; that is without conversion to a first-order system. This type of specification leads to a complicated calling sequence. If you are an inexperienced user two simpler routines are supplied. D02JAF solves a single regular linear differential equation of any order whereas D02JBF solves a system of regular linear first-order differential equations.

3.4 Eigenvalue Problems

Two routines, D02KAF and D02KDF, may be used to find the eigenvalues of second-order Sturm–Liouville problems. D02KAF is designed to solve simple problems with regular boundary conditions. D02KAF calls D02KDF, which is designed to solve more difficult problems, for example with singular boundary conditions or on infinite ranges or with discontinuous coefficients.

If the eigenfunctions of the Sturm–Liouville problem are also required, D02KEF should be used. (D02KEF solves the same types of problem as D02KDF.)

3.5 Summary of Recommended Routines

Problem	**Routine**		
	RK Method	Adams Method	BDF Method
Initial-value Problems **Driver Routines**			
Integration over a range with optional intermediate output and optional determination of position where a function of the solution becomes zero	D02BJF	D02CJF	D02EJF
Integration of a range with intermediate output	D02BJF	D02CJF	D02EJF
Integration of a range until function of solution becomes zero	D02BJF	D02CJF	D02EJF
Comprehensive Integration Routines	D02PCF, D02PDF, D02PVF, D02PWF, D02PXF, D02PYF and D02PZF	D02QFF, D02QGF, D02QWF, D02QXF, D02QYF and D02QZF	sub-chapter D02M–N routines, D02XJF, D02XKF and D02ZAF
Package for Solving Stiff Equations	sub-chapter D02M–N routines		
Package for Solving Second-order Systems of Special Form	D02L routines		
Boundary value Problems **Collocation Method, Mixed Order**	D02TKF, D02TVF, D02TXF, D02TYF and D02TZF		
Boundary value Problems **Shooting Method**			
simple parameter	D02HAF		
generalized parameters	D02AGF and D02HBF		
additional facilities	D02SAF		
Boundary value Problems **Finite-difference Method**			
simple parameter	D02GAF		
linear problem	D02GBF		
full nonlinear problem	D02RAF		
Chebyshev Collocation, Linear Problems			
single equation	D02JAF		
first-order system	D02JBF		
general system	D02TGF		
Sturm–Liouville Eigenvalue Problems			
regular problems	D02KAF		
general problems	D02KDF		
eigenfunction calculation	D02KEF		

4 Decision Trees

Tree 1: Initial Value Problems

Is the problem first order?
— yes → Is the problem known to be stiff?
— yes → Use routines described in the D02M–N sub-chapter Introduction.
— no → Backward Difference Formula: Use routines described in the D02M–N sub-chapter Introduction.
Adams method with driver routine: D02CJF
Adams method with comprehensive suite: D02QFF, D02QGF, D02QWF, D02QXF, D02QYF and D02QZF
Runge–Kutta method with driver routine: D02BJF
Runge–Kutta method with comprehensive suite: D02PCF, D02PDF, D02PVF, D02PWF, D02PXF and D02PYF

no

Is the problem of the form: $y'' = f(x, y)$
— yes → Use the D02L routines

no

Convert to first order problem: is the problem known to be stiff?
— yes → Use routines described in the D02M–N sub-chapter Introduction.

no

Backward Difference Formula: use routines described in the D02M–N sub-chapter Introduction.
Adams method with driver routine: D02CJF
Adams method with comprehensive suite: D02QFF, D02QGF, D02QWF, D02QXF, D02QYF and D02QZF
Runge–Kutta method with driver routine: D02BJF
Runge–Kutta method with comprehensive suite: D02PCF, D02PDF, D02PVF, D02PWF, D02PXF and D02PYF

Tree 2: Boundary Value Problems

Is the problem simply of the form $y' = f(x, y)$?
— yes → Are only boundary values to be determined?
— yes → Shooting method: D02HAF
Finite differences: D02GAF
Collocation: D02TKF
— no → Shooting method: D02HBF
Finite differences: D02GBF
Collocation: D02TKF

no

Shooting method: D02SAF
Finite differences: D02RAF
Collocation: D02TKF

5 Index

6 Routines Withdrawn or Scheduled for Withdrawal

Withdrawn Routine	**Mark of Withdrawal**	**Replacement Routine(s)**
D02BAF	18	D02PCF and associated D02P routines
D02BBF	18	D02PCF and associated D02P routines
D02BDF	18	D02PCF and associated D02P routines
D02CAF	18	D02CJF
D02CBF	18	D02CJF
D02CGF	18	D02CJF
D02CHF	18	D02CJF
D02EAF	18	D02EJF
D02EBF	18	D02EJF
D02EGF	18	D02EJF
D02EHF	18	D02EJF
D02PAF	18	D02PDF and associated D02P routines
D02QDF	17	D02NBF or D02NCF
D02QQF	17	D02NBF or D02NCF
D02XAF	18	D02PXF and associated D02P routines
D02XBF	18	D02PXF and associated D02P routines
D02YAF	18	D02PDF and associated D02P routines

7 References

Ascher U M and Bader G (1987) A new basis implementation for a mixed order boundary value ODE solver *SIAM J. Sci. Stat. Comput.* **8** 483–500

Ascher U M, Christiansen J and Russell R D (1979) A collocation solver for mixed order systems of boundary value problems *Math. Comput.* **33** 659–679

Ascher U M, Mattheij R M M and Russell R D (1988) *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* Prentice–Hall

Berzins M, Brankin R W and Gladwell I (1988) Design of the stiff integrators in the NAG Library *SIGNUM Newsl.* **23** 16–23

Gladwell I (1979a) The development of the boundary value codes in the ordinary differential equations chapter of the NAG Library *Codes for Boundary Value Problems in Ordinary Differential Equations. Lecture Notes in Computer Science* (eds B Childs, M Scott, J W Daniel, E Denman and P Nelson) **76** Springer–Verlag

Gladwell I (1979b) Initial value routines in the NAG Library *ACM Trans. Math. Software* **5** 386–400

Gladwell I (1987) The NAG Library boundary value codes *Numerical Analysis Report* **134** Manchester University

Gladwell I and Sayers D K (ed.) (1980) *Computational Techniques for Ordinary Differential Equations* Academic Press

Hall G and Watt J M (ed.) (1976) *Modern Numerical Methods for Ordinary Differential Equations* Clarendon Press, Oxford

Keller H B (1992) *Numerical Methods for Two-point Boundary-value Problems* Dover, New York

Pryce J D (1986) Error estimation for phase-function shooting methods for Sturm–Liouville problems *IMA J. Numer. Anal.* **6** 103–123