# NAG Toolbox for MATLAB

# 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). Functions 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 functions 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

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

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

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

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 function to solve a system of second-order ordinary differential equations of the special form

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

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 function 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 functions 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 e02ak. 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 functions for many common standard functions are already available in Chapter S.)

## 2.4 Eigenvalue Problems

Sturm–Liouville problems of the form

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

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 Functions

There are no functions 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 functions provided presently.

## 3.1 Initial Value Problems

In general, for non-stiff first-order systems, Runge–Kutta (RK) functions should be used. For the usual requirement of integrating across a range the appropriate functions are d02pv and d02pc; d02pv is a setup function for d02pc. For more complex tasks there are a further five related functions: d02pd, d02pw, d02px, d02py and d02pz. 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 function in this case is d02cj. For more complex tasks using an Adams code there are a further six related functions: d02qf, d02qg, d02qx, d02qw, d02qy and d02qz.

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 function in this case is d02ej. For more complex tasks using a BDF code there are a collection of functions in sub-chapter D02M/N. These functions 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 d02pc, which can indicate whether the system is stiff. We also advise performing some trial calculations with d02pc (RK), d02cj (Adams) and d02ej (BDF) so as to determine which type of function 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) function d02la should be used.

### 3.1.1 Runge–Kutta functions

The basic RK function is d02pd which takes one integration step at a time. An alternative is d02pc, which provides output at user-specified points. The initialization of either d02pc or d02pd and the setting of optional inputs, including choice of method, is made by a call to the setup function d02pv. Optional output information about the integration and about error assessment, if selected, can be obtained by calls to the diagnostic functions d02py and d02pz respectively. d02px may be used to interpolate on information produced by d02pd to give solution and derivative values between the integration points. d02pw may be used to reset the end of the integration range whilst integrating using d02pd.

There is a simple driving function d02bj, 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 functions

The general Adams variable-order variable-step function is d02qf, 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 d02qg, which otherwise has the same facilities as d02qf. A reverse communication function makes a return to the calling (sub)program for evaluations of equations rather than calling a user-supplied procedure. The initialization of either of d02qf and d02qg and the setting of optional inputs is made by a call to the setup function d02qw. Optional output information about the integration and any roots detected can be obtained by calls to the diagnostic

functions d02qx and d02qy respectively. d02qz may be used to interpolate on information produced by d02qf or d02qg to give solution and derivative values between the integration points.

There is a simple driving function d02cj, 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 functions

General functions 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 functions of the sub-chapter.

There is a simple driving function d02ej, 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 function d02cj 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 functions

The Runge–Kutta–Nystrom function d02la 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 function d02lx. Optional output information about the integration can be obtained by a call to the diagnostic function d02ly. When the low-order method has been employed d02lz may be used to interpolate on information produced by d02la 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 (d02tk and its associated functions) can be used. Problems of a more general nature can often be transformed into a suitable form for treatment by d02tk, for example nonseparated boundary conditions or problems with unknown parameters (see Section 8 of the document for d02tv 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 functions with simple calling sequences (d02ha and d02ga).

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 (d02sa) or finite-differences (d02ra).

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

### 3.2.1 Collocation methods

The collocation function d02tk 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 function d02tv. Optional output information about the final mesh and estimated maximum error can be obtained by a call to the diagnostic function d02tz. The solution anywhere on the mesh can be computed by a call to the interpolation function d02ty. If d02tk is being used to solve a sequence of related problems then the continuation function d02tx should also be used.

### 3.2.2 Shooting methods

d02ha 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 d02hb. This function allows for a generalized parameter structure, and is fairly complicated. The older function d02ag

has been retained for use when an interior matching-point is essential; otherwise the newer function d02hb 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 function is d02sa, which extends the facilities provided by d02hb. If you are a sophisticated user d02sa permits you much more control over the calculation than does d02hb; in particular you are permitted precise control of solution output and intermediate monitoring information.

#### 3.2.3 Finite-difference methods

d02ga may be used for simple boundary-value problems with assigned boundary-values. The calling sequence of d02ga is very similar to that for d02ha discussed above.

You may find that convergence is difficult to achieve using d02ga since only specifying the unknown boundary-values and the position of the finite-difference mesh is permitted. In such cases you may use d02ra, 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. d02ra is designed to solve a general nonlinear two-point boundary-value problem with nonlinear boundary conditions.

A function, d02gb, 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 functions from Chapter E01 to obtain solution values at points not on the final mesh.

### 3.3 Chebyshev Collocation Method

d02tg may be used to obtain the approximate solution of a system of differential equations in the form of a Chebyshev-series. The function treats linear differential equations directly, and makes no distinction between initial value and boundary-value problems. This function 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 d02tg as long as their polynomial-like solutions are required.

d02tg 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 functions are supplied. d02ja solves a single regular linear differential equation of any order whereas d02jb solves a system of regular linear first-order differential equations.

### 3.4 Eigenvalue Problems

Two functions, d02ka and d02kd, may be used to find the eigenvalues of second-order Sturm–Liouville problems. d02ka is designed to solve simple problems with regular boundary conditions. d02ka calls d02kd, 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, d02ke should be used. (d02ke solves the same types of problem as d02kd.)

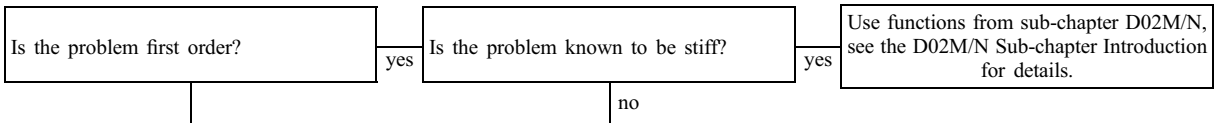
### 3.5 Summary of Recommended Functions

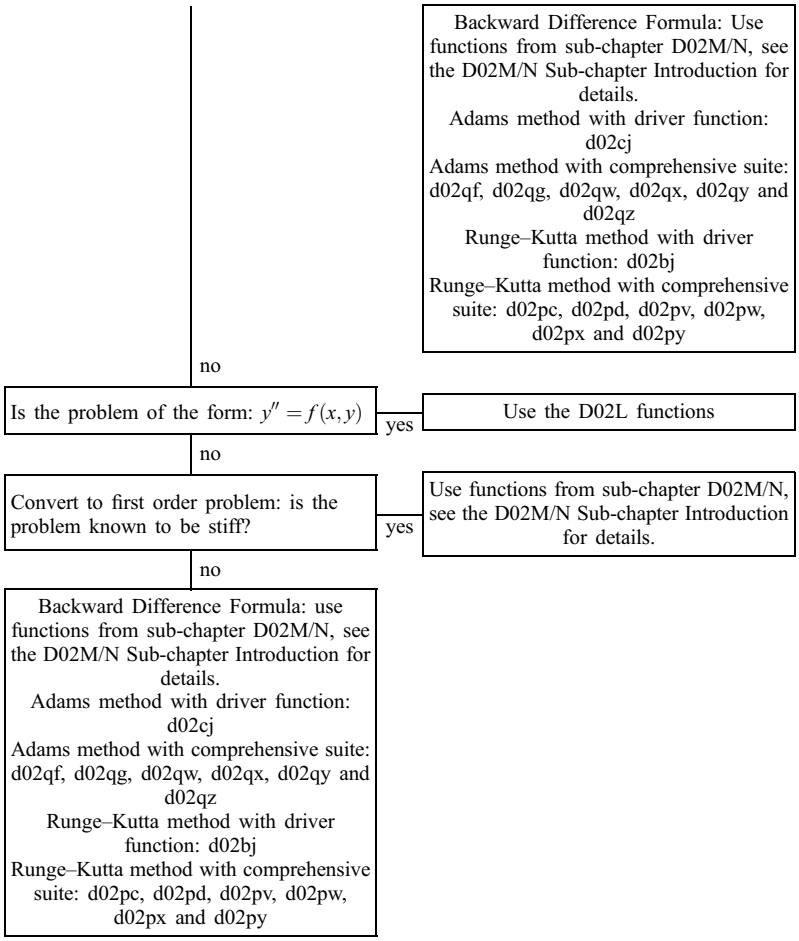
| **Problem** | |
|---|---|
| | |
| **Initial-value Problems**<br>**Driver Functions** | |
| Integration over a range with optional intermediate output and optional determination of position where a function of the solution becomes zero | |

| | |
|---|---|
| Integration of a range with intermediate output | |
| Integration of a range until function of solution becomes zero | |
| **Comprehensive Integration Functions** | d02pc, d02p |
| **Package for Solving Stiff Equations** | |
| **Package for Solving Second-order Systems of Special Form** | |
| **Boundary-value Problems**<br>**Collocation Method, Mixed Order** | |
| **Boundary-value Problems**<br>**Shooting Method** | |
| simple parameter | |
| generalized parameters | |
| additional facilities | |
| **Boundary-value Problems**<br>**Finite-difference Method** | |
| simple parameter | |
| linear problem | |
| full nonlinear problem | |
| **Chebyshev Collocation, Linear Problems** | |
| single equation | |
| first-order system | |
| general system | |
| **Sturm–Liouville Eigenvalue Problems** | |
| regular problems | |
| general problems | |
| eigenfunction calculation | |

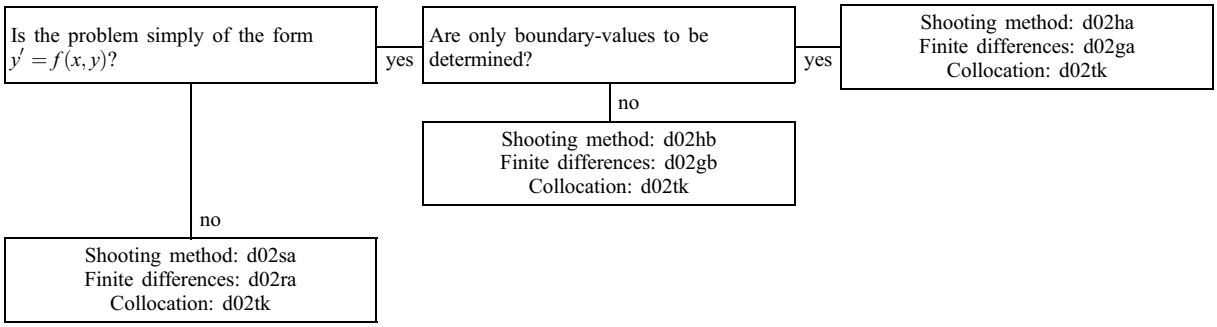
# 4 Decision Trees

**Tree 1: Initial Value Problems**






**Tree 2: Boundary Value Problems**




# 5 Index

## 6 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* (ed 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