BOUNDPAK : numerical software for linear boundary value problems

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

BOUNDPAK

Numerical Software for Linear Boundary Value Problems Part One

by

R.M.M. Mattheij and G.W.M. Staarink

EUT Report 92-WSK-01
Eindhoven, February 1992

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven, The Netherlands
ISBN 9038600224
ISSN 0167-9708
Coden: TEUEDE

CONTENTS

PART ONE

PART TWO

PREFACE

The work on the routines in this booklet started some years ago when a number of codes were written to study BVP phenomena. Due to interest from other users we updated and diversified these codes time and again. As a result their descriptions became more detailed. When we realized that the specialisation to certain subclasses of BVPs was gradually producing an entire family of problems oriented codes, the idea was born to collect their description, supplemented with some mathematical analysis. From the foregoing it follows that the present volume is not a course book; it rather contains mathematical backgrounds and computational details of a number of algorithms for solving linear boundary value problems. These algorithms are based on a special implementation of a multiple shooting approach (although the name sequential shooting would be more appropriate). Their important common feature is that they employ a special stable linear algebra solver, based on a decoupling of the multiple shooting recursion. These methods have been found to be at least as robust and efficient as other (sparse) solvers. In fact for some special cases (like multipoint problems) they are more efficient. Therefore we have devoted a separate chapter to these linear solvers, describing routines that can be used in combination with other (discretization) methods as well.

We are well aware of the fact that often problems are nonlinear rather than linear. However, the mathematical descriptions and the codes treated in this book can be used almost directly in a quasilinearization approach. On the other hand, for nonlinear BVPs a similar diverse range of subproblems can be distinguished. The ideas given in the various chapters may be a source of inspiration for implementing nonlinear counterparts.

We like to say a word about the philosophy of this package: Although it is often possible to reformulate various classes of BVPs into a standard form (we give some hints how to achieve this), such a formulation often leads to more costly computations than are necessary. Moreover, as it will turn out, special problems have special characteristics: for instance, dichotomy, that plays such a crucial role in any well-conditioned two-point BVP may lose its meaning in a multipoint BVP. For certain applications one is often interested in the specific problem characteristics (like estimates for the fundamental solution or the Green's function). Our package makes such information available. We also strongly believe in the idea that a code should provide as much additional information as possible in order to enable the user to give a meaningful diagnosis. At minor points therefore we have traded efficiency for robustness. Consequently we make a distinction between various two-point boundary conditions, between two-point and multipoint problems and between finite and infinite intervals. Special attention is being paid to ODEs with parameters and BVPs with jump conditions (where, incidentally, multiple shooting is a natural approach, requiring not even continuity at a shooting point). Finally we also consider eigenvalue problems.

Eindhoven,
February 1992

R.M.M. Mattheij
G.W.M. Staarink

CHAPTER I

INTRODUCTION

1. ODEs, BCs and BVPs

In this chapter we give a brief overview of the various types of boundary value problems which will be discussed later. We also include a general introduction to the solution methods on which the algorithms in the next chapters are based.

Consider the following ordinary differential equation (ODE) :

(1.1) $$\frac{d}{dt}x(t) = L(t)x(t) + r(t) \quad , \alpha \le t \le \beta ,$$

where $L(t)$ is an $n \times n$-matrix function (assumed to be sufficiently smooth in our aplications) and $x(t)$, $r(t)$ n-vector functions. Sometimes we shall have to consider the *homogeneous* case ($r(t) \equiv 0$) separately.
The solution $x(t)$ is subject to a *boundary condition* (BC). In its most general form we have a *multipoint* BC,

(1.2) $$\sum_{j=1}^{m+1} M_j \, x(\alpha_j) = b \ ,$$

where $M_1, \ldots, M_{m+1}$ are $n \times n$-matrices, b is an n-vector and $\alpha_1, \ldots, \alpha_{m+1} \in [\alpha, \beta]$ are ordered, such that $\alpha = \alpha_1 < \alpha_2 < \cdots < \alpha_{m+1} = \beta$.
A problem (1.1), (1.2) is called a (linear) *boundary value problem* (BVP). Most often we have $m = 1$, i.e. a two-point BC, which we usually shall write as

(1.3) $$M_\alpha x(\alpha) + M_\beta x(\beta) = b \ .$$

In CHAPTER II we shall discuss methods for BVPs with two-point BC; as it will turn out, situations where M_α and/or M_β have some zero rows allow for a particular, more efficient, treatment.
A somewhat different situation occurs when $\beta = \infty$. For such BVPs on *infinite intervals* we have to truncate the interval to a finite one in a deliberate way; moreover, the terminal condition matrix M_β is often absent, thus leading to a "conditional" initial value problem. This is discussed in CHAPTER III.

For a genuine *multipoint case* (i.e. $m \geq 2$) the BVP may have inherently different properties, which calls for a special treatment. As a special limiting case of (1.2) we consider an *integral condition* of the form

(1.4) $$\int_{\alpha}^{\beta} M(\tau)x(\tau)d\tau = b \ .$$

Multipoint and integral BC are considered in CHAPTER IV.
Sometimes the solution should obey certain relations which we may collectively indicate as singular. A singularity in the ODE is usually treated by analytical means and so does not have a special treatment here.
If we have at a certain point γ

(1.5) $$x(\gamma^{+}) = x(\gamma^{-}) + c \ ,$$

where $x(\gamma^{+})$ and $x(\gamma^{-})$ have to be understood as right and left limits, we obviously meet a problem at γ. Such *side conditions* (and more general ones) are dealt with in CHAPTER VI.
For yet another type of problem we may let the ODE and/or the BC depend on some *parameters*, which are either supplemented by sufficient additional BCs, or are to be chosen such that the solution of the BVP is unique, apart from a multiplicative constant, to mention the simplest case of an *eigenvalue problem:*

(1.6) $$\frac{d}{dt}x(t) = L(t)x(t) + \lambda x(t) \ .$$

Here the BC (1.3) is assumed to be homogeneous, i.e. $b = 0$; see CHAPTER VII.
If

(1.7) $$\frac{d}{dt}x(t) = L(t)x(t) + K(t)\lambda + r(t) \ ,$$

where $K(t)$ is an $n \times l$-matrix function and λ a fixed l-vector, we have a so called *parameter problem.* For the l unknown parameters we need l additional BCs. Such problems are considered in CHAPTER V.

Several of the routines that are developed to solve the various BVPs are useful in their own right. In particular this holds true for the linear algebra routines. We have adapted some of them in such a way that they can be used to solve certain sparse systems. In CHAPTER VIII we shall indicate more precisely which kind.

In the introduction of each chapter an explicit reference is being made to the appropriate routines.
The documentation of these routines, in particular their parameter list, the table of error messages and an example, to demonstrate their use, is given in CHAPTER IX.

2. Notational conventions

For efficiency's sake we briefly review here some conventions that will be used throughout the book.
We shall frequently meet partitioned matrices. As we shall also meet recursions, we adopt the convention that subscripts denote the iteration index, as in

(2.1) $$x_{i+1} = A_i x_i + d_i \ .$$

Superscripts exclusively refer to partitioning, as in

(2.2) $$A = \begin{bmatrix} A^{11} & A^{12} \\ A^{21} & A^{22} \end{bmatrix} ,$$

where A^{11}, A^{22} are assumed to be square blocks; trivially, when the order of A^{11} is given, the sizes of the other blocks are determined. Corresponding to a matrix partitioning (2.2), we can have a vector partitioning. Let A_i^{11} be a $k \times k$ matrix (A_i as in (2.2)) then in

(2.3) $$x_i = \begin{bmatrix} x_i^1 \\ x_i^2 \end{bmatrix} ,$$

x_i^1 is assumed to have k coordinates. This induces a consistent partitioning in the recursion (2.1)

(2.4a) $$x_{i+1}^1 = A_i^{11} x_i^1 + A_i^{12} x_i^2 + d_i^1 \ ,$$

(2.4b) $$x_{i+1}^2 = A_i^{21} x_i^1 + A_i^{22} x_i^2 + d_i^2 \ .$$

For matrices we also use the following partitioning

(2.5a) $$A = [A^1 | A^2] ,$$

to indicate a partitioning into *columns* and

(2.5b) $$A = \begin{bmatrix} {}^1A \\ {}^2A \end{bmatrix} ,$$

to indicate a partitioning into *rows*.

Because of their favourable numerical properties we use orthogonal matrices as much as possible. Three important matrix factorizations are used throughout:

(2.6) $\quad A = Q\,U$,

where Q is an orthogonal matrix and U is an upper triangular matrix (*Gram-Schmidt* or *QU-factorization* cf. [2]);

(2.7) $\quad A = U\,Q$,

where U is an upper triangular matrix and Q an orthogonal matrix (*UQ-factorization*); and

(2.8) $\quad A = U\,\Sigma V^T$,

where U and V are orthogonal matrices and Σ a diagonal matrix with semi-positive diagonal elements (*singular value decomposition* , cf. [2]).

Regularly we shall use norms to measure matrices and vectors, i.e. $\|A\|$ and $\|x\|$ for a matrix A and a vector x, respectively. Usually one may use any norm for this, but sometimes we give preference to the maximum norm (∞-norm) as this is easy to compute, or to the Euclidean norm (2-norm) because of its orthogonal invariance, i.e. for orthogonal Q_1, Q_2,

(2.9) $\quad \| Q_1 A\, Q_2\|_2 = \| A \|_2, \ \| Q_1 x \|_2 = \| x \|_2$.

3. General description of (multiple) shooting and decoupling

The algorithms that will be described in the subsequent chapters are all based on a special implementation of two basic methods: *multiple shooting* and *decoupling*. We shall briefly outline these methods here.
For the ODE (1.1) let a two-point BC

(3.1) $\quad M_\alpha x(\alpha) + M_\beta x(\beta) = b$,

be given (cf. (1.3)) (for multipoint BC the derivation is similar, though more complicated, cf. chapter IV).
Let $F(t)$ be a *fundamental solution* of (1.1) (i.e. an $n \times n$-matrix solution of the homogeneous part of (1.1)) and $w(t)$ some *particular solution* of (1.1). Then because of the linearity, we may find the solution $x(t)$ by *superposition*. That is: there exists some (unknown) vector c, such that

(3.2) $\quad x(t) = F(t)\,c + w(t)$.

This c is (uniquely) determined by the BC (3.1), i.e.

$$(3.3)\qquad [M_\alpha F(\alpha)+M_\beta F(\beta)]\,c = b - M_\alpha w(\alpha) - M_\beta w(\beta).$$

A natural way to determine $F(t)$ and $w(t)$ is starting at $t=\alpha$ (with an arbitrary, or (preferably) simple looking initial value) and using an initial value integrator. An algorithm based on this principle is called *single shooting*. This method is notorious for giving bad results for problems where, say, $\exp((\beta-\alpha)\max\|L(t)\|)$ is large. By applying a superposition idea repeatedly on subsequent subintervals we obtain a *multiple shooting method*: Let $[\alpha,\beta]$ be divided into $N-1$ subintervals, say, $[t_i, t_{i+1}]$, $i=1,\ldots,N-1$ ($t_1=\alpha$, $t_N=\beta$). On each subinterval a fundamental solution $F_i(t)$ and a particular solution $w_i(t)$ is computed (often $w_i(t_i)=0$). So for some vectors c_i we have

$$(3.4)\qquad x(t)=F_i(t)\,c_i + w_i(t)\,,\ i=1,\ldots,N\,.$$

Here we have added the solutions $F_N(t)$ and $w_N(t)$ for esthetic reasons.
By requiring continuity at the *shooting points* t_i (a condition that might be relaxed in certain applications cf. chapter VI) we obtain a recurrence relation for the c_i :

$$(3.5)\qquad F_i(t_{i+1})\,c_i + w_i(t_{i+1}) = F_{i+1}(t_{i+1})\,c_{i+1} + w_{i+1}(t_{i+1})\,.$$

Together with the relation obtained from the BC (3.1),viz.

$$(3.6)\qquad M_\alpha F_1(t_1)\,c_1 + M_\beta F_N(t_N)\,c_N = b - M_\alpha w_1(t_1) - M_\beta w_N(t_N)\,,$$

this gives rise to N linear equations for the unknown $c_1,\ldots,c_N$.

Although multiple shooting seems to be more complicated than single shooting, the initial value instability is exponentially reduced by the length of the (maximal) subinterval (i.e. errors are expected to grow by not more than a factor $\exp((t_{i+1}-t_i)\max\limits_{t\in[t_i,t_{i+1}]}\|L(t)\|)$ on $[t_i, t_{i+1}]$).

The *discrete* BVP (3.5), (3.6) leads to the following linear system:

$$(3.7)\qquad \begin{bmatrix} F_1(t_2) & -F_2(t_2) & & \\ & \ddots & \ddots & \\ & & F_{N-1}(t_N) & -F_N(t_N) \\ M_\alpha F_1(t_1) & & & M_\beta F_N(t_N) \end{bmatrix} \mathbf{c} = \mathbf{f}$$

with $\quad \mathbf{c}=[c_1^T,\ldots,c_N^T]^T$, $\mathbf{f}=[f_1^T,\ldots,f_N^T]^T$ and where
$f_i = w_{i+1}(t_{i+1}) - w_i(t_{i+1})$, $\quad i=1,\ldots,N-1$,
$f_N = b - M_\alpha w_1(t_1) - M_\beta w_N(t_N)$.

The solution of this system can be obtained by using any general linear algebraic solver. However the sparsity of this system requires a special treatment for efficiency reasons. Therefore we shall describe a method which solves such a discrete BVP by decoupling; we shall do this for the formulation (3.5), (3.6): Let e.g. $F_i(t_i)=I$ and write $A_i := F_i(t_{i+1})$, then

(3.8a) $\quad c_{i+1} = A_i c_i - f_i \quad , i=1, \ldots , N-1 ,$

(3.8b) $\quad M_1 c_1 + M_N c_N = f_N .$

Further,let T_1 be an orthogonal matrix. Then compute recursively for $i=1, \ldots , N-1$

(3.9a) $\quad \hat{A}_i := A_i T_i \quad ,$

(3.9b) $\quad \hat{A}_i =: T_{i+1} U_{i+1} \quad ,$

where T_{i+1} is an orthogonal and U_{i+1} an upper triangular matrix (i.e. (3.9b) is a QU-decomposition). By defining

(3.10a) $\quad a_i := T_i^{-1} c_i \qquad , i=1, \ldots , N ,$

(3.10b) $\quad d_i := T_{i+1}^{-1} f_i \qquad , i=1, \ldots , N-1 ,$

(3.10c) $\quad \hat{M}_i := M_i T_i \qquad , i=1,N ,$

we obtain the *decoupled* recursion

(3.11) $\quad a_{i+1} = U_{i+1} a_i + d_i \ ,$

where a_i satisfies the BC

(3.12) $\quad \hat{M}_1 a_1 + \hat{M}_N a_N = f_N \ .$

For well-conditioned problems, it can be shown that the solution space S (of the homogeneous problem) is *dichotomic* , i.e. there exists a subspace S_1 (of dimension k say) of solutions that do *not increase* significantly for *decreasing t* and a complementary subspace S_2 (of dimension $n-k$) of solutions that do *not increase* significantly for *increasing t*; in fact both subspaces may contain exponentially growing modes and in particular the exponentially growing modes (for increasing t) of the first subspace may cause instabilities for (single) shooting. Avoiding technical details, it can be shown that the dichotomy has visible effects on the decoupled recursion matrices U_i. Under fairly general conditions (dealing with the choice of T_1) the $k \times k$ left upper blocks in the U_i reflect the incremental growth of the modes $\in S_1$ and the $(n-k)\times(n-k)$ right lower blocks the growth of modes $\in S_2$. One may compare this idea with probably more familiar results in power methods, where the A_i are constant. The algorithm (3.9) then is essentially equivalent to subspace iteration (a predecessor of the QR algorithm without shifts). Partitioning U_i, a_i and d_i as

(3.13) $$U_i = \begin{bmatrix} B_i & C_i \\ \varnothing & E_i \end{bmatrix} , \quad a_i = \begin{bmatrix} a_i^1 \\ a_i^2 \end{bmatrix} , \quad d_i = \begin{bmatrix} d_i^1 \\ d_i^2 \end{bmatrix} ,$$

respectively, we can write

(3.14a) $\quad a_{i+1}^2 = E_{i+1}\, a_i^2 + d_{i+1}^2$,

(3.14b) $\quad a_{i+1}^1 = B_{i+1}\, a_i^1 + C_{i+1}\, a_i^2 + d_{i+1}^1$.

Because of the said properties of E_i, it should be expected that (3.14a) is a stable recursion, i.e. if a_1^2 is given no significant error growth in the a_i^2 will be present. On the other hand, (3.14b) will be stable given a value of a_N^1 (and assuming $a_{N-1}^2, \ldots, a_1^2$ are known, so they just add to the source term d_{i+1}^1). This combination of forward and backward sweeps in appropriate directions is then used to stably compute both some fundamental solution of (3.14) and some particular solution. These are used in turn with a superposition principle in (3.12), after which the c_i essentially follow from (3.10a).

References

[1] U.M. Ascher, R.M.M. Mattheij, R.D. Russell, *Numerical Solutions of Boundary Value Problems for ODE*, Prentice-Hall, Englewood Cliffs, 1987.

[2] G.H. Golub, C.F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1983.

[3] R.M.M. Mattheij, *Stable computation of solutions of unstable initial value recursions*, BIT 22 (1982), 79-83.

[4] R.M.M. Mattheij, G.W.M. Staarink, *An efficient algorithm for solving general linear two-point BVP*, SIAM J. Sci. Stat. Comput. 5 (1984), 745-763.

CHAPTER II

TWO-POINT BVP

1. Introduction

Consider the ODE

$$(1.1) \qquad \frac{d}{dt}x(t) = L(t)x(t) + r(t)\,, \quad \alpha \le t \le \beta$$

and the two-point BC

$$(1.2) \qquad M_\alpha x(\alpha) + M_\beta x(\beta) = b\,.$$

The algorithm combines multiple shooting with decoupling (cf. §I.3). In particular it computes the fundamental solutions *sequentially* by choosing them such that

$$(1.3) \qquad F_i(t_{i+1}) = F_{i+1}(t_{i+1})\,U_{i+1} = Q_{i+1}\,U_{i+1}\,,$$

where Q_{i+1} is an orthogonal and U_{i+1} an upper triangular matrix.
On the subinterval $[\,t_i\,, t_{i+1}\,]$ we have

$$(1.4) \qquad x(t) = F_i(t)\,a_i + w_i(t).$$

Matching at the endpoints of the subintervals leads to

$$(1.5) \qquad F_i(t_{i+1})\,a_i + w_i(t_{i+1}) = F_{i+1}(t_{i+1})\,a_{i+1} + w_{i+1}(t_{i+1})\,,$$

which results into the recursion

$$(1.6) \qquad a_{i+1} = U_{i+1}\,a_i + Q_{i+1}^{-1}\,[\,w_i(t_{i+1}) - w_{i+1}(t_{i+1})\,]\,.$$

Denoting $d_{i+1} = Q_{i+1}^{-1}\,[\,w_i(t_{i+1}) - w_{i+1}(t_{i+1})\,]$ we have

$$(1.7) \qquad a_{i+1} = U_{i+1}\,a_i + d_{i+1}\,.$$

It is easy to see that

$$(1.8) \qquad x(t_i) = Q_i\,a_i + w_i(t_i)\,.$$

Now any solution $\{a_i\}$ of recursion (1.7) can be written as

(1.9) $\qquad a_i = \Phi_i\, c + z_i$,

where $\{\Phi_i\}_{i=1}^{N}$ is a fundamental solution of (1.7), i.e.

(1.10) $\qquad \Phi_{i+1} = U_{i+1}\,\Phi_i$

and $\{z_i\}_{i=1}^{N}$ a particular solution of (1.7).
The computation of $\{\Phi_i\}_{i=1}^{N}$ and $\{z_i\}_{i=1}^{N}$ is employing the decoupling in (1.7), which in turn is related to the dichotomy for a well-conditioned problem. Using $\{\Phi_i\}_{i=1}^{N}$ and $\{z_i\}_{i=1}^{N}$, we can compute c from (cf.(1.2))

(1.11)
$$[M_\alpha Q_1 \Phi_1 + M_\beta Q_N\, \Phi_N\,]\, c = b - M_\alpha w_1(t_1) - M_\beta w_N(t_N) \\ - M_\alpha Q_1 z_1 - M_\beta Q_N\, z_N .$$

Then $x(t_i)$ follows from (1.9) and (1.8).

Remark 1.12

If the matrix $[M_\alpha Q_1 \Phi_1 + M_\beta Q_N\, \Phi_N\,]$ is ill-conditioned, computing c from (1.11) may result in an inaccurate computation of the $x(t_i)$. The routines compute a *condition number CN* which indicates whether this matrix is ill-conditioned or not (cf.(3.12)). Another problem is that errors might be propagated in an unstable way when the recursion (1.7) is used (although this should not be any problem in a well-conditioned case). The routines compute an estimate of the amplification of errors, which we call the *amplification factor* (in fact another condition number).

Quite often the matrices M_α, M_β have more structure. In particular M_α or M_β may have some systematically zero rows. This will be referred to as *partially separated* BC. If both M_α and M_β have zero rows (but for different row indices) and such that there total number equals n, the BC is referred to as *(completely) separated.*

The methods discussed in this chapter are implemented in the routines MUTSGE (for general BC), MUTSPS (for partially separated BC), MUTSSE (for separated BC).

2. Global description of the algorithms

In this section we shall give an outline of the various algorithms for the various types of two-point BC.

2.1 BVPs with general BC

Consider the ODE (1.1) and the general two-point BC

(2.1) $$M_\alpha x(\alpha) + M_\beta x(\beta) = b \ .$$

Any solution of the ODE (1.1) can be written as

(2.2) $$x(t) = F(t)c + w(t) \ ,$$

where $F(t)$ is a fundamental solution of the homogeneous part of (1.1), i.e.

(2.3) $$\frac{d}{dt}F(t) = L(t)F(t) \ ,$$

$w(t)$ a particular solution of (1.1) and c a constant n-vector.
After substituting (2.2) in (2.1) determine c from

(2.4) $$[M_\alpha F(\alpha) + M_\beta F(\beta)]\, c = b - M_\alpha w(\alpha) - M_\beta w(\beta).$$

So the solution x of (1.1) and (2.1) may be computed by *superposition* as follows:

(2.5a) find a particular solution $w(t)$ of the ODE (1.1),

(2.5b) find a fundamental solution $F(t)$ of the ODE (2.3),

(2.5c) find the n-vector c from equation (2.4).

This method is mathematically equivalent to what would have been found by *single shooting* . However, in many interesting problems, the homogeneous part of the ODE (1.1) has fast growing modes, which makes e.g. the computation of the fundamental solution $F(t)$ an unstable affair, cf. the remarks about dichotomy made in §I.1.3. To reduce this instability, the interval $[\alpha, \beta]$ is divided into subintervals $[t_i, t_{i+1}]$, $i = 1,2, \ldots, N-1$, say; then on each subinterval a particular solution $w_i(t)$ and a fundamental solution $F_i(t)$ is computed. This is called *multiple shooting* . Now, any solution of (1.1) on the subinterval can be written as

(2.6) $$x(t) = F_i(t)\, a_i + w_i(t) \ , i = 1, \ldots, N.$$

There are several possibilities for choosing the fundamental solution $F_i(t), i=1,2, \ldots, N$. For the methods discussed here the $F_i(t)$ are chosen such that

(2.7) $$F_i(t_{i+1}) = F_{i+1}(t_{i+1})\, U_{i+1} = Q_{i+1}\, U_{i+1} \ , \quad i = 1, \ldots, N-1 \ ,$$

where Q_{i+1} is an orthogonal matrix and U_{i+1} an upper triangular matrix. By letting $U_1 = I$, we may include the case $i=0$, if we choose $F_1(t_1) = Q_1$, some orthogonal matrix.
By matching the relations (2.6) at the points t_{i+1}, $i = 1, \ldots, N-1$, we then obtain

(2.8) $$x(t_{i+1}) = F_{i+1}(t_{i+1})\, a_{i+1} + w_{i+1}(t_{i+1}) = Q_{i+1}\, U_{i+1}\, a_i + w_i(t_{i+1}).$$

If we denote

(2.9) $\quad d_{i+1} = Q_{i+1}^{-1} \, [\, w_i(t_{i+1}) - w_{i+1}(t_{i+1}) \,] \;,$

we thus obtain the following upper triangular recursion:

(2.10) $\quad a_{i+1} = U_{i+1}\, a_i + d_{i+1} \quad, i=1,2,\ldots,N-1.$

By our choice of the F_i we immediately see that

(2.11) $\quad a_i = Q_i^{-1}(\, x(t_i) - w_i(t_i)\,).$

Now let $\{\Phi_i\}_{i=1}^N$ be a fundamental solution of (2.10), i.e.

(2.12) $\quad \Phi_{i+1} = U_{i+1}\,\Phi_i \;,\; i = 1,2,\ldots,N-1$

and let $\{z_i\}_{i=1}^N$ be some particular solution of (2.10). Then there should exist some vector c such that

(2.13) $\quad a_i = \Phi_i\, c + z_i \;,\; i = 1,2,\ldots,N.$

From (2.11) and (2.13) we therefore obtain the relation

(2.14) $\quad x(t_i) = w_i(t_i) + Q_i\,(\, z_i + \Phi_i\, c\,)\,,\; i = 1,2,\ldots,N.$

After substituting $x(t_1) = x(\alpha)$ and $x(t_N) = x(\beta)$ in the BC (2.1) we thus find:

(2.15)
$$[\, M_\alpha Q_1 \Phi_1 + M_\beta Q_N \Phi_N \,]\, c = b - M_\alpha w_1(\alpha) - M_\beta w_N(\beta) - M_\alpha Q_1 z_1 - M_\beta Q_N z_N.$$

The vector c which follows from (2.15) gives us the desired solution values $x(t_i)$ via (2.14).

Remark 2.16

In the case that the ODE (2.1) is homogeneous, i.e. $r(t) = 0$, $t \in [\alpha,\beta]$, there is no particular solution to be computed. Then (2.6), (2.8), (2.10), (2.11), (2.14) and (2.15) are to be replaced by:

(2.6)′ $\quad x(t) = F_i(t)\, a_i \quad,$

(2.8)′ $\quad x(t_{i+1}) = F_{i+1}(t_{i+1})\, a_{i+1} = Q_{i+1}\, U_{i+1}\, a_i \quad,$

(2.10)′ $\quad a_{i+1} = U_{i+1}\, a_i \quad,$

(2.11)′ $\quad a_i = Q_i^{-1}\, x(t_i) \quad,$

(2.14)′ $\quad x(t_i) = Q_i\, \Phi_i\, c \quad,$

(2.15)' $[M_\alpha Q_1 \Phi_1 + M_\beta Q_N \Phi_N]c = b$,

respectively (for relevant indices i).

2.2 BVPs with partially separated BC

If we have a partially separated BC, i.e. where in (2.1) the matrix M_α and/or M_β have a few zero rows, this fact can be utilized to reduce the computational labour, in that a smaller number of basis solutions has to be computed. For our discussion the following typical BC is to be considered:

(2.17a) $^1M_\alpha x(\alpha) + {}^1M_\beta x(\beta) = b^1$,

(2.17b) $^2M_\alpha x(\alpha) = b^2$.

Here $^1M_\alpha$ and $^1M_\beta$ are $k_s \times n$-matrices, $^2M_\alpha$ is an $(n-k_s)\times n$-matrix and b^1 and b^2 are k_s-vector and $(n-k_s)$-vector, respectively; i.e. only M_β has systematically zeros, viz. in its last $(n-k_s)$ rows.

Remark 2.18

If M_α happens to have a number of zero rows instead of M_β, the arguments below are essentially the same.

The reduction in computing $F_i(t)$ consists of the fact that we only compute its first k_s columns, viz. ($F_i^1(t)$), by requiring that

(2.19a) $^2M_\alpha F_1^1(\alpha) = 0$.

The particular solution $w_1(t)$ is then chosen such that it satisfies the decoupled initial value part, i.e.

(2.19b) $^2M_\alpha w_1(\alpha) = b^2$.

Formally we thus see that the desired solution x should lie in a linear variety $w_1(t) \oplus \text{span}(F_1^2(t))$, where $F_1^2(t)$ is just some complementary part of the fundamental solution $F_1^1(t)$. From (2.17) and (2.19) we see that span($w_1(t)$)$\perp$ span($F_1^1(t)$). Now we can proceed as in the general case, i.e. we can divide $[\alpha,\beta]$ into subintervals $[t_i, t_{i+1}]$, $i=1,2,\ldots,N-1$. On each subinterval $[t_i, t_{i+1}]$ a partial fundamental solution $F_i^1(t)$ and a particular solution $w_i(t)$ is computed such that at the initial point of the interval:

$$\operatorname{span}(F_i^1(t_i)) = \operatorname{span}(F_{i-1}^1(t_i)),$$

(2.20) $$w_i(t_i) \perp \operatorname{span}(F_i^1(t_i)),$$

$$w_i(t_i) \in w_{i-1}(t_i) \oplus \operatorname{span}(F_{i-1}(t_i)).$$

This then means that there exist k_s-vectors a_i^1, such that for any i,

(2.21) $$x(t) = F_i^1(t)\, a_i^1 + w_i(t)\,.$$

In our algorithm we choose $F_i^1(t_i)$ such that its columns are orthogonal. The analogue of (2.6) reads therefore:

(2.22) $$F_i^1(t_{i+1}) = F_{i+1}^1(t_{i+1})\, V_{i+1} = Q_{i+1}^1\, V_{i+1},$$

where the $n \times k_s$-matrix Q_{i+1}^1 has orthogonal columns and V_{i+1} is a $k_s \times k_s$ upper triangular matrix. Now if we denote (cf. (2.9))

(2.23) $$d_{i+1}^1 = (Q_{i+1}^1)^T\, [\, w_i(t_{i+1}) - w_{i+1}(t_{i+1})\,]\,,$$

then we obtain the following reduced upper triangular recursion:

(2.24) $$a_{i+1}^1 = V_{i+1}\, a_i^1 + d_{i+1}^1 \;\; , i = 1, \ldots, N-1.$$

Remark 2.25

Since we choose $w_{i+1}(t_{i+1})$ orthogonal to $\operatorname{span}(F_{i+1}^1(t_{i+1})) = \operatorname{span}(Q_{i+1}^1)$, we see that we can actually simplify (2.23) to

(2.26) $$d_{i+1}^1 = (Q_{i+1}^1)^T\, w_i(t_{i+1})\,.$$

Remark 2.27

$w_{i+1}(t_{i+1})$ is uniquely determined by the requirements (2.20). We apparently should project $w_i(t_{i+1})$ onto $\operatorname{span}(Q_{i+1}^1)$ and subtract this from $w_i(t_{i+1})$. Hence we find

(2.28) $$w_{i+1}(t_{i+1}) = w_i(t_{i+1}) - Q_{i+1}^1\, (Q_{i+1}^1)^T\, w_i(t_{i+1})\,.$$

The computation of the a_i^1 from the BC is done in a similar way as in the preceding subsection; we compute a fundamental solution $\{\Phi_i^1\}_{i=1}^N$ and a particular solution $\{z_i^1\}_{i=1}^N$ of (2.24). Since for some k_s-vector c^1 there must hold

(2.29) $$a_i^1 = \Phi_i^1\, c^1 + z_i^1\,,$$

we obtain the desired solution from

(2.30) $\quad x(t_i) = w_i(t_i) + Q_i^1(z_i^1 + \Phi_i^1 c^1)$.

After substituting $x(t_1) = x(\alpha)$ and $x(t_N) = x(\beta)$ in the BC (2.17a) we thus find c^1 from

(2.31) $$[{}^1M_\alpha Q_1^1 \Phi_1^1 + {}^1M_\beta Q_N^1 \Phi_N^1]\, c^1 = b^1 - {}^1M_\alpha w_1(\alpha) - {}^1M_\beta w_N(\beta) - {}^1M_\alpha Q_1^1 z_1^1 - {}^1M_\beta Q_N^1 z_N^1 .$$

Remarks 2.32

(i) If the ODE is homogeneous we still have to compute solutions $w_i(t)$ (but now of the homogeneous ODE) such that (2.19b) is satisfied.

(ii) If the ODE is homogeneous and moreover $b^2 = 0$, then we can skip the computation of w_i and put $d_i = 0$ for all i. In such a case we have to replace (2.21), (2.24), (2.29), (2.30) and (2.31) by

(2.21)′ $\quad x(t) = F_i^1(t)\, a_i^1$,

(2.24)′ $\quad a_{i+1}^1 = V_{i+1}\, a_i^1$,

(2.29)′ $\quad a_i^1 = \Phi_i^1 c^1$,

(2.30)′ $\quad x(t_i) = w_i(t_i) + Q_i^1 \Phi_i^1 c^1$,

(2.31)′ $\quad [{}^1M_\alpha Q_1^1 \Phi_1^1 + {}^1M_\beta Q_N^1 \Phi_N^1]\, c^1 = b^1$,

respectively.

2.3 BVP with (completely) separated BC

If we have (completely) separated BC then ${}^1M_\alpha = \varnothing$ in (2.17) as well. So

(2.33a) $\quad {}^1M_\beta x(\beta) = b^1$,

(2.33b) $\quad {}^2M_\alpha x(\alpha) = b^2$,

where ${}^1M_\beta$ is a $k_s \times n$-matrix and ${}^2M_\alpha$ is an $(n-k_s)\times n$-matrix.
We can use a similar approach as in § 2.2. However (2.29) until (2.31) are not needed. Indeed, as can be expected we have an explicit terminal value for the recursion (2.24) to compute the sequence $\{a_N^1, \ldots, a_1^1\}$. From (2.21) we derive

(2.34) $\quad x(t_i) = Q_i^1 a_i^1 + w_i(t_i)$.

After substitution in (2.33) we obtain

(2.35) $\quad {}^1M_\beta Q_N^1 a_N^1 = b^1 - {}^1M_\beta w_N(\beta)$.

Remark 2.36

The same remarks as 2.32 apply to the separated case, i.e. if the problem is homogeneous and $b^2 = 0$, we skip the computation of the $\{w_i(t)\}$ and $\{z_i^1\}$.
Instead of (2.34) and (2.35) we then have

(2.34)′ $\quad x(t_i) = Q_i^1 a_i^1$,

(2.35)′ $\quad {}^1M_\beta Q_N^1 a_N^1 = b^1$.

3. Special features of the methods

There are several aspects which make our routines different from other Multiple Shooting strategies. In the following subsections we shall describe some of them. This may help to understand the power and also the limitations of the method.

3.1 Numerical realization of the integration

Since the numerical integration accounts for the bulk of the computational labour, it is of fairly great importance to have this computation done efficiently. A first gain can be achieved quite simply. Realizing that the unstable solutions will inevitably dictate the stepsize if an absolute tolerance is given (and won't do for less if a relative tolerance is required), we need to use the adaptive integration control only for one solution on each subinterval. The other solutions are found at the thus determined grid. The grid is determined by the particular solution $w_i(t)$, or, if the problem is homogeneous, by the first column of $F_i(t)$ (or $F_i^1(t)$). The latter choice is induced by the wish to have points such that the most unstable solution is still integrated correctly (i.e. up to the required tolerance). See also [7].

3.2 Computing fundamental and particular solutions of recursions

For solving a BVP with general BC or partially separated BC we have to compute a fundamental solution and a particular solution of recursions (2.10) and (2.24), respectively. As both recursions are of the same nature, we only discuss recursion (2.10).

The important idea behind the decoupling method of §2 is that in well-posed linear BVP, the homogeneous solution space of (2.1) is *dichotomic* , i.e. is such that for some integer k_p ("partitioning index") there exist a k_p-dimensional subspace of increasing solutions and an $(n-k_p)$-dimensional subspace of non-increasing solutions. Using this property and starting with a proper Q_1 $(=F_1(t_1))$, we can compute a set of U_i for which the first k_p columns represent the subspace of increasing solutions and the last $(n-k_p)$ columns the subspace of the non-increasing solutions. In this way we have decoupled the increasing solutions and the

non-increasing solutions. This decoupling enables us to compute a fundamental solution of the upper triangular recursion (2.10) in a stable way as follows:
We partition matrices and vectors as

$$U_i = \begin{bmatrix} B_i & C_i \\ \varnothing & E_i \end{bmatrix} \quad , \quad a_i = \begin{bmatrix} a_i^1 \\ a_i^2 \end{bmatrix}, \tag{3.1}$$

where B_i is a $k_p \times k_p$-upper triangular matrix, E_i an $(n-k_p)\times(n-k_p)$-upper triangular matrix, C_i a $k_p \times (n-k_p)$-matrix, a_i^1 a k_p-vector and a_i^2 an $(n-k_p)$-vector.
The recursion (2.10) can be rewritten as

$$a_{i+1}^2 = E_{i+1}\, a_i^2 + d_{i+1}^2 \ , \tag{3.2a}$$

$$a_{i+1}^1 = B_{i+1}\, a_i^1 + C_{i+1}\, a_i^2 + d_{i+1}^1 \ . \tag{3.2b}$$

As the B_i represent the increasing solutions, the absolute value of the diagonal elements of B_i can be expected to be greater than 1, making forward computation of (3.2b) unstable. The E_i represent the non-increasing solutions, so the absolute value of the diagonal elements of E_i can be expected to be less than or equal to 1, making forward computation of (3.2a) stable. Hence the obvious strategy for computing a fundamental solution $\{\Phi_i\}_{i=1}^N$ and a particular solution $\{z_i\}_{i=1}^N$ of recursion (2.10) is to use (3.2a) in forward direction and (3.2b) in backward direction. So for the particular solution $\{z_i\}_{i=1}^N$ we have the BC

$$z_1^2 = 0 \quad , \quad z_N^1 = 0 \ . \tag{3.3}$$

Then $z_i^2,\ i=2,3,\ldots,N$, using (3.2a) in forward direction, and $z_i^1,\ i=N-1,N-2,\ldots,1$ using (3.2b) in backward direction, is computed.
For the fundamental solution we have the recursion

$$\Phi_{i+1}^2 = E_{i+1}\,\Phi_i^2 \ , \tag{3.4a}$$

$$\Phi_{i+1}^1 = B_{i+1}\,\Phi_i^1 + C_{i+1}\,\Phi_i^2 \tag{3.4b}$$

and the BC

$$\Phi_1^2 = (\varnothing \mid I) \ ; \ \Phi_N^1 = (I \mid \varnothing). \tag{3.5}$$

Now $\{\Phi_i^2\}_{i=1}^N$ is computed via (3.4a) and $\{\Phi_i^1\}_{i=N}^1$ is then computed via (3.4b).

3.3 Choosing Q_1 and $w_i(t_i)$

As in fact the matrix Q_1 generates the sequences of $\{Q_i\}$ and $\{U_i\}$ it is important to have a proper choice for Q_1. Indeed as was shown in [4] the desired splitting of the solution space into increasing and non-increasing solutions may not be achieved for general initial matrices Q_1, though in practice it is most likely that an arbitrary choice will do eventually. Nevertheless for a good stability of the recursion some effort to obtain a good guess is worth

paying for. For general BC no information about k_p nor the direction of the increasing solutions is available, so we just take $Q_1 = I$. If, after a few normalizations, a disorder of eigenvalues of the matrices U_i becomes visible, we perform a permutation of the columns of Q_1 to hopefully restore an ordering in decreasing absolute magnitude. If needed this process is repeated a finite number of times. In § 4.3 we return to this.
If the BC are partially separated, one has to realize that k_s and k_p may be different ($k_s \geq k_p$). Hence, in general one should try to obtain an ordering of the diagonal elements of the V_i, at least to such an extent that the $k_p \times k_p$ left upper part contains the eigenvalues which are in absolute value greater than 1; of course this can only be found by guessing and correcting as in the general case.
Finally, if the BC are completely separated we necessarily have that $k_s = k_p$ (or at least a reasonable choice of k_s, if there is no exponential but only an ordinary dichotomy). For this, however, we presuppose the problem to be well-conditioned, which will be explained in the next subsection.
As far as the $w_i(t_i)$ are concerned, we already remarked that they were in fact determined by our desire to keep $w_i(t_i)$ in the same linear variety as $w_{i-1}(t_i)$. Of course this only makes sense in case the BC are (partially or completely) separated. If we use the strategy for general BC we have a complete freedom again. We have chosen for the option $w_i(t_i) = 0$ because, in general, this gives $O(1)$ components of all solutions involved, notably the desired particular one and the most unstable one. It was discussed in [7] that this was a sensible choice.

3.4 Conditioning and stability

The accuracy of the solution $x(t)$ of a BVP, using the method as described in § 2, depends on:

(i) The accuracy by which the fundamental solution $F_i(t_i)$ and the particular solution $w_i(t_i)$ are computed. (This accuray is determined by the user.)

(ii) The accuracy by which the vector c in equation (2.15) is computed.

(iii) The accuracy by which the fundamental solution $\{\Phi_i\}_{i=1}^N$ or $\{\Phi_i^1\}_{i=1}^N$ and the particular solution $\{z_i\}_{i=1}^N$ or $\{z_i^1\}_{i=1}^N$ of the recursion (2.10) and (2.24), respectively, is computed.

First we will discuss point (ii).
Since (2.15) resulted from the boundary conditions we have to investigate the effect of perturbations in the BC on the computed solution. Suppose we have a BC with a perturbed right-hand side, i.e. instead of (2.1) we have

(3.6) $$M_\alpha \hat{x}(\alpha) + M_\beta \hat{x}(\beta) = b + \delta b \;.$$

As x and $\hat{x}$ are both solutions of the ODE of the BVP, there exists a vector v such that

(3.7) $\hat{x}(t) - x(t) = F(t)\,v$,

where $F(t)$ is a fundamental solution.

Subtracting (3.2) from (3.6) and using (3.7) we obtain:

(3.8) $[M_\alpha F(\alpha) + M_\beta F(\beta)]\,v = \delta b$.

So we have

(3.9) $\hat{x}(t) - x(t) = F(t)[M_\alpha F(\alpha) + M_\beta F(\beta)]^{-1}\delta b$

and

(3.10) $\max_{t\in(\alpha,\beta)} \|\hat{x}(t) - x(t)\| \le \max_{t\in(\alpha,\beta)} \|F(t)[M_\alpha F(\alpha) + M_\beta F(\beta)]^{-1}\|\,\|\delta b\|$.

Therefore we define a *condition number CN* of a BVP as

(3.11) $CN = \max_{t\in(\alpha,\beta)} (\|F(t)[M_\alpha F(\alpha) + M_\beta F(\beta)]^{-1}\|)$.

(Notice that CN is independent of the fundamental solution $F(t)$, as for any other fundamental solution $G(t)$, say, there is a constant matrix P such that $G(t) = F(t)P$).
As is shown in [8] if $\{\Phi_i\}$ is defined as in (3.4), then an estimate of CN is given by

(3.12) $\kappa = \|[M_\alpha Q_1 \Phi_1 + M_\beta Q_N \Phi_N]^{-1}\|_1 \le 2\,CN$.

Basically the information to compute κ is available (cf. (2.15)). However when the BVP has (partially) separated BC, only k_s $(< n)$ columns of Q_1, Q_N, Φ_1, Φ_N are computed. The separated BC can be written as

(3.13) $$\begin{bmatrix} {}^1M_\alpha \\ {}^2M_\alpha \end{bmatrix} x(\alpha) + \begin{bmatrix} {}^1M_\beta \\ \emptyset \end{bmatrix} x(\beta) = \begin{bmatrix} b^1 \\ b^2 \end{bmatrix} .$$

For the condition number CN we have

(3.14) $$CN = \max_{t\in[\alpha,\beta]} \|F(t) \left[\begin{bmatrix} {}^1M_\alpha \\ {}^2M_\alpha \end{bmatrix} \left[F^1(\alpha) \,\middle|\, F^2(\alpha) \right] + \begin{bmatrix} {}^1M_\beta \\ \emptyset \end{bmatrix} \left[F^1(\beta) \,\middle|\, F^2(\beta) \right] \right]^{-1} \|$$

$$=\max_{t\in(\alpha,\beta)}\|F(t)\begin{bmatrix} {}^1M_\alpha F^1(\alpha)+{}^1M_\beta F^1(\beta) & {}^1M_\alpha F^2(\alpha)+{}^1M_\beta F^2(\beta) \\ {}^2M_\alpha F^1(\alpha) & {}^2M_\alpha F^2(\alpha)\end{bmatrix}^{-1}\| .$$

As CN is independent of $F(t)$ and we have taken $F(t)$ such that ${}^2M_\alpha F^1(\alpha)=\varnothing$, it is easy to see that if either $[\,{}^1M_\alpha F^1(\alpha)+{}^1M_\beta F^1(\beta)\,]$ or ${}^2M_\alpha F^2(\alpha)$ is ill-conditioned also the BVP will be ill-conditioned. Hence we compute

(3.15) $\qquad \kappa_1=\|\,[\,{}^1M_\alpha Q_1^1\,\Phi_1^1+{}^1M_\beta Q_N^1\,\Phi_N^1\,]^{-1}\|_1\,,$

(3.16) $\qquad \kappa_2=\|\,[\,{}^2M_\alpha Q_1^2\,]^{-1}\|_1 .$

Although a large κ_1 or a large κ_2 indicates that the BVP is ill-conditioned, it is possible to have an ill-conditioned BVP for which both κ_1 and κ_2 are of order one. For well-conditioned BVP with separated BC it is necessary that $F^2(t)$ contains only non-growing modes (in case of completely separated BC, all non-growing modes). To find out whether $F^2(\alpha)$ would result in computing a growing solution, we recall that for the solution $x(t)$ we had (cf. e.g. (2.29))

$$x(t)=F^1(t)\,c^1+w(t)\ ,$$

and completing $F^1(t)$ to a fundamental solution $F(t)=(F^1(t)\,|\,F^2(t))$ we thus see that

(3.17a) $\qquad w(t)=F^2(t)\,c^2+z(t)\,,$

where $z(t)$ is a particular solution of the ODE of the BVP and c^2 an $(n-k_s)$-vector. Supposing that $z(t)$ is a smooth solution, a dominant mode in $F^2(t)$ will influence the growth of $w(t)$, unless $c^2=0$. However, by computing another particular solution $v(t)$ say, where

(3.17b) $\qquad v(t)=F^2(t)\,e^2+w(t)\ ,\ e^2\neq 0\,,$

and thus

(3.18) $\qquad w(t)-v(t)=F^2(t)\,e^2\ ,$

we have a way to find out whether $F^2(t)$ contains dominant modes or not (see §4.4).

For BVP with a dichotomic solution space we have the recursion (cf. (3.2)):

(3.19a) $\qquad a_{i+1}^2=E_{i+1}\,a_i^2+d_{i+1}^2\quad ,$

(3.19b) $\qquad a_i^1=B_{i+1}^{-1}\,(a_{i+1}^1-C_{i+1}\,a_i^2-d_{i+1}^1\,)\ \ ,\ i=1,\ldots,N-1\,.$

To investigate the stability of (3.19) we examine the effects of additive perturbations $\{p_i^2\}$ and $\{p_i^1\}$ of respectively (3.19a) and (3.19b), i.e. suppose $\{\hat{a}_i^1\}$ and $\{\hat{a}_i^2\}$ satisfy

(3.20a) $\quad \hat{a}_{i+1}^2 = E_{i+1}\hat{a}_i^2 + d_{i+1}^2 + p_{i+1}^2$,

(3.20b) $\quad \hat{a}_i^1 = B_{i+1}^{-1}(\hat{a}_{i+1}^1 - C_{i+1}\hat{a}_i^2 - d_{i+1}^1) + p_i^1$.

Then for $g_{i+1}^1 = \hat{a}_{i+1}^1 - a_{i+1}^1$, $g_{i+1}^2 = \hat{a}_{i+1}^2 - a_{i+1}^2$ we have

(3.21a) $\quad g_{i+1}^2 = E_{i+1}g_i^2 + p_{i+1}^2 \quad ; \quad g_1^2 = p_1^2$,

(3,21b) $\quad g_i^1 = B_{i+1}^{-1}(g_{i+1}^1 - C_{i+1}g_i^2) + p_i^1 \quad ; \quad g_N^1 = p_N^1$,

which results in

(3.22a) $$g_i^2 = \sum_{l=1}^{i}[(\prod_{j=l+1}^{i} E_j)p_l^2] ,$$

(3.22b) $$g_i^1 = \sum_{l=1}^{i}[\Omega_{i,N}(\prod_{j=l+1}^{i} E_j)p_l^2] + \sum_{l=i+1}^{N-1}[(\prod_{j=i+1}^{l} B_j)^{-1}\Omega_{l,N}\,p_l^2]$$

$$+\sum_{l=i}^{N}[(\prod_{j=i+1}^{l} B_j)^{-1}p_l^1] ,$$

where $\Omega_{m,q}$ is a shorter notation for

(3.23a) $$\Omega_{m,q} = -\sum_{l=m+1}^{q}[(\prod_{j=m+1}^{l} B_j)^{-1}C_l(\prod_{j=m+1}^{l} E_j)] ,$$

where

(3.23b) $$\prod_{p}^{q} M_j = \begin{matrix} M_q M_{q-1}\cdots M_p & \text{if } q \ge p \\ I & \text{if } q < p \end{matrix} ,$$

(3.23c) $$\sum_{p}^{q} M_j = \begin{matrix} M_p+\ldots+M_q & \text{if } q \ge p \\ \varnothing & \text{if } q < p \end{matrix} .$$

If the permutations p_i^1, p_i^2 are of the same order, i.e. $\|p_i^1\| \le \delta$, $\|p_i^2\| \le \delta$ for some δ, we have

(3.24a) $$\|g_i^2\| \le [\sum_{l=0}^{i}\|\prod_{j=l+1}^{i} E_j\|]\,\delta ,$$

(3.24b) $$\|g_i^1\| \le [(\sum_{l=0}^{i}\|\Omega_{i,N}(\prod_{j=l+1}^{i} E_j)\|) + (\sum_{l=i+1}^{N-1}\|(\prod_{j=i+1}^{l} B_j)^{-1}\Omega_{l,N}\|)$$

$$+(\sum_{l=i}^{N}\|(\prod_{j=i+1}^{l} B_j)^{-1}\|)]\,\delta .$$

One easily checks that a proper dichotomy implies reasonably bounded $\|\Omega_{m,p}\|$ as well as such bounds for $\|\prod E_j\|$ and $\|(\prod B_j)^{-1}\|$. This then establishes the stability of the computation of $\{\Phi_i\}_{i=1}^N$ and $\{z_i\}_{i=1}^N$.

4. Computational aspects of the methods

There are a number of aspects which have not been filled in yet. In this chapter we shall therefore treat some particular implementations as they are realized in the various routines.

4.1 The use of RKF45

A very reliable and fairly inexpensive integrator is RKF45, written by L.F. Shampine and H.A. Watts, a Runge Kutta Fehlberg routine which uses fifth order estimates combined with fourth order approximations (cf. [1]). This routine is the working horse in our codes and as long as the system is not stiff (in the sense that there is high activity of some modes) we have found it to work very well indeed (cf. [8]). We have changed the original routines to make that it only uses the combined fourth-fifth order integrator for the grid determining solution, see § 3.1. A special routine computes solutions on a given grid by the fifth order only. Another special feature is that it terminates the calculations if five consecutive new points are found. Then an orthogonalization of the solution is performed and a new cycle is started. This QU-decomposition is carried out with elementary hermitians (Householder's method, cf. [2]). Rather than in the form ($A\ Q_i =$) $Q_{i+1}\, U_{i+1}$ we obtain Q_{i+1}^T in factored form. It is obvious that we only need to evaluate the first k_s columns of Q_{i+1} if we have (partially) separated BC. In the next subsection we consider how this will work out in the global computations.
In the original routine RKF45 both a relative and an absolute tolerance has to be supplied. Because of the fact that for general BVP on finite intervals one is mainly interested in absolute accuracy and our strategy makes signifant growth per shooting interval unlikely anyway, we recommend to set the relative tolerance sufficiently smaller than the absolute tolerance.

4.2 The choice of shooting points

The idea to have shooting intervals consisting of 5 steps only was induced by considerations of optimal efficiency, cf. [8]. It is obvious that this strategy may give many more points for output than is needed by the user. Therefore a special device takes care of assembling these so called *minor shooting intervals* to *major shooting intervals* ; the latter are such that the initial and terminal points coincide with user requested output points. Here another powerful feature of the decoupling method is revealed. Because of the fact that the k-partitioning (k_p) coincides with the decoupling into increasing and decreasing modes, forward assembling of increments on minor intervals is relatively stable. Such an assembly may be described as follows:
Let t_{i_j} be the initial point of the j^{th} *major shooting interval*, i.e. t_{i_j} is the j^{th} output point. Define

$$(4.1) \qquad W_0 := I \ ; G_0 := 0 \, .$$

Now compute for $s = 1,2, \ldots ,$

(4.2) $\quad W_s := U_{i_j+s}\, W_{s-1}\ ;\ G_s := U_{i_j+s}\, G_{s-1} + d_{i_j+s}$.

If s is large enough, then W_s describes the increment on the major interval $[t_{i_j}, t_{i_j+s}]$ and G_s the forcing term on that interval, so that

(4.3) $\quad a_{i_j+s} = W_s\, a_{i_j} + G_s$

(of course s is only a local index for W_s and G_s).
Now we have five possible options for the $(j+1)^{\text{th}}$ output point $t_{i_{j+1}} = t_{i_j+s}$:

(i) choose s such that $\|W_s\| \le \rho$, ρ prescribed;

(ii) choose s such that $|\, t_{i_j+s} - t_{i_j} \,| = \frac{\beta-\alpha}{N}$

(N the number of intervals);

(iii) choose s such that t_{i_j+s} equals the first next specified output point;

(iv) choose s such that either $\|W_s\| \le \rho$, ρ prescribed or $|\, t_{i_j+s} - t_{i_j} \,| = \frac{\beta-\alpha}{N}$;

(v) choose s such that either $\|W_s\| \le \rho$, ρ prescribed or t_{i_j+s} equals the first next specified output point.

Remark 4.4

Of course, it may be that these criteria above need shorter minor shooting intervals at the end of the major shooting interval. This is taken care of by the routines.

Remark 4.5

Criterion (i) is of interest if one suspects the maximal incremental growth to be changing on $[\alpha,\beta]$ and likes to monitor this so that the solution is equidistributed with respect to this. However, one should realize that it may lead to (undesirably) large intervals if there are mildly growing solutions only.
Criteria (ii) and (iii) may cause overflow problems if the given major shooting intervals are too large. Therefore only criteria (i), (iv) and (v) are implemented, allowing a ρ which is smaller than the square root of the largest positive real number that can be represented by the used computer.

4.3 The computation of Q_1 and Q_1^1 and the proper splitting

Suppose we find the diagonal of the matrix U_2 not to be ordered properly (to recall: we need to have the diagonal elements appear more or less in non increasing absolute value). Then we use a permutation matrix P, which permutes the columns of U_2 according to the ordering of the absolute value of these diagonal elements. Of course U_2P is no longer upper triangular, so we perform another QU-decomposition, i.e.

(4.6) $$U_2(old)\,P =: R\; U_2(new)\,.$$

The matrix $U_2(new)$ replaces $U_2(old)$, whilst $Q_1(old)$ is replaced by

(4.7) $$Q_1(new) := Q_1(old)\,P$$

and Q_2 by

(4.8) $$Q_2(new) := Q_2(old)\,R\ .$$

If U_2 is still not found in order we repeat this procedure. In fact we do the same with the assembled product $U_s U_{s-1} \cdots U_2$ on the first major shooting interval. On subsequent major intervals this reordering is no longer feasible. One should realize that neat problems have to be dichotomic (cf. [3]), i.e. after reaching the endpoint of the first major interval, we should have a good idea of k_p. Indeed the routines choose k_p equal to the position of that diagonal element of U_2 which is the smallest one (in absolute value) being larger than 1. Of cource this only makes sense for an ordered diagonal. Although U_2 etc. are expected to be ordered in general, there might be situations where this is not the case. Therefore a global check on the increment on the whole interval $[\alpha,\beta]$ is made. If the ordering is found not to be satisfactory, a global reordering is performed using permutation matrices according to this. In fact this is rather cheap as it only requires matrix-matrix multiplications plus one QU-decomposition at each output point. This process is moreover stable if the norm of the assembled matrices does not outgrow TOL / EPS, where TOL is the absolute tolerance and EPS the machine constant.

If the BC are (partially) separated we have to determine a Q_1^2 such that ${}^2M_\alpha Q_1^2 = \varnothing$ (cf. (2.19a)). This can be done conveniently as follows:
Compute elementary hermitians $P_1, \ldots, P_n$ such that

(4.9) $$R := P_n \cdots P_1\, {}^2M_\alpha^T\,,$$

is upper triangular. Now take Q_1^2 as the last k_s columns of

(4.10) $$Q_1 = P_1 \cdots P_n\ .$$

(It is easily seen that this results in the desired matrix as

$$\mathrm{span}({}^2M_\alpha^T) = \mathrm{span}(P_1 \cdots P_{n-k_s}\begin{bmatrix} I_{n-k_s} \\ 0 \end{bmatrix}).$$

Sometimes it is not clear beforehand whether $\mathrm{rank}(M_\alpha) < n$ or $\mathrm{rank}(M_\beta) < n$. (Note that when M_β has some zero rows, say $n-k_s$, $\mathrm{rank}(M_\beta)$ may be smaller than k_s.) In such a case we may invoke the singular value decomposition (SVD) of these matrices to determine the *numerical* rank. So consider

(4.11) $$M_\alpha = U_\alpha \Sigma_\alpha V_\alpha^T \quad , \quad M_\beta = U_\beta \Sigma_\beta V_\beta^T \ ,$$

where U_α, V_α, U_β, V_β are orthogonal matrices and Σ_α, Σ_β diagonal matrices. Suppose Σ_α has k_{s1} non-zero diagonal elements and Σ_β has k_{s2} non-zero diagonal elements. If both $k_{s1} = k_{s2} = n$ we do not have separated BC. If $k_{s2} < n$ we have

(4.12) $$U_\beta^T M_\beta = \Sigma_\beta V_\beta = \begin{bmatrix} \Sigma_\beta^1 V_\beta \\ \emptyset \end{bmatrix} .$$

So multiplying (2.2) by U_β^T we obtain

(4.13) $$U_\beta^T M_\alpha x(\alpha) + U_\beta^T M_\beta x(\beta) = U_\beta^T b \ ,$$

which, denoting $U_\beta^T M_\alpha = \hat{M}_\alpha$, $U_\beta^T M_\beta = \hat{M}_\beta$, $U_\beta^T b = \hat{b}$, can be written as

(4.14a) $${}^1\hat{M}_\alpha x(\alpha) + {}^1\hat{M}_\beta x(\beta) = \hat{b}^1 \ ,$$

(4.14b) $${}^2\hat{M}_\alpha x(\alpha) = \hat{b}^2 \ .$$

This is of the form (2.17).
Of course it may be that $k_{s1} \le k_{s2}$, in which case it would be more profitable to regard the BVP as a problem on $[\beta, \alpha]$, instead of on $[\alpha, \beta]$. Therefore we compute both the SVD of M_α and of M_β and take the smallest of k_{s1} and k_{s2} with the coresponding initial and terminal points (i.e. either $[\alpha, \beta]$ or $[\beta, \alpha]$).

4.4 The computation of the stability constants

The actual solution of (2.15), (2.31) and (2.35) is done using a Crout routine (LU-decomposition). From this it follows that for general BC the quantity κ in (3.12) can be computed without much additional effort, using this LU-decompostion. As we remarked κ is at most a factor 2 amiss in comparison with the actual condition number (cf. (3.12)). If the BC are (partially) separated we do not have all necessary information about the E_i available. It may be even so that κ_1 and κ_2 (see (3.15) and (3.16)) are moderate since the ill-conditioning is concealed by the particular solution w_i. In order to detect this we also compute another sequence of particular solutions $\{v_i\}$ such that

(4.15) $$v_1(t_1) = w_1(t_1) + F_1^2(t_0)\, e^2 \ ,$$

$$e^2 = (n - k_s)^{-1/2} (1,1,\ldots,1)^T \ .$$

Then a κ_3 is computed as

$$(4.16) \qquad \kappa_3 = \kappa_2 \max_i (\| w_i(t_i) - v_i(t_i) \|_2) .$$

As an estimate for the condition number CN we now better take

$$(4.17) \qquad \kappa = \max(\kappa_1, \kappa_2, \kappa_3) .$$

The user may find the κ as an output parameter ER(4).

Of course it is possible that the matrices $[M_\alpha Q_1 \Phi_1 + M_\beta Q_N \Phi_N]$, $[{}^1M_\alpha Q_1^1 \Phi_1^1 + {}^1M_\beta Q_N^1 \Phi_N^1]$ or ${}^2M_\alpha Q_1^2$ (cf. (2.5), (2.31), (2.35), respectively) happen to be numerically singular. In that case a terminal error, IERROR = 320 is given.

Apart from this condition number another quantity is of importance. In fact we need to compute the maximal value in norm of suitable Green's functions (cf. [5]). This is an almost impossible task and therefore we are satisfied with a somewhat heuristical estimate of them. Note that in (3.24) the magnitude of the quantities $\| (\prod E_j) \|$ and $\| (\prod B_j^{-1}) \|$ may be blamed if the local errors are blown up significantly. Hence it makes sense to monitor the diagonal elements of the product matrices $E_p \cdots E_q$ and $B_q^{-1} \cdots B_p^{-1}$ for arbritary p and q $(p \geq q)$, as they essentially reflect the growth of the basis solutions. Thinking of (3.24) we therefore also compute

$$(4.18) \qquad A_f^2 = \max_k (\max_i (1 + \sum_{l=2}^{i} (\prod_{j=2}^{i} | E_j^k |))) ,$$

where E_j^k denotes the k-th diagonal element of E_j,

$$(4.19) \qquad a_{f1}(k) = \max_i (1 + \sum_{l=i+1}^{N} (\prod_{j=i}^{l-1} | B_j^k |^{-1})) ,$$

$$(4.20) \qquad a_{f2}(k) = \max_i ((\prod_{l=i}^{N} | B_l^k |)^{-1}, \ldots ,(\prod_{l=N-1}^{N} | B_l^k |)^{-1}, | B_N^k |^{-1}),$$

$$(4.21) \qquad a_{f3}(k) = \max_i (\prod_{l=1}^{i} | E_l^k | , \ldots , \prod_{l=i-1}^{i} | E_l^k | , | E_i^k |),$$

$$(4.22) \qquad A_f^1 = \max_k (a_{f1}(k) + a_{f2}(k) \times a_{f3}(k)) ,$$

where B_j^k denotes the k-th diagonal element of B_j .

As an estimate of the amplification factor A_f (being a bound for the Green's functions in turn) we take

$$(4.23) \qquad A_f = \max(A_f^1 , A_f^2).$$

The user may find A_f as an output parameter ER(5).

If A_f is such that the global rounding error is larger than the discretization error, a warning error, IERROR = 300, is given.

Remark 4.24

If there are constant modes or very slowly growing modes or very slowly decreasing modes, A_f will be of the order of the number of output points.

Remark 4.25

The computation of A_f depends on the number of output points. If the problem is dichotomic, the influence of the number of output points on the estimate A_f is small. However, if there is no dichotomy on the interval $[\alpha, \beta]$, the choice of the output points determines whether A_f is a good estimate for the amplification factor or not. If the problem is not dichotomic, it will be locally dichotomic on subintervals $[\alpha, \alpha_2]$, $[\alpha_3, \alpha_4], \ldots, [\alpha_m, \beta]$, say, with different subspaces of growing modes and nongrowing modes on each subinterval. In order to detect these changes of the dichotomy on $[\alpha, \beta]$ and to get a reasonable estimate A_f for the amplification factor, the output points should be chosen such that, besides α and β, each subinterval $[\alpha, \alpha_1], \ldots, [\alpha_m, \beta]$ contains at least one output point.

References

[1] G.F. Forsythe, M.A. Malcolm, C.B. Moler, *Computer Methods for Mathematical Computation*, Prentice Hall, Englewood Cliffs, 1977.

[2] G.H. Golub, C.F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1983.

[3] F.R. de Hoog, R.M.M. Mattheij, *On dichotomy and well-conditioning in BVP*, SIAM J. Numer. Anal. 24 (1987), 89-105.

[4] R.M.M. Mattheij, *Characterization of dominant and dominated solutions of linear recursions*, Numer. Math. 35 (1980), 421-442.

[5] R.M.M. Mattheij, *Estimates for the errors in the solutions of linear boundary value problems, due to perturbations*, Computing 27 (1981), 299-318.

[6] R.M.M. Mattheij, *The conditioning of linear boundary value problems*, SIAM J. Numer. Anal. 19 (1982), 963-978.

[7] R.M.M. Mattheij, G.W.M. Staarink, *On optimal shooting intervals*, Math. Comp. 42 (1984), 25-40.

[8] R.M.M. Mattheij, G.W.M. Staarink, *An efficient algorithm for solving general linear two-point BVP*, SIAM J. Sci. Stat. Comput. 5 (1984), 745-763.

[9] R.M.M. Mattheij, *Decoupling and stability of BVP algorithms*, SIAM Review 27 (1985), 1-44.

CHAPTER III

BVP ON INFINITE INTERVALS

1. Introduction

If for an ODE

$$(1.1)\qquad \frac{d}{dt}x(t) = L(t)x(t) + r(t), \quad \alpha \le t < \infty,$$

a BC is given

$$(1.2)\qquad M_\alpha x(\alpha) + M_\infty x(\infty) = b,$$

then it can be shown that x can be written as

$$(1.3)\qquad x(t) = F^2(t)c^2 + w(t),$$

where $w(t)$ is a bounded particular solution and $F^2(t)$ is a matrix solution ($F^2(t)$ an $n \times k_b$-matrix say) of bounded homogeneous solutions (see [1]). Let us denote the complementary part of the fundamental solution by $F^1(t)$. If $F^1(t)$ consists of exponentially increasing modes exclusively, then it is possible using the decoupling idea to effectively "remove" them doing the backward sweep of the multiple shooting recursion. To this end it is assumed that the particular interval $[\alpha, \beta]$ is specified where the output values are wanted. The shooting process then continues over an interval $[\beta, \gamma]$, where γ is such that the modes in $F^1(t)$ have grown sufficiently large to expect the backward sweep of the recursion algorithm, cf. §II.3.2, to damp their effect to (user specified) accuracy.
For some problems there may be some slower growing modes (like polynomially growing) present. This requires a special technique, like extrapolation. The routine MUTSIN for solving the BVP (1.1), (1.2), has therefore some special provisions for doing this efficiently.

Remark 1.4

Although the algorithm computes c^2 from the (usually) singular system $[M_\alpha F^2(\alpha) + M_\infty F^2(\beta)]c^2 = \hat{b}$ (where $\hat{b}$ is derived from the BC in a least-squares sense) we can still determine a quantity like the condition number. As a consequence often a diagnosis can still be given if something goes wrong or when output variables should not be trusted.

2. Global description of the algorithm

Consider the ODE

(2.1) $\frac{d}{dt}x(t) = L(t)x(t) + r(t), \quad \alpha \le t < \infty,$

where $L(t)$ is an $n \times n$-matrix and $r(t), x(t)$ are n-vectors for all t. Let the BC be given by

(2.2) $M_\alpha x(\alpha) + M_\infty x(\infty) = b.$

If we assume that the solution space is dichotomic (cf. §II.3.2), then there exist integers k_u and k_b $(k_u + k_b = n)$ and a fundamental solution $F(t)$, such that

(2.3) $F(t) = [F^1(t) \mid F^2(t)],$

where $F^1(t)$ contains k_u columns and $F^2(t)$ k_b columns such that $F^2(t)$ precisely represents the bounded homogeneous solutions. Under suitable conditions, cf. [2], there exist at least one *bounded particular solution* of (2.1), $w(t)$. Hence for some constant k_b-vector c^2 we find

(2.4) $x(t) = F^2(t)c^2 + w(t).$

Upon substituting (2.4) in (2.2) we find

(2.5) $[M_\alpha F^2(\alpha) + M_\infty F^2(\infty)]c^2 = b - M_\alpha w(\alpha) - M_\infty w(\infty).$

Note that in case $F^2(t), w(t) \to 0, t \to \infty$, the condition above reduces to an initial value condition (though rank deficient!). Because of the requirements on $F^2(t)$ and $w(t)$, the problem (2.1), (2.2) is sometimes also called a *conditionally stable initial value problem.* The main question therefore is how to find the non-increasing ("stable") manifold.
With some adaptations this can be done along the lines of the method described in chapter II. Suppose we like to have output values for x on the interval $[\alpha,\beta]$ within an accuracy TOL. Let us assume that F^1 consists of exponentially increasing solutions only. Then there certainly exists a point γ, such that

(2.6) $\|F(\gamma)P\,F(\beta)^{-1}\| > \mathrm{TOL}^{-1}, \quad \text{where} \quad P = \begin{bmatrix} I_{k_u} & \varnothing \\ \varnothing & \varnothing \end{bmatrix};$

in other words, each of the increasing solutions has grown at least by TOL^{-1}. We then proceed as follows: use a multiple shooting strategy as in §II.2.1, with at least $\alpha = t_1$, $\beta = t_M$ and $\gamma = t_N$ as output points, resulting in an upper triangular recursion

(2.7) $a_{i+1} = U_{i+1}a_i + d_{i+1}, \; i = 1,2, \ldots, N-1,$

(cf. (II.2.11)), with

(2.8) $\quad x(t_i) = Q_i\, a_i + w_i(t_i).$

Then compute a particular solution $\{z_i\}$ of (2.7), satisfying

(2.9) $\quad z_1^2 = 0\,,\ z_N^1 = 0$

and a partial fundamental solution $\{\Psi_i\}$ (Ψ_i is $n \times k_b$), satisfying

(2.10) $\quad \Psi_1^2 = I\ ;\ \Psi_N^1 = \varnothing\,,$

Clearly for some k_b-vector c^2 we have (within accuracy TOL!)

(2.11) $\quad a_i = \Psi_i\, c^2 + z_i\,.$

From (2.8), (2.11) and (2.2) we thus derive the following relation for c^2:

(2.12)
$$[M_\alpha Q_1 \Psi_1 + M_\infty Q_M \Psi_M]\, c^2 = b - M_\alpha w_1(\alpha) - M_\infty w_M(\beta) - M_\alpha Q_1 z_1 - M_\infty Q_M z_M\,.$$

The matrix appearing in (2.12) on the left is $n \times k_b$. Therefore we solve this system in a least-squares sense.

3. Special features

The previously outlined algorithm is implemented as MUTSIN. For the computation of the multiple shooting recursion on the interval $[\alpha, \gamma]$ the same strategy is used as in §II.3.1-II.3.3 for BVP with general BC.

3.1 Errors introduced by finite choice of γ

In §2 we considered the case of exponentially increasing solutions in $F^1(t)$. For our upper triangular shooting recursion (2.7) this means that in

(3.1)
$$U_i = \begin{bmatrix} B_i & C_i \\ \varnothing & E_i \end{bmatrix},$$

we may assume that $\|B_{i+1}^{-1}\| \geq \kappa\, e^{\lambda(t_{i+1} - t_i)}$ for some negative λ and (not large) positive κ. That means that on $[\beta, \gamma] = [t_M, t_N]$ we expect

(3.2)
$$\|[\prod_{M+1}^{N} B_j]^{-1}\| \leq \kappa\, e^{\lambda(\gamma - \beta)}$$

Since we do not know the bounded (and non increasing) solutions at t_N exactly we choose their component in span($F^1(t_N)$) to be zero, cf. (2.9) and (2.10). Hence we introduce a *truncation error* $T_i^{(N)}$ cf. [2], which satisfies the homogeneous part of (II.3.2b):

(3.3) $\quad T_{i+1}^{(N)} = B_{i+1}\, T_i^{(N)}.$

Because of the boundedness of those solutions we have

(3.3a) $\quad \|T_N^{(N)}\| = O(1),$

whence,

(3.3b) $\quad \|T_i^{(N)}\| = O(e^{\lambda(\gamma - t_i)}).$

Hence if $e^{\lambda(\gamma-\beta)} \le$ TOL (TOL the required accuracy) this truncation error is not significant.

3.2 Conditioning

The system (2.5) is rank deficient, so the conditioning with respect to the BC (as was introduced in §II.3.4) has to be redefined here. Since we virtually rule out the increasing components we may define the *subcondition numbers* cf. [2] :

(3.4) $$CN_p(\beta) = \max_{t \in (\alpha,\beta)} \| F(t)(I-P)\,[M_\alpha F(\alpha) + M_\infty F(\beta)]^+ \|,$$

where $I-P = \begin{bmatrix} \varnothing & \varnothing \\ \varnothing & I_{k_b} \end{bmatrix}$ and + denotes a pseudo-inverse. By making use of the approximate $\{Q_i\, \Psi_i\}$ instead of $F^2(t_i)$, we can estimate $CN_p(\beta)$ by (cf. (II.3.13))

(3.5) $$\kappa_p(\beta) = \|\,[M_\alpha Q_1 \Psi_1 + M_\infty Q_M\, \Psi_M]^+\|.$$

3.3 Problems with polynomially increasing modes

If there exist increasing modes that grow "slower" than an exponential function of t, the construction in §3.1 to find a terminal point may result in exceedingly large values of γ. Under certain circumstances, however, we do not need to go that far.
In order to describe them, let $F^1(t)$ be split further into

(3.6) $$F^1(t) = [\,G^1(t) \mid G^2(t)\,],$$

where $G^2(t)$ is an $n \times k_q$-matrix representing the polynomially increasing modes, $G^1(t)$ an $n \times k_e$-matrix representing the exponentially increasing modes. We now consider two (non exclusive!) possibilities:

(i) $\lim_{t\to\infty} L(t),\ \lim_{t\to\infty} r(t)$ exists.

This means that both $w(t)$ and $G^2(t)$ have asymptotically constant directions. If we partition the truncation error $T_i^{(N)}$ in two components, viz.

$$(3.7)\qquad T_i^{(N)} = \begin{bmatrix} [T_i^{(N)}]^1 \\ [T_i^{(N)}]^2 \end{bmatrix},$$

where $[T_i^{(N)}]^1$ has k_e components, then it makes sense to try some asymptotic expansion for $[T_0^{(N)}]^2$, e.g.

$$(3.8)\qquad [T_0^{(N)}]^2 = v_0 + v_1 [t_N]^{-\omega} + v_2 [t_N]^{-2\omega} + \cdots,$$

where $\omega > 0$ and $v_0, v_1, \cdots$ are independent of t_N; obviously the user should provide the model for this.
If we apply this idea we see that the point γ is mainly determined by the exponential behaviour of $G^1(t)$ (cf. (3.3b)). On the other hand, in order to employ (3.8), one should choose several terminal conditions instead.

(ii) $\lim_{t \to \infty} w(t)$ exists.

This still allows fairly general ODE (in particular with a fundamental solution of which the directions are not asymptotically constants). Because of boundedness of $w(t)$ we may try an asymptotic expansion like

$$(3.9)\qquad x(t) = u_0 + u_1 t^{-\omega} + u_2 t^{-2\omega} + \cdots,$$

where ω and $u_0, u_1, \cdots$ are independent of t (we assume $t - \alpha$ large enough); again the user should provide the proper model. If we choose γ large enough, so that exponentially increasing modes have been damped out within TOL on $[\alpha, \beta]$, we can employ (3.9) in combination with (2.8) (note that $w_i(t_i) = 0$). Indeed, within TOL, we may write for the actually found solution $\hat{x}$:

$$(3.10a)\qquad \hat{x}(t_i) := x(t_i) + e(t_i),$$

with

$$(3.10b)\qquad \hat{x}(t_i) = Q_i (\Psi_i c^2 + z_i)$$

$$(3.10c)\qquad e(t_i) = Q_i \hat{\Psi}_i \hat{c},$$

where $\hat{\Psi}_i$ is an $n \times k_q$-matrix, representing the polynomially increasing modes and $\hat{c}$ a constant k_q-vector, only depending on the choice of γ. Now one should realize that $\{\hat{\Psi}_i\}$ can be computed in much the same way as $\{\Psi_i\}$. The only difference is that we use a recursion like (2.7) with B_i as the incremental matrix instead and a partitioning such that the left upper block is k_e. From this we see that $e(t_i)$ is in fact completely determined by the unknown $\hat{c}$; $\hat{c}$ in turn can in principle be found together with the vectors $u_0, u_1, \cdots$ from monitoring $\hat{x}(t_i)$ for various values of t_i. Note that we only need k_q points t_i to find $\hat{c}$ in case x is a constant vector.

4. Computational aspects

The code MUTSIN is based on the computational framework as outlined in chapter II. Some special aspects are considered below.

4.1 Determination of γ and bounded solutions

In order to find a suitable value for γ, MUTSIN keeps track of the diagonal elements of the B_i (cf. §4.3). In order to estimate a λ as in (3.2) it takes

$$\lambda := (\ln m) / (\beta - \alpha), \tag{4.1}$$

where m is the absolutely smallest diagonal element of $\prod_1^N B_i$. From this a value of γ is computed as

$$\gamma := \beta - \frac{\ln \mathrm{TOL}}{\lambda}, \tag{4.2}$$

Arriving at $t = \gamma$ it is checked whether the increment is large enough indeed, and if necessary a new (larger) γ is computed, using an updated λ. If the latter value of γ is still insufficient to give large enough increments, a warning error IERROR = 335 occurs. It may happen that γ as defined by (4.2) is already quite large (due to a pessimistic choice of the partitioning parameter k_u). Therefore the user should provide a maximum value of γ, γ_{max} say, If γ becomes larger than γ_{max}, γ_{max} is taken as the value for γ and a warning error IERROR = 330 occurs.

4.2 Use of BC and determination of conditioning constants

System (2.12) can be written as

$$[M_\alpha Q_1 [\varnothing \mid \Psi_1] + M_\infty Q_M [\varnothing \mid \Psi_M]] \begin{bmatrix} 0 \\ c_2 \end{bmatrix} = \hat{b}, \tag{4.3}$$

where $\hat{b} = b - M_\alpha Q_1 z_1 - M_\infty Q_M z_M$. To solve (4.3) a singular value decomposition (SVD) is used, that is we determine orthogonal matrices U, V and a semi-positive diagonal matrix Σ, such that

$$M_\alpha Q_1 [\varnothing \mid \Psi_1] + M_\infty Q_M [\varnothing \mid \Psi_M] = U \Sigma V^T, \tag{4.4}$$

where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, with $\sigma_1 \geq \cdots \geq \sigma_{k_b} \geq 0$, $\sigma_{k_b+1} = \cdots = \sigma_n = 0$, and $\begin{bmatrix} 0 \\ c_2 \end{bmatrix} = V y$. Then (4.3) can be rewritten as

(4.5) $\quad \Sigma y = U^T \hat{b}$.

To have a meaningful solution of (4.5) it is necessary that the vector $U^T \hat{b} = (\xi_1, \ldots, \xi_n)^T$ satisfies the conditions

(4.6) $\quad \sigma_i = 0 \Longrightarrow \xi_i = 0\,,\ i=1, \ldots, n$.

We call the problem *inconsistent* with respect to the BC if (4.6) is false. Numerically we consider σ_i to be zero if the computed $\sigma_i \leq TOL$ and hence we check whether

(4.7) $\quad \sigma_i \leq \mathrm{TOL} \Longrightarrow \xi_i \leq \mathrm{TOL}\ \ , i=1, \ldots, n$

is true or false. If (4.7) is false a warning error IERROR = 340 is given. It is possible that IERROR = 340 occurs after the warning error IERROR = 335. In that case IERROR = 335 is likely to cause IERROR = 340 too.

If we write $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_l, 0, \ldots, 0)$ $(l \leq k_b)$, we can define its pseudo-inverse as $\Sigma^+ = \mathrm{diag}(\sigma_1^{-1}, \ldots, \sigma_l^{-1}, 0, \ldots, 0)$ and hence solve (4.5) in a straightforward manner. For a well-posed problem we should expect $l = k_b$, so we have as an estimate for the condition number:

(4.8) $\quad \kappa = [\sigma_{k_b}]^{-1}$.

If $[\sigma_{k_b}]^{-1} > TOL^{-1}$ we should call the problem ill-conditioned (as TOL means numerically zero) and a warning error IERROR = 345 is given. In such a case, and -more generally- if $\sigma_{l+1}, \ldots, \sigma_{k_b}$ are smaller than or equal to TOL, we choose

(4.9) $\quad \kappa = \sigma_l^{-1}$,

unless $l+1 = 1$. Although clearly we cannot give a unique solution then, we can still give a basis of a meaningful manifold, viz. those components that can be found from singular vectors corresponding to $\sigma_{l+1}, \ldots, \sigma_{k_b}$. Let us write

(4.10) $\quad V = [\, v_1 | \ \cdots \ | \, v_{k_b}]$,

then these basis solutions are defined by

(4.11) $\quad \{Q_i \Psi_i v_j\}_{i=1}^{N}\,,\ j = l+1, \ldots, k_b$.

From the pseudo-inverse we get some bounded particular solution as well.
Clearly uniqueness requires more independent conditions in (2.2).

4.3 Use of MUTSIN for problems with slowly increasing modes

For problems without an exponential dichotomy MUTSIN may fail to compute a bounded solution being accurate up to TOL. If the warning error IERROR = 335 occurs, there might be some non-exponential growing modes. It is also possible that the problem is not dichotomic (in which case ER(5) should be large). When there are non-exponentially growing modes MUTSIN can still be used in combination with asymptotic expansions.
First consider case (i) of §3.3. One should then set IEXT equal to 1 and C equal to a desired new value of γ. A new call to MUTSIN results in the computation of a new solution using the new value of γ. This means that one can use approximate solutions for various γ and hence utilize asymptotics. Because of the variety of possible expansions the user should write himself a program that calls MUTSIN and then uses Richardson extrapolation (for instance). Obviously, denoting the approximate value of $x(\alpha)$ obtained from using γ as a terminal point by $x_\gamma(\alpha)$, it follows from an assumption like (3.8) that also $x_\gamma(\alpha)$ has an expansion in $\gamma^{-\omega}$.
In case (ii) of §3.3 the fundamental solutions $\hat{\Psi}_i$ and Ψ_i are stored in the array PHIREC. Then not only an approximate $x_\gamma(\alpha)$ is given but also the values of the non-exponentially increasing solutions at the output points.
When applying the previous idea, one should realize that all computations are exact within $O(\text{TOL})$. This implies that under circumstances it is advisable to choose the parameter TOL fairly small in order to have a vector for which Richardson extrapolation is still meaningful. Also, the code is designed to choose γ as small as possible when slowly increasing modes (that should not influence its choice !) are detected. If γ happens to be equal to γ_{max}, the actual found partitioning integer k_e is based on the criterion that exponentially growing modes should at least correspond to a λ (cf. (4.1)) such that (4.2) is satisfied. Hence the value C − A $(=\gamma_{max}-\alpha)$ should not be chosen too small compared to the interval length B − A $(=\beta-\alpha)$, the latter being considered to be relevant for the problem as such.

References

[1] R.M.M. Mattheij, *On the computation of solutions of BVP on infinite intervals*, Math. Comp. 48 (1987), 533-549.

[2] F.R. de Hoog, R.M.M. Mattheij, *On non-invertible boundary value problems*, Numerical Boundary Value ODE's, (U. Ascher, R. Russell, eds.), Birkhäuser (1985), 55-76.

CHAPTER IV

MULTIPOINT BVP AND INTEGRAL BVP

1. Introduction

In this section we first describe the problem briefly.
Consider the ODE:

$$(1.1) \qquad \frac{d}{dt}x(t)=L(t)x(t)+r(t)\,, \quad \alpha \le t \le \beta\,,$$

where $L(t)$ is an $n \times n$-matrix function and $x(t)$ and $r(t)$ are n-vector functions. Let for $x(t)$ the boundary condition (BC) be given:

$$(1.2) \qquad M_1 x(\alpha_1)+M_2 x(\alpha_2)+\cdots+M_{m+1} x(\alpha_{m+1})=b\,,$$

where $M_1, \ldots, M_{m+1}$ are $(n \times n)$-matrices, b is an n-vector; the points $\alpha_1, \ldots, \alpha_{m+1}$, with $\alpha=\alpha_1<\alpha_2<\cdots<\alpha_{m+1}=\beta$, are the so called *switching points*.
A possible way to solve a multipoint BVP (1.1), (1.2) is to map the intervals $[\alpha_i, \alpha_{i+1}]$, $i=1, \ldots, m$ onto one and the same interval $[0,1]$ say and solve for the solution on these intervals simultanously. Denoting the solution at $[\alpha_i, \alpha_{i+1}]$ by $x_i(t)$ we thus have

$$(1.1)' \qquad \frac{d}{dt}\mathbf{x}(t)=\mathbf{L}(t)\mathbf{x}(t)+\mathbf{r}(t),$$

where $\quad \mathbf{x}(t)=[x_1^T(t), \ldots, x_m^T(t)]^T$, $\mathbf{r}(t)=[r_1^T, \ldots, r_m^T(t)]^T$ and

$$\mathbf{L(t)}=\begin{bmatrix} L_1(t) & & \\ & \cdot & \\ & & \cdot \quad L_m(t) \end{bmatrix},$$

(where $L_i(t)$ and $r_i(t)$ are properly transformed from $[\alpha_1, \alpha_{i+1}] \rightarrow [0,1]$). A coresponding two-point BC is now given by

$$(1.2)' \qquad \mathbf{M}_0\mathbf{x}(0)+\mathbf{M}_1\mathbf{x}(1)=\mathbf{b}$$

where $\quad \mathbf{b}=[0^T, \cdots, 0^T, b^T]^T$ and

$$\mathbf{M}_0 = \begin{bmatrix} \varnothing & I & & \\ & & \cdot & \\ & & & I \\ M_1 & . & . & M_m \end{bmatrix}, \quad \mathbf{M}_1 = \begin{bmatrix} -I & & \\ & \cdot & \\ & & -I \\ & & & M_{m+1} \end{bmatrix}.$$

For (1.1)′ and (1.2)′ one can use a routine of chapter II. Note however that this system has order $n \times m$ now! Hence we look for a cheaper solution.

Because of the linearity of (1.1) we may write the solution $x(t)$ as:

(1.3) $$x(t) = F(\alpha_i, t)\, c_i + w(\alpha_i, t), \quad \alpha_i \le t \le \alpha_{i+1},$$

where $F(\alpha_i, t)$ is a fundamental solution on $[\alpha_i, \alpha_{i+1}]$ and $w(\alpha_i, t)$ a particular solution of (1.1) on $[\alpha_i, \alpha_{i+1}]$. In principle we may identify $F(\alpha_i, t)$ with $F(\alpha_j, t)$ for $i \neq j$, thus reducing (1.3) to the well known superposition of solutions. However, as was shown in [1] the dichotomy character might be different on each subinterval: that is the dimension of the non decreasing mode subspace may become smaller after such a point a_i; this is called *polychotomy*. Hence it makes sence to consider the $F(\alpha_i, t)$ separately, at least computationally, cf. [2]. Matching in the usual way gives us the relation for the c_i. We obtain:

(1.4a) $$F(\alpha_i, \alpha_{i+1})\, c_i = F(\alpha_{i+1}, \alpha_{i+1})\, c_{i+1} + w(\alpha_{i+1}, \alpha_{i+1}) - w(\alpha_i, \alpha_{i+1})$$

and the BC

(1.4b) $$M_1 F(\alpha_1, \alpha_1) c_1 + \cdots + [M_m F(\alpha_m, \alpha_m) + M_{m+1} F(\alpha_m, \alpha_m)]\, c_m = \hat{b},$$

$$\hat{b} := b - M_1 w(\alpha_1, \alpha_1) - \cdots - M_m(\alpha_m, \alpha_m) - M_{m+1}(\alpha_m, \alpha_{m+1}).$$

The method now uses multiple shooting on each interval $[\alpha_i, \alpha_{i+1}]$. In this way we obtain a discrete analogue of (1.4a) and (1.4b) which constitutes a linear system A of order $m \times n$. The conditioning of the problem can be measured by $\|A^{-1}\|$ as well as by monitoring the growth behaviour of the fundamental solutions. These quantities are actually accounted for by the routine, see §4.

Remark 1.5.

If the dichotomy does not change on consecutive intervals $[\alpha_i, \alpha_{i+1}], \ldots, [\alpha_{i+k}, \alpha_{i+k+1}]$ say, the fundamental solutions $F(\alpha_{i+l}, t)$ $l = 1, \ldots, k$ can be identified with $F(\alpha_i, t)$, the particular solutions $w(\alpha_{i+l}, t)$, $l = 1, \ldots, k$ with $w(\alpha_i, t)$ and the c_{i+l}, $l = 1, \ldots, k$ with c_i. As a consequence (1.4a,b) change into

(1.6a) $$F(\alpha_j, \alpha_{j+1})\, c_j = F(\alpha_{j+1} \alpha_{j+1})\, c_{j+1} + w(\alpha_{j+1}, \alpha_{j+1}) - w(\alpha_j, \alpha_{j+1}),$$

$$j = 1, \ldots, i-1 \text{ and } j = i+k+1, \ldots, m,$$

$$(1.6b) \qquad F(\alpha_i,\alpha_{i+k+1})c_i = F(\alpha_{i+k+1},\alpha_{i+k+1})c_{i+k+1} + w(\alpha_{i+k+1},\alpha_{i+k+1}) - w(\alpha_i,\alpha_{i+k+1}).$$

$$(1.6c) \qquad \sum_{l=1}^{i-1} M_l F(\alpha_l,\alpha_l)c_l + [\sum_{l=i}^{i+k} M_l F(\alpha_i,\alpha_l)]c_i +$$

$$+ \sum_{l=i+k+1}^{m} M_l F(\alpha_l,\alpha_l)c_l + M_{m+1}F(\alpha_m,\alpha_{m+1})c_m = \hat{b},$$

$$\hat{b} = b - \sum_{l=1}^{i-1} M_l w_l(\alpha_l,\alpha_l) - \sum_{l=i}^{i+k} M_l w(\alpha_i,\alpha_l) - \sum_{l=i+k+1}^{m+1} M_l w(\alpha_l,\alpha_l).$$

This gives a linear system of order $(m-k)\times n$.

If we consider the limit case where the number of switching points goes to infinity (and the weight M_i are scaled appropiately), we arrive at an *integral condition*

$$(1.7) \qquad \int_\alpha^\beta M(t)x(t)\,dt = b,$$

where $M(t)$ is an $n\times n$ matrix function and b an n-vector. This requires an extra discretisation for casting the problem into a form compatible with multipoint BC. Another way, though often more costly than the method we shall outline below, is to augment (1.1) with $\frac{d}{dt}y(t) = M(t)x(t)$, $y(\alpha)=0$, so that we have an ODE

$$(1.1)'' \qquad \frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} L(t) \\ M(t) \end{bmatrix} x(t) + \begin{bmatrix} r(t) \\ 0 \end{bmatrix}$$

and a (two-point) BC

$$(1.2)'' \qquad \begin{bmatrix} \varnothing & \varnothing \\ \varnothing & I \end{bmatrix}\begin{bmatrix} x(\alpha) \\ y(\alpha) \end{bmatrix} + \begin{bmatrix} \varnothing & I \\ \varnothing & \varnothing \end{bmatrix}\begin{bmatrix} x(\beta) \\ y(\beta) \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

Obviously, the ODE (1.1)" is of order $2n$.

Finally, it is possible to have a combination of a multipoint (including two-point) and integral BC. A mixed condition has the form (1.8a,b)

$$(1.8a) \qquad \sum_{i=1}^{m+1} {}^1M_i\, x(\alpha_i) = b^1$$

$$(1.8b) \qquad \int_\alpha^\beta {}^2M(t)x(t) = b^2,$$

where for some $l<n$, ${}^1M_1,\ldots,{}^1M_{m+1}$ are $l\times n$ matrices and ${}^2M(t)$ is an $(n-l)\times n$ matrix function.

Remark 1.9

Sometimes a BC describing e.g. discontinuities at certain points is confusingly called a multipoint BC. However, as can be checked those discontinuities would increase the total number of BC beyond n. If this is the case one should use the methods described in chapter VI.

The algorithm discussed in this chapter have been implemented in the routines MUTSMP for BC of type (1.2) and MUTSMI for BC of type (1.7) or (1.8).

2. Global description of the algorithms

We shall consider the multipoint and integral case separately.

2.1 BVP with multipoint BC

As mentioned in § 1, multiple shooting is used on each interval $[\alpha_i, \alpha_{i+1}]$ to compute a fundamental solution and a particular solution. Each interval $[\alpha_i, \alpha_{i+1}]$ is divided into say N_i-1 subintervals. To simplify the notation we shall use a *local index* j to describe them; i.e. let the interval $[\alpha_i, \alpha_{i+1}]$ be split up into subintervals $[t_{j-1}, t_j]$, $j=2, \ldots, N_i$, $t_1=\alpha_i$ and $t_{N_i}=\alpha_{i+1}$.
Like in the algorithm described in [3] for two-point BVP, fundamental solutions $F_j(\alpha_i, t)$ and particular solutions $w_j(\alpha_i, t)$ are computed such that:

$$(2.1) \qquad F_j(\alpha_i, t_{j+1}) = F_{j+1}(\alpha_i, t_{j+1})\, U_{j+1}(i) = Q_{j+1}(i)\, U_{j+1}(i),\ j=1, \ldots, N_i-1,$$

where the $Q_{j+1}(i)$ are orthogonal and the $U_{j+1}(i)$ upper triangular and $w_j(\alpha_i, t_j)=0$. (Here we identify $F_1(\alpha_i, \alpha_i)$ with $F(\alpha_i, \alpha_i)$ and $w_1(\alpha_i, \alpha_i)$ with $w(\alpha_i, \alpha_i)$).
For the solution $x(t)$ we have:

$$(2.2) \qquad x(t) = F_j(\alpha_i, t)\, a_j(i) + w_j(\alpha_i, t),$$

from which the following upper triangular recursion for the $a_j(i)$ is obtained:

$$(2.3) \qquad a_{j+1}(i) = U_{j+1}(i)\, a_j(i) + d_{j+1}(i), \quad j=1, \ldots, N_i-1,$$

where

$$(2.4) \qquad d_{j+1}(i) = Q_{j+1}^{-1}(i)\, [w_j(\alpha_i, t_{j+1}) - w_{j+1}(\alpha_i, t_{j+1})].$$

Now assume that $\{\Phi_j(i)\}_{j=1}^{N_i}$ is a fundamental solution of (2.3) and $\{z_j(i)\}_{j=1}^{N_i}$ a particular solution. Then for some vector c_i we should have:

(2.5) $$a_j(i) = \Phi_j(i)\, c_i + z_j(i)\,, \quad j = 1, \ldots, N_i \,.$$

By matching at the points α_i we obtain a recursion for the $\{c_i\}$ in the usual way. So for the solution of the BVP at the switching points $\alpha_1, \alpha_2, \ldots, \alpha_{m+1}$ we have:

(2.6a) $$x(\alpha_i) = w_1(\alpha_i\,, \alpha_i) + Q_1(i)\,[\, z_1(i) + \Phi_1(i)\, c_i \,] \quad, i = 1, \ldots, m$$

and

(2.6b) $$x(\alpha_{i+1}) = w_{N_i}(\alpha_i, \alpha_{i+1}) + Q_{N_i}(i)\,[\, z_{N_i}(i) + \Phi_{N_i}(i)\, c_i \,]\,, i = 1, \ldots, m \,.$$

Substituting (2.6) in the *BC* gives a *BC* for the sequence $\{c_i\}_{i=1}^{m}$ (cf. (1.4b)) viz.

(2.7) $$M_1 Q_1(1)\Phi_1(1)\, c_1 + \cdots + [\, M_m Q_1(m)\Phi_1(m) + M_{m+1} Q_{N_m}(m)\Phi_{N_m}(m)\,]\, c_m = \hat{b}\,,$$

$$\hat{b} = b - \sum_{i=1}^{m} M_i\, Q_1(i)\, z_1(i) - M_{m+1}\, Q_{N_m}(m)\, z_{N_m}(m)$$

$$- \sum_{i=1}^{m} M_i\, w_1(\alpha_i\,, \alpha_i) - M_{m+1}\, w_{N_m}(\alpha_m\,, \alpha_{m+1}).$$

Denoting:

(2.8a) $$\hat{M}_i = M_i\, Q_1(i)\, \Phi_1(i) \quad i = 1, \ldots, m-1,$$

(2.8b) $$\hat{M}_m = M_m\, Q_1(m)\, \Phi_1(m) + M_{m+1}\, Q_{N_m}(m)\, \Phi_{N_m}(m)\,,$$

(2.8c) $$\Pi_i = \Phi_{N_i}(i)\,, \quad i = 1, \ldots, m-1\,,$$

(2.8d) $$\Omega_{i+1} = Q_{N_i}^{-1}(i)\, Q_1(i+1)\, \Phi_1(i+1)\,, \quad i = 1, \ldots, m-1\,,$$

(2.8e) $$q_i = Q_{N_i}^{-1}(i)\,[\, w(\alpha_{i+1}\,, \alpha_{i+1}) - w_{N_i}(\alpha_i\,, \alpha_{i+1})\,] +$$

$$Q_{N_i}^{-1}(i)\, Q_1(i+1)\, z_1(i+1) - z_{N_i}(i) \quad, \quad i = 1, \ldots, m-1 \,.$$

we obtain the linear system:

(2.9a) $$\mathbf{A}\,\mathbf{c} = \mathbf{q}\,,$$

where

(2.9b)
$$A=\begin{bmatrix} \Pi_1 & -\Omega_2 & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \Pi_{m-1} & -\Omega_m \\ \hat{M}_1 & \hat{M}_2 & \cdot & \hat{M}_{m-1} & \hat{M}_m \end{bmatrix},$$

$$\mathbf{c}=[c_1^T,\ldots,c_{m-1}^T,c_m^T]^T,\ \mathbf{q}=[q_1^T,\ldots,q_{m-1}^T,\hat{b}^T]^T.$$

Remark 2.10

In the case the ODE (1.1) is homogeneous, i.e. $r(t)=0, t\in[\alpha,\beta]$, the computation of particular solutions is skipped. Then (2.2), (2.3), (2.5), (2.6) have to be replaced by:

(2.2)′ $\quad x(t_{j+1})=F_j(\alpha_i,t_{j+1})\,a_j(i)=F_{j+1}(\alpha_i,t_{j+1})\,a_{j+1}(i)$,

(2.3)′ $\quad a_{j+1}(i)=U_{j+1}(i)\,a_j(i)$,

(2.5)′ $\quad a_j(i)=\Phi_j(i)\,c_i\ ,j=1,\ldots,N_i$,

(2.6a)′ $\quad x(\alpha_i)=Q_1(i)\,\Phi_1(i)\,c_i\ ,i=1,\ldots,m$,

(2.6b)′ $\quad x(\alpha_{m+1})=Q_{N_m}(m)\,\Phi_{N_m}(m)\,c_m$,

respectively.
Moreover, the vector $\hat{b}$ in (2.7) equals b and the vector $\mathbf{q}$ in (2.9a) becomes:

$$\mathbf{q}=[0^T,0^T,\ldots,0^T,b^T]^T.$$

2.2 BVP with integral BC

When we have a BC like (1.7) the situation becomes more complicated in two ways: First, there are no natural candidates for switching points and second we need to use a quadrature formula to implement the integral conditiion practically.

By using a marching technique and orthogonalisation after a fairly small number of gridpoints, cf. (2.1), we have a means to check the growth behaviour of the various modes. When a change is noted at such a minor shooting point, we basically choose it as a switching point (the refinement of this idea is discussed in §3.2).
A more complicated problem is to discretise the BC. Assuming we have a quadrature formula of appropriate order (i.e. compatible with the integrator of the ODE), we determine approximations

(2.11) $$\int_{t_j}^{t_{j+1}} M(t)\,F_j(\alpha_i,t)\,dt \doteq M_j(i)$$

$$(2.12)\qquad \int_{t_j}^{t_{j+1}} M(t)\,w_j(t)\,dt \doteq v_j(i)$$

In discrete form the BC then results in

$$(2.13)\qquad \sum_{i=1}^{m}\sum_{j=1}^{N_i-1} M_j(i)\,\alpha_j(i) = b - \sum_{i,} j\ v_j(i) =: \bar{b}\ ,$$

(where we use the same notation for indices as in §2.1)
By substituting (2.5) in (2.13) we find the multipoint BC

$$(2.14)\qquad \sum_{i=1}^{m}\Big[\sum_{j=1}^{N_i-1} M_j(i)\,\Phi_j(i)\Big]c_i = \hat{b} := \bar{b} - \sum_{i=1}^{m}\sum_{j=1}^{N_i-1} M_j(i)\,z_j(i)\ .$$

If we denote

$$(2.15)\qquad \hat{M}_i := \sum_{j=1}^{N_i-1} M_j(i)\,\Phi_j(i)\ ,$$

and Π_i, Ω_i, q_i as in (2.8c,d,e), then we end up with a system like (2.9a,b) for the unknown vector **c**.

3. Special features of the methods

The actual computation of the solutions $F(\alpha_i, t)$ and $w(\alpha_i, t)$ on each interval is basically the same as described in [§II.3], i.e. the algorithm uses the adaptivity feature for the integration of the particular mode only. It also uses the decoupled form of the recursion (2.3) for the computation of $\Phi_j(i)$ and $z_j(i)$. Below we summarize some more aspects.

3.1 Computation of the $\Phi_j(i)$

As was shown in [1] a well conditioned multipoint boundary value problem is dichotomic on each interval $[\alpha_i, \alpha_{i+1}]$. However, we basically should reckon with a different partitioning integer k_p (cf. §II.3.2), indicating the dimension of the nondecreasing solution space, on each such interval. If we denote this integer at the i^{th} interval by $k(i)$, then we know from [1] that for well conditioned multipoint boundary value problems, $k(i)$ is a non-increasing set, i.e. $k(1) \geq k(2) \geq \cdots \geq k(m)$. The fundamental solution $\{\Phi_j(i)\}_{j=1}^{N_i}$ cf.(2.3) on the i^{th} interval is then computed using the BC:

$$(3.1)\qquad \Phi_1^2(i) = [\varnothing \mid I_{n-k(i)}]\ ;\quad \Phi_{N_i}^1(i) = [I_{k(i)} \mid \varnothing]\ ,$$

where the superscript refers to an obvious local partitioning involving the integer $k(i)$.

3.2 Choosing $F_1(\alpha_i, \alpha_i)$ and $w_j(\alpha_i, t_j)$

Like in the two-point case there is, in general, no information available for choosing the particular solution $w_j(\alpha_i, t)$ in a special way. Hence $w_j(\alpha_i, t_j) = 0$ is a good one, simplifying the formulae in (2.4)-(2.9) substantially. At $t = \alpha_1$ the algorithm initially chooses $Q_1(1) = F_1(\alpha_1, \alpha_1) = I$ and checks the ordering of the diagonal elements of the first upper triangular matrices $U_j(1)$, computed after reaching the endpoint of a minor shooting interval. If this ordering is found to be improper it performs a permutation of columns like in §II.3.3. Arriving at $t=\alpha_2$ we have a complete freedom to choose $F_1(\alpha_2, \alpha_2)$. A very useful choice is:

(3.2) $$F_1(\alpha_2, \alpha_2) = Q_{N_1}(1).$$

Indeed, if the dichotomy is invariant on $[\alpha_1, \alpha_3]$ we may proceed on $[\alpha_2, \alpha_3]$ like we did on the previous interval, thus computing an upper triangular recursion for the superposition vectors $a_j(1)$ and $a_j(2)$ combined. By formally writing

(3.3) $$a_j(2) =: a_{j+N_1}(1),$$

we may the extend the recursion (2.3) for $i = 1$ over the index range $j = 1, \ldots, N_1+N_2-1$. If $Q_{N_1}(1)$ is found not to be a good starting value on the interval $[\alpha_2, \alpha_3]$ (for similar reasons as the identity might be an improper starting matrix on $[\alpha_1, \alpha_2]$) a permutation of its columns is carried out until some satisfactory ordering on the diagonal of the upper triangular matices $U_j(2)$ has been found. Since for well conditioned multipoint BVP, $\{k(i)\}_{i=1}^{m}$ is a non-increasing set, a permutation is carried out on the first $k(i)$ columns of $Q_{N_1}(1)$ only.

Since the number of minor shooting intervals may be fairly large (cf. §II.4.2) assembling of these into major shooting intervals causes an additional problem for integral BC.
By using the notation in §II.4.2 of W_s and G_s, we see that we may write

(3.4) $$a_{j+1}(i) = W_j a_j(i) + G_j,$$

with $W_1 = U_2(i)$, $G_1 = d_2(i)$.
Hence for $l \leq N_i-1$

(3.5) $$\sum_{i=1}^{l} M_j(i)\, a_j(i) = \left(\sum_{i=1}^{l} M_j(i) W_j\right) a_{1(1)} + \sum_{i=1}^{l} M_j(i) G_j .$$

Whether l may be taken as large as N_i-1 depends on $\max_j \| M_j(i) W_j \|$. Indeed although W_j may be found in a relatively stable way, forming the (partial) sum $\sum_{i=1}^{l} M_j(i) W_j$ will invoke errors of the order of $\sum_{i=1}^{l} \| M_j(i) \| \, \| W_j \| \, EPS$ (where EPS is the machine constant). Since we expect $\| M_j(i) \|$ to be of a moderate size, the assembling to major shooting intervals should be confined to cases where $\| W_j \|$ does not exceed the characteristic stability constant $TOL \,/\, EPS$ (TOL being the required accuracy).

3.3 Reduction of the system (2.9)

If the choice (3.2) is a proper one then we can identify c_1 and c_2 in (2.5). so the system (2.9a) is of order $(m-1)\times n$ only, being of the form:

(3.6a) $$\hat{\mathbf{A}}\,\hat{\mathbf{c}} = \hat{\mathbf{q}}$$

where

(3.6b) $$\hat{\mathbf{A}} = \begin{bmatrix} \hat{\Pi}_1 & -\hat{\Omega}_3 & & & \\ & \Pi_3 & -\Omega_4 & & \\ & & \cdot & \cdot & \\ & & & \Pi_{m-1} & -\Omega_m \\ B_1 & B_3 & \cdot & \cdot & B_m \end{bmatrix}$$

$$\hat{\mathbf{c}} = [\,c_1^T, c_3^T, \ldots, c_m^T\,]^T \quad \mathbf{q} = [\,\hat{q}_1^T, q_3^T, \ldots, q_{m-1}^T, \hat{b}^T\,]^T$$

and where we have denoted for short ($L=N_1+N_2-1$) :

(3.6c) $\hat{\Pi}_1 = \Phi_L(1)\,;\ \ \hat{\Omega}_3 = Q_L^{-1}\,Q_1(3)\,\Phi_1(3)\,,$

(3.6d) $B_1 = M_1 Q_1(1)\Pi_1(1) + M_2 Q_{N_1}(1)\Pi_{N_1}(1)\,,$

$B_j = \hat{M}_j\,,\ j = 3,\ldots,m\,,$

(3.6e) $\hat{q}_1 = Q_L^{-1}(1)[\,w(\alpha_3,\alpha_3) - w_L(\alpha_1,\alpha_3)\,] + Q_L^{-1}(1)\,Q_1(3)\,z_1(3) - z_L(1)\,.$

Hopefully it will be clear how further reductions can be carried out now. Such a further reduction may arise either from an even longer interval $[\,\alpha_1, \alpha_l\,]$, $l>3$ where the dichotomy is invariant or from an invariance on other consecutive intervals. In particular it may happen that the order of the thus obtained matrix $\hat{\mathbf{A}}$ is just n; in such a situation we virtually have reduced the procedure to that of the two-point case.

Remark 3.7

Note that this reduction would make sense for integral BC as well (since assembling does not increase the norms of the BC matrices significantly), were it not that the sequential approach (cf. (3.2)) would also cause the $\|W_j\|$ (cf. (3.4)) to grow.

3.4 Special solution of the algebraic system (2.9)

Instead of solving the system (2.9) (or its condensed variant (3.6)) by LU-decomposition, we do the following: Rewrite the matrix $\mathbf{A}$ for simplicity as:

$$(3.8) \qquad \mathbf{A} = \begin{bmatrix} S_1 & -R_2 & & & & \\ & S_2 & -R_3 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & S_{N-1} & -R_N \\ T_1 & T_2 & \cdot & \cdot & T_{N-1} & T_N \end{bmatrix}$$

$$\mathbf{q}^T = [\, q_1^T, q_2^T, \ldots, q_N^T \,]$$

At the i^{th} switching point interval, let $k(i)$ be the partitioning integer, i.e. there are $k(i)$ increasing solutions at that interval. From [1] we know that $\{k(i)\}$ is a non-increasing set, i.e. we expect $k(1) \geq k(2) \geq \cdots \geq k(N-1) = k(N)$.
In the recursion (cf. (2.9) and (3.7))

$$(3.9) \qquad R_{i+1}\, c_{i+1} = S_i\, c_i - q_i \,,$$

we have

$$(3.10a) \qquad R_{i+1} = \begin{bmatrix} R_{i+1}^{11} & R_{i+1}^{12} \\ \varnothing & I \end{bmatrix},$$

where R_{i+1}^{11} is a $k(i) \times k(i)$ matrix and the identity matrix I is of order $n - k(i+1)$, and

$$(3.10b) \qquad S_i = \begin{bmatrix} I & \varnothing \\ \varnothing & S_i^{22} \end{bmatrix},$$

where S_i is a $(n - k(i)) \times (n - k(i))$ matrix and the identity matrix I is of order $k(i)$.

We now like to solve (3.9) plus BC again by superposition. Since we do not have a uniform dichotomy on $[\alpha, \beta]$ we use a more refined fundamental solution $\{\Psi_i\}_{i=1}^N$ (cf. §3.1). By assumption we let the partitioning depend on the index.

$$(3.11) \qquad \Psi_i = \begin{bmatrix} \Psi_i^{11} & \Psi_i^{12} \\ \varnothing & \Psi_i^{22} \end{bmatrix}, \quad \Psi_i^{11} \text{ of order } k(i)\,.$$

(At $i = N$ we have the same partitioning as for $i = N - 1$)
At $i = 1$ we define:

$$(3.12) \qquad [\, \Psi_1^{21} \mid \Psi_1^{22} \,] = [\, \varnothing \mid I_{n-k(1)} \,]\,,$$

and compute

$$(3.13a) \qquad \hat{\Psi}_2^{22} = S_1^{22}\, \Psi_1^{22}\,,$$

(For S_1^{22}, the right lower block of S_1, see (3.10)), where $\hat{\Psi}_2^{22}$ has the same order as S_1^{22} and $\hat{\Psi}_1^{22}$.

Now compute Ψ_2^{22} as follows:

$$\Psi_2^{22} = \begin{bmatrix} I_{k(1)-k(2)} & \varnothing \\ \varnothing & \hat{\Psi}_2^{22} \end{bmatrix}, \tag{3.13b}$$

if $k(1) > k(2)$ and $\Psi_2^{22} = \hat{\Psi}_2^{22}$ otherwise.

and from this $\hat{\Psi}_3^{22}$ etc.. In general we have

$$\hat{\Psi}_{i+1}^{22} = S_i^{22}\Psi_i^{22}, \tag{3.14a}$$

$$\Psi_{i+1}^{22} = \begin{bmatrix} I_{k(i)-k(i+1)} & \varnothing \\ \varnothing & \hat{\Psi}_{i+1}^{22} \end{bmatrix}, \tag{3.14b}$$

if $k(i) > k(i+1)$ and $\Psi_{i+1}^{22} = \hat{\Psi}_{i+1}^{22}$ otherwise.

At $i=N$ we set

$$[\Psi_N^{11} \mid \Psi_N^{12}] = [I_{k(N-1)} \mid \varnothing]. \tag{3.15}$$

Then we have

$$\Psi_N = \begin{bmatrix} \Psi_N^{11} & \varnothing \\ \varnothing & \Psi_N^{22} \end{bmatrix} = \begin{bmatrix} \hat{\Psi}_N^{11} & \hat{\Psi}_N^{12} \\ \varnothing & \hat{\Psi}_N^{22} \end{bmatrix}, \tag{3.16}$$

where $\hat{\Psi}_N^{11}$ is of order $k(N-1)$, $\hat{\Psi}_N^{12}$ $k(N-1)\times(n-k(N-1))$ and $\hat{\Psi}_N^{22}$ is of order $n-k(N-1)$ (the latter already being computed in the forward sweep). Next we have

$$\Psi_{N-1}^{12} = R_N^{11}\hat{\Psi}_N^{12} + R_N^{12}\hat{\Psi}_N^{22}, \tag{3.17a}$$

$$\Psi_{N-1}^{11} = R_N^{11}\hat{\Psi}_N^{11}. \tag{3.17b}$$

And in general:

$$\Psi_i = \begin{bmatrix} \Psi_i^{11} & \Psi_i^{12} \\ \varnothing & \Psi_i^{22} \end{bmatrix} = \begin{bmatrix} \hat{\Psi}_i^{11} & \hat{\Psi}_i^{12} \\ \varnothing & \hat{\Psi}_i^{22} \end{bmatrix}, \tag{3.18}$$

where Ψ_i^{11} is of order $k(i)$ and $\hat{\Psi}_i^{11}$ is of order $k(i-1)$.
Then:

$$\Psi_{i-1}^{12} = R_i^{11}\hat{\Psi}_i^{12} + R_i^{12}\hat{\Psi}_i^{22}, \tag{3.19a}$$

$$\Psi_{i-1}^{11} = R_i^{11}\hat{\Psi}_i^{11}, \tag{3.19b}$$

Note that this scheme to compute $\{\Psi_i\}$ is a generalisation of the dichotomic case dealt with in chapter II.

Finally we compute a particular solution $\{p_i\}$, which is done in a similar way as the commputation of the fundamental solution. We start with

(3.20a) $$p_1^2 = 0 \quad , \quad p_N^1 = 0$$

(again the partioning here and below is local!). At each of the switching points where $k(i+1) < k(i)$ we add a sufficient number of zeros to obtain a larger second component vector, so for $i = 1, \ldots, N$

(3.20b) $$\hat{p}_{i+1}^2 := S_i^{22} p_i^2 - q_i^2 \ ;$$

(3.20c) $$p_{i+1}^2 = \hat{p}_{i+1}^2 \ , \quad \text{if } k(i) = k(i+1) \, ,$$

$$p_{i+1}^2 = \begin{bmatrix} \varnothing \\ \hat{p}_{i+1} \end{bmatrix} , \quad \text{if } k(i) > k(i+1) \, ,$$

i.e. the first $k(i) - k(i+1)$ elements of p_{i+1}^2 are 0.

At the backward sweep we typically compute

(3.20d) $$p_i = \begin{bmatrix} p_i^1 \\ p_i^2 \end{bmatrix} = \begin{bmatrix} \hat{p}_i^1 \\ \hat{p}_i^2 \end{bmatrix} ,$$

where $\hat{p}_i^1$ is a vector of order $k(i-1)$.

(3.20e) $$p_{i-1}^1 = R_i^{11} \hat{p}_i^1 + R_i^{12} \hat{p}_i^2 + q_{i-1}^1 \ ,$$

where q_{i-1}^1 represents the first $k(i-1)$ elements of q_{i-1}.
The solution $\{c_i\}$ of (2.9) is then given by:

(3.21) $$c_i = \Psi_i \, v + p_i \ ,$$

where the vector v can be found from:

(3.22) $$[\sum_{i=1}^{N} T_i \, \Psi_i \,] \, v = \hat{b} - \sum_{i=1}^{N} T_i \, p_i$$

3.5 Conditioning and stability

Since multipoint problems are essentially more complicated than two-point ones, the algorithm outlined before and - as a consequence - also its stability analysis is more difficult. As we already indicated, the homogeneous solution space is *polychotomic* , that is dichotomic on each interval $[\alpha_i , \alpha_{i+1}]$ and moreover such that non-decreasing basis solutions may become non-increasing at one of the switching points at most. Since the algorithm is tuned to monitor the particular dichotomy on each interval, it follows from arguments in § II.3.2 that the recursions are used in stable directions only (that is if we assume well-conditioning, so polychotomy cf. [1]). The only remaining problem then is the conditioning of the system in

(3.22), that is of the matrix W defined by

$$W := \sum_{j=1}^{m} \hat{M}_i \Psi_i \,. \tag{3.23}$$

One can show that in general, given m actually used switching points

$$\| W^{-1} \| \leq (m+1)\, CN \,, \tag{3.24}$$

where for a multipoint BC

$$CN := \max_{t \in [\alpha,\beta]} \| F(t) [\sum_{j=1}^{m+1} M_j F(\alpha_j)]^{-1} \| \,, \tag{3.25a}$$

or for an integral BC

$$CN := \max_{t \in [\alpha,\beta]} \| F(t) [\int_{\alpha}^{\beta} M(t) F(t)\, dt \,]^{-1} \| \tag{3.25b}$$

with $F(t)$ any fundamental solution. Note that (3.25) is a straightforward generalization of (II.3.12) and is a measure for amplifications of perturbations in the BC. For stability with respect to perturbations in the ODE as such we may monitor appropiate blocks of the upper triangular matrices, just as in the two-point case, cf. chapter II.

4. Computational aspects

The routine MUTSMP basically uses the same strategy for computing the upper triangular recursion on the intervals $[\alpha_i, \alpha_{i+1}]$, i=1,...,m as the routine MUTSGE for two-point BVP (see chapter II). Only the choice of the $Q_1(i)$, $i=2, \ldots, m$ (that is the orthogonal value for $F(\alpha_i, \alpha_i)$) and the computation of the k-partitionings are different (see next section). The computations of the $\{c_i\}_{i=1}^{m}$ is decribed in §3. Once knowing the c_i, the computation of the solution at the i^{th} interval $[\alpha_i \,, \alpha_{i+1}]$ is the same as in the two-point case (see chapter II). The routine MUTSMI computes a solution of a BVP with a mixed integral multipoint BC.

4.1 The computation of Q_1(i)

On the first interval $[\alpha_1 \,, \alpha_2]$ we do the same as in the two-point case, i.e. $Q_1(1) = I$ and if this is not a satifactory choice, the columns of $Q_1(1)$ are permuted such that diagonal($\prod_{j=2}^{N_1} U_j(1)$) is ordered. As a first choice for $Q_1(i)$, $i = 2, \ldots, m$ we take (see §3.2)

$$Q_1(i) = Q_{N_{i-1}}(i-1). \tag{4.1}$$

Since the dichotomic character of the solution space may change at each switching point, it may be necessary to carry out a permutation of columns of $Q_i(1)$. Anticipating that the problem is well-conditioned (i.e. the partitioning parameters satisfy $k(i-1) \geq k(i)$) no column interchanges are necessary for the last $n-k(i-1)$ columns. So an initial choice of $Q_1(i)$ is accepted if the first $k(i-1)$ elements of diagonal($\prod_{j=2}^{N_i} U_j(i)$) are ordered; otherwise a permutation of the first $k(i-1)$ columns of $Q_1(i)$ is carried out. At this stage the partitioning parameter $k(i)$ is computed as the number of elements of the first $k(i-1)$ elements of diagonal($\prod_{j=2}^{N_i} U_j(i)$) which are greater than 1 If no permutations are needed and $k(i-1)=k(i)$ then the two succesive intervals $[\alpha_{i-1}, \alpha_i]$ and $[\alpha_i, \alpha_{i+1}]$ are assembled (see §3.2). However, it is possible that, due to discretization errors, the computed $k(i)$ does not correspond to the proper partitioning. Therefore, after the above described procedure, globally correct partitioning parameters are determined.

4.2 The computation of $M_j(i)$ and $w_j(i)$

One of the problems for integral BC is to obtain sufficiently accurate approximations for $M_j(i)$ and $w_j(i)$ (cf. (2.11),(2.12)), that is such that their errors commensurate with errors caused by discretizing the ODE. The simplest way to do this is to apply the same integration formular for (2.11), (2.12) as used in RKF45: We apply RKF45 to the augmented particular problems (cf. §2.1)

$$\frac{d}{dt}\begin{bmatrix} F_j(\alpha_i,t) \\ M_j(\alpha_i,t) \end{bmatrix} = \begin{bmatrix} L(t) \\ M(t) \end{bmatrix} f_j(\alpha_i,t), \tag{4.2}$$

with $F_j(\alpha_i,t)=Q_j(i)$, $M_j(\alpha_i,t)=\varnothing$ and

$$\frac{d}{dt}\begin{bmatrix} w_j(\alpha_i,t) \\ v_j(\alpha_i,t) \end{bmatrix} = \begin{bmatrix} L(t) \\ M(t) \end{bmatrix} w_j(\alpha_i,t) + \begin{bmatrix} r(t) \\ \varnothing \end{bmatrix}, \tag{4.3}$$

with $w_j(\alpha_i,t_j)=0$, $v_j(\alpha_i,t_j)=0$. One should note that this yields

$$M_j(i) = M_j(\alpha_i, t_{j+1}) \tag{4.4}$$

$$v_j(i) = v_j(\alpha_i, t_{j+1}) . \tag{4.5}$$

As for other routines in this package, the adaptivity is used when computing $w_j(\alpha_i,t)$ only.

4.3 Determination of switching points α_i for integral BC

If we have integral BC (or a mixed integral multipoint BC) we do not know whether there are switching points nor where they possibly are. In view of the delicate way we have to choose the initial values of the fundamental solutions $F(\alpha_i,t)$, cf. §3.2, it is important to find a balance between checking incremental growth and concluding that a switch in the dichotomy patern has taken place.

We start off with the strategy as outlined in §3.2. An output point is certainly chosen if the accumulated sidepoint condition matrix $\sum_{j=1}^{l} M_j(1)$, cf.(3.5), is found to be larger than or equal to TOL/EPS, or any time before, when user requested. Initially, the method finds a partitioning $k(1)$ at the first minor shooting point and basically updates this index at each new (minor) shooting point; if necessary a permutation is carried out to obtain a correct ordering. For a switching point α_i, $1 < i < m+1$, we have: there is a mode which is growing on $[\alpha_1, \alpha_i]$ and is decreasing on $[\alpha_i, \alpha_m]$. Using this property a minor shooting point t_l, say, is considered to be a switching point α_i, say, if there is a diagonal element of $\prod_{j=2}^{l} U_j$ greater than 2 and the same diagonal element of U_{l+1} is less than 1. Here U_j is the incremental matrix of the fundamental solution on the minor shooting interval $[t_{j-1}, t_j]$. Because a constant mode may result in a diagonal element alternative greater then 1 and less than 1, due to discretization errors, only modes with an incremental growth greater than 2 on $[\alpha_1, \alpha_i]$ are considered.
Anticipating polychotomy only the first $k(i-1)$ diagonal elements have to be checked and a permutation on the next subinterval $[\alpha_i, \alpha_{i+1}]$ should be restricted to the first $k(i)$ columns only.
Note that there can be at most n switching point between α_1 and α_{m+1}.

4.4 Finding a globally correct partitioning

Although the algorithm tries to determine a correct partitioning parameter $k(i)$ on each interval $[\alpha_i, \alpha_{i+1}]$, its resolution of the growth behaviour of the various modes may be fairly small (e.g. if $\alpha_{i+1} - \alpha_i$ is small) and/or it may be misled by non growing- non decreasing modes. Since a normal (that is a well-conditioned) situation implies the existence of a non increasing sequence $\{k(i)\}$, we need a check on this and - if this ordering turn out not to be monotonic - an update. This is done by the following procedure:

step 1: Compute on each interval $[\alpha_i, \alpha_{i+1}]$, $i = 1, \ldots, m$, a partitioning parameter $k(i)$, where $k(i)$ is the number of elements of diagonal($\prod_{j=2}^{N_i} U_j(i)$), which are greater than 1.

step 2: Determine the lowest index l, where $k(l) > k(l-1)$. If no such index exists, goto step 8.

step 3: Determine the lowest index $j < l$, where $k(j) < k(l)$.

step 4: Determine the index $p > l$, where
$k(l) = k(l+1) = \cdots = k(p) \neq k(p+1)$

step 5: Compute a global partitioning parameter $\hat{k}(l)$ say, for the interval $[\alpha_j, \alpha_{p+1}]$ by checking the increments over $[\alpha_j, \alpha_{p+1}]$ in an obvious way, taking into account the various permutations at the switching points.

step 6: The new updated sequence $\{k(i)\}_{i=1}^{m}$ is defined as

$$k(i) := \left[\begin{array}{ll} k(i) & i = 1, \ldots, j-1, p+1, \ldots, m \\ max(k(i), \hat{k}(l)) & i = j, \ldots, l-1 \\ \hat{k}(l) & i = l, \ldots, p \end{array}\right. .$$

step 7: Go back to step 2.

step 8: The current sequence $\{k(i)\}_{i=1}^{m}$ is correct.

With this procedure we get, at least theoretically, a good choice for the sequence of the $k(i)$. However, if the problem is not polychotomic also this procedure may not be satisfactory, naturally, and a large amplification factor may result (as is to be expected of course).

4.5 The computation of stability constants

Since the algorithm computes fundamental solutions at (possibly "enlarged") switching intervals, it does some bookkeeping of stability constants. The computations of the stability constant CN (see §3.5) is a straightforward matter and its value can be found in ER(4). If in (3.22) the matrix $[\sum_{i=1}^{N} T_i \Psi_i]$ is numerically singular a terminal error IERROR = 320 is given. Concerning the "amplification factor", which is an estimate for the Green's functions, the algorithm computes an estimate for this on each interval. Therefore the output value in ER(5) is the maximum of such factors over the entire region. If the amplification factor is such the the global rounding error is greater than the discretization error, a warning error, IERROR = 300, is given.

Remark 4.4

If the partitioning is incorrect, we may expect at least ER(5) to be "large". On the other hand, due to the special way the algorithm tries to seek the appropiate partitionings, it should be expected that a large value of ER(5) has to be attributed to the problem.

References

[1] F.R. de Hoog, R.M.M. Mattheij, *An algorithm for solving multipoint boundary value problems*, Computing, 38 (1987) pp. 219-234.

[2] F.R. de Hoog, R.M.M. Mattheij, *On the conditioning of multipoint and integral boundary value problems*, SIAM J. Math. Anal. 20 (1989) pp. 200-214.

[3] R.M.M. Mattheij, G.W.M. Staarink, *An efficient algorithm for solving genereal linear two-point BVP*, SIAM J. Sci. Stat. Comp. 5 (1984), pp. 745-763.

CHAPTER V

BVP WITH PARAMETERS

1. Introduction

Some ODE contain one or more parameters which are to be determined along with the solution. They can be described by the ODE

$$(1.1) \qquad \frac{d}{dt}x(t) = L(t)x(t) + C(t)z + r(t), \quad \alpha \le t \le \beta,$$

where $L(t)$ is an $n \times n$-matrix function, $C(t)$ an $n \times l$-matrix function ($l \ge 1$), $x(t)$ and $r(t)$are n-vector functions and z is a constant l-vector, the vector of parameters. Note the linearity in x and z. (In the next chapter we shall consider ODE that contain products of $x(t)$ and scalar z, so-called *eigenvalue problems*.) Since both $x(t)$ and z are unknown, we need $n+l$ BC, which we assume to be two-point BC of the following form:

$$(1.2a) \qquad [M_\alpha | P_\alpha]\begin{bmatrix} x(\alpha) \\ z \end{bmatrix} + [M_\beta | P_\beta]\begin{bmatrix} x(\beta) \\ z \end{bmatrix} = \begin{bmatrix} b_x \\ b_z \end{bmatrix} = b,$$

where M_α, M_β are $(n+l)\times n$-matrices, P_α, P_β are $(n+l)\times l$-matrices, b_x is an n-vector and B_z is an l-vector. Since z is constant, the BC (1.2a) can also be written as

$$(1.2b) \qquad M_\alpha x(\alpha) + M_\beta x(\beta) + M_z z = b,$$

where $M_z = P_\alpha + P_\beta$.

We can augment (1.1) with

$$(1.3) \qquad \frac{d}{dt}z = 0,$$

thus having an ODE of order $n+l$:

$$(1.4) \qquad \frac{d}{dt}\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} L(t) & C(t) \\ \varnothing & \varnothing \end{bmatrix}\begin{bmatrix} x(t) \\ z \end{bmatrix} + \begin{bmatrix} r(t) \\ 0 \end{bmatrix}.$$

The BVP (1.2a), (1.4) is actually a two-point BVP of order $n+l$, and can be solved using the routines from chapter II. However, we rather like to preserve the lower-order form (1.1) and this requires some manipulations reminiscent of the multipoint case, chapter IV. In particular the homogeneous problem may be *skew polychotomic*, i.e. have *switching points* where the dichotomy splitting changes; here, however, the dimension of the subspace of non-decreasing modes is *in*creasing. As in the integral BC case these switching points are not known in advance. Actually it can be shown that the parameter BVP is the adjoint of a suitable integral/multipoint BVP, cf. [1].

In order to compute the solution of (1.1), (1.2), we apply a multiple shooting strategy as before. Denoting the switching points as $\alpha_1, \dots, \alpha_{m+1}$ ($\alpha_1 = \alpha$, $\alpha_{m+1} = \beta$), then the following three types of solutions are computed on each subinterval $[\alpha_i, \alpha_{i+1}]$:

(i) $F(\alpha_i, t)$, being a fundamental solution of (1.1);

(ii) $Z(\alpha_i, t)$, being an $n \times l$-matrix function satisfying

(1.5) $$\frac{d}{dt} Z(\alpha_i, t) = L(t) Z(\alpha_i, t) + C(t);$$

(iii) a particular solution $w(\alpha_i, t)$ of (1.1) for $z = 0$, i.e. satisfying

(1.6) $$\frac{d}{dt} w(\alpha_i, t) = L(t) w(\alpha_i, t) + r(t).$$

It follows then that there exists a vector c_i such that

(1.7) $$x(t) = F(\alpha_i, t) c_i + Z(\alpha_i, t) z + w(\alpha_i, t) \quad , \alpha_i \le t \le \alpha_{i+1}.$$

Matching at the switching points yields the following relation for the c_i:

(1.8) $$F(\alpha_{i+1}, \alpha_{i+1}) c_{i+1} = F(\alpha_i, \alpha_{i+1}) c_i + [Z(\alpha_i, \alpha_{i+1}) - Z(\alpha_{i+1}, \alpha_{i+1})] z + w(\alpha_i, \alpha_{i+1}) - w(\alpha_{i+1}, \alpha_{i+1})$$

and the BC

(1.9) $$M_\alpha F(\alpha_1, \alpha_1) c_1 + M_\beta F(\alpha_m, \alpha_{m+1}) c_m + [M_\alpha Z(\alpha_1, \alpha_1) + M_\beta Z(\alpha_m, \alpha_{m+1}) + M_z] z = b - M_\alpha w(\alpha_1, \alpha_1) - M_\beta w(\alpha_m, \alpha_{m+1}).$$

The relations (1.8), (1.9) constitute a linear system for the unknowns $c_1, \dots, c_m$ and z; the order of the matrix is $m \times n + l$.
The algorithm discussed in this chapter has been implemented in the routine MUTSPA.

2. Global description of the algorithm

As in the multipoint case, cf. §IV.2.1, we use multiple shooting with minor shooting points t_j on each interval $[\alpha_i, \alpha_{i+1}]$. (So again the index j is local). We start the integration at $t_1 = \alpha_1$ with $w_1(\alpha_1, t_1) = 0$, $F_1(\alpha_1, t_1) = I$ and $Z_1(\alpha_1, t_1) = \varnothing$. At the next (minor) shooting point t_{j+1} ($j = 1, \ldots, N_i - 1$) we similarly choose $w_{j+1}(\alpha_1, t_{j+1}) = 0$, $Z_{j+1}(\alpha_1, t_{j+1}) = \varnothing$ and the initial value for $F_{j+1}(\alpha_1, t_{j+1})$ via

(2.1) $$F_j(\alpha_1, t_{j+1}) = F_{j+1}(\alpha_1, t_{j+1})\, U_{j+1}(1) = Q_{j+1}(1)\, U_{j+1}(1) \ ,$$

where $Q_{j+1}(1)$ is orthogonal and $U_{j+1}(1)$ is upper triangular.
When, for $j > 1$ it is found that the growth of any of the various modes (as can be monitored from the diagonal of the $U_j(1)$) is changing from decreasing to increasing, a switching point α_2 is chosen and the marching is continued, etc.
On a general interval $[\alpha_i, \alpha_{i+1}]$ we have for suitable $a_j(i)$

(2.2) $$x(t) = F_j(\alpha_i, t)\, a_j(i) + Z_j(\alpha_i, t)\, z + w_j(\alpha_i, t) \, ,$$

which gives the following recursion for the $a_j(i)$:

(2.3) $$a_{j+1}(i) = U_{j+1}(i)\, a_j(i) + C_{j+1}(i)\, z + d_{j+1}(i) \ , j = 1, \ldots, N_i - 1 \, ,$$

where

(2.4a) $$d_{j+1}(i) = Q_{j+1}^{-1}(i)\, [\, w_j(\alpha_i, t_{j+1}) - w_{j+1}(\alpha_i, t_{j+1})\,] \, ,$$

(2.4b) $$C_{j+1}(i) = Q_{j+1}^{-1}(i)\, [\, Z_j(\alpha_i, t_{j+1}) - Z_{j+1}(\alpha_i, t_{j+1})\,] \, .$$

Let $\{\Phi_j(i)\}_{j=1}^{N_i}$ be a fundamental solution of (2.3), $\{Y_j(i)\}_{j=1}^{N_i}$ a particular matrix solution of

(2.5) $$Y_{j+1}(i) = U_{j+1}(i)\, Y_j(i) + C_{j+1}(i) \, ,$$

and $\{z_j(i)\}_{j=1}^{N_i}$ a particular solution of (2.3) with $z = 0$, then for some suitable vector c_i we have

(2.6) $$a_j(i) = \Phi_j(i)\, c_i + Y_j(i)\, z + z_j(i) \, , j = 1, \ldots, N_i \, .$$

The sequence of vectors now can be found by matching at the points α_i and using the BC. We find

(2.7a) $$x(\alpha_i) = w_1(\alpha_i, \alpha_i) + Q_1(i)\, [\, z_1(i) + Y_1(i)\, z + \Phi_1(i)\, c_i\,] + Z_1(\alpha_i, \alpha_i)\, z \ ,$$

(2.7b) $$x(\alpha_{i+1}) = w_{N_i}(\alpha_i, \alpha_{i+1}) + Q_{N_i}(i)\,[\, z_{N_i}(i) + Y_{N_i}(i)\, z + \Phi_{N_i}(i)\, c_i \,] + Z_{N_i}(\alpha_i, \alpha_{i+1})\, z \ .$$

So for the BC we find

(2.8) $$B_\alpha c_1 + B_\beta c_m + B_z z = \hat{b},$$

where

(2.9a) $$B_\alpha = M_\alpha Q_1(1)\, \Phi_1(1),$$

(2.9b) $$B_\beta = M_\beta Q_{N_m}(m)\, \Phi_{N_m}(m),$$

(2.9c) $$B_z = M_\alpha [Q_1(1) Y_1(1) + Z_1(\alpha_1, \alpha_1)] + M_\beta [Q_{N_m}(m) Y_{N_m}(m) + Z_{N_m}(\alpha_m, \alpha_{m+1})] + M_z \ ,$$

(2.9d) $$\hat{b} = b - M_\alpha w_1(\alpha_1, \alpha_1) - M_\alpha Q_1(1) z_1(1) - M_\beta w_{N_m}(\alpha_m, \alpha_{m+1}) - M_\beta Q_{N_m}(m) z_{N_m}(m).$$

By finally denoting for $i = 1, \ldots, m-1$,

(2.10a) $$\Psi_i = \Phi_{N_i}(i)\ ,$$

(2.10b) $$\Omega_{i+1} = Q_{N_i}^{-1}(i)\, Q_1(i+1)\, \Phi_1(i+1)\ ,$$

(2.10c) $$D_i = Q_{N_i}^{-1}(i)\, Q_1(i+1)\, Y_1(i+1) - Y_{N_i}(i)\ ,$$

(2.10d) $$q_i = Q_{N_i}^{-1}(i)\, [w_1(\alpha_{i+1}, \alpha_{i+1}) - w_{N_i}(\alpha_i, \alpha_{i+1})] + Q_{N_i}^{-1}(i) Q_1(i+1)\, z_1(i+1) - z_{N_i}(i)\ ,$$

we obtain the linear system

(2.11a) $$\mathbf{A}\,\mathbf{c} = \mathbf{q}\ ,$$

where

(2.11b) $$\mathbf{A} = \begin{bmatrix} \Psi_1 & -\Omega_2 & & & & D_1 \\ & \cdot & \cdot & & & \cdot \\ & & \cdot & \cdot & & \cdot \\ & & & \Psi_{m-1} & -\Omega_m & D_{m-1} \\ B_\alpha & & & & B_\beta & B_z \end{bmatrix},$$

$$\mathbf{c}^T = [\, c_1^T, \ldots, c_{m-1}^T, c_m^T, z^T \,], \quad \mathbf{q}^T = [\, q_1^T, \ldots, q_{m-1}^T, \hat{b}^T \,].$$

Remark 2.12

If no switching point is detected, i.e. if $m = 1$, the matrix **A** simplifies to an $(n+l)^{\text{th}}$ order matrix

(2.11b)′ $\quad \mathbf{A} = [\, B_\alpha + B_\beta \mid B_z \,].$

Remark 2.13

If the ODE is homogeneous, i.e. $r(t) = 0, t \in [\alpha, \beta]$, there is no need to compute the particular solution of the ODE and the recursion. The expressions (2.3), (2.6), (2.7) and (2.9) should then be simplified accordingly, cf. remark IV.2.10.

3. Special features of the method

Many special aspects that were described for the multipoint and integral BC case in chapter IV also apply to the parameter problem considered in this chapter. They will be briefly indicated below, along with some other ones.

3.1 Computation of the $\Phi_j(i)$ and $Y_j(i)$

It can be shown that a well-conditioned parameter problem is skew polychotomic, with a dichotomic structure of the fundamental solution on each interval $[\, \alpha_i, \alpha_{i+1} \,]$. The dimension of the non-decreasing solution space at $[\, \alpha_i, \alpha_{i+1} \,]$, say $k(i)$, forms a non-decreasing sequence, i.e. $k(1) \le k(2) \le \cdots \le k(m)$. The fundamental solution $\{\Phi_j(i)\}_{i=1}^{N_i}$ is then found from (2.3) using the BC

(3.1) $\quad \Phi_1^2(i) = [\, \varnothing \mid I_{n-k(i)} \,] \;;\; \Phi_{N_i}^1(i) = [\, I_{k(i)} \mid \varnothing \,]\,.$

The particular matrix solution $\{Y_j(i)\}_{i=1}^{N_i}$ is similarly computed using the decoupled form of the recursion, cf. (2.5), and has the BC

(3.2) $\quad Y_1^2(i) = \varnothing \;;\; Y_{N_i}^1(i) = \varnothing\,.$

Note that $Y_1^2(i)$ is an $(n - k(i)) \times l$-matrix and $Y_{N_i}^1(i)$ a $k(i) \times l$-matrix.

3.2 Choosing $F_1(\alpha_i, \alpha_i)$, $Z_j(\alpha_i, t_j)$ and $w_j(\alpha_i, t_j)$

As before, the particular solution $w_j(\alpha_i, t_j)$ is chosen such that $w_j(\alpha_i, t_j) = 0$. Similarly, we choose $Z_j(\alpha_i, t_j) = \varnothing$.

The computation of $F_1(\alpha_1, t)$ is essentially the same as described in §IV.3.2. If a change of k-partitioning is noticed (here such that the subspace of non-decreasing modes is increased, rather than decreased as in the integral BC case) a new switching point α_2 is chosen. As initial value for $F_1(\alpha_2, \alpha_2)$ we take

$$(3.3) \qquad F_1(\alpha_2, \alpha_2) = Q_{N_1}(1) .$$

If $Q_{N_1}(1)$ is found not to be a good starting value on the interval $[\alpha_2, t]$, t suficiently large, a permutation of the last $n-k(1)$ columns of $Q_{N_1}(1)$ may be carried out to obtain a more appropriate ordering of the diagonal of the $U_j(2)$; this is of course a strategy complementary to the one outlined in §IV.3.2.

3.3 Special solution of the linear system (2.11)

The sparse system (2.11) is solved by a special technique in order to save both memory and computer time. Instead of (2.11) we rather consider the augmented system. Define

$$\hat{c}_i = \begin{bmatrix} z \\ c_i \end{bmatrix}, \quad \hat{q}_i = \begin{bmatrix} 0 \\ q_i \end{bmatrix}, \quad S_i = \begin{bmatrix} I & \varnothing \\ D_i & \Psi_i \end{bmatrix}, \quad R_{i+1} = \begin{bmatrix} -I & \varnothing \\ \varnothing & -\Omega_{i+1} \end{bmatrix}, B_1 = [\varnothing | B_\alpha], B_m = [B_z \,|\, B_\beta].$$

Then we have for the augmented system:

$$(3.4a) \qquad \hat{\mathbf{A}}\,\hat{\mathbf{c}} = \hat{\mathbf{q}},$$

where

$$(3.4b) \qquad \hat{\mathbf{A}} = \begin{bmatrix} S_1 & R_2 & & \\ & \ddots & \ddots & \\ & & S_{m-1} & R_m \\ B_1 & & & B_m \end{bmatrix}, \quad \hat{\mathbf{c}} = \begin{bmatrix} \hat{c}_1 \\ \vdots \\ \hat{c}_{m-1} \\ \hat{c}_m \end{bmatrix}, \quad \hat{\mathbf{q}} = \begin{bmatrix} \hat{q}_1 \\ \vdots \\ \hat{q}_{m-1} \\ \hat{b} \end{bmatrix}.$$

This linear system has the same structure as the linear system (I.3.7) which resulted from the discrete BVP (I.3.5),(I.3.6). In fact applying multiple shooting to the two-point BVP

$$(3.5a) \qquad \frac{d}{dt}\begin{bmatrix} z \\ x \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing \\ C(t) & L(t) \end{bmatrix}\begin{bmatrix} z \\ x \end{bmatrix} + \begin{bmatrix} \varnothing \\ r(t) \end{bmatrix}, \quad \alpha \le t \le \beta ,$$

$$(3.5b) \qquad B_1\begin{bmatrix} z \\ x(\alpha) \end{bmatrix} + B_m\begin{bmatrix} z \\ x(\beta) \end{bmatrix} = \begin{bmatrix} b_x \\ b_z \end{bmatrix},$$

using the switching points $\alpha_1, \ldots, \alpha_m$ as shooting points and starting on each subinterval $[\alpha_i, \alpha_{i+1}]$ with a fundamental solution $H(t)$, where

$$H(t) = \begin{bmatrix} I_l & \emptyset \\ \emptyset & Q_i(1) \end{bmatrix}$$

would lead to the linear system (3.4). Note that (3.5) is equivalent to (1.1), (1.2) and (1.4). Although the S_i and R_{i+1} in (3.4) have a special structure, we will solve (3.4) in a general way; that is, we will consider the S_i and R_{i+1} to be full matrices. In this case system (3.4) is called a *general discrete two-point BVP*, which can be written as

$$\text{(3.6)} \qquad S_i \hat{c}_i + R_{i+1} \hat{c}_{i+1} = \hat{q}_{i+1}, \quad i = 1, \ldots, m-1$$

$$B_1 \hat{c}_1 + B_m \hat{c}_m = \hat{b}.$$

For well-conditioned two-point BVP the solution space of the homogeneous problem is dichotomic. In order to use the ideas outlined in chapter II for two-point BVP, we shall now show how to transform S_i and R_{i+1} appropriately for use in a forward-backward algorithm.
Let O_1 and T_1 be orthogonal matrices such that

$$\text{(3.7)} \qquad S_1 O_1 = -T_1 V_1,$$

where V_1 is upper triangular. Then let O_2 be an orthogonal matrix such that

$$\text{(3.8)} \qquad T_1^{-1} R_2 O_2 = W_2,$$

where W_2 is upper triangular.
This process gives in general

$$\text{(3.9)} \qquad S_i O_i = T_i V_i,$$

$$T_i^{-1} R_{i+1} O_{i+1} = -W_{i+1},$$

where T_i, O_i are orthogonal matrices and V_i, W_{i+1} are upper triangular matrices. Finally define

$$\text{(3.10)} \qquad f_{i+1} = T_i^{-1} \hat{q}_i \text{ and } e_i = O_i^{-1} \hat{c}_i,$$

then we have the transformed system

$$\text{(3.11a)} \qquad W_{i+1} e_{i+1} = V_i e_i + f_{i+1}, \quad i = 1, \ldots, m-1,$$

and a BC

$$\text{(3.11b)} \qquad \hat{B}_1 e_1 + \hat{B}_m e_m = \hat{b},$$

where $\hat{B}_1 = B_1 O_1$ and $\hat{B}_m = B_m O_m$.

If system (3.11) is well-conditioned, it is dichotomic, i.e. for some integer k_p there exist a k_p-dimensional subspace of increasing solutions and an $(n-k_p)$-dimensional subspace of non-increasing solutions. Moreover these two subspaces are disjoint. Using this property and starting with a proper O_1, we can compute a set of V_i and W_{i+1} for which the first k_p columns represent the subspace of increasing solutions and the last $(n-k_p)$ columns the subspace of non-increasing solutions. Partitioning of the matrices and vectors results in

(3.12a) $\qquad W_{i+1}^{22}\, e_{i+1}^2 = V_i^{22} e_i^2 + f_{i+1}^2 \ ,$

(3.12b) $\qquad W_{i+1}^{11}\, e_{i+1}^1 + W_{i+1}^{12}\, e_{i+1}^2 = V_i^{11} e_i^1 + V_i^{12} e_i^2 + f_{i+1}^1 \ ,$

which can also be written as

(3.13a) $\qquad e_{i+1}^2 = (W_{i+1}^{22})^{-1} [\, V_i^{22} e_i^2 + f_i^2 \,] \ ,$

(3.13b) $\qquad e_i^1 = (V_i^{11})^{-1} [\, W_{i+1}^{11}\, e_{i+1}^1 + W_{i+1}^{12}\, e_{i+1}^2 - V_i^{12} e_i^2 - f_{i+1}^1 \,],$

where W_{i+1}^{11}, V_i^{11} are $k_p \times k_p$-matrices, W_{i+1}^{22}, V_i^{22} are $(n-k_p)\times(n-k_p)$-matrices, e_i^1, f_{i+1}^1 are k_p-vectors and e_i^2, f_{i+1}^2 $(n-k_p)$-vectors.

Forward computation of (3.12a) and backward computation of (3.12b) are stable. Hence the obvious strategy for computing a fundamental solution $\{\Theta_i\}_{i=1}^m$ and a particular solution $\{p_i\}_{i=1}^m$ of recursion (3.11) is to use (3.12a) in forward direction and (3.12b) in backward direction. So for the particular solution $\{p_i\}_{i=1}^m$ we have the BC

(3.14) $\qquad p_1^2 = 0 \ , p_m^1 = 0.$

Then p_i^2 , $i = 2,3, \ldots , m$, using (3.13a), and p_i^1 , $i = m-1, m-2, \ldots , 1$, using (3.13b), is computed.

For the fundamental solution we have the recursion

(3.15a) $\qquad \Theta_{i+1}^2 = (W_{i+1}^{22})^{-1} V_i^{22} \Theta_i^2 \ ,$

(3.15b) $\qquad \Theta_i^1 = (V_i^{11})^{-1} [\, W_{i+1}^{11}\, \Theta_{i+1}^1 + W_{i+1}^{12}\, \Theta_{i+1}^2 - V_i^{12} \Theta_i^2 \,]$

and the BC

(3.16) $\qquad \Theta_1^2 = [\, \varnothing \mid I \,] \ ; \Theta_m^1 = [\, I \mid \varnothing \,].$

Now $\{\Theta_i^2\}_{i=2}^m$ is computed via (3.14a) and $\{\Theta_i^1\}_{i=m-1}^1$ is then computed via (3.14b).

The solution of (3.11) can be written as

(3.17) $\qquad e_i = \Theta_i\, a + p_i \ , \quad i = 1, \ldots , m \ ,$

for some $(n+l)$-vector a. Substituting (3.17) into (3.11b) we have

(3.18) $\qquad [\hat{B}_1 \Theta_1 + \hat{B}_m \Theta_m \,]\, a = \hat{b} - \hat{B}_1 p_1 - \hat{B}_m p_m ,$

from which a can be computed. Then the e_i can be computed via (3.17) and then the $\hat{c}_i$ via (3.10).

Remark 3.19

In order to compute a solution of (3.13) in a stable way, it is necessary that the W_{i+1}^{22} and the V_i^{11} are nonsingular. Moreover the diagonal elements of $(W_{i+1}^{22})^{-1}V_i^{22}$ and $(V_i^{11})^{-1}W_{i+1}^{11}$ should be less than or equal to 1.

Remark 3.20

It is not necessary that the W_{i+1}^{11} and V_i^{22} are nonsingular, i.e. it is not necessary that all S_i and R_{i+1} are nonsingular. If the dichotomy induces a splitting such that the V_i^{11} and W_{i+1}^{22} are nonsingular and $[\hat{B}_1\Theta_1 + \hat{B}_m\Theta_m]$ is nonsingular, we still have a solution for the general discrete BVP (3.5)

3.4 Conditioning and stability

As a BVP with parameters can be written as a two-point BVP (3.5), it is obvious that we have for the condition number CN:

(3.21) $$CN = \max_t \| H(t)[B_1 H(\alpha) + B_m H(\beta)]^{-1} \| ,$$

where $H(t)$ is a fundamental solution of (3.5a). Moreover we have

(3.22) $$\kappa := \| [\hat{B}_1\Theta_1 + \hat{B}_m\Theta_m]^{-1} \| \le 2\,CN.$$

For stability we have to investigate the (growth of) solutions between two successive switching points; this is essentially similar to investigating the recursion of the two-point BVP, and recursions (3.11) or (3.13). For stability only the homogeneous part of a recursion is of interest; for (3.13) the latter can be written as

(3.23a) $$e_{i+1}^2 = (W_{i+1}^{22})^{-1} V_i^{22} e_i^2 ,$$

(3.23b) $$e_i^1 = (V_i^{11})^{-1} [W_{i+1}^{11} e_{i+1}^1 - [V_i^{12} - W_{i+1}^{12} (W_{i+1}^{22})^{-1} V_i^{22}] e_i^2] .$$

Denoting $E_{i+1} := (W_{i+1}^{22})^{-1} V_i^{22}$, $B_{i+1}^{-1} := (V_i^{11})^{-1} W_{i+1}^{11}$ and

$B_{i+1}^{-1} C_{i+1} := (V_i^{11})^{-1} [V_i^{12} - W_{i+1}^{12} (W_{i+1}^{22})^{-1} V_i^{22}]$ we have

(3.24a) $$e_{i+1}^2 = E_{i+1} e_i^2$$

(3.24b) $$e_i^1 = B_{i+1}^{-1} e_{i+1}^1 - B_{i+1}^{-1} C_{i+1} e_i^2 .$$

This is similar to the recursion derived from a two-point BVP and therefore the same formula can be used to compute the amplification factor, cf. §II.3.4.

Remark (3.25)

Note that the effect of accumulated errors as given in (II.3.22) depends on B_{i+1}^{-1} and $B_{i+1}^{-1}\ C_{i+1}$ and not on B_{i+1} itself. So even if W_{i+1}^{11} is singular and therefore B_{i+1} is not defined, B_{i+1}^{-1} and the quantity "$B_{i+1}^{-1}\ C_{i+1}$" are still meaningful.

4. Computational aspects

The routine MUTSPA basically uses the same strategy for computing the upper triangular recursion on the intervals $[\alpha_i, \alpha_{i+1}]$, $i = 1, \ldots, m$, as the routine MUTSGE does for two-point BVP (see chapter I). Only the choice of the $Q_1(i), i = 2, \ldots, m$ (that is the "orthogonalized" $F_1(\alpha_i, \alpha_i)$) and the computation of the k-partitionings are different. The computation of the $\{c_i\}_{i=1}^{m}$ is decribed in §3. Once knowing the c_i, the computation of the solution at the i^{th} interval $[\alpha_i, \alpha_{i+1}]$ is the same as in the two-point case (see chapter II). In the next sections we discuss how to find the switching points, the choice of $Q_1(i)$, how to find a correct global partitioning and how to find a correct partitioning for the general discrete two-point BVP (cf. system (3.11)).

4.1 The computation of switching points

A well-conditioned parameter problem is skew polychotomic, that is the dimension $k(i)$, say, of the non-decreasing solution space on $[\alpha_i, \alpha_{i+1}]$ forms a non-decreasing sequence, i.e. $k(1) \leq k(2) \leq \cdots \leq k(m)$.
For a switching point α_i, say, we potentially have a mode which is decreasing on $[\alpha_1, \alpha_i]$ and increasing on $[\alpha_i, \alpha_{m+1}]$. Using this property a minor shooting point, t_l say, is considered to be a switching point α_i, say, if there is a diagonal element of $\prod_{j=2}^{l} U_j$ less than 0.5 and if the same diagonal element of U_{l+1} is greater than 1. Here U_j is the incremental matrix of the fundamental solution on the minor shooting interval $[t_{j-1}, t_j]$.
Because a more or less constant mode may result in a diagonal element fluctuating around 1, only modes with an incremental growth less than 0.5 on $[\alpha_1, t_l]$ are considered.
Anticipating skew polychotomy, only the last $n - k(i)$ diagonal elements have to be checked; there are at most n switching points between α_1 and α_{m+1}, i.e. $m \leq n+1$.

4.2 The computation of $Q_1(i)$

On the first interval $[\alpha_1, \alpha_2]$ we do the same as in the two-point case, i.e. $Q_1(1) = I$ and if this is not a satisfactory choice, the columns of $Q_1(1)$ are permuted such that diagonal $U_2(1)$ is ordered. As a first choice for $Q_1(i)$, $i = 2, \ldots, m$, we take

$$(4.1) \qquad Q_1(i) = Q_{N_{i-1}}(i-1).$$

This choice is satisfactory if the diagonal of the incremental matrix $V_2(i)$ of the fundamental solution on the first minor shooting interval on $[\alpha_i, \alpha_{i+1}]$ is ordered. Otherwise the columns

of $Q_1(i)$ are permuted such that the diagonal of $|V_2(i)|$ is ordered. At this stage the partitioning parameter k_i is computed as the number of diagonal elements of $|V_2(i)|$ which are greater than 1.
Although this stategy results in a set of actual switching points and an increasing sequence of k-partitioning parameters $k(i)$, it is possible that, due to discretization errors, the computed $k(i)$ does not correspond to the proper partitioning. Therefore, after the above described procedure, globally correct partitioning parameters are determined.

4.3 Finding a globally correct partitioning

Although the algorithm tries to determine a correct partitioning parameter $k(i)$ on each interval $[\alpha_i, \alpha_{i+1}]$, its resolution of the growth behaviour of the various modes may be fairly small (e.g. if $\alpha_{i+1}-\alpha_i$ is small) and/or it may be misled by non-growing non-decreasing modes. Since a normal (that is a well-conditioned) situation implies the existence of a non-decreasing sequence $\{k(i)\}$, we need a check on this and - if this ordering turns out not to be monotonic - an update. This is done by the following procedure:

step 1: Compute on each interval $[\alpha_i, \alpha_{i+1}]$, $i = 1, \ldots, m$, a partitioning parameter $k(i)$, where $k(i)$ is the number of elements of diagonal$(\prod_{j=2}^{N_i} U_j(i))$, which are greater than 1.

step 2: Determine the highest index l, where $k(l) > k(l+1)$. If no such index exists, goto step 8.

step 3: Determine the highest index $j > l$, where $k(j) < k(l)$.

step 4: Determine the index $p < l$, where
$k(l) = k(l-1) = \cdots = k(p) \neq k(p-1)$.

step 5: Compute a global partitioning parameter $\hat{k}(l)$ say, for the interval $[\alpha_p, \alpha_{j+1}]$ by checking the increments over $[\alpha_p, \alpha_{j+1}]$ in an obvious way, taking into account the various permutations at the switching points.

step 6: The new updated sequence $\{k(i)\}_{i=1}^{m}$ is defined as

$$k(i) := \begin{cases} k(i) & , i = 1, \ldots, p-1, j+1, \ldots, m \\ \max(k(i), \hat{k}(l)) & , i = l+1, \ldots, j \\ \hat{k}(l) & , i = p, \ldots, l \end{cases} .$$

step 7: Go back to step 2.

step 8: The current sequence $\{k(i)\}_{i=1}^{m}$ is correct.

With this procedure we get, at least theoretically, a good choice for the sequence of the $k(i)$. However, if the problem is not skew polychotomic also this procedure may not be satisfactory, naturally, and a large amplification factor may result (as is to be expected of course).

4.4 The computation of O_1 and k_p of system (3.6)

Generally there is no information for choosing O_1, so we start with $O_1 = I$ and compute a V_1 and a W_2. If the diagonal of $W_2^{-1} V_1$ is not ordered, the columns of O_1 are permuted such that the diagonal of $W_2^{-1} V_1$ is ordered. The k-partitioning (k_p) is defined in a similar way as in the two-point BVP case, i.e. k_p is equal to the position of that diagonal element of $W_2^{-1}V_1$ which is the smallest one (in absolute value) being greater than 1. However, this k_p may not be the globally best one for the recursion. Therefore a global check of the increment $\prod_{j=1}^{m} W_{j+1}^{-1} V_j$ is made. If the ordering of this product is not found to be satisfactory, a global reordering is performed using permutation matrices according to this.
The question remains what to do when some of the W_{i+1} or V_i are singular. There still may be a stable solution (see §3.4) if the singularity of W_{i+1} occurs in the $k_p \times k_p$ left upper block of W_{i+1} (i.e. W_{i+1}^{11}) and if the singularity of V_i occurs in the right $(n-k_p)\times(n-k_p)$ lower block of V_i (i.e. V_i^{22}). Therefore each zero diagonal element of V_i and W_{i+1} will be given the value of the machine constant (i.e. the value of ER(3)). If there is a proper dichotomy this will result in a correct global partitioning. If there is no proper dichotomy this will result in either a large amplification factor or either a numerically singular V_i^{11} or W_{i+1}^{22}. In the latter case a terminal error IERROR=315 is given.

4.5 The computation of the stability constants

Since the algorithm computes fundamental solutions at switching intervals, it does some bookkeeping of stability constants. The computation of the condition number CN (see §3.5) is a straightforward matter and its value can be found in ER(4).
Concerning the "amplification factor", which is an estimate for the Green's functions, the algorithm computes an estimate for this on each interval and also for system (3.11). The largest of these values can be found in ER(5)

Remark 4.2

If the partitioning is incorrect, we may expect at least ER(5) to be "large". On the other hand, due to the special way the algorithm tries to seek the appropriate partitionings, it should be expected that a large value of ER(5) has to be attributed to the problem.

References

[1] R.M.M. Mattheij, *On boundary value problems for ODE with parameters*, EQUADIFF, Differential Equations (C.M. Dafermos et al., eds.), Marcel Dekker (1989), 481 - 489.

CHAPTER VI

ODE WITH DISCONTINUOUS DATA

1. Introduction

In the preceding chapters we descibed BVP for which the right-hand side of the ODE and the solution were both continuous with respect to the independent variable. In this chapter we will consider BVP for which the solution or the right-hand side of the ODE is discontinuous at certain points.
Let $\alpha = \alpha_1 < \alpha_2 < \cdots < \alpha_{m+1} = \beta$ be switching points. Consider the ODE

(1.1) $$\frac{d}{dt} x(t) = L_i(t)x(t) + r_i(t), \qquad \alpha_i \le t < \alpha_{i+1}, \ i = 1, \ldots, m,$$

where the $L_i(t)$ are bounded continuous $n \times n$-matrix functions and the $r_i(t)$ are bounded continuous n-vector functions.

For a solution $x(t)$ of (1.1) we define:

(1.2a) $$x(\alpha_i^-) := \lim_{\varepsilon \downarrow 0} x(\alpha_i - \varepsilon) \ ; \ x(\alpha_1^-) = x(\alpha_1^+),$$

(1.2b) $$x(\alpha_i^+) := \lim_{\varepsilon \downarrow 0} x(\alpha_i + \varepsilon) \ ; \ x(\alpha_{m+1}^+) = x(\alpha_{m+1}^-).$$

Although the ODE (1.1) is discontinuous at $\alpha_2, \ldots, \alpha_m$, there are continuous solutions of (1.1). For specifying a discontinuous solution of (1.1) at $\alpha_2, \ldots, \alpha_m$, we need *side conditions* at $\alpha_2, \ldots, \alpha_m$, which have the form

(1.3) $$Z_i^- x(\alpha_i^-) + Z_i^+ x(\alpha_i^+) = b_i, \quad i = 2, \ldots, m,$$

where Z_i^-, Z_i^+ are $n \times n$-matrices, b_i an n-vector.
These side conditions are completed by a (multipoint) BC, i.e.

(1.4) $$\sum_{i=1}^{m+1} M_i \, x(\alpha_i^+) = b,$$

where the M_i are $n \times n$-matrices and b is an n-vector.

Two cases can be distinguished for the *side conditions* :

i) *jump conditions* at α_i, like

$$(1.5a)\qquad \begin{bmatrix} I_p & \varnothing \\ \varnothing & I_{n-p} \end{bmatrix} x(\alpha_i^-) = \begin{bmatrix} I_p & \varnothing \\ \varnothing & I_{n-p} \end{bmatrix} x(\alpha_i^+) + \begin{bmatrix} s_i \\ 0 \end{bmatrix}, \quad s_i \neq 0 .$$

E.g. if both Z_i^- and Z_i^+ are nonsingular, we have a jump condition.

ii) *internal boundary conditions* at α_i, like

$$(1.5b)\qquad \begin{bmatrix} I_p & \varnothing \\ \varnothing & I_{n-p} \end{bmatrix} x(\alpha_i^-) = \begin{bmatrix} \varnothing & \varnothing \\ \varnothing & I_{n-p} \end{bmatrix} x(\alpha_i^+) + \begin{bmatrix} s_i \\ 0 \end{bmatrix}, \quad s_i \neq 0 .$$

Jump conditions just make the solution discontinuous and are not genuine BC, whereas internal BC in part determine the solution locally.

As in chapter IV, we compute fundamental solutions $F(\alpha_i, t)$ and particular solutions $w(\alpha_i, t)$ consecutively on the intervals $[\alpha_i, \alpha_{i+1}]$ and try to determine the vectors c_i in

$$(1.6)\qquad x(t) = F(\alpha_i, t)\, c_i + w(\alpha_i, t), \qquad \alpha_i \le t \le \alpha_{i+1} .$$

The major difference with both the two-point and the multipoint case is that we have to use the side condition (1.3) at $t = \alpha_j$, $j = 2, \ldots, m$, instead of employing continuity there as before. This gives for $i = 1, \ldots, m-1$,

$$(1.7a)\qquad Z_{i+1}^- F(\alpha_i, \alpha_{i+1}^-)\, c_i + Z_{i+1}^+ F(\alpha_{i+1}, \alpha_{i+1}^+)\, c_{i+1} =$$

$$= b_i - Z_{i+1}^- w(\alpha_i, \alpha_{i+1}^-) - Z_{i+1}^+ w(\alpha_{i+1}, \alpha_{i+1}^+) .$$

Together with the BC (cf. (1.4)),

$$(1.7b)\qquad \sum_{i=1}^{m-1} M_i F(\alpha_i, \alpha_i^+)\, c_i + [M_m F(\alpha_m, \alpha_m^+) + M_{m+1} F(\alpha_m, \alpha_{m+1})]\, c_m = \hat{b} ,$$

where

$$\hat{b} = b - \sum_{i=1}^{m} M_i w(\alpha_i, \alpha_i^+) - M_{m+1} w(\alpha_m, \alpha_{m+1}) .$$

We have a linear system to be solved for the unknown $c_1, \ldots, c_m$.
The algorithm described below has been implemented as the routine MUTSDD.

2. Global description of the algorithm

The basic part of the algorithm essentially follows the ideas outlined in chapter IV, i.e. it determines minor shooting intervals and assembles them into major shooting intervals. Boundary points of such a major shooting interval are either user requested output points or switching points; in contrast to the regular multipoint case, however, no assembly across a switching point is being made.
Let us use the terminology of § IV.2 again: On each interval $[\alpha_i, \alpha_{i+1}]$ orthogonal matrices $Q_j(i)$ and upper triangular matrices $U_j(i)$ are computed. For the solution $x(t)$ we have

(2.1) $$x(t) = F_j(\alpha_i, t)\, a_j(i) + w_j(\alpha_i, t)\,.$$

This gives the following recursion

(2.2) $$a_{j+1}(i) = U_{j+1}(i)\, a_j(i) + d_{j+1}(i)\,, \qquad j = 1, \ldots, N_i - 1\,.$$

Moreover, let $\{\Phi_j(i)\}_{j=1}^{N_i-1}$ and $\{z_j(i)\}_{j=1}^{N_i-1}$ be a fundamental and particular solution of (2.2). Then for some vector c_i we have

(2.3) $$a_j(i) = \Phi_j(i)\, c_i + z_j(i)\,, \qquad j = 1, \ldots, N_i\,.$$

At the switching points we have

(2.4a) $$x(\alpha_i^+) = w_1(\alpha_i, \alpha_i^+) + Q_1(i)\,[\, z_1(i) + \Phi_1(i)\, c_i\,]\,, \qquad i = 1, \ldots, m\,,$$

(2.4b) $$x(\alpha_{i+1}^-) = w_{N_i}(\alpha_i, \alpha_{i+1}^-) + Q_{N_i}(i)\,[\, z_{N_i}(i) + \Phi_{N_i}(i)\, c_i\,]\,, \quad i = 1, \ldots, m\,.$$

Substituting (2.4) in (1.3), we obtain

(2.5) $$K_i\, c_i + L_{i+1}\, c_{i+1} = q_i\,, \qquad i = 1, \ldots, m-1\,,$$

where

(2.6a) $$K_i = Z_{i+1}^-\, Q_{N_i}(i)\, \Phi_{N_i}(i)\,,$$

(2.6b) $$L_{i+1} = Z_{i+1}^+\, Q_1(i+1)\, \Phi_1(i+1)\,,$$

(2.6c) $$q_i = b_i - Z_{i+1}^-\,[\, w_{N_i}(\alpha_i, \alpha_{i+1}^-) + Q_{N_i}(i)\, z_{N_i}(i)\,] -$$
$$- Z_{i+1}^+\,[\, w_1(\alpha_{i+1}, \alpha_{i+1}^+) + Q_1(i+1)\, z_1(i+1)\,]\,.$$

Substituting in (1.4) we obtain

(2.7) $$\sum_{i=1}^{m} \hat{M}_i \, c_i = \hat{b} \,,$$

where

(2.8a) $$\hat{M}_i = M_i \, Q_1(i) \, \Phi_1(i) \,, \qquad i = 1, \ldots, m-1 \,,$$

(2.8b) $$\hat{M}_m = M_m \, Q_1(m) \, \Phi_1(m) + M_{m+1} \, Q_{N_m}(m) \, \Phi_{N_m}(m) \,,$$

(2.8c) $$\hat{b} = b - \sum_{i=1}^{m} M_i \, Q_1(i) \, z_1(i) - M_{m+1} \, Q_{N_m}(m) \, z_{N_m}(m) -$$

$$- \sum_{i=1}^{m} M_i \, w_1(\alpha_i, \alpha_i^+) - M_{m+1} \, w_{N_m}(\alpha_m, \alpha_{m+1}^-) \,.$$

This gives the linear system

(2.9a) $$\mathbf{A} \, \mathbf{c} = \mathbf{q} \,,$$

where

(2.9b) $$\mathbf{A} = \begin{bmatrix} K_1 & L_2 & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & K_{m-1} & L_m \\ \hat{M}_1 & \hat{M}_2 & \cdot\cdot & \hat{M}_{m-1} & \hat{M}_m \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_{m-1} \\ c_m \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} q_1 \\ \cdot \\ \cdot \\ \cdot \\ q_{m-1} \\ \hat{b} \end{bmatrix}.$$

This system resembles the multipoint system obtained in (IV.2.9), but for a different form of the blocks K_i, L_{i+1}, as compared to Π_i, Ω_{i+1} there. In general K_i, L_{i+1} are not upper triangular and therefore we call systems like (2.9) a *general discrete multipoint* BVP. In the next section we descibe how to solve these systems.

3. Special features of the methods

For most aspects we can refer to chapters II and IV. What is really different here is the solution of the linear system (2.9).

3.1. Solution of the system (2.9)

There is no special structure for the K_i and L_{i+1} in system (2.9). Moreover some of the $\hat{M}_i$ may be singular. Therefore we will describe how to solve general discrete multipoint BVPs. There is a strong similarity between discrete multipoint BVPs and continuous multipoint BVPs. Therefore we can make use of the ideas of chapter IV.
Consider the recursion

$$(3.1a) \qquad B_{i+1}x_{i+1} + A_i x_i = g_{i+1}\,, \quad i = 1, \ldots, N-1\,,$$

and a multipoint BC

$$(3.1b) \qquad \sum_{j=1}^{m+1} M_j x_{i_j} = b\,,$$

where A_i, B_{i+1}, M_j are $n \times n$-matrices, x_i, g_{i+1}, b are n-vectors and $1 = i_1 < i_2 < \cdots < i_{m+1} = N$.
Recursion (3.1a) can be split up into m subrecursions:

$$(3.2) \qquad B_{j+1}(l)x_{j+1}(l) + A_j(l)x_j(l) = g_{j+1}(l)\,, \quad j = 1, \ldots, N_l - 1\,,$$

$$\text{where} \qquad A_j(l) = A_{i_l+j-1}\,; \quad B_{j+1}(l) = B_{i_l+j}\,; \quad x_j(l) = x_{i_l+j-1}\,;$$

$$g_{j+1}(l) = g_{i_l+j}; \quad N_l = i_{l+1} - i_l + 1\,.$$

A solution $\{x_j(l)\}_{j=1}^{N_l}$ of (3.2) can be written as

$$(3.3) \qquad x_j(l) = F_j(l)\,a_l + w_j(l)\,,$$

where $F_j(l)$ is a fundamental solution of (3.2), $w_j(l)$ a particular solution of (3.2) and a_l some constant vector.
For $l = 2, \ldots, m$, we have $x_{i_l} = x_1(l) = x_{N_{l-1}}(l-1)$, which gives the recursion for the a_l:

$$(3.4a) \qquad F_1(l+1)\,a_{l+1} = F_{N_l}(l)\,a_l + w_{N_l}(l) - w_1(l+1)\,,$$

and a multipoint BC for the a_l:

$$(3.4b) \qquad \sum_{j=1}^{m} M_j F_1(j)\,a_j + M_{m+1}F_{N_m}(m)\,a_m = b - \sum_{j=1}^{m} M_j\, w_1(j) - M_{m+1}\, w_{N_m}(m)\,.$$

This system is similar to system (IV.1.4). The i_j can be considered as the discrete version of the switching points of chapter IV. Similar to continuous multipoint BVPs we have that, if the problem is well-conditioned, the problem is polychotomic, which means that recursion (3.1) is polychotomic, so the subrecursions (3.2) are dichotomic and for the so called k-partitionings $k(l)$ of the subrecursions we have

(3.5) $\qquad k(1) \geq k(2) \geq \cdots \geq k(m).$

To compute a fundamental solution and a particular solution of the subrecursions (3.2), the same method is used as in the case of discrete two-point BVPs (cf. §V.3.3). That is, the recursions are transformed into appropriate upper triangular recursions and the fundamental solutions and particular solutions are computed using the forward-backward algorithm.
Let $\{\Psi_j(l)\}_{j=1}^{N_l}$ be the fundamental solution and $\{p_j(l)\}_{j=1}^{N_l}$ the particular solution of the upper triangular recursion; let $\{O_j(l)\}_{j=1}^{N_l}$ be the orthogonal transformation matrices. Then for some c_l we have

(3.6) $\qquad x_j(l) = O_j(l)\,[\,\Psi_j(l)\,c_l + p_j(l)\,].$

As the problem is polychotomic, the $O_1(l+1)$ are chosen such that

(3.7) $\qquad O_1(l+1) = O_{N_l}(l)\,P_l \;,\; l = 1, \ldots, m-1$

where P_l is a permutation matrix, which only permutes the first $k(l)$ columns of $O_{N_l}(l)$, where $k(l)$ is the k-partitioning of the l^{th} subrecursion (3.2) (cf. §IV.4.3).
Matching at the "switching points i_j" and substituting (3.6) in the BC (3.1b), we obtain the linear system for the c_l :

(3.8) $\qquad \mathbf{A}\,\mathbf{c} = \mathbf{q}$
where

$$\mathbf{A} = \begin{bmatrix} \Pi_1 & -\Omega_2 & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \Pi_{m-1} & -\Omega_m \\ \hat{M}_1 & \hat{M}_2 & \cdot & \hat{M}_{m-1} & \hat{M}_m \end{bmatrix},$$

$$\mathbf{c}^T = [\,c_1^T, \ldots, c_{m-1}^T, c_m^T\,] \quad \mathbf{q}^T = [\,q_1^T, \ldots, q_{m-1}^T, \hat{b}^T\,],$$

and for $l = 1, \ldots, m-1$,

$$\Pi_l = \Psi_{N_l}(l),$$

$$\Omega_{l+1} = O_{N_l}^{-1}(l)\,O_1(l+1)\,\Psi_1(l+1),$$

$$q_l = O_{N_l}^{-1}(l)\,O_1(l+1)\,p_1(l+1) - p_{N_l}(l),$$

$$\hat{M}_l = M_l\,O_1(l)\,\Psi_1(l),$$

$$\hat{M}_m = M_m\,O_1(m)\,\Psi_1(m) + M_{m+1}\,O_{N_m}(m)\,\Psi_{N_m}(m),$$

$$\hat{b} = b - \sum_{l=1}^{m} M_l\, O_1(l)\, p_1(l) - M_{m+1}\, O_{N_m}(m)\, p_{N_m}(m).$$

This system is similar to system (IV.2.9). The method for solving such systems is described in §IV.3.4. Having the solution for the c_l, (3.6) is used to find the solutions x_i of (3.1).

3.2 Conditioning and stability

The condition number CN for BVP with discontinuous data is defined as follows: Let $F(t)$ be a fundamental solution of ODE (1.1) and let $\{G(i)\}_{i=1}^{m}$ be a fundamental solution of recursion (1.7a), i.e. of the recursion

(3.9) $$Z_{i+1}^{-} F(\alpha_{i+1}^{-})\, c_i + Z_{i+1}^{+} F(\alpha_{i+1}^{+})\, c_{i+1} = 0, \quad i = 1, \ldots, m-1.$$

Define the matrix solution $H(t)$ of ODE (1.1) as

(3.10) $$H(t) := F(t)\, G(i), \quad \alpha_i^{+} \le t \le \alpha_{i+1}^{-},\ i = 1, \ldots, m-1.$$

Then

(3.11) $$CN = \max_{t \in [\alpha,\beta]} \| H(t) [\sum_{j=1}^{m+1} M_j\, H(\alpha_j^{+})]^{-1} \| .$$

If the ODE is polychotomic, we can choose the $F(t)$ such that $\max_{t \in [\alpha,\beta]} \| F(t) \| \le 1$. For such an $F(t)$ we have

(3.12) $$CN \le \max_{i=1,\ldots,m} \| G(i) [\sum_{j=1}^{m+1} M_j\, F(\alpha_j^{+})\, G(i)]^{-1} \| .$$

Conditioning of the discrete multipoint BVP (3.1) is similar to the conditioning of (continuous) multipoints BVPs. Let $\{ G(i) \}_{i=1}^{N}$ be a fundamental solution of recursion (3.1a), then the condition number CN_D is defined as

(3.13) $$CN_D = \max_{i=1,\ldots,N} \| G(i) [\sum_{j=1}^{m+1} M_j\, G(i_j)]^{-1} \| .$$

If the recursion (3.1) is polychotomic, the $G(i)$ can be chosen such that $\| G(i) \| \le 1$ and for CN_D we have

(3.14) $$CN_D \le \| [\sum_{j=1}^{m+1} M_j\, G(i_j)]^{-1} \| .$$

Remark 3.15

The right-hand side of (3.12) is the condition number of the discrete multipoint BVP (1.7). Therefore the estimate of the CN_D of (1.7) is also an estimate for CN.

4. Computational aspects

The routine MUTSDD basically uses the same strategy for computing the upper triangular recursion, the fundamental and particular solutions of the upper triangular recursion and the k-partitioning on the intervals $[\alpha_i^+, \alpha_{i+1}^-]$, $i = 1, \ldots, m$, as the routine MUTSGE uses for the two-point BVP. As a first choice for the $Q_1(i)$ we use: $Q_1(1) = I$, $Q_1(i+1) = Q_{N_i}(i)$, $i = 1, \ldots, m-1$.
For the resulting discrete multipoint BVP, the routine MUTSDD basically uses the same strategy for computing the upper triangular recursions of the subrecursions as is used for discrete two-point BVP (see §V.3.3, §V.4.4). For the choice of the $O_1(i)$ and the global k-partitioning basically the same strategy is used as in the case of multipoint BVP (see §IV.4.1, §IV.4.4).

4.1 The computation of the stability constants

As an estimate for the condition number CN of the problem, we take the estimate for the condition number CN_D of the discrete multipoint BVP. The algorithm for solving system (3.8) delivers the matrix, from which the estimate is computed. If this matrix is singular a terminal error IERROR=320 is given. The output value of ER(4) is the estimated value for CN.
For each interval $[\alpha_i^+, \alpha_{i+1}^-]$ $i = 1, \ldots, m$, an error amplification factor, which is an estimate for the Green's functions, is computed. The output value of ER(5) is the maximum of these amplification factors.
For the discrete multipoint BVP an error amplification factor, being the estimate for the discrete Green's functions, is computed for each subrecursion. The output value of ER(6) is the maximum of these error amplification factors.
If the value of ER(5) or ER(6) is such that the global rounding error is greater than the discretization error, warning errors IERROR = 300 or IERROR = 305 are given.

Remark 4.1

If the partitioning on the intervals $[\alpha_i^+, \alpha_{i+1}^-]$ is incorrect, we may expect at least ER(5) to be "large". If the partitioning of the discrete multipoint BVP is incorrect, we may expect at least ER(6) to be "large". However, due to the special way the algorithm tries to seek the appropriate partitionings, large values for ER(5) or ER(6) have to be attributed to the problem.

4.2 Internal BC

If there are internal BCs, then for some i either Z_{i+1}^{-} or Z_{i+1}^{+} is singular, and therefore either K_i or L_{i+1} is singular. In general, we may have singular matrices A_i or B_{i+1} in the discrete multipoint BVP (3.1). If it is impossible to compute a fundamental and particular solution of the subrecursions (3.2), because of a singular A_i or B_{i+1}, a terminal error IERROR = 315 is given.

On the other hand, realizing that an internal BC at α_{i+1}^{-} should control only growing modes on $[\alpha_i^{+}, \alpha_{i+1}^{-}]$ and an internal BC at α_{i+1}^{+} should control only decreasing modes on $[\alpha_{i+1}^{+}, \alpha_{i+2}^{-}]$ and the special way the algorithm tries to seek appropriate partitionings and fundamental solutions, a terminal error IERROR = 315 should be attributed to the problem.

CHAPTER VII

EIGENVALUE PROBLEMS

1. Introduction

Consider the ODE

(1.1) $$\frac{d}{dt}x(t,\lambda) = L(t)x(t,\lambda) + \lambda K(t)x(t,\lambda) \quad , \alpha \le t \le \beta ,$$

where $K(t)$ is an $n \times n$-matrix function. Let a homogeneous BC

(1.2) $$M_\alpha x(\alpha,\lambda) + M_\beta x(\beta,\lambda) = 0$$

be given. Then (1.1), (1.2) is called an eigenvalue problem, where λ is an *eigenvalue* and the nontrivial solution $x(t,\lambda)$ an *eigensolution*. Formulated this way we obviously do not have uniqueness of x (any multiple of $x(t,\lambda)$ is also an eigensolution). By viewing both x and λ as unknowns it can be seen that (1.1) is in fact a nonlinear equation for the "solution" $(x^T, \lambda)^T$, despite the linearity in x. This makes it suitable for using a nonlinear BVP solver. We augment (1.1) by the simple equation $\dot{\lambda} = 0$ and (1.2) by fixing the solution $x(t,\lambda)$ somewhere (thus making it unique). Here we shall use a method based on successively computing approximations found from integrating (1.1) with a fixed (though recursively updated) λ. Let in the neighbourhood of the eigenvalue λ_e, $F(t,\lambda)$ be a fundamental solution of (1.1). Then any solution $x(t,\lambda)$ can be written as

(1.3) $$x(t,\lambda) = F(t,\lambda)\,c(\lambda) ,$$

for $c(\lambda) \in I\!R^n$.
After substitution of this in the BC (1.2) we should have for λ_e:

(1.4) $$R(\lambda_e)\,c(\lambda_e) = 0 ,$$
where

(1.5) $$R(\lambda) := M_\alpha F(\alpha,\lambda) + M_\beta F(\beta,\lambda) .$$

Apparently, for an eigenvalue λ_e we should have that $R(\lambda_e)$ is singular. By applying an iterative rootfinding algorithm to the latter property we can employ the type of multiple shooting approach of chapter II to (1.1), having only a *nonlinear* algebraic problem via $R(\lambda)$. It should be realized that (1.1), (1.2) can constitute a very complicated problem, potentially : the eigenvalue λ_e can be multiple. If this multiplicity is only algebraic, the method below is certainly not necessarily reliable; if the multiplicity is geometric, then it may give results under special circumstances only.

The algorithm decribed in this chapter is implemented in the routine MUTSEI.

2. Global description of the algorithm

Our algorithm will be based on two ideas: in the first place a method to determine an approximate solution manifold and in the second place a nonlinear *scalar* solver. Assume for a given value λ, $F(t,\lambda)$ has been obtained using a multiple shooting approach with decoupling. Rather than using a classical way of updating λ, based on zeroing $\det(R(\lambda))$ (see (1.5)) we shall use appropriate information from the singular value decomposition

(2.1) $\qquad R(\lambda) = U(\lambda)\Sigma(\lambda)V^T(\lambda)$,

where $U(\lambda)$, $V(\lambda)$ are orthogonal matrices and $\Sigma(\lambda)$ is a diagonal matrix with semi-positve diagonal entries $\sigma_1(\lambda), \ldots, \sigma_n(\lambda)$, where

(2.2) $\qquad \sigma_1(\lambda) \geq \sigma_2(\lambda) \geq \cdots \geq \sigma_n(\lambda) \geq 0$.

Since the number of (numerical) nonzero singular values is equal to the (numerical) rank of $R(\lambda)$, it follows that (aiming initially at a rank $(n-1)$ matrix $R(\lambda_e)$) it makes sense to use $\sigma_n(\lambda)$ as a function of λ that should be zeroed. Realizing that $\sigma_n(\lambda)$ might be a complicated function we use an interval method applied to

(2.3) $\qquad \rho(\lambda) := \mathrm{sgn}\,(\det(R(\lambda)))\,\sigma_n(\lambda)$.

The factor $\mathrm{sgn}(\det(R(\lambda)))$ is employed to make sure that $\rho(\lambda)$ switches sign at least once (in the case of a single eigenvalue). Note that a lower and an upper bound for λ_e has to be supplied. An advantage of an interval method is that the iteration can be stopped when sufficient accuracy has been achieved, viz. by controlling the interval width via a tolerance parameter.
Given a single eigenvalue λ_e, a solution $x(t,\lambda_e)$ can be found directly using $v_n(\lambda_e)$, the last column of $V(\lambda_e)$, i.e.

(2.4) $\qquad x(t,\lambda_e) := F(t,\lambda_e)\,v_n(\lambda_e)$.

For multiple eigenvalues the iteration function (2.3) cannot be guaranteed to work satisfactorily. Moreover, if the numerical rank of the null-space of $R(\lambda_e)$ is larger than one, say l, an eigensolution may be any linear combination of the solutions $F(t,\lambda_e)v_j(\lambda_e)$, with $j = n, n-1, \ldots, n-l+1$, where $v_j(\lambda_e)$ denotes the singular vector in $V(\lambda_e)$ corresponding to $\sigma_j(\lambda_e)$.

Remark 2.5

For quite a few Sturm-Liouville problems the homogeneous system (1.1) does not have strongly increasing or decreasing modes, but rather rapidly oscillating ones. Consequently, although instability, caused by growth of certain modes, is not a likely problem, sufficient accuracy may be a problem as this oscillation requires very many grid points.

3. Special features: conditioning

Usually an iteration is performed on $\det(R(\lambda))$. Although it is undeniably true that $\det(R(\lambda))=0$ whenever λ is an eigenvalue of the problem, one should realize that $\det(R(\lambda))$ is the product of eigenvalues of the matrix R. If some of these are very large (in magnitude) or behave erratically in a neighbourhood of λ_e, the iteration may be far from efficient, or even lead to a numerically unsatisfactory result. On the other hand, it is not unrealistic to use the sign of $R(\lambda)$ as a mean to determine on which side of the "zero" λ_e we are working. This fact, combined with the robustness of a singular value decomposition (and in particular the measure for singularity as indicated by the magnitude of $\sigma_n(\lambda)$, cf. [2]) make the iteration function $\rho(\lambda)$ to be our favorite. Below we shall give a pertubation analysis.

Let the BC (1,1) be perturbed by small matrices δM_α, δM_β. Then we obtain a perturbed matrix $R(\lambda_e)+\delta R(\lambda_e)$, where

(3.1) $$R(\lambda)+\delta R(\lambda)=(M_\alpha+\delta M_\alpha)F(\alpha,\lambda)+(M_\beta+\delta M_\beta)F(\beta,\lambda).$$

Note that $[R(\hat{\lambda}_e)+\delta R(\hat{\lambda}_e)]$ being singular in general means $\hat{\lambda}_e\neq\lambda_e$ and $F(t,\hat{\lambda}_e)\neq F(t,\lambda_e)$. However, given enough regularity with respect to λ, we may say that

(3.2) $$\delta R(\hat{\lambda}_e)\approx\delta M_\alpha F(\alpha,\lambda_e)+\delta M_\beta F(\beta,\lambda_e)\,.$$

Due to the normalisation of the fundamental solutions (as we computed them via the algorithm of chapter II) it follows that

(3.3) $$\|\delta R(\hat{\lambda}_e)\|\leq\|\delta M_\alpha\|+\|\delta M_\beta\|\,.$$

Moreover, from what we just said we may assume that $R(\hat{\lambda}_e)+\delta R(\hat{\lambda}_e)\approx R(\lambda_e)+\delta R(\hat{\lambda}_e)$. By ordering the singular values of the latter perturbed matrix in decreasing order (as for $R(\lambda_e)$) it follows that they differ from the corresponding singular values of the unperturbed $R(\lambda_e)$ by $\|\delta R(\hat{\lambda}_e)\|$ at most. It can also be shown that the perturbation of $v_n(\lambda_e)$ (cf. (2.4)) in the direction orthogonal to v is $\approx\dfrac{\|\delta R(\hat{\lambda}_e)\|}{\sigma_{n-1}(\lambda_e)}$ (given multiplicity 1). Hence, as a measure for the condition number we shall use

(3.4) $$\kappa:=[\sigma_{n-1}(\lambda_e)]^{-1}\,.$$

This is a meaningful estimate of the "condition number"

(3.5) $$CN := \max_t \| F(t,\lambda_e)\,[R(\lambda_e)]^+ \| ,$$

where $[R(\lambda)]^+$ is the pseudo-inverse $V(\lambda)\Sigma^+(\lambda)U^T(\lambda)$
$(\Sigma^+(\lambda) = \text{diagonal}(\sigma^{-1}(\lambda), \ldots, \sigma_{n-1}(\lambda), 0)$.
Note that this "condition number" is a straightforward analogue of that defined in (2.3.12). If the null-space of $R(\lambda_e)$ is of rank larger than one, the condition number is apparently infinite (or very large, if it concerns the numerical rank). However, for geometrical multiplicity $l, l > 1$, it was remarked in §2 that the potential eigenspace was of rank l. Hence the condition number estimate should then read

(3.6) $$\kappa := [\sigma_{n-l}(\lambda_e)]^{-1} ,$$

being an obvious upper bound for (3.5) with appropriately defined $[R(\lambda_e)]^+$.

4. Computational aspects

The routine MUTSEI basically uses the strategy employed in MUTSGE. The extra feature is the use of the nonlinear solver ZEROIN.

4.1 The use of ZEROIN

A reliable method for approximately determining the zero of a nonlinear function, for which an interval is given where it has opposite signs at the endpoints, is usually based on the secant method (or something alike) stabilized with bisection. A successful implementation of this idea is the routine ZEROIN, cf. [1]. This routine is used to "solve" $\rho(\lambda) = 0$, cf. (2.3). Hence the user should supply two (interval end-) points $\lambda_{\min}$ and $\lambda_{\max}$, where he presumes that $\rho(\lambda_{\min}) \times \rho(\lambda_{\max}) < 0$. If, after evaluation of $\rho(\lambda_{\min})$ and $\rho(\lambda_{\max})$ the routine detects that this assumption is violated, a terminal error is given, with the actual value of ρ being printed. From this a better idea of suitable points $\lambda_{\min}$ and $\lambda_{\max}$ might be obtained in order to restart the routine.

4.2 Accuracy of the result

Since the integrator is working with tolerances given in ER(1) and ER(2), one cannot expect - in general - that the eigenvalue is obtained with significantly higher accuracy than ER(2).

4.3 The solution space

As decribed in §2 we may have an eigenspace of dimension > 1. In this case the algorithm may fail. Our iteration function $\rho(\lambda)$ is implicitly assuming that $\det(R(\lambda))$ is changing sign at λ_e, which may no longer be true for (algebraic) multiplicity > 1. Nevertheless, given the absolute tolerance ER(2), all singular values smaller or equal to this value are considered to be zero. When this number turns out to be larger than 1, a more-dimensional space of basic solutions is given, cf. the discussion in §3.4.2.

References

[1] J.C.P. Bus, T.J. Dekker, *Two efficient algorithms with guaranteed convergence for finding a zero of a function*, Mathematical Centre, report NW 13/74, Amsterdam (1974).

[2] R.M.M. Mattheij, F.R. de Hoog, *On non-invertible boundary value problems*, Numerical Boundary Value ODEs (U. Ascher, R. Russell, eds.), Birkhäuser (1985), 55-76.

CHAPTER VIII

SPECIAL LINEAR SOLVERS

1. Introduction

Using multiple shooting techniques to compute approximate solutions of linear BVPs, results into solving sparse linear systems, as decribed in the preceding chapters. These sparse linear systems can be considered as general discrete BVPs. Three sparse linear systems can be distinguished:

i) Linear systems resulting from two-point BVPs.

ii) Linear systems resulting from multipoint BVPs.

iii) Linear systems resulting from two-points BVPs with parameters.

For these three cases we have the routines SPLS1, SPLS2 and SPLS3, respectively.

2. General discrete BVPs

In this section we will descibe the three types of discrete BVPs.

2.1 General discrete two-point BVPs

Consider the sparse linear system

(2.1) $\mathbf{A}\,\mathbf{x} = \mathbf{b}$,
where

$$\mathbf{A} = \begin{bmatrix} A_1 & B_2 & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & A_{N-1} & B_N \\ M_1 & & & & M_N \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ \cdot \\ \cdot \\ x_{N-1} \\ x_N \end{bmatrix}, \mathbf{b} = \begin{bmatrix} g_2 \\ \cdot \\ \cdot \\ g_N \\ b \end{bmatrix}.$$

Here A_i, B_{i+1} are (full) $n \times n$-matrices, M_1, M_N are $n \times n$-matrices, x_i, g_{i+1}, b are n-vectors.

Writing problem (2.1) in a recursive way, we have to consider the recursion

(2.2a) $$A_i x_i + B_{i+1} x_{i+1} = g_{i+1} , \quad i = 1, \ldots , N-1 ,$$

and a BC

(2.2b) $$M_1 x_1 + M_N x_N = b \ .$$

The method for solving this type of linear system is described in §V.3.3 and is implemented in routine SPLS1.

2.2 General discrete multipoint BVPs

Consider the sparse linear system

(2.3) $$\mathbf{A}\mathbf{x} = \mathbf{b} ,$$
where

$$\mathbf{A} = \begin{bmatrix} A_1 & B_2 & & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & A_{m-1} & B_m \\ M_1 & \cdot & \cdot & \cdot & M_m \end{bmatrix} , \mathbf{x} = \begin{bmatrix} x_1 \\ \cdot \\ \cdot \\ x_{m-1} \\ x_m \end{bmatrix} , \mathbf{b} = \begin{bmatrix} g_2 \\ \cdot \\ \cdot \\ g_m \\ b \end{bmatrix} .$$

Here A_i, B_{i+1} are (full) $n \times n$-matrices, $M_1, \ldots, M_m$ are $n \times n$-matrices, x_i, g_{i+1}, b are n-vectors.
Writing problem (2.3) in a recursive way we have to consider the recursion

(2.4a) $$A_i x_i + B_{i+1} x_{i+1} = g_{i+1} , \quad i = 1, \ldots , m-1 ,$$

and a multipoint BC

(2.4b) $$\sum_{j=1}^{k} M_{i_j} x_{i_j} = b \ ,$$

where $1 = i_1 < i_2 < \cdots < i_k = m$, i.e. $M_l = \varnothing$ if $l \neq i_j$, $j = 1, \ldots , k$. (Here we have taken into account that some of the M_i are $\varnothing$.)
The i_j can be considered as the discrete version of the switching points of the continuous multipoint BVP.
The method for computing an approximate solution is decribed in §VI.3.1.
For discrete multipoint BVP we have the routine SPLS2.

2.3 General discrete two-point BVP with parameters

Consider the sparse linear system

(2.5) $\mathbf{A}\,\mathbf{x} = \mathbf{b}$

where

$$\mathbf{A} = \begin{bmatrix} A_1 & B_2 & & & D_2 \\ & \cdot & \cdot & & \cdot \\ & & \cdot & \cdot & \cdot \\ & & A_{N-1} & B_N & D_N \\ M_1 & & & M_N & M_z \end{bmatrix},$$

$$\mathbf{x}^T = [\,x_1^T, \ldots, x_{N-1}^T, x_N^T, z^T\,], \quad \mathbf{b}^T = [\,g_2^T, \ldots, g_N^T, b_x^T, b_z^T\,],$$

A_i, B_{i+1} are (full) $n \times n$-matrices, D_{i+1} are $n \times l$-matrices, M_1, M_N are $(n+l)\times n$-matrices, M_z is an $(n+l)\times l$-matrix, x_i, g_{i+1}, b_x are n-vectors and z, b_z are l-vectors.

Writing system (2.5) in a recursive way we have to consider the recursion

(2.6a) $A_i\,x_i + B_{i+1}x_{i+1} + D_{i+1}\,z = g_{i+1}, \quad i = 1, \ldots, N-1,$

and a BC

(2.6b) $M_1 x_1 + M_N\,x_N + M_z\,z = \begin{bmatrix} b_x \\ b_z \end{bmatrix}.$

The l-vector z can be considered as a vector of l parameters. The method for computing an approximate solution of (2.5) is described in §V.3.3.
For discrete two-point BVPs with parameters we have the routine SPLS3.

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

BOUNDPAK

Numerical Software for Linear Boundary Value Problems Part Two

by

R.M.M. Mattheij and G.W.M. Staarink

EUT Report 92–WSK–01
Eindhoven, February 1992

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven, The Netherlands
ISBN 9038600224
ISSN 0167-9708
Coden: TEUEDE

CONTENTS

PART TWO

CHAPTER IX

DOCUMENTATION

1. Introduction

BOUNDPAK is a package containing Fortran '77 subroutines for solving linear BVP, using the algorithms which are described in the preceding chapters. There are nine subroutines for various BVP of ODE and three subroutines for discrete BVP.
BOUNDPAK is designed for non-stiff problems and uses a multiple shooting technique to compute an approximation of the solution of the BVP at given output points.

The important subroutines of BOUNDPAK for the various types of problems are:

MUTSGE	for two-point BVP with general BC
MUTSPS	for two-point BVP with partially separated BC
MUTSSE	for two-point BVP with completely separated BC
MUTSIN	for two-point BVP with BC at infinity
MUTSMP	for multipoint BVP
MUTSMI	for BVP with an integral BC
MUTSPA	for two point BVP with parameters
MUTSDD	for BVP with discontinuous data
MUTSEI	for eigenvalue problems
SPLS1	for discrete two-point BVP
SPLS2	for discrete multipoint BVP
SPLS3	for discrete two-point BVP with parameters

In §2 - §13 the documentation of these subroutines is given, §14 contains the list of error messages and §15 the names of all the subroutines in BOUNDPAK.

Remark 1.1

The subroutines require a value for the machine constant EPS. In general the machine epsilon is a suitable value for EPS.
However, in the case of a discrete BVP, the EPS is used to determine whether a matrix is singular or not, by checking the diagonal elements of the upper triangular matrix, obtained from the QU-decomposition or the UQ-decomposition. Due to rounding errors, the machine epsilon might be too small to detect a singular matrix, which will result in an improper partitioning and a rather large amplification factor. In such cases a multiple of the machine epsilon will be a more suitable value for EPS.

For the machine epsilon we have the subroutine EPSMAC:

```
SUBROUTINE EPSMAC(EPS)
DOUBLE PRECISION EPS
```

On exit EPS contains the value of the machine epsilon.

Remark 1.2

In the documentation of the subroutines an example of their use is given. The programs for these examples have been run on an Olivetti M24 personal computer, operating under MS-DOS V2.11, using the Olivetti MS-Fortran V3.13 R1.0 compiler and the MS Object Linker V2.01 (large).

2. Subroutine MUTSGE

****

SPECIFICATION

****

```
        SUBROUTINE MUTSGE(FLIN, FINH, N, IHOM, A, B, MA, MB, BCV, ALI, ER,
     1     NRTI, TI, NTI, X, U, NU, Q, D, KPART, PHI, W, LW, IW, LIW, IERROR)
C       INTEGER N, IHOM, NRTI, NTI, NU, LW, IW(LIW), LIW, IERROR
C       DOUBLE PRECISION A, B, MA(N,N), MB(N,N), BCV(N), ALI, ER(5), TI(NTI),
C    1     X(N,NTI), U(NU,NTI), Q(N,N,NTI), D(N,NTI), PHI(NU,NTI), W(LW)
C       EXTERNAL FLIN, FINH
```

****

Purpose

****

MUTSGE solves the two-point BVP:

$$\frac{d}{dt}x(t) = L(t)x(t) + r(t) \ , \ \ A \le t \le B \ \ or \ \ B \le t \le A \ ,$$

with BC:

$$M_A \, x(A) + M_B \, x(B) = BCV$$

where M_A and M_B are the BC matrices and BCV the BC vector.

****

Parameters

****

FLIN SUBROUTINE, supplied by the user with specification:

```
SUBROUTINE FLIN(N, T, FL)
DOUBLE PRECISION T, FL(N,N)
```

where N is the order of the system. FLIN must evaluate the matrix $L(t)$ of the differential equation for t = T and place the result in the array FL(N,N).
FLIN must be declared as EXTERNAL in the (sub)program from which MUTSGE is called.

FINH SUBROUTINE, supplied by the user, with specification:

```
SUBROUTINE FINH(N, T, FR)
DOUBLE PRECISION T, FR(N)
```

where N is the order of the system. FINH must evaluate the vector $r(t)$ of the differential equation for t = T and place the result in FR(1), FR(2), . . . , FR(N).
FINH must be declared as EXTERNAL in the (sub)program from which MUTSGE is called.
In the case that the system is homogeneous FINH is a dummy and one can use FLIN for FINH in the call to MUTSGE.

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

A,B DOUBLE PRECISION, the two boundary points.
Unchanged on exit.

MA,MB DOUBLE PRECISION array of dimension (N, N).
On entry : MA and MB must contain the matrices in the BC:
$M_A\, x(A) + M_B\, x(B) = BCV$.
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector.
Unchanged on exit.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2*ALI, a new output point is inserted.
If ALI ≤ 1 the defaults are:
If NRTI = 0 : ALI := max(ER(1), ER(2)) / (2*ER(3)),
if NRTI ≠ 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.
On exit ALI contains the actually used incremental factor.

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into

ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation.
On entry ER(3) must contain the machine constant EPS (see Remark 1.1)
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimate of the condition number of the BVP.
On exit ER(5) contains an estimate of the amplification factor.

NRTI INTEGER.
On entry NRTI is used to specify the required output points. There are three ways to specify the required output points:
1) NRTI = 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.
2) NRTI = 1, the output points are supplied by the user in the array TI.
3) NRTI > 1, the subroutine computes the (NRTI+1) output points TI(k) by:
$$TI(k) = A + (k-1) * (B - A) / NRTI ;$$
so TI(1) = A and TI(NRTI+1) = B .
Depending on the allowed incremental factor ALI, more output points may be inserted in the cases 2 and 3. On exit NRTI contains the total number of output points.

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: $A = TI(1) < \cdots < TI(k) = B$ or $A = TI(1) > \cdots > TI(k) = B$
(k denotes the total number of required output points).
On exit: TI(i), i = 1,2, . . . , NRTI, contains the output points.

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, D, PHI. When NOTI denotes the total number of output points then
NTI ≥ max(5, NOTI + 1). If the routine was called with NRTI > 1 and ALI ≤ 1, the total number of required output points is (the entry value of NRTI) + 1,
so NTI ≥ max(5, NRTI + 2).
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NTI).
On exit X(i,k) , i=1,2, . . . , N contains the solution of the BVP at the output point TI(k), k=1, . . . , NRTI.

U DOUBLE PRECISION array of dimension (NU, NTI).
On exit U(i,k) i=1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NRTI. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k) , . . . , U(nj + j, k), where nj = (j – 1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to N * (N+1) / 2.
Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N, N, NTI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal matrix Q_k, k = 1, . . . , NRTI.

D DOUBLE PRECISION array of dimension (N, NTI).
If IHOM = 0 the array D has no real use and the user is recommended to use the same array for the X and the D.
If IHOM = 1 : on exit D(i,k) i = 1,2, . . . , N contains the inhomogeneous term d_k, k = 1,2, . . . , NRTI, of the multiple shooting recursion.

KPART INTEGER.
On exit KPART contains the global k-partition of the upper triangular matrices U_k.

PHI DOUBLE PRECISION array of dimension (NU, NTI).
On exit PHI contains a fundamental solution of the multiple shooting recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
If IHOM=0 : LW ≥ 8*N + 7*N*N ; if IHOM=1 : LW ≥ 9*N + 7*N*N.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ 4*N + 1.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See § 14 for the other errors.

****

Auxiliary Routines

****

This routine calls the BOUNDPAK library routines AMTES, APLB, BCMAV, CDI, CNRHS, COPMAT, COPVEC, CONDW, CROUT, CWISB, DEFINC, DUR, FCBVP, FC2BVP, FQUS, FUNPAR, FUNRC, GTUR, INPRO, INTCH, KPCH, LUDEC, MATVC, PSR, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SBVP, SOLDE, SOLUPP, SORTD, TAMVC, TUR, UPUP, UPVECP.

****

Remarks

****

MUTSGE is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****

Method

****

See chapter II.

****

Example of the use of MUTSGE

****

Consider the ordinary differential equation

$$\frac{d}{dt}x(t) = L(t)\,x(t) + r(t)\,, \quad 0 \le t \le 6$$

and a boundary condition $M_0 x(0) + M_N x(6) = C$ with

$$L(t) = \begin{bmatrix} 1-2\cos(2t) & 0 & 1+2\sin(2t) \\ 0 & 2 & 0 \\ -1-2\sin(2t) & 0 & 1+2\cos(2t) \end{bmatrix}$$

$$r(t) = \begin{bmatrix} (-1+2\cos(2t)-2\sin(2t))e^t \\ -e^t \\ (1-2\cos(2t)-2\sin(t))e^t \end{bmatrix}, \quad C = \begin{bmatrix} 1+e^6 \\ 1+e^6 \\ 1+e^6 \end{bmatrix}$$

and $M_A = M_B = I$.

The solution of this problem is: $x(t) = (e^t, e^t, e^t)^T$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      DOUBLE PRECISION A,B,MA(3,3),MB(3,3),BCV(3),ALI,ER(5),TI(12),
     1 X(3,12),U(6,12),Q(3,3,12),D(3,12),PHIREC(6,12),W(90),
     2 EXSOL,AE
      INTEGER IW(13)
      EXTERNAL FLIN,FINH
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 3
      IHOM = 1
      ALI = 0
      ER(1) = 1.D-11
      ER(2) = 1.D-6
      CALL EPSMAC(ER(3))
      NRTI = 10
      NTI = 12
      NU = 6
      LW = 90
      LIW = 13
      A = 0.D0
      B = 6.D0
C
C     SETTING THE BC MATRICES MA AND MB
C
      DO 1100 I = 1 , N
        DO 1000 J = 1 , N
          MA(I,J) = 0.D0
          MB(I,J) = 0.D0
 1000   CONTINUE
        MA(I,I) = 1.D0
        MB(I,I) = 1.D0
 1100 CONTINUE
C
C     SETTING THE BC VECTOR BCV
C
      BCV(1) = 1.D0 + DEXP(6.D0)
      BCV(2) = BCV(1)
      BCV(3) = BCV(1)
C
C     CALL MUTSGE
C
```

```
      CALL MUTSGE(FLIN,FINH,N,IHOM,A,B,MA,MB,BCV,ALI,ER,NRTI,TI,NTI,
     1        X,U,NU,Q,D,KPART,PHIREC,W,LW,IW,LIW,IERROR)
      IF ((IERROR.NE.0).AND.(IERROR.NE.200).AND.(IERROR.NE.213).AND.
     1 (IERROR.NE.300)) GOTO 5000
C
C     COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND
C     WRITING OF THE SOLUTION AT THE OUTPUTPOINTS
C
      WRITE(6,200)
      WRITE(6,190) ER(4),ER(5)
      WRITE(6,210)
      WRITE(6,200)
      DO 1500 K = 1 , NRTI
        EXSOL = DEXP(TI(K))
        AE = EXSOL - X(1,K)
        WRITE(6,220) K,TI(K),X(1,K),EXSOL,AE
        DO 1300 I = 2 , N
          AE = EXSOL - X(I,K)
          WRITE(6,230) X(I,K),EXSOL,AE
 1300   CONTINUE
 1500 CONTINUE
      STOP
 5000 WRITE(6,300) IERROR
      STOP
C
  190 FORMAT(' CONDITION NUMBER     = ',D10.3,/,
     1 ' AMPLIFICATION FACTOR = ',D10.3,/)
  200 FORMAT(' ')
  210 FORMAT(' I ',6X,'T',8X,'APPROX. SOL.',9X,'EXACT SOL.',8X,
     1 'ABS. ERROR')
  220 FORMAT(' ',I3,3X,F7.4,3(3X,D16.9))
  230 FORMAT(' ',13X,3(3X,D16.9))
  300 FORMAT(' TERMINAL ERROR IN MUTSGE: IERROR = ',I4)
C
      END
C
      SUBROUTINE FLIN(N,T,FL)
C
      DOUBLE PRECISION T,FL(N,N)
      DOUBLE PRECISION TI,SI,CO
C
      TI = 2.D0 * T
      SI = 2.D0 * DSIN(TI)
      CO = 2.D0 * DCOS(TI)
      FL(1,1) = 1.D0 - CO
      FL(1,2) = 0.D0
```

```
      FL(1,3) = 1.D0 + SI
      FL(2,1) = 0.D0
      FL(2,2) = 2.D0
      FL(2,3) = 0.D0
      FL(3,1) = -1.D0 + SI
      FL(3,2) = 0.D0
      FL(3,3) = 1.D0 + CO
C
      RETURN
C     END OF FLIN
      END
C
      SUBROUTINE FINH(N,T,FR)
C
      DOUBLE PRECISION T,FR(N)
      DOUBLE PRECISION TI,SI,CO
C
      TI = 2.D0 * T
      SI = 2.D0 * DSIN(TI)
      CO = 2.D0 * DCOS(TI)
      TI = DEXP(T)
      FR(1) = (-1.D0 + CO - SI)*TI
      FR(2) = - TI
      FR(3) = (1.D0 - CO - SI)*TI
C
      RETURN
C     END OF FINH
      END
```

```
CONDITION NUMBER       = 0.133D+01
AMPLIFICATION FACTOR   = 0.221D+01
```

I	T	APPROX. SOL.	EXACT SOL.	ABS. ERROR
1	0.0000	0.100000001D+01	0.100000000D+01	-0.120756514D-07
		0.100000001D+01	0.100000000D+01	-0.149754604D-07
		0.100000001D+01	0.100000000D+01	-0.130719151D-07
2	0.6000	0.182211882D+01	0.182211880D+01	-0.230910355D-07
		0.182211882D+01	0.182211880D+01	-0.186150286D-07
		0.182211880D+01	0.182211880D+01	0.276479217D-08
3	1.2000	0.332011694D+01	0.332011692D+01	-0.162950000D-07
		0.332011695D+01	0.332011692D+01	-0.299702672D-07
		0.332011690D+01	0.332011692D+01	0.253190855D-07
4	1.8000	0.604964745D+01	0.604964746D+01	0.189447806D-07
		0.604964752D+01	0.604964746D+01	-0.521154062D-07

		0.604964743D+01	0.604964746D+01	0.319208493D-07
5	2.4000	0.110231763D+02	0.110231764D+02	0.450974791D-07
		0.110231764D+02	0.110231764D+02	-0.360646266D-07
		0.110231764D+02	0.110231764D+02	0.539664380D-08
6	3.0000	0.200855369D+02	0.200855369D+02	0.716164905D-08
		0.200855369D+02	0.200855369D+02	-0.169556351D-07
		0.200855369D+02	0.200855369D+02	-0.136451952D-07
7	3.6000	0.365982345D+02	0.365982344D+02	-0.159334164D-07
		0.365982345D+02	0.365982344D+02	-0.192572500D-07
		0.365982344D+02	0.365982344D+02	-0.500945774D-08
8	4.2000	0.666863311D+02	0.666863310D+02	-0.193062100D-07
		0.666863311D+02	0.666863310D+02	-0.313411270D-07
		0.666863310D+02	0.666863310D+02	0.170771948D-07
9	4.8000	0.121510418D+03	0.121510418D+03	0.102888684D-07
		0.121510418D+03	0.121510418D+03	-0.503274649D-07
		0.121510417D+03	0.121510418D+03	0.372506967D-07
10	5.4000	0.221406416D+03	0.221406416D+03	0.489649175D-07
		0.221406416D+03	0.221406416D+03	-0.360825183D-07
		0.221406416D+03	0.221406416D+03	0.207052722D-07
11	6.0000	0.403428793D+03	0.403428793D+03	0.120757022D-07
		0.403428793D+03	0.403428793D+03	0.149755124D-07
		0.403428793D+03	0.403428793D+03	0.130721105D-07

3. Subroutine MUTSPS

****
SPECIFICATION
****

```
      SUBROUTINE MUTSPS(FLIN, FINH, N, IHOM, KSP, A, B, MA, MB, BCV, ALI,
     1     ER, NRTI, TI, NTI, X, U, NU, Q, NQD, ZI, D, KPART, PHI, W, LW,
     2     IW, LIW, IERROR)
C     INTEGER N, IHOM, KSP, NRTI, NTI, NU, NQD, LW, IW(LIW), LIW, IERROR
C     DOUBLE PRECISION A, B, MA(N,N), MB(N,N), BCV(N), ALI, ER(5), TI(NTI),
C    1     X(N,NTI), U(NU,NTI), Q(N,NQD,NTI), ZI(NQD,NTI), D(NQD,NTI),
C    2     PHI(NU,NTI), W(LW)
C     EXTERNAL FLIN, FINH
```

****
Purpose
****

MUTSPS solves the two-point BVP with partially separated BC:

$$\frac{d}{dt}x(t)=L(t)x(t)+r(t)\ ,\quad A\le t\le B\ or\ B\le t\le A\ ,$$

with BC:

$$^1M_A\,x(A)+{}^1M_B\,x(B)=BCV^1$$
$$^2M_A\,x(A)+{}^2M_B\,x(B)=BCV^2$$

where ^1M_A, ^1M_B are KSP×N BC matrices, ^2M_A, ^2M_B are (N−KSP)×N BC matrices and either $^2M_A=\varnothing$ or $^2M_B=\varnothing$, BCV^1 an KSP BC vector and BCV^2 an (N−KSP) BC vector. Moreover, if KSP equals N, MUTSPS checks whether the BC are partially separated or not. If not MUTSGE is used to compute the solution, otherwise a KSP < N is determined and the BC are transformed such that the last N − KSP rows of either M_A or M_B are zero.

****
Parameters
****

FLIN SUBROUTINE, supplied by the user with specification:

```
SUBROUTINE FLIN(N, T, FL)
DOUBLE PRECISION T, FL(N,N)
```

where N is the order of the system. FLIN must evaluate the matrix $L(t)$ of the differential equation for t = T and place the result in the array FL(N,N).

FLIN must be declared as EXTERNAL in the (sub)program from which MUTSPS is called.

FINH SUBROUTINE, supplied by the user, with specification:

SUBROUTINE FINH(N, T, FR)
DOUBLE PRECISION T, FR(N)

where N is the order of the system. FINH must evaluate the vector $r(t)$ of the differential equation for t = T and place the result in FR(1), FR(2), . . . , FR(N).

FINH must be declared as EXTERNAL in the (sub)program from which MUTSPS is called.
In the case that the system is homogeneous FINH is a dummy and one can use FLIN for FINH in the call to MUTSPS.

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

KSP INTEGER
KSP denotes the k-separation, i.e. the number of rows of ${}^1M_A$ and ${}^1M_B$
On entry:
if 0 < KSP < N the BC are partially separated and if on entry IERROR = 0, the last N−KSP rows of M_B are supposed to be zero. If on entry IERROR = 1, the last N−KSP rows of M_A are supposed to be zero.
If KSP = N, the routine checks whether the BC are partially separated or not. If not MUTSGE is called to compute the solution, otherwise the BC are transformed appropriately.
On exit KSP contains the used k-separation. (If KSP = N we have general BC).

A,B DOUBLE PRECISION, the two boundary points.
Unchanged on exit.

MA,MB DOUBLE PRECISION array of dimension (N,N).
On entry : MA and MB must contain the matrices in the BC:
$M_A\, x(A) + M_B\, x(B) = BCV$, where

$$M_A = \begin{bmatrix} {}^1M_A \\ {}^2M_A \end{bmatrix} \text{ and } M_B = \begin{bmatrix} {}^1M_B \\ {}^2M_B \end{bmatrix}.$$

If on entry 0 < KSP < N and IERROR = 0, the last (N−KSP) rows of MB are supposed to be zero and if IERROR = 1 the last (N−KSP) rows of MA are supposed to be zero.
On exit: if on entry KSP=N and the BC are found to be partially separated, MA and MB will contain the transformed BC matrices, otherwise the MA and MB are unchanged.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector; BCV=$(BCV^1, BCV^2)^T$.
On exit: if on entry KSP=N and the BC are found to be partially separated, BCV will contain the transformed BC vector, otherwise BCV is unchanged.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2*ALI, a new output point is inserted.
If ALI ≤ 1 the defaults are:
If NRTI = 0 : ALI := max(ER(1), ER(2)) / (2*ER(3)),
if NRTI > 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.
On exit ALI contains the actually used incremental factor.

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into
ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation.
On entry ER(3) must contain the machine constant EPS (see Remark 1.1).
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimate of the condition number of the BVP.
On exit ER(5) contains an estimate of the amplification factor.

NRTI INTEGER.
On entry NRTI is used to specify the required output points. There are three ways to specify the required output points:
1) NRTI = 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.
2) NRTI = 1, the output points are supplied by the user in the array TI.
3) NRTI > 1, the subroutine computes the (NRTI+1) output points TI(k) by:
TI(k) = A + (k−1) * (B − A) / NRTI ;
so TI(1) = A and TI(NRTI+1) = B .
Depending on the allowed incremental factor ALI, more output points may be inserted in the cases 2 and 3. On exit NRTI contains the total number of output points.

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: A = TI(1) < · · · < TI(k) = B or A = TI(1) > · · · > TI(k) = B
(k denotes the total number of required output points).
On exit: TI(i), i = 1,2, . . . , NRTI, contains the output points.

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, ZI, D, PHI.
Let NOTI be the total number of output points, then NTI ≥ max(5, NOTI + 1). If the routine was called with NRTI > 1 and ALI ≤ 1, the total number of required output points is (the entry value of NRTI) + 1, so NTI ≥ max(5, NRTI + 2).
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NTI).
On exit X(i,k) , i = 1,2, . . . , N contains the solution of the BVP at the output point TI(k), k = 1, . . . , NRTI.

U DOUBLE PRECISION array of dimension (NU, NTI).
On exit U(i,k) i = 1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NRTI. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k), . . . , U(nj + j, k), where nj = (j − 1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to KSP * (KSP + 1) / 2.
Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N, NQD, NTI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , KSP contains the N columns of the orthogonal matrix Q_k^1, k = 1, . . . , NRTI.

NQD INTEGER
NQD is one of the dimension of Q, ZI, D. NQD ≥ KSP.
Unchanged on exit.

ZI DOUBLE PRECISION array of dimension (NQD, NTI) If the BC are partially separated the array ZI is used for storing the particular solution z_i, i = 1, . . . , NRTI of the multiple shooting recursion. Otherwise the array ZI is not used.

D DOUBLE PRECISION array of dimension (NQD, NTI).
On exit D(i,k) i = 1,2, . . . , KSP contains the inhomogeneous term d_k^1, k = 1,2, . . . , NRTI, of the multiple shooting recursion.

KPART INTEGER.
On exit KPART contains the global k-partition of the upper triangular matrices U_k.

PHI DOUBLE PRECISION array of dimension (NU, NTI).
On exit PHI contains a fundamental solution of the multiple shooting recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W. LW ≥ 10*N + 6*N*N + N*KSP.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ 3*N + KSP + 2.
Unchanged on exit.

IERROR INTEGER
On entry IERROR is used as a type indicator for the BC.
If on entry 0 < KSP < N then
IERROR = 0 indicates that ${}^2M_B = \varnothing$,
IERROR = 1 indicates that ${}^2M_A = \varnothing$.
On exit IERROR is an error indicator.
If IERROR = 0 then there are no errors detected.
See § 14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, BCMAV, CDI, CNRHS, COPMAT, COPVEC, CONDW, CQIZI, CROUT, CWISB, DEFINC, DUR, FCBVP, FC2BVP, FQUS, FUNPAR, FUNRC, GOPBC, GTUR, INPRO, INTCH, KPCH, LUDEC, MATVC, MUTSGE, MTSP, PSR, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SBVP, SOLDE, SOLUPP, SORTD, TAMVC, TUR, UPUP, UPVECP.

****

Remarks

****

MUTSPS is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****

Method

****

See chapter II

****

Example of the use of MUTSPS

****

Consider the ordinary differential equation

$$\frac{d}{dt}x(t) = L(t)\,x(t) + r(t)\,,\quad 0 \le t \le 6$$

and a boundary condition $M_0 x(0) + M_N x(6) = C$ with

$$L(t) = \begin{bmatrix} 1-2cos(2t) & 0 & 1+2sin(2t) \\ 0 & 2 & 0 \\ -1-2sin(2t) & 0 & 1+2cos(2t) \end{bmatrix},\quad r(t) = \begin{bmatrix} (-1+2cos(2t)-2sin(2t))e^t \\ -e^t \\ (1-2cos(2t)-2sin(t))e^t \end{bmatrix},$$

$$M_A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix},\quad M_B = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 1+e^6 \\ 1+e^6 \\ 1 \end{bmatrix}.$$

The solution of this problem is: $x(t) = (e^t, e^t, e^t)^T$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on an Olivetti M24 personal computer (see Remark 1.2).

```
      DOUBLE PRECISION A,B,MA(3,3),MB(3,3),BCV(3),ALI,ER(5),TI(15),
     1 X(3,12),U(3,12),Q(3,2,12),ZI(2,12),D(2,12),PHI(3,12),W(90),
     2 EXSOL,AE
      INTEGER IW(13)
      EXTERNAL FLIN,FINH
C
C     SETTING OF THE INPUT PARAMETERS
```

```
C
      N = 3
      KSP = 2
      IERROR = 0
      IHOM = 1
      ALI = 0
      ER(1) = 1.D-11
      ER(2) = 1.D-6
      CALL EPSMAC(ER(3))
      NRTI = 10
      NTI = 12
      NU = 3
      NQD = 2
      LW = 90
      LIW = 13
      A = 0.D0
      B = 6.D0
C
C     SETTING THE BC MATRICES MA AND MB
C
      DO 1000 I = 1 , N
      DO 1000 J = 1 , N
        MA(I,J) = 0.D0
        MB(I,J) = 0.D0
 1000 CONTINUE
      MA(1,3) = 1.D0
      MA(2,2) = 1.D0
      MA(3,1) = 1.D0
      MB(1,3) = 1.D0
      MB(2,2) = 1.D0
C
C     SETTING THE BC VECTOR BCV
C
      BCV(1) = 1.D0 + DEXP(6.D0)
      BCV(2) = BCV(1)
      BCV(3) = 1.D0
C
C     CALL MUTSPS
C
      CALL MUTSPS(FLIN,FINH,N,IHOM,KSP,A,B,MA,MB,BCV,ALI,ER,NRTI,TI,
     1        NTI,X,U,NU,Q,NQD,ZI,D,KPART,PHI,W,LW,IW,LIW,IERROR)
      IF ((IERROR.NE.0).AND.(IERROR.NE.200).AND.(IERROR.NE.213).AND.
     1 (IERROR.NE.300)) GOTO 5000
C
C     COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND WRITING
C     OF THE SOLUTION AT THE OUTPUTPOINTS
```

```
C
      WRITE(*,200)
      WRITE(*,190) ER(4),ER(5)
      WRITE(*,210)
      WRITE(*,200)
      DO 1500 K = 1 , NRTI
       EXSOL = DEXP(TI(K))
       AE = EXSOL - X(1,K)
       WRITE(6,220) K,TI(K),X(1,K),EXSOL,AE
       DO 1300 I = 2 , N
        AE = EXSOL - X(I,K)
        WRITE(*,230) X(I,K),EXSOL,AE
1300  CONTINUE
1500  CONTINUE
      STOP
5000  WRITE(6,300) IERROR
      STOP
C
190   FORMAT(' CONDITION NUMBER     = ',D10.3,/,
     1 ' AMPLIFICATION FACTOR = ',D10.3,/)
200   FORMAT(' ')
210   FORMAT(' I ',6X,'T',8X,'APPROX. SOL.',9X,'EXACT SOL.',8X,
     1 'ABS. ERROR')
220   FORMAT(' ',I3,3X,F7.4,3(3X,D16.9))
230   FORMAT(' ',13X,3(3X,D16.9))
300   FORMAT(' TERMINAL ERROR IN MUTSPS: IERROR = ',I4)
C
      END
C
      SUBROUTINE FLIN(N,T,FL)
C
      DOUBLE PRECISION T,FL(N,N)
      DOUBLE PRECISION TI,SI,CO
C
      TI = 2.D0 * T
      SI = 2.D0 * DSIN(TI)
      CO = 2.D0 * DCOS(TI)
      FL(1,1) = 1.D0 - CO
      FL(1,2) = 0.D0
      FL(1,3) = 1.D0 + SI
      FL(2,1) = 0.D0
      FL(2,2) = 2.D0
      FL(2,3) = 0.D0
      FL(3,1) = -1.D0 + SI
      FL(3,2) = 0.D0
      FL(3,3) = 1.D0 + CO
```

```
C
      RETURN
C     END OF FLIN
      END
C
      SUBROUTINE FINH(N,T,FR)
C
      DOUBLE PRECISION T,FR(N)
      DOUBLE PRECISION TI,SI,CO
C
      TI = 2.D0 * T
      SI = 2.D0 * DSIN(TI)
      CO = 2.D0 * DCOS(TI)
      TI = DEXP(T)
      FR(1) = (-1.D0 + CO - SI)*TI
      FR(2) = - TI
      FR(3) = (1.D0 - CO - SI)*TI
C
      RETURN
C     END OF FINH
      END
```

```
CONDITION NUMBER       = 0.100D+01
AMPLIFICATION FACTOR   = 0.143D+01
```

I	T	APPROX. SOL.	EXACT SOL.	ABS. ERROR
1	.0000	.100000000D+01	.100000000D+01	.000000000D+00
		.100000002D+01	.100000000D+01	-.171180516D-07
		.100000002D+01	.100000000D+01	-.160840654D-07
2	.6000	.182211882D+01	.182211880D+01	-.209659907D-07
		.182211882D+01	.182211880D+01	-.176029289D-07
		.182211880D+01	.182211880D+01	.955206580D-09
3	1.2000	.332011694D+01	.332011692D+01	-.145581911D-07
		.332011695D+01	.332011692D+01	-.254962655D-07
		.332011690D+01	.332011692D+01	.242195828D-07
4	1.8000	.604964745D+01	.604964746D+01	.193012015D-07
		.604964751D+01	.604964746D+01	-.430982885D-07
		.604964744D+01	.604964746D+01	.283331465D-07
5	2.4000	.110231763D+02	.110231764D+02	.540218572D-07
		.110231764D+02	.110231764D+02	-.664868463D-07
		.110231764D+02	.110231764D+02	-.180926403D-07
6	3.0000	.200855369D+02	.200855369D+02	-.122056782D-07
		.200855369D+02	.200855369D+02	-.214101092D-07
		.200855369D+02	.200855369D+02	-.216627782D-07

7	3.6000	.365982345D+02	.365982344D+02	-.315469819D-07
		.365982345D+02	.365982344D+02	-.196939780D-07
		.365982344D+02	.365982344D+02	.107361586D-08
8	4.2000	.666863311D+02	.666863310D+02	-.249469991D-07
		.666863311D+02	.666863310D+02	-.270732272D-07
		.666863310D+02	.666863310D+02	.290659301D-07
9	4.8000	.121510418D+03	.121510418D+03	.122443566D-07
		.121510418D+03	.121510418D+03	-.418312851D-07
		.121510417D+03	.121510418D+03	.405908480D-07
10	5.4000	.221406416D+03	.221406416D+03	.560881404D-07
		.221406416D+03	.221406416D+03	-.633252739D-07
		.221406416D+03	.221406416D+03	.228180852D-08
11	6.0000	.403428794D+03	.403428793D+03	-.755363772D-08
		.403428793D+03	.403428793D+03	.171179977D-07
		.403428793D+03	.403428793D+03	.160840727D-07

4. Subroutine MUTSSE

****

SPECIFICATION

****

```
      SUBROUTINE MUTSSE(FLIN, FINH, N, IHOM, KSP, A, B, MA, BCV, ALI, ER,
     1      NRTI, TI, NTI, X, U, NU, Q, NQD, D, ZI, W, LW, IW, LIW, IERROR)
C     INTEGER N, IHOM, KSP, NRTI, NTI, NU, NQD, LW, IW(LIW), LIW, IERROR
C     DOUBLE PRECISION A, B, MA(N,N), BCV(N), ALI, ER(5), TI(NTI), X(N,NTI),
C    1      U(NU,NTI), Q(N,NQD,NTI), D(NQD,NTI), ZI(NQD,NTI), W(LW)
C     EXTERNAL FLIN, FINH
```

****

Purpose

****

MUTSSE solves the two-point BVP with completely separated BC:

$$\frac{d}{dt}x(t) = L(t)x(t) + r(t) \ , \ \ A \le t \le B \ or \ B \le t \le A \ ,$$

with BC:

$$\begin{array}{lcl} & {}^1M_B \, x(B) & = BCV^1 \\ {}^2M_A \, x(A) & & = BCV^2 \end{array}$$

where ${}^1M_B$ is a KSP $\times$N BC matrix, ${}^2M_A$ an (N$-$KSP) $\times$N BC matrix, BCV^1 an KSP BC vector and BCV^2 an (N$-$KSP) BC vector.

****

Parameters

****

FLIN SUBROUTINE, supplied by the user with specification:

SUBROUTINE FLIN(N, T, FL)
DOUBLE PRECISION T, FL(N,N)

where N is the order of the system. FLIN must evaluate the matrix $L(t)$ of the differential equation for t = T and place the result in the array FL(N,N).
FLIN must be declared as EXTERNAL in the (sub)program from which MUTSSE is called.

FINH SUBROUTINE, supplied by the user, with specification:

```
SUBROUTINE FINH(N, T, FR)
DOUBLE PRECISION T, FR(N)
```

where N is the order of the system. FINH must evaluate the vector $r(t)$ of the differential equation for t = T and place the result in FR(1), FR(2), . . . , FR(N).

FINH must be declared as EXTERNAL in the (sub)program from which MUTSSE is called.
In the case that the system is homogeneous FINH is a dummy and one can use FLIN for FINH in the call to MUTSSE.

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

KSP INTEGER
KSP denotes the k-separation, i.e. the number of rows of ${}^1M_B$.
On entry: 0 < KSP < N.
Unchanged on exit.

A,B DOUBLE PRECISION, the two boundary points.
Unchanged on exit.

MA DOUBLE PRECISION array of dimension (N,N).
MA is used to supply the boundary condition matrices ${}^1M_B$ and ${}^2M_A$.
On entry the first KSP rows of MA must contain the matrix ${}^1M_B$ and the last (N − KSP) rows of MA must contain the matrix ${}^2M_A$
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector; BCV=$(BCV^1, BCV^2)^T$.
Unchanged on exit.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2*ALI, a new output point is inserted.

If ALI $\leq$ 1 the defaults are:
If NRTI = 0 : ALI := max(ER(1), ER(2)) / (2*ER(3)),
if NRTI > 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.
On exit ALI contains the actually used incremental factor.

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into
ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation.
On entry ER(3) must contain the machine constant (EPS).
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimate of the condition number of the BVP.
On exit ER(5) contains an estimate of the amplification factor.

NRTI INTEGER.
On entry NRTI is used to specify the required output points. There are three ways to specify the required output points:
1) NRTI = 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.
2) NRTI = 1, the output points are supplied by the user in the array TI.
3) NRTI > 1, the subroutine computes the (NRTI+1) output points TI(k) by:

$$TI(k) = A + (k-1) * (B - A) / NRTI \, ;$$

so TI(1) = A and TI(NRTI+1) = B .
Depending on the allowed incremental factor ALI, more output points may be inserted in the cases 2 and 3. On exit NRTI contains the total number of output points.

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: $A = TI(1) < \cdots < TI(k) = B$ or $A = TI(1) > \cdots > TI(k) = B$
(k denotes the total number of required output points).
On exit: TI(i), i = 1,2, . . . , NRTI, contains the output points.

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, ZI, D, PHI.
Let NOTI be the total number of output points, then NTI $\geq$ max(5, NOTI + 1). If the routine was called with NRTI > 1 and ALI $\leq$ 1 the total number of required output points is (the entry value of NRTI) + 1, so NTI $\geq$ max(5, NRTI + 2).
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NTI).
On exit X(i,k) , i = 1,2, . . . , N contains the solution of the BVP at the output point TI(k), k = 1, . . . , NRTI.

U DOUBLE PRECISION array of dimension (NU,NTI).
On exit U(i,k) i = 1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NRTI. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k), . . . , U(nj + j, k), where nj = (j − 1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to KSP * (KSP + 1) / 2.
Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N, NQD, NTI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , KSP contains the N columns of the orthogonal matrix Q_k^1, k = 1, . . . , NRTI.

NQD INTEGER
NQD is one of the dimension of Q, ZI, D. NQD ≥ KSP.
Unchanged on exit.

D DOUBLE PRECISION array of dimension (NQD, NTI).
On exit D(i,k) i = 1,2, . . . , KSP contains the inhomogeneous term d_k^1,
k = 1,2, . . . , NRTI, of the multiple shooting recursion.

ZI DOUBLE PRECISION array of dimension (NQD, NTI)
The array ZI is used for storing the particular solution z_i, i = 1, . . . , NRTI of the multiple shooting recursion.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W. LW ≥ 10*N + 6*N*N + N*KSP.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ 3*N + KSP + 2.
Unchanged on exit.

IERROR INTEGER
Error indicator.
If IERROR = 0 then there are no errors detected; integration from A to B.
If IERROR = 1 then there are no errors detected; integration from B to A.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, CDI, CNRHS, COPMAT, COPVEC, CONDW, CQIZI, CROUT, CWISB, DEFINC, DUR, FCBVP, FC2BVP, FQUS, FUNPAR, INPRO, INTCH, KPCH, LUDEC, MATVC, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SOLDE, SOLUPP, SORTD, TAMVC, UPUP, UPVECP.

****
Remarks
****

MUTSSE is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****
Method
****

See chapter II

****
Example of the use of MUTSSE
****

Consider the ordinary differential equation

$$\frac{d}{dt}x(t) = L(t)\,x(t) + r(t)\,, \quad 0 \le t \le 6$$

and a boundary condition $M_A\,x(0) + M_B\,x(6) = C$ with

$$L(t) = \begin{bmatrix} 1-2cos(2t) & 0 & 1+2sin(2t) \\ 0 & 2 & 0 \\ -1-2sin(2t) & 0 & 1+2cos(2t) \end{bmatrix}, \quad r(t) = \begin{bmatrix} (-1+2cos(2t)-2sin(2t))e^t \\ -e^t \\ (1-2cos(2t)-2sin(t))e^t \end{bmatrix},$$

$$M_A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad M_B = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } C = \begin{bmatrix} e^6 \\ e^6 \\ 1 \end{bmatrix}.$$

The solution of this problem is: $x(t) = (e^t, e^t, e^t)^T$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a Olivetti M24 personal computer (see Remark 1.2).

```
      DOUBLE PRECISION A,B,MA(3,3),BCV(3),ALI,ER(5),TI(15),
     1 X(3,12),U(3,12),Q(3,2,12),D(2,12),ZI(2,12),W(90),
     2 EXSOL,AE
      INTEGER IW(13)
      EXTERNAL FLIN,FINH
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 3
      KSP = 2
      IHOM = 1
      ALI = 0
      ER(1) = 1.D-11
      ER(2) = 1.D-6
      CALL EPSMAC(ER(3))
      NRTI = 10
      NTI = 12
      NU = 3
      NQD = 2
      LW = 90
      LIW = 13
      A = 0.D0
      B = 6.D0
C
C     SETTING THE BC MATRICES MA AND MB
C
      DO 1000 I = 1 , N
      DO 1000 J = 1 , N
        MA(I,J) = 0.D0
 1000 CONTINUE
      MA(1,3) = 1.D0
      MA(2,2) = 1.D0
      MA(3,1) = 1.D0
C
C     SETTING THE BC VECTOR BCV
C
      BCV(1) = DEXP(6.D0)
      BCV(2) = BCV(1)
      BCV(3) = 1.D0
```

```
C
C       CALL MUTSSE
C
        CALL MUTSSE(FLIN,FINH,N,IHOM,KSP,A,B,MA,BCV,ALI,ER,NRTI,TI,NTI,
     1         X,U,NU,Q,NQD,D,ZI,W,LW,IW,LIW,IERROR)
        IF ((IERROR.GT.1).AND.(IERROR.NE.200).AND.(IERROR.NE.213).AND.
     1 (IERROR.NE.300)) GOTO 5000
C
C       COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND WRITING
C       OF THE SOLUTION AT THE OUTPUTPOINTS
C
        WRITE(*,200) ER(4),ER(5)
        WRITE(*,210)
        DO 1500 K = 1 , NRTI
         EXSOL = DEXP(TI(K))
         AE = EXSOL - X(1,K)
         WRITE(6,220) K,TI(K),X(1,K),EXSOL,AE
         DO 1300 I = 2 , N
          AE = EXSOL - X(I,K)
          WRITE(*,230) X(I,K),EXSOL,AE
 1300   CONTINUE
 1500   CONTINUE
        STOP
 5000   WRITE(6,300) IERROR
        STOP
C
 200    FORMAT(' CONDITION NUMBER    = ',D10.3,/,
     1  ' AMPLIFICATION FACTOR = ',D10.3,/)
 210    FORMAT(' I ',6X,'T',8X,'APPROX. SOL.',9X,'EXACT SOL.',8X,
     1  'ABS. ERROR',/)
 220    FORMAT(' ',I3,3X,F7.4,3(3X,D16.9))
 230    FORMAT(' ',13X,3(3X,D16.9))
 300    FORMAT(' TERMINAL ERROR IN MUTSSE: IERROR = ',I4)
C
        END
C
        SUBROUTINE FLIN(N,T,FL)
C
C
        DOUBLE PRECISION T,FL(N,N)
        DOUBLE PRECISION TI,SI,CO
C
        TI = 2.D0 * T
        SI = 2.D0 * DSIN(TI)
        CO = 2.D0 * DCOS(TI)
        FL(1,1) = 1.D0 - CO
```

```
      FL(1,2) = 0.D0
      FL(1,3) = 1.D0 + SI
      FL(2,1) = 0.D0
      FL(2,2) = 2.D0
      FL(2,3) = 0.D0
      FL(3,1) = -1.D0 + SI
      FL(3,2) = 0.D0
      FL(3,3) = 1.D0 + CO
C
      RETURN
C     END OF FLIN
      END
C
      SUBROUTINE FINH(N,T,FR)
C
C
      DOUBLE PRECISION T,FR(N)
      DOUBLE PRECISION TI,SI,CO
C
      TI = 2.D0 * T
      SI = 2.D0 * DSIN(TI)
      CO = 2.D0 * DCOS(TI)
      TI = DEXP(T)
      FR(1) = (-1.D0 + CO - SI)*TI
      FR(2) = - TI
      FR(3) = (1.D0 - CO - SI)*TI
C
      RETURN
C     END OF FINH
      END
```

```
CONDITION NUMBER        = 0.100D+01
AMPLIFICATION FACTOR    = 0.226D+01
```

I	T	APPROX. SOL.	EXACT SOL.	ABS. ERROR
1	.0000	.100000000D+01	.100000000D+01	.000000000D+00
		.999999880D+00	.100000000D+01	.119845530D-06
		.999999910D+00	.100000000D+01	.898404952D-07
2	.6000	.182211866D+01	.182211880D+01	.144821880D-06
		.182211875D+01	.182211880D+01	.461684040D-07
		.182211887D+01	.182211880D+01	-.738497004D-07
3	1.2000	.332011687D+01	.332011692D+01	.544821295D-07
		.332011688D+01	.332011692D+01	.399801263D-07
		.332011706D+01	.332011692D+01	-.136792865D-06

4	1.8000	.604964751D+01	.604964746D+01	-.425415418D-07
		.604964741D+01	.604964746D+01	.534903659D-07
		.604964757D+01	.604964746D+01	-.101300240D-06
5	2.4000	.110231765D+02	.110231764D+02	-.877975967D-07
		.110231763D+02	.110231764D+02	.981698403D-07
		.110231764D+02	.110231764D+02	-.626071639D-09
6	3.0000	.200855369D+02	.200855369D+02	-.183782873D-07
		.200855367D+02	.200855369D+02	.177060244D-06
		.200855368D+02	.200855369D+02	.144276381D-06
7	3.6000	.365982342D+02	.365982344D+02	.256026752D-06
		.365982344D+02	.365982344D+02	.725795601D-07
		.365982345D+02	.365982344D+02	-.728410896D-07
8	4.2000	.666863309D+02	.666863310D+02	.127783622D-06
		.666863310D+02	.666863310D+02	.529505257D-07
		.666863312D+02	.666863310D+02	-.201877768D-06
9	4.8000	.121510418D+03	.121510418D+03	-.316227045D-07
		.121510417D+03	.121510418D+03	.580042609D-07
		.121510418D+03	.121510418D+03	-.174474522D-06
10	5.4000	.221406416D+03	.221406416D+03	-.123719332D-06
		.221406416D+03	.221406416D+03	.101443163D-06
		.221406416D+03	.221406416D+03	-.426184954D-07
11	6.0000	.403428793D+03	.403428793D+03	.240764280D-07
		.403428793D+03	.403428793D+03	.000000000D+00
		.403428793D+03	.403428793D+03	.000000000D+00

5. Subroutine MUTSIN

****
SPECIFICATION
****

```
      SUBROUTINE MUTSIN(FLIN, FINH, N, IHOM, A, B, C, BMA, BMINF, BCV,
     1 ALI, ER, NRTI, TI, NTI, IEXT, X, NRSOL, U, NU, Q, D, KU, KE,
     2 KEXT, KPART, PHI, W, LW, IW, LIW, IERROR)
C     INTEGER N, IHOM, NRTI, NTI, IEXT, NRSOL, NU, KU, KE, KEXT, LW,
C    1 IW(LIW), LIW, IERROR
C     DOUBLE PRECISION A, B, BMA(N,N), BMINF(N,N), BCV(N), ALI, ER(5),
C    1 TI(NTI), X(N,NTI,N), U(NU,NTI), Q(N,N,NTI), D(N,NTI),
C    2 PHI(NU,NTI), W(LW)
C     EXTERNAL FLIN, FINH
```

****
Purpose
****

MUTSIN solves the two-point BVP defined on an infinite interval:

$$\frac{d}{dt}x(t)=L(t)x(t)+r(t)\ ,\quad t>A\ ,$$

with BC:

$$M_A\,x(A)+M_\infty x(\infty)=BCV$$

where M_A and M_∞ are the BC matrices and BCV the BC vector.
MUTSIN gives output on a subinterval [A, B], specified by the user.

****
Parameters
****

FLIN SUBROUTINE, supplied by the user with specification:

```
SUBROUTINE FLIN(N, T, FL)
DOUBLE PRECISION T, FL(N,N)
```

where N is the order of the system. FLIN must evaluate the matrix $L(t)$ of the differential equation for t = T and place the result in the array FL(N,N).
FLIN must be declared as EXTERNAL in the (sub)program from which MUTSIN is called.

FINH SUBROUTINE, supplied by the user, with specification:

```
SUBROUTINE FINH(N, T, FR)
DOUBLE PRECISION T, FR(N)
```

where N is the order of the system. FINH must evaluate the vector $r(t)$ of the differential equation for t = T and place the result in FR(1), FR(2), . . . , FR(N).
FINH must be declared as EXTERNAL in the (sub)program from which MUTSIN is called.
In the case that the system is homogeneous FINH is a dummy and one can use FLIN for FINH in the call to MUTSIN.

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

A,B DOUBLE PRECISION.
A,B denotes the interval $[\alpha, \beta]$ (see § III.2). If $M_\infty \neq \emptyset$, B should be taken sufficiently large. Unchanged on exit.

C DOUBLE PRECISION.
When IEXT = 0 C must contain the value for γ_{max} (see §III.4). The actually used value for γ is stored in TI(KEXT).
When IEXT $\neq$ 0, the routine computes an solution using the given value in C as the new value for γ. If TI(1) < TI(KEXT) then C must be greater than TI(KEXT) and C must be smaller than TI(KEXT) if TI(KEXT) < TI(1).
Note that on subsequent call to MUTSIN with IEXT $\neq$ 0, the value of KE may change.
Unchanged on exit.

BMA DOUBLE PRECISION array of dimension (N, N).
On entry BMA must contain the BC matrix M_A.
Unchanged on exit.

BMINF DOUBLE PRECISION array of dimension (N, N)
On entry BMINF must contain the BC matrix M_∞.
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector.

Unchanged on exit.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2*ALI, a new output point is inserted.
If ALI ≤ 1 the defaults are:
If NRTI ≤ 0 : ALI := max(ER(1), ER(2)) / ER(3),
if NRTI > 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.
On the extension interval [B , C], an allowed incremental factor equal to SQRT(RMAX) is used.
On exit ALI contains the actually used incremental factor on the interval [A , B].

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into
ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation.
On entry ER(3) must contain the machine constant EPS (see Remark 1.1).
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimation of the condition number of the BVP.
On exit ER(5) contains an estimated error amplification factor.

NRTI INTEGER.
On entry NRTI is used to specify the required output points on the interval [A,B].
There are three ways to specify the required output points:
1) NRTI ≤ 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.
2) NRTI = 1, the output points are supplied by the user in the array TI.
3) NRTI > 1, the subroutine computes the (NRTI+1) output points TI(k) by:

$$TI(k) = A + (k-1) * (B - A) / NRTI ;$$

so TI(1) = A and TI(NRTI+1) = B .
Depending on the allowed incremental factor ALI, more output points may be inserted in the cases 2 and 3. On exit NRTI contains the total number of output points on the interval [A,B].

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: $A = TI(1) < \cdots < TI(k) = B$ or $A = TI(1) > \cdots > TI(k) = B$
(k denotes the total number of required output points).
On exit: TI(i), i = 1,2, . . . , NRTI, contains the output points and TI(j),

j = NRTI + 1, . . . , KEXT the points used on the interval [B , γ].

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, D, PHI.
If k denotes the number of output points on the interval [A,B] and m denotes the number of points used on the extension interval (B , γ], then
NTI $\geq$ max(5, k + 1) + m.
If the routine was called with NRTI > 1 and ALI $\leq$ 1 then k = NRTI + 1 and m $\geq$ 1, as at least one point is needed on the extension interval, i.e. γ, so
NTI $\geq$ max(5, NRTI + 2) + 1.
If the incremental factor of a homogeneous solution on the interval [B , γ] becomes greater than SQRT(RMAX) an additional point is used on the extension interval. In this case m > 1.
Unchanged on exit.

IEXT INTEGER.
IEXT is a flag concerning the extension interval. On the first call to MUTSIN, IEXT must be zero. When the extension interval [B , C] is too small, a new call to MUTSIN with IEXT = 1 and a new value for C results in the computation of a new solution with the new value for C. In this case MUTSIN continues the integration from the old value of C to the new value of C, so only the value for IEXT and C may be changed between succesive calls.
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NTI, N).
On exit X(i,k,1), i = 1,2, . . . , N contains the solution of the BVP at the output point TI(k), k = 1, . . . , NRTI. If there is no unique solution the base of the manifold is given in X(i,k,j), j = 2, . . . , NRSOL.

NRSOL INTEGER.
On exit NRSOL contains the information concerning the uniqueness of the solution. If NRSOL = 1 the solution is unique, otherwise the solution of the problem is a manifold for which the base is given in X(i,k,j), j = 2, . . . , NRSOL.

U DOUBLE PRECISION array of dimension (NU, NTI).
On exit U(i,k) i = 1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , KEXT. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k),... ,U(nj + j, k), where nj = (j − 1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to N * (N+1) / 2.

Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N, N, NTI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal matrix Q_k, k = 1, . . . , KEXT.

D DOUBLE PRECISION array of dimension (N, NTI).
If IHOM = 0 the array D has no real use and the user is recommended to use the same array for the X and the D.
If IHOM = 1 : on exit D(i,k) i = 1,2, . . . , N contains the inhomogeneous term d_k, k = 1,2, . . . , KEXT, of the multiple shooting recursion.

KEXT INTEGER.
KEXT denotes the total number of points used to compute the solution. If k denotes the number of output points on the interval [A , B] and m the number of points used on the extension interval [B , C], then KEXT = k + m.
On entry: if IEXT = 0, no value for KEXT is needed; if IEXT = 1, KEXT must contain the exit value of the previous call to MUTSIN.
On exit: KEXT contains the value for k + m.

KU INTEGER.
On exit KU is the number of detected unbounded growing modes on the interval [A , C]. Growing modes with an increment greater than 2 are considered to be unbounded modes.

KE INTEGER.
On entry: when IEXT $\neq$ 0, KE must contain the value from the previous call to MUTSIN.
On exit: KE contains the detected number of exponentially growing modes on the interval [B , C]. Growing modes are considered to be exponentially increasing when there increment on the interval [B , C] is greater than
1 / max(ER(1), ER(2)).

KPART INTEGER.
On exit KPART contains the global k-partition of the upper triangular matrices U_k.

PHI DOUBLE PRECISION array of dimension (NU, NTI).
On exit PHI contains the (KE + 1)th till the Nth columns of the fundamental solution of the multiple shooting recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
If IHOM = 0: LW ≥ 8*N + 7*N*N; if IHOM = 1: LW ≥ 9*N + 7*N*N.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ 4*N + 1.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, BCMAV, CDI, CEVIN, CNRHS, COPMAT, COPVEC, CONDW, CROUT, CWISB, DEFINC, DUR, FCBVP, FC2BVP, FQUS, FUNPAR, FUNRC, GTURI, INPRO, INTCH, KPCH, LUDEC, MATVC, PSR, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SBVP, SOLDE, SOLUPP, SORTD, TAMVC, TUR, UPUP, UPVECP.

****
Remarks
****

MUTSIN is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****
Method
****

See chapter III.

****
Example of the use of MUTSIN
****

Consider the ordinary differential equation

$$\frac{d}{dt}x(t)=\begin{bmatrix}2 & 2+0.4t\\ 0 & -0.4t\end{bmatrix}x(t)+\begin{bmatrix}-4-0.4t\\ 0.4t\end{bmatrix}$$

and a boundary condition

$$\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}x(0)+\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}x(\infty)=\begin{bmatrix}1\\ 2\end{bmatrix}.$$

The solution of this problem is:

$$x(t)=[\,1-exp(-0.2t^2)\,,1+exp(-0.2t^2)\,]^T\,.$$

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      DOUBLE PRECISION A,B,C,MA(2,2),MINF(2,2),BCV(2),AMP,ER(5),TI(13),
     1         X(2,13,2),U(3,13),Q(2,2,13),D(2,13),PHIREC(3,13),
     2         W(46),XEX,E,ERR
      INTEGER IW(9)
      EXTERNAL FLIN,FINH
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 2
      IHOM = 1
      A = 0.D0
      B = 10.D0
      C = 20.D0
      ER(1) = 1.1D-12
      ER(2) = 1.D-6
      CALL EPSMAC(ER(3))
      NRTI = 10
      NTI = 13
      IEXT = 0
      NU = 3
      LW = 46
      LIW = 9
C
C     SETTING THE BC MATRICES MA AND MINF AND THE BC VECTOR BCV
C
      MA(1,1) = 0.D0
      MA(1,2) = 0.D0
```

```
      MA(2,1) = 0.D0
      MA(2,2) = 1.D0
      MINF(1,1) = 1.D0
      MINF(1,2) = 0.D0
      MINF(2,1) = 0.D0
      MINF(2,2) = 0.D0
      BCV(1) = 1.D0
      BCV(2) = 2.D0
C
C     CALL TO MUTSIN
C
      CALL MUTSIN(FLIN,FINH,N,IHOM,A,B,C,MA,MINF,BCV,AMP,ER,NRTI,TI,NTI,
     1    IEXT,X,NRSOL,U,NU,Q,D,KU,KE,KEXT,KPART,PHIREC,W,LW,
     2    IW,LIW,IERROR)
      IF ((IERROR.EQ.0).OR.((IERROR.GE.200).AND.(IERROR.LE.213)).OR.
     1 (IERROR.EQ.300).OR.((IERROR.GE.330).AND.
     2 (IERROR.LE.340))) THEN
C
C     PRINTING  A, B ,THE ACTUAL USED VALUE FOR GAMMA, TOLERANCE,
C     CONDITION NUMBER AND AMPLIFICATION FACTOR.
C
        WRITE(*,100) A,B,TI(KEXT),ER(2),ER(4),ER(5)
C
C     COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND PRINTING
C     THE SOLUTION AT THE OUTPUT POINTS.
C
        WRITE(*,110)
        DO 1100 K = 1 , NRTI
          E = DEXP(-0.2d0*TI(K)*TI(K))
          XEX = 1.D0 - E
          ERR = XEX - X(1,K,1)
          WRITE(*,120) TI(K),X(1,K,1),XEX,ERR
          XEX = 1.D0 + E
          ERR = XEX - X(2,K,1)
          WRITE(*,130) X(2,K,1),XEX,ERR
 1100   CONTINUE
        IF (NRSOL.GT.1) THEN
          WRITE(*,140)
          DO 1200 K = 1 , NRTI
            WRITE(*,150) TI(K),X(1,K,2)
            WRITE(*,160) X(2,K,2)
 1200     CONTINUE
        ENDIF
C      ENDIF NRSOL
        ELSE
        WRITE(*,300) IERROR
```

```
      ENDIF
C     ENDIF IERROR
C
 100  FORMAT(' A   ',D12.5,2X,'B = ',D12.5,2X,'C = ',D12.5,/,
     1 ' TOL = ',D12.5,2X,' COND = ',D12.5,2X,'AMPLI = ',D12.5,/)
 110  FORMAT(' ',3X,'T',9X,'X APPROX',11X,'X EXACT',11X,'ERROR',/)
 120  FORMAT(' ',F7.3,3(2X,D16.9))
 130  FORMAT(' ',7X,3(2X,D16.9))
 140  FORMAT(' SOLUTION IS OF THE FORM X + LAMBDA * PHI',/,' ',3X,'T',
     1 12X,'PHI',/)
 150  FORMAT(' ',F7.5,2X,D16.9)
 160  FORMAT(' ',9X,D16.9)
 300  FORMAT(' TERMINAL ERROR IN MUTSIN: IERROR = ',I3)
C
      STOP
      END
      SUBROUTINE FLIN(N,T,F)
C
C
      DOUBLE PRECISION T,F(2,2)
C
      F(1,1) = 2.D0
      F(1,2) = 2.D0 + 0.4D0 * T
      F(2,1) = 0.D0
      F(2,2) = - 0.4D0 * T
      RETURN
      END
      SUBROUTINE FINH(N,T,R)
C
C
      DOUBLE PRECISION T,R(2)
C
      R(1) = -0.4D0 * T - 4.D0
      R(2) =  0.4D0 * T
      RETURN
      END
```

```
A =  .00000D+00  B =  .10000D+02  C =  .16955D+02
TOL =  .10000D-05  COND =  .10000D+01  AMPLI =  .19981D+01

   T        X APPROX          X EXACT           ERROR

  .000   .222044605D-15   .000000000D+00   -.222044605D-15
         .200000000D+01   .200000000D+01    .199840144D-14
 1.000   .181269247D+00   .181269247D+00   -.569665703D-10
```

	.181873075D+01	.181873075D+01	.569801983D-10
2.000	.550671036D+00	.550671036D+00	.189427252D-09
	.144932896D+01	.144932896D+01	-.189325000D-09
3.000	.834701112D+00	.834701112D+00	-.691497193D-09
	.116529889D+01	.116529889D+01	.692252700D-09
4.000	.959237798D+00	.959237796D+00	-.192215954D-08
	.104076220D+01	.104076220D+01	.192774530D-08
5.000	.993262055D+00	.993262053D+00	-.160490565D-08
	.100673795D+01	.100673795D+01	.164617830D-08
6.000	.999253414D+00	.999253414D+00	-.210793272D-09
	.100074659D+01	.100074659D+01	.515759879D-09
7.000	.999944546D+00	.999944548D+00	.216747531D-08
	.100005545D+01	.100005545D+01	.859421423D-10
8.000	.999997223D+00	.999997239D+00	.166426903D-07
	.100000276D+01	.100000276D+01	.793609622D-11
9.000	.999999785D+00	.999999908D+00	.123031966D-06
	.100000009D+01	.100000009D+01	.434541292D-12
10.000	.999999089D+00	.999999998D+00	.909093262D-06
	.100000000D+01	.100000000D+01	.139888101D-13

6. Subroutine MUTSMP

```
****************
SPECIFICATION
****************

      SUBROUTINE MUTSMP(FLIN, FINH, N, IHOM, TBP, NBP, BCM, BCV, ALI,
     1               ER, NRTI, TI, NTI, X, U, NU, Q, D,
     2               KPART, PHI, W, LW, IW, LIW, IERROR)
C     INTEGER N, IHOM, NBP, NRTI(NBP), NTI, NU, KPART(NBP), LW, IW(LIW),
C    1      LIW, IERROR
C     DOUBLE PRECISION TBP(NBP), BCM(NBP), BCV(N), ALI, ER(5), TI(NTI),
C    1 X(N,NTI), U(NU,NTI), Q(N,N,NTI), D(N,NTI), PHI(NU,NTI), W(LW)
C     EXTERNAL FLIN, FINH
```

****
Purpose
****

MUTSMP solves the multipoint BVP:

$$\frac{d}{dt}x(t) = L(t)x(t) + r(t) \ , \quad \alpha_1 \le \alpha_k \ or \ \alpha_k \le t \le \alpha_1 \, ,$$

with BC:

$$M_1 x(\alpha_1) + M_2 x(\alpha_2) + \cdots + M_k x(\alpha_k) = BCV, \ k > 1 \, ,$$

where $M_1, j = 1, \ldots, k$ are the BC matrices, BCV the BC vector and $\alpha_1 < \cdots < \alpha_k$ or $\alpha_1 > \cdots > \alpha_k$ the switching points.

****
Parameters
****

FLIN SUBROUTINE, supplied by the user with specification:

```
SUBROUTINE FLIN(N, T, FL)
DOUBLE PRECISION T, FL(N,N)
```

where N is the order of the system. FLIN must evaluate the matrix $L(t)$ of the differential equation for t = T and place the result in the array FL(N,N).
FLIN must be declared as EXTERNAL in the (sub)program from which MUTSMP is called.

FINH SUBROUTINE, supplied by the user, with specification:

```
SUBROUTINE FINH(N, T, FR)
DOUBLE PRECISION T, FR(N)
```

where N is the order of the system. FINH must evaluate the vector $r(t)$ of the differential equation for t = T and place the result in FR(1), FR(2), . . . , FR(N).
FINH must be declared as EXTERNAL in the (sub)program from which MUTSMP is called.
In the case that the system is homogeneous FINH is a dummy and one can use FLIN for FINH in the call to MUTSMP.

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

TBP DOUBLE PRECISION array of dimension (m), m ≥ NBP.
On entry TBP must contain the switching points α_j, $j = 1, \ldots$, NBP in monotone order, i.e. TBP(j) = α_j, $j = 1, \ldots$, NBP.
Unchanged on exit.

NBP INTEGER. NBP is the number of switching points.
Unchanged on exit.

BCM DOUBLE PRECISION array of dimension (N, N, m), m ≥ NBP.
On entry : BCM(. , . , j) must contain the BC matrix M_j, $j = 1, \ldots$, NBP.
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector.
Unchanged on exit.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2*ALI, a new output point is inserted.
If ALI ≤ 1 the defaults are:
If NRTI(1) = 0 : ALI := max(ER(1), ER(2)) / (2*ER(3)),
if NRTI(1) ≠ 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.

On exit ALI contains the actually used incremental factor.

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into
ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation.
On entry ER(3) must contain the machine constant EPS (see Remark 1.1).
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimate of the condition number of the BVP.
On exit ER(5) contains an estimate of the amplification factor.

NRTI INTEGER array of dimension (m), $m \geq$ NBP
On entry NRTI is used to specify the required output points. There are three ways to specify the required output points:
1) NRTI(1) = 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.
2) NRTI(1) = 1, the output points are supplied by the user in the array TI.
3) NRTI(1) > 1, in this case the intervals [TBP(j − 1 , TBP(j)], j = 2, . . . , NBP are divided into NRTI(j) subintervals of equal length. The endpoints of these intervals are the required output points.
Depending on the allowed incremental factor ALI, more output points may be inserted in the cases 2 and 3.
On exit: NRTI(1) contains the total number of output points.
For j = 2, . . . , NBP: if NRTI(j) < 0 then no change of dichotomy is detected on the succesive intervals [TBP(j − 1) , TBP(j)] and [TBP(j) , TBP(j+1)].
If NRTI(j) > 0 then a change of dichotomy is dectected at TBP(j) and NRTI(j) contains the number of output points on the interval [TBP(i) , TBP(j)], where
i < j, NRTI(i) > 0, NRTI(k) < 0, i < k < j, i.e. TBP(i) is the previous point where a change of dichotomy was detected.

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: α_1 = TI(1) < $\cdots$ < TI(k) = α_k or α_1 = TI(1) > $\cdots$ > TI(k) = α_k (k denotes the total number of required output points). The output points must include all switching points α_j, $j = 1, \ldots,$ NBP.
On exit: TI(i), i = 1,2, . . . , NRTI(1), contains the output points.

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, D, PHI. When m(j) denotes the number of output points on the interval
[TBP(j − 1) , TBP(j)], j = 2, . . . , NBP, and m the number of output points on the interval [TBP(1) , TBP(NBP)], i.e. m = m(2) + $\cdots$ +m(NBP) − NBP + 2, then

NTI ≥ m + 1 + max(4 − m(NBP) , 0).
If the routine was called with NRTI(1) > 1 and ALI ≤ 1 then
m = NRTI(2) + · · · + NRTI(NBP)+1 , m(NBP)=NRTI(NBP)+1 ; so
NTI ≥ 2 + NRTI(2) + · · · + NRTI(NBP) + max(3 − NRTI(NBP) , 0).
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NTI).
On exit X(i,k) , i = 1,2, . . . , N contains the solution of the BVP at the output point TI(k), k = 1, . . . , NRTI(1).

U DOUBLE PRECISION array of dimension (NU, NTI).
On exit U(i,k) i = 1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NRTI(1) The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k), . . . , U(nj + j, k), where nj = (j − 1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to N * (N+1) / 2.
Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N, N, NTI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal matrix Q_k, k = 1, . . . , NRTI(1).

D DOUBLE PRECISION array of dimension (N,NTI).
If IHOM = 0 the array D has no real use and the user is recommended to use the same array for the X and the D.
If IHOM = 1 : on exit D(i,k) i = 1,2, . . . , N contains the inhomogeneous term d_k, k = 1,2, . . . , NRTI(1), of the multiple shooting recursion.

KPART INTEGER array of dimension (m), m ≥ NBP
On exit KPART(j) contains the global partitioning parameter on the interval [TBP(i_j) , TBP(i_{j+1})], j = 1, . . . , where the TBP(i_j) are the points where a change of dichotomy has been detected; $i_1 < i_2 < \cdots$ and NRTI(i_j) > 0.

PHI DOUBLE PRECISION array of dimension (NU, NTI).
On exit PHI contains a fundamental solution of the multiple shooting recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
If IHOM=0 : LW ≥ (8 + 2.5*NBP)*N + (7 + 1.5*NBP)*N*N.
If IHOM=1 : LW ≥ (9 + 2.5*NBP)*N + (7 + 1.5*NBP)*N*N.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ (4 + NBP)*N + NBP + 2.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, CDI, CNRHS, COPMAT, COPVEC, CONDW, CROUT, CWISB, DEFINC, DUR, FCBVP, FC2BVP, FQUS, FUNPAR, FUNRC, GKPMP, GTUR, INPRO, INTCH, KPCH, LUDEC, MATVC, MTSMP, PSR, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SMBVP, SOLDE, SOLUPP, SORTD, TAMVC, TUR, UPUP, UPVECP.

****
Remarks
****

MUTSMP is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****
Method
****

See chapter IV.

****

Example of the use of MUTSMP

****

Consider the ordinary differential equation

$$\frac{d}{dt}x(t) = L(t)\,x(t) + r(t)\ ,\quad -1 \le t \le 1$$

and a boundary condition:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x(-1) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x(0) + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} x(1) = \begin{bmatrix} e \\ 1+e^{-1} \end{bmatrix},$$

where

$$L(t) = \begin{bmatrix} -t + ½ - (t+½)\cos(2t) & 1 + (t+½)\sin(2t) \\ -1 + (t+½)\sin(2t) & -t + ½ + (t+½)\cos(2t) \end{bmatrix},$$

$$r(t) = \begin{bmatrix} (-3 + \cos(t)(\cos(t) - \sin(t))(2t+1))\,e^{-t} \\ (-1 + \sin(t)(\sin(t) - \cos(t))(2t+1))\,e^{-t} \end{bmatrix}.$$

The solution of this problem is: $x(t) = (\,e^{-t}\,,\,e^{-t}\,)^T$. The ODE has fundamental solutions growing like $exp(-t^2)$ and $exp(t)$, so there is a change of dichotomy at $t = 0$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      DOUBLE PRECISION TBP(3),BCM(2,2,3),BCV(3),ALI,ER(5),TI(10),
     1 X(2,10),U(3,10),Q(2,2,10),D(2,10),PHIREC(3,10),W(79),
     2 EXSOL,AE
      INTEGER KPART(3),NRTI(3),IW(19)
      EXTERNAL FLIN,FINH
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 2
      IHOM = 1
      NBP = 3
      TBP(1) = -1.D0
      TBP(2) = 0.D0
      TBP(3) = 1.D0
      ALI = 0
      ER(1) = 1.D-11
```

```
      ER(2) = 1.D-6
      CALL EPSMAC(ER(3))
      NRTI(1) = 2
      NRTI(2) = 4
      NRTI(3) = 4
      NTI = 10
      NU = 3
      LW = 79
      LIW = 19
C
C     SETTING THE BC MATRICES
C
      DO 1100 I = 1 , NBP
      DO 1100 J = 1 , N
      DO 1100 L = 1 , N
       BCM(J,L,I) = 0.D0
1100  CONTINUE
      BCM(1,1,1) = 1.D0
      BCM(2,1,2) = 1.D0
      BCM(2,2,3) = 1.D0
C
C     SETTING THE BC VECTOR BCV
C
      BCV(1) = DEXP(1.D0)
      BCV(2) = 1.D0 + DEXP(-1.D0)
C
C     CALL MUTSMP
C
      CALL MUTSMP(FLIN,FINH,N,IHOM,TBP,NBP,BCM,BCV,ALI,ER,NRTI,TI,NTI,
     1            X,U,NU,Q,D,KPART,PHIREC,W,LW,IW,LIW,IERROR)
      IF ((IERROR.NE.0).AND.(IERROR.NE.200).AND.(IERROR.NE.213).AND.
     1 (IERROR.NE.240)) GOTO 5000
C
C     COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND
C     WRITING OF THE SOLUTION AT THE OUTPUTPOINTS
C
      WRITE(6,200) (TBP(I),I=1,NBP)
      WRITE(6,190) ER(4),ER(5)
      WRITE(6,210)
      DO 1500 K = 1 , NRTI(1)
       EXSOL = DEXP(TI(K))
       AE = EXSOL - X(1,K)
       WRITE(6,220) K,TI(K),X(1,K),EXSOL,AE
       DO 1300 I = 2 , N
        AE = EXSOL - X(I,K)
        WRITE(6,230) X(I,K),EXSOL,AE
```

```
1300    CONTINUE
1500   CONTINUE
       STOP
5000   WRITE(6,300) IERROR
       STOP
C
190    FORMAT(' CONDITION NUMBER     = ',D10.3,/,
     1  ' AMPLIFICATION FACTOR = ',D10.3,/)
200    FORMAT(' SWITCHING POINTS: ',3(F5.2,3X),/)
210    FORMAT('  I ',6X,'T',8X,'APPROX. SOL.',9X,'EXACT SOL.',8X,
     1  'ABS. ERROR',/)
220    FORMAT(' ',I3,3X,F7.4,3(3X,D16.9))
230    FORMAT(' ',13X,3(3X,D16.9))
300    FORMAT(' TERMINAL ERROR IN MUTSMP: IERROR = ',I4)
C
       END
C
       SUBROUTINE FLIN(N,T,FL)
C
       DOUBLE PRECISION T,FL(N,N)
       DOUBLE PRECISION TI,SI,CO
C
       T1 = 2.D0 * T
       SI = (T+0.5D0)*DSIN(T1)
       CO = (T+0.5D0)*DCOS(T1)
       TI = -T + 0.5D0
       FL(1,1) = T1 - CO
       FL(1,2) = 1.D0 + SI
       FL(2,1) = -1.D0 + SI
       FL(2,2) = TI + CO
C
       RETURN
C      END OF FLIN
       END
C
       SUBROUTINE FINH(N,T,FR)
C
       DOUBLE PRECISION T,FR(N)
       DOUBLE PRECISION TI,ET,SI,CO
C
       SI = DSIN(T)
       CO = DCOS(T)
       TI = (CO - SI) * (2*T + 1.D0)
       ET = DEXP(-T)
       FR(1) = (-3.D0 + CO*TI) * ET
       FR(2) = (-1.D0 - SI*TI) * ET
```

```
C
      RETURN
C     END OF FINH
      END
```

SWITCHING POINTS: -1.00 .00 1.00

CONDITION NUMBER = 0.613D+01
AMPLIFICATION FACTOR = 0.543D+01

I	T	APPROX. SOL.	EXACT SOL.	ABS. ERROR
1	-1.000	.271828183D+01	.271828183D+01	.000000000D+00
		.271828175D+01	.271828183D+01	.735283456D-07
2	-.750	.211699998D+01	.211700002D+01	.392049313D-07
		.211699991D+01	.211700002D+01	.108340283D-06
3	-.500	.164872118D+01	.164872127D+01	.933285536D-07
		.164872114D+01	.164872127D+01	.128283102D-06
4	-.250	.128402527D+01	.128402542D+01	.150341578D-06
		.128402529D+01	.128402542D+01	.127439107D-06
5	.000	.999999808D+00	.100000000D+01	.191680895D-06
		.999999891D+00	.100000000D+01	.109373630D-06
6	.250	.778800571D+00	.778800783D+00	.211765096D-06
		.778800694D+00	.778800783D+00	.886011160D-07
7	.500	.606530374D+00	.606530660D+00	.285309541D-06
		.606530718D+00	.606530660D+00	-.580605503D-07
8	.750	.472366284D+00	.472366553D+00	.268479161D-06
		.472366790D+00	.472366553D+00	-.237313363D-06
9	1.000	.367879306D+00	.367879441D+00	.134962732D-06
		.367879633D+00	.367879441D+00	-.191680895D-06

7. Subroutine MUTSMI

****

SPECIFICATION

****

```
      SUBROUTINE MUTSMI(FLIN, FINH, FMT, N, IHOM, A, B, NRTI, ALI, TI,
     1              NTI, ER, BCV, X, TSW, NSW, NRSW, U, NU, Q, D,
     2              KP, PHI, BMI, W, LW, IW, LIW, IERROR)
C     INTEGER N, IHOM, NRTI, NTI, NSW, NRSW, NU, KP(NSW), LW, IW(LIW),
C    1              LIW, IERROR
C     DOUBLE PRECISION A, B, ALI, TI(NTI), ER(5), BCV(N), X(N,NTI),
C    1              TSW(NSW), U(NU,NTI), Q(N,N,NTI), D(N,NTI),
C    2              BMI(N,N,NTI), PHI(NU,NTI), W(LW)
C     EXTERNAL FLIN, FINH, FMT
```

****

Purpose

****

MUTSMI solves BVP with integral BC:

$$\frac{d}{dt}x(t)=L(t)x(t)+r(t)\ ,\quad A\leq B\ ,$$

with BC:

$$\int_A^B M(t)x(t)\,dt=BCV\ ,$$

where $M(t)$ is an N×N matrix function and BCV an N-vector.

****

Parameters

****

FLIN SUBROUTINE, supplied by the user with specification:

```
SUBROUTINE FLIN(N, T, FL)
DOUBLE PRECISION T, FL(N,N)
```

where N is the order of the system. FLIN must evaluate the matrix $L(t)$ of the differential equation for t = T and place the result in the array FL(N,N).
FLIN must be declared as EXTERNAL in the (sub)program from which MUTSMI is called.

FINH SUBROUTINE, supplied by the user, with specification:

```
SUBROUTINE FINH(N, T , FR)
DOUBLE PRECISION T, FR(N)
```

where N is the order of the system. FINH must evaluate the vector $r(t)$ of the differential equation for t = T and place the result in FR(1), FR(2), . . . , FR(N).
FINH must be declared as EXTERNAL in the (sub)program from which MUTSMI is called.
In the case that the system is homogeneous FINH is a dummy and one can use FLIN for FINH in the call to MUTSMI.

FMT SUBROUTINE supplied by the user, with specification:

```
SUBROUTINE FMT(N, T, FM)
DOUBLE PRECISION T, FM(N,N)
```

where N is the order of the system. FMT must evaluate the matrix $M(t)$ of the integral BC for t = T and place the result in the array FM(N,N).
FMT must be declared as EXTERNAL in the (sub)program from which MUTSMI is called.

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

A,B DOUBLE PRECISION, the two boundary points.
Unchanged on exit.

NRTI INTEGER
On entry NRTI is used to specify the required output points. There are three ways to specify the required output points:
1) NRTI = 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.
2) NRTI = 1, the output points are supplied by the user in the array TI.
3) NRTI > 1, the subroutines computes the (NRTI+1) output points TI(k) by:

TI(k) = A + (k − 1) * (B − A) / NRTI

so TI(1) = A and TI(NRTI+1) = b.

More output points may be inserted in the cases 2 and 3, depending on the allowed incremental factor ALI. Also if a new switching point is detected or if $\| \int M(t)x(t)\,dt \|$ becomes larger than ER(2) / ER(3), a new output point is inserted.

On exit NRTI contains the total number of output points.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2*ALI, a new output point is inserted. If ALI $\leq$ 1 the defaults are:
If NRTI = 0 : ALI := max(ER(1), ER(2)) / (2*ER(3)),
if NRTI $\neq$ 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.
On exit ALI contains the actually used incremental factor.

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: A = TI(1) $< \cdots <$ TI(k) = B or B = TI(1) $> \cdots >$ TI(k) = B
(k denotes the total number of required output points).
On exit: TI(k), k = 1,2, . . . , NRTI, contains the output points.

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, D, BMI, PHI.
Let m be the total number of output points then NTI $\geq$ max(5, m + 1).
If the routine was called with NRTI > 1 and ALI $\leq$ 1 the total number of required output points is NRTI + 1, so NTI $\geq$ max(5, NRTI + 2), if the required output points include possible switching points, otherwise NTI $\geq$ max(5, NRTI + 2) + k, where k denotes the number of switching points between A and B (k $\leq$ N).
Unchanged on exit.

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into
ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation.
On entry ER(3) must contain the machine constant EPS (see Remark 1.1).
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimate of the condition number of the BVP.
On exit ER(5) contains an estimate of the amplification factor.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector.
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NTI).
On exit X(i,k) , i = 1,2, . . . , N contains the solution of the BVP at the output point TI(k), k = 1, . . . , NRTI.

TSW DOUBLE PRECISION array of dimension (m), m ≥ N + 2.
On exit TSW contains the NRSW detected switching points. Note that the boundary points A and B are also switching points and that the maximum number of switching points is N + 2.

NSW INTEGER. NSW denotes the number of possible switching points.
On entry NSW ≥ N + 2.
Unchanged on exit.

NRSW INTEGER.
On exit NRSW contains the number of detected switching points.

U DOUBLE PRECISION array of dimension (NU, NTI).
On exit U(i,k) i = 1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NRTI. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k), . . . , U(nj + j, k), where nj = (j − 1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to N * (N + 1) / 2.
Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N, N, NTI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal matrix Q_k, k = 1, . . . , NRTI.

D DOUBLE PRECISION array of dimension (N, NTI).
If IHOM = 0 the array D has no real use and the user is recommended to use the same array for the X and the D.
If IHOM = 1 : on exit D(i,k) i = 1,2, . . . , N contains the inhomogeneous term d_k, k = 1,2, . . . , NRTI, of the multiple shooting recursion.

KP INTEGER
On exit KP(j) contains the global partitioning parameter of the interval [TSW(j) , TSW(j + 1)], j = 1, . . . , NRSW − 1.

PHI DOUBLE PRECISION array of dimension (NU,NTI).
On exit PHI contains a fundamental solution of the multiple shooting recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

BMI DOUBLE PRECISION array of dimension (N,N,NTI).
On exit BMI(. , . , j) contains the BC matrix of the discretised integral BC at the output point TI(j), j = 1, . . . , NRTI − 1.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
If $N < 8 : LW \geq 15 * N * N + 21 * N$.
If $N \geq 8 : LW \geq (3 * N * N * N + 11 * N * N) / 2 + 5 * N$.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ N*N + 6*N + NTI.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, ANORM1, APLB, CDI, CHDIAU, CKPSW ,CNRHS, COPMAT, COPVEC, CONDW, CPRDIA, CROUT, CWISB, DEFINC, DETSWP, DURIN, FCBVP, FCIBVP, FQUS, FUNPAR, FUNRC, GKPMP, INPRO, INTCH, KPCH, LUDEC, MATVC, PSR, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SMBVP, SOLDE, SOLUPP, SORTD, TAMVC, UPUP, UPVECP.

****
Remarks
****

MUTSMI is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991

****
Method
****

See chapter IV.

****
Example of the use of MUTSMI
****

Consider the ordinary differential equation

$$\frac{d}{dt}x(t) = L(t)\,x(t) + r(t) \ , \quad -4 \le t \le 4$$

and an integral boundary condition:

$$\int_{-4}^{4} M(t)x(t)\,dt = b \ ,$$

where

$$L(t) = \begin{bmatrix} 1 & 0 \\ 0 & -2t \end{bmatrix} , \ r(t) = \begin{bmatrix} -e^{-t} \\ (2t-1)\,e^{-t} \end{bmatrix} \ M(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} , \ b = \begin{bmatrix} 2\sinh 4 \\ 2\sinh 4 \end{bmatrix} .$$

The solution of this problem is: $x(t) = [\,\cosh t \,,\, e^{-}t\,]^T$.
The ODE has fundamental solutions growing like $\sim e^{-t^2}$ and $\sim e^{t}$, so there is a change of dichotomy at $t = 0$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION TI(10),ER(5),X(2,10),BCV(2),TSW(4),Q(2,2,10),U(3,10),
     1 D(2,10),BMI(2,2,10),PHI(3,10),W(102)
      INTEGER KP(4),IW(27)
      EXTERNAL FLIN,FINH,FMT
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 2
      NU = 3
      NTI = 10
      NSW = 4
      LW = 102
      LIW = 27
```

```
      IHOM = 1
      ER(1) = 1.1D-12
      ER(2) = 1.D-6
      CALL EPSMAC(ER(3))
      A = -4.D0
      B = 4.D0
      ALI = 0.D0
      NRTI = 8
C
C     SETTING THE BOUNDARY CONDITION VECTOR
C
      BCV(1) = 2.D0 * DSINH(4.D0)
      BCV(2) = BCV(1)
C
C     CALL TO MUTSMI
C
      CALL MUTSMI(FLIN,FINH,FMT,N,IHOM,A,B,NRTI,ALI,TI,NTI,ER,BCV,X,
     1   TSW,NSW,NRSW,U,NU,Q,D,KP,PHI,BMI,W,LW,IW,LIW,IERROR)
      IF (IERROR.NE.0) GOTO 2000
C
C     WRITING OF THE SWITCHING POINTS, THE CONDITION NUMBER AND
C     THE ERROR AMPLIFICATION ERROR.
C
      WRITE(*,200) (TSW(I),I=1,NRSW)
      WRITE(*,210) ER(4),ER(5)
      WRITE(*,220)
      DO 1300 I = 1 , NRTI
       E = DCOSH(TI(I))
       AE = X(1,I) - E
       WRITE(*,230) I,TI(I),X(1,I),E,AE
       E = DEXP(-TI(I))
       AE = X(2,I) - E
       WRITE(*,240) X(2,I),E,AE
 1300 CONTINUE
      STOP
 2000 WRITE(*,300) IERROR
      STOP
C
  200 FORMAT(' SWITCHING POINTS:',4(F10.6,4X),/)
  210 FORMAT(' CONDITION NUMBER     = ',D12.5,/
     1 ' AMPLIFICATION FACTOR = ',D12.5,/)
  220 FORMAT('  I',6X,'T',8X,'APPROX. SOL.',7X,'EXACT SOL.',9X,
     1 'ABS. ERROR',/)
  230 FORMAT(' ',I3,3X,F7.3,3(3X,D16.9))
  240 FORMAT(' ',13X,3(3X,D16.9))
  300 FORMAT(' TERMINAL ERROR IN MUTSMI: IERROR = ',I4)
```

```
C
      END
      SUBROUTINE FLIN(N,T,FL)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION FL(N,N)
C
      FL(1,1) = 1.D0
      FL(1,2) = 0.D0
      FL(2,1) = 0.D0
      FL(2,2) = -2.D0*T
      RETURN
      END
      SUBROUTINE FINH(N,T,FR)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION FR(N)
C
      E = DEXP(-T)
      FR(1) = -E
      FR(2) = (2.D0*T - 1.D0) * E
      RETURN
      END
      SUBROUTINE FMT(N,T,FM)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION FM(N,N)
      FM(1,1) = 1.D0
      FM(1,2) = 0.D0
      FM(2,1) = 0.D0
      FM(2,2) = 1.D0
      RETURN
      END
```

```
SWITCHING POINTS: -4.000000     .000000    4.000000
CONDITION NUMBER         = 0.10003D+01
AMPLIFICATION FACTOR     = 0.17067D+01
```

I	T	APPROX. SOL.	EXACT SOL.	ABS. ERROR
1	-4.000	.273082328D+02	.273082328D+02	.328726202D-08
		.545981500D+02	.545981500D+02	.535869304D-08
2	-3.000	.100676620D+02	.100676620D+02	.891424357D-08
		.200855369D+02	.200855369D+02	.109678666D-07
3	-2.000	.376219572D+01	.376219569D+01	.241194678D-07

		.738905615D+01	.738905610D+01	.507391977D-07
4	-1.000	.154308070D+01	.154308063D+01	.640715478D-07
		.271828220D+01	.271828183D+01	.375098707D-06
5	.000	.100000011D+01	.100000000D+01	.107381681D-06
		.100000002D+01	.100000000D+01	.173592072D-07
6	1.000	.154308067D+01	.154308063D+01	.325343779D-07
		.367879197D+00	.367879441D+00	-.243884238D-06
7	2.000	.376219570D+01	.376219569D+01	.123411108D-07
		.135335178D+00	.135335283D+00	-.104949590D-06
8	3.000	.100676620D+02	.100676620D+02	.110139773D-07
		.497870526D-01	.497870684D-01	-.157823572D-07
9	4.000	.273082328D+02	.273082328D+02	-.445510295D-10
		.183156315D-01	.183156389D-01	-.743454046D-08

-

8. Subroutine MUTSPA

****
SPECIFICATION
****

```
      SUBROUTINE MUTSPA(FLIN, FINH, FCT, N, L, NPL, IHOM, A, B, MA, MB, BCV,
     1    ALI, ER, NRTI, TI, NTI, X, Z, TSW, NSW, NRSW, U, NU, Q, D,
     2    KPART, CI, PHI, YI, W, LW, IW, LIW, IERROR)
C     INTEGER N, L, NLP, IHOM, NRTI, NTI, NU, NSW, NRSW, KPART(NSW), LW,
C         IW(LIW), LIW, IERROR
C     DOUBLE PRECISION A, B, MA(NPL,NPL), MB(NPL,NPL), BCV(NPL), ALI, ER(5),
C    1    TI(NTI), X(N,NTI), Z(L), TSW(NSW), U(NU,NTI), Q(N,N,NTI),
C    2    D(N,NTI), CI(N,NTI,L), PHI(NU,NTI), YI(N,NTI,L), W(LW)
C     EXTERNAL FLIN, FINH, FCT
```

****
Purpose
****

MUTSPA solves the two-point BVP with parameters:

$$\frac{d}{dt}x(t) = L(t)x(t) + C(t)z + r(t) \ , \quad A \le t \le B \ or \ B \le t \le A \ ,$$

with BC:

$$[M_A \mid P_A]\begin{bmatrix} x(A) \\ z \end{bmatrix} + [M_B \mid P_b]\begin{bmatrix} x(B) \\ z \end{bmatrix} = \begin{bmatrix} b_x \\ b_z \end{bmatrix},$$

where z is an L-vector containing the L parameters, M_A and M_B are NPL×N matrices, P_A and P_B are NPL×L matrices, B_x an N-vector and B_z an L-vector.

****
Parameters
****

FLIN SUBROUTINE, supplied by the user with specification:

```
SUBROUTINE FLIN(N, T, FL)
DOUBLE PRECISION T, FL(N,N)
```

where N is the order of the system. FLIN must evaluate the matrix $L(t)$ of the differential equation for t = T and place the result in the array FL(N,N).

FLIN must be declared as EXTERNAL in the (sub)program from which MUTSPA is called.

FINH SUBROUTINE, supplied by the user, with specification:

```
SUBROUTINE FINH(N, T, FR)
DOUBLE PRECISION T, FR(N)
```

where N is the order of the system. FINH must evaluate the vector $r(t)$ of the differential equation for t = T and place the result in FR(1), FR(2), . . . , FR(N).
FINH must be declared as EXTERNAL in the (sub)program from which MUTSPA is called.
In the case that the system is homogeneous FINH is a dummy and one can use FLIN for FINH in the call to MUTSPA.

FCT SUBROUTINE, supplied by the user, with specification:

```
SUBROUTINE FCT(N, L, T, FC)
DOUBLE PRECISION T, FC(N,L)
```

where N is the order of the system and L the number of parameters. FCT must evaluate the $N \times L$ matrix $C(t)$ of the differential equation for t = T and place the result in the array FC(N,L).
FCT must be declared as EXTERNAL in the (sub)program from which MUTSPA is called.

N INTEGER, the order of the system.
Unchanged on exit.

L INTEGER, the number of parameters
Unchanged on exit.

NPL INTEGER.
NPL is the dimension of the arrays MA, MB and BCV. NPL must have the value N+L.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

A,B DOUBLE PRECISION, the two boundary points.
Unchanged on exit.

MA,MB DOUBLE PRECISION array of dimension (NPL, NPL).
On entry : MA and MB must contain the BC matrices : $[M_A \mid P_A]$ and $[M_B \mid P_B]$ respectively.
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (NPL).

On entry BCV must contain the BC vector $\begin{bmatrix} b_x \\ b_z \end{bmatrix}$.
Unchanged on exit.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2*ALI, a new output point is inserted.
If ALI ≤ 1 the defaults are:
If NRTI = 0 : ALI := max(ER(1), ER(2)) / (2*ER(3)),
if NRTI $\neq$ 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.
On exit ALI contains the actually used incremental factor.

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into
ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation.
On entry ER(3) must contain the machine constant EPS (see Remark 1.1).
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimate of the condition number of the BVP.
On exit ER(5) contains an estimate of the amplification factor.

NRTI INTEGER.
On entry NRTI is used to specify the required output points. There are three ways to specify the required output points:
1) NRTI = 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.
2) NRTI = 1, the output points are supplied by the user in the array TI.
3) NRTI > 1, the subroutine computes the (NRTI+1) output points TI(k) by:
TI(k) = A + (k − 1) * (B − A) / NRTI ;
so TI(1) = A and TI(NRTI+1) = B .
Depending on the allowed incremental factor ALI, more output points may be inserted in the cases 2 and 3. Furthermore detected switching points are also inserted. On exit NRTI contains the total number of output points.

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: A = TI(1) < · · · < TI(k) = B or A = TI(1) > · · · > TI(k) = B
(k denotes the total number of required output points).
On exit: TI(i), i = 1,2, . . . , NRTI, contains the output points, including possible switching points.

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, D, CI, PHI, YI.
Let m be the total number of output points then NTI ≥ max(5, m + 1).
If the routine was called with NRTI > 1 and ALI ≤ 1 the total number of required output points is NRTI + 1 , so NTI ≥ max(5, NRTI + 2), if the required output points include possible switching points, otherwise NTI ≥ max(5,NRTI+2) + k, where k is the number of switching points between A and B (k ≤ N).
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NTI).
On exit X(i,k) , i = 1,2, . . . , N contains the solution of the BVP at the output point TI(k), k = 1, . . . , NRTI.

Z DOUBLE PRECISION array of dimension (L)
On exit the array Z contains the values of the L parameters.

TSW DOUBLE PRECISION array of dimension (NSW)
On exit TSW contains the NRSW switching points:
A = TSW(1), . . . , TSW(NRSW) = B.

NSW INTEGER.
NSW is the dimension of array TSW and array KPART. NSW ≥ N + 2 !
Unchanged on exit

NRSW INTEGER.
On exit NRSW contains the total number of detected switching points.

U DOUBLE PRECISION array of dimension (NU, NTI).
On exit U(i,k) i = 1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NRTI. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k), . . . , U(nj + j, k), where nj = (j − 1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to N * (N + 1) / 2.

Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N, N, NTI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal matrix Q_k, k = 1, . . . , NRTI.

D DOUBLE PRECISION array of dimension (N, NTI).
If IHOM = 0 the array D has no real use and the user is recommended to use the same array for the X and the D.
If IHOM = 1 : on exit D(i,k) i = 1,2, . . . , N contains the inhomogeneous term d_k, k = 1,2, . . . , NRTI, of the multiple shooting recursion.

KPART INTEGER array of dimension (NSW)
On exit KPART(j) contains the global partitioning parameter of the interval [TSW(j) , TSW(j+1)], j = 1, . . . , NRSW − 1.

CI DOUBLE PRECISION array of dimension (N, NTI, L)
On exit CI(i,j,k) i = 1, . . . , N, k = 1, . . . , L contains the N×L matrix C_j, j = 2, . . . , NRTI.

PHI DOUBLE PRECISION array of dimension (NU, NTI).
On exit PHI contains a fundamental solution of the multiple shooting recursion (V.2.3). The fundamental solution is upper triangular and is stored in the same way as the U_k.

YI DOUBLE PRECISION array of dimension (N, NTI, L).
On exit YI contains the particular matrix solution Y_j of recursion (V.2.5). The particular N×L matrix solution is stored in the same way as the C_j.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
LW ≥ 7 * NRSW * NPL * (NPL + 1) / 2 + 4 * NPL * (NPL + 1)
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ N*N + 8*N + 4*L + 2.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, BCMAV, CAMPF, CCI, CDI, CFUNRC, CHDIAU, CKPSW, CNRHS, COPMAT, COPVEC, CONDW, CPRDIA, CPSRC, CROUT, CUVRC, CGTURC, CWISB, DEFINC, DETSWP, DURPA, FCBVP, FC2BVP, FQUS, FUNPAR, FUNRC, GKPPA, CPABC, CPARC, CSPABV, INPRO, INTCH, KPCH, LUDEC,MATVC, PSR, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SBVP, SOLDE, SOLUPP, SORTD, SPARC, SPLS1, TAMVC, UPUP, UPVECP.

****
Remarks
****

MUTSPA is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****
Method
****

See chapter V.

****
Example of the use of MUTSPA
****

Consider the ordinary differential equation with parameter z

$$\frac{d}{dt}x(t) = L(t)\,x(t) + C(t)\,z + r(t)\ ,\quad -5 \le t \le 5$$

and a boundary condition $M_\alpha \begin{bmatrix} x(-5) \\ z \end{bmatrix} + M_\beta \begin{bmatrix} x(5) \\ z \end{bmatrix} = b$, where

$$L(t) = \begin{bmatrix} 2 & 0 \\ 0 & tanh(t) \end{bmatrix},\quad C(t) = \begin{bmatrix} 0 \\ 1/cosh(t) \end{bmatrix},\quad r(t) = \begin{bmatrix} -2 \\ (1-sinh(t))/cosh(t) \end{bmatrix},$$

$$M_\alpha = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & \frac{1}{2} \\ 0 & 1 & -\frac{1}{2} \end{bmatrix},\quad M_\beta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & -1 & \frac{1}{2} \end{bmatrix}$$

and $b = [\,2, 2\,cosh(5), 2\,sinh(5)\,]^T$.

This problem has a switching point at $t = 0$ and the solution is:

$x(t) = (1 - exp(2(t-5)),\, 1 + exp(-t))^T$ and $z = -2$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION BCMA(3,3),BCMB(3,3),BCV(3),ER(5),TI(13),TSW(4),
     1 X(2,13),Z(1),U(3,13),Q(2,2,13),D(2,13),PHI(3,13),CI(2,13,1),
     2 YI(2,13,1),W(174)
      INTEGER KP(4),IW(26)
      EXTERNAL FLIN,FINH,FCT
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 2
      L = 1
      NPL = 3
      NSW = 4
      IHOM = 1
      NTI = 13
      NU = 3
      LW = 174
      LIW = 26
      ER(1) = 1.1D-12
      ER(2) = 1.D-6
      CALL EPSMAC(ER(3))
      A = -5.D0
      B = 5.D0
      ALI = 0.D0
      NRTI = 10
C
C     SETTING THE BOUNDARY CONDITIONS
C
      BCMA(1,1) = 0.D0
      BCMA(1,2) = 0.D0
      BCMA(1,3) = -1.D0
      BCMA(2,1) = 0.D0
      BCMA(2,2) = 1.D0
      BCMA(2,3) = 0.5D0
      BCMA(3,1) = 0.D0
```

```
      BCMA(3,2) = 1.D0
      BCMA(3,3) = -0.5D0
      BCMB(1,1) = 1.D0
      BCMB(1,2) = 0.D0
      BCMB(1,3) = 0.D0
      BCMB(2,1) = 0.D0
      BCMB(2,2) = 1.D0
      BCMB(2,3) = 0.5D0
      BCMB(3,1) = 0.D0
      BCMB(3,2) = -1.D0
      BCMB(3,3) = 0.5D0
      BCV(1) = 2.D0
      BCV(2) = 2.D0 * DCOSH(5.D0)
      BCV(3) = 2.D0 * DSINH(5.D0)
C
C     CALL MUTSPA
C
      CALL MUTSPA(FLIN,FINH,FCT,N,L,NPL,IHOM,A,B,BCMA,BCMB,BCV,AMP,ER,
     1      NRTI,TI,NTI,X,Z,TSW,NSW,NRSW,U,NU,Q,D,KP,CI,PHI,YI,W,
     2      LW,IW,LIW,IERROR)
      IF (IERROR.NE.0) GOTO 5000
C
C     PRINTING OF THE SWITCHING POINTS, CONDITION NUMBER AND
C     AMPLIFICATION FACTOR
C
        WRITE(*,105) (TSW(J),J=1,NRSW)
        WRITE(*,110) ER(4),ER(5)
C
C     COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND WRITING OF
C     THE SOLUTION AT THE OUTPUTPOINTS
C
      WRITE(*,*) ' Z = ',Z(1)
      WRITE(*,120)
      DO 1200 I = 1 , NRTI
       E1 = 1.D0 - DEXP(2.D0*(TI(I)-5.D0))
       E2 = E1 - X(1,I)
       WRITE(*,130) TI(I),X(1,I),E1,E2
       E1 = 1.D0 + DEXP(-TI(I))
       E2 = E1 - X(2,I)
       WRITE(*,135) X(2,I),E1,E2
1200  CONTINUE
      STOP
5000  WRITE(*,100) IERROR
      STOP
100   FORMAT(' TERMINAL ERROR IN MUTSPA: IERROR = ',I4)
105   FORMAT(' SWITCHING POINTS: ',4(F7.2,3X))
```

```
110     FORMAT(' CONDITION NUMBER     = ',D12.5,/,
     1  ' AMPLIFICATION FACTOR = ',D12.5,/)
120     FORMAT(' ',/,5X,'T',5X,'APPROX.SOL.',7X,'EXACT SOL.',8X,
     1  'ABS. ERROR',/)
130     FORMAT(' ',F7.3,3(2X,D16.9))
135     FORMAT(' ',7X,3(2X,D16.9))
        END
        SUBROUTINE FLIN(N,T,F)
C
        DOUBLE PRECISION T,F(N,N),TI
C
        F(1,1) = 2.D0
        F(1,2) = 0.D0
        F(2,1) = 0.D0
        F(2,2) = DTANH(T)
        RETURN
        END
        SUBROUTINE FINH(N,T,F)
C
        DOUBLE PRECISION T,F(N)
C
        F(1) = -2.D0
        F(2) = ( 1.D0 - DSINH(T)) / DCOSH(T)
        RETURN
        END
        SUBROUTINE FCT(N,L,T,F)
C
        DOUBLE PRECISION T,F(N,L)
C
        F(1,1) = 0.D0
        F(2,1) = 1.D0 / DCOSH(T)
        RETURN
        END
```

```
 SWITCHING POINTS:  -5.00     .00    5.00
CONDITION NUMBER      =  .10068D+01
AMPLIFICATION FACTOR  =  .20633D+01

 Z =     -1.99999998386800

   T       APPROX. SOL.       EXACT SOL.        ABS. ERROR

-5.000   .999999998D+00   .999999998D+00   .105471187D-13
         .149413159D+03   .149413159D+03   .806599587D-08
-4.000   .999999985D+00   .999999985D+00   .693889390D-13
```

	.555981495D+02	.555981500D+02	.552555562D-06
-3.000	.999999887D+00	.999999887D+00	.454525306D-12
	.210855365D+02	.210855369D+02	.400276733D-06
-2.000	.999999168D+00	.999999168D+00	.293742808D-11
	.838905589D+01	.838905610D+01	.207147576D-06
-1.000	.999993856D+00	.999993856D+00	.185887972D-10
	.371828175D+01	.371828183D+01	.746298814D-07
.000	.999954600D+00	.999954600D+00	.114339760D-09
	.199999998D+01	.200000000D+01	.201335302D-07
1.000	.999664537D+00	.999664537D+00	.674805989D-09
	.136787944D+01	.136787944D+01	.611146622D-08
2.000	.997521244D+00	.997521248D+00	.372963660D-08
	.113533528D+01	.113533528D+01	.192635374D-08
3.000	.981684343D+00	.981684361D+00	.182738354D-07
	.104978707D+01	.104978707D+01	.159673852D-08
4.000	.864664650D+00	.864664717D+00	.664215662D-07
	.101831564D+01	.101831564D+01	.312881987D-08
5.000	.161319893D-07	.000000000D+00	-.161319893D-07
	.100673794D+01	.100673795D+01	.806598388D-08

9. Subroutine MUTSDD

****
SPECIFICATION
****

```
      SUBROUTINE MUTSMP(FLIN, FINH, N, IHOM, TSP, NSP, BCM, BCV, ZM, ZP,
     1                BI, ALI, ER, NRTI, TI, NTI, X, U, NU, Q, D,
     2                KPART, PHI, W, LW, IW, LIW, IERROR)
C      INTEGER N, IHOM(NSP), NSP, NRTI(NSP), NTI, NU, KPART(NSP), LW, IW(LIW),
C     1       LIW, IERROR
C      DOUBLE PRECISION TBP(NBP), BCM(NBP), BCV(N), ZM(N,N,NSP), ZP(N,N,NSP),
C     1       BI(N,NSP), ALI, ER(6), TI(NTI), X(N,NTI), U(NU,NTI),
C     2       Q(N,N,NTI), D(N,NTI), PHI(NU,NTI), W(LW)
C      EXTERNAL FLIN, FINH
```

****
Purpose
****

MUTSDD solves the BVP with discontinuous data:

$$\frac{d}{dt}x(t)=L(t)x(t)+r(t) \qquad \alpha_i \le t < \alpha_{i+1}\ ,\ i=1,\ldots,m\ ,$$

with side conditions

$$Z_{i+1\_}\, x(\alpha_{i+1}^-)+Z_{i+1}^+\, x(\alpha_{i+1}^+)=b_{i+1}\ \ ,\ i=1,\ldots,m-1\ ,$$

and a BC

$$\sum_{i=1}^{m+1} M_i\, x(\alpha_i^+)=b\ ,$$

where the $L_i(t)$ are bounded continuous matrix functions,the $r_i(t)$ are bounded continuous vector functions, the Z_{i+1}^-, Z_{i+1}^+ are the side conditions matrices, the b_{i+1} are the side conditions vectors, the M_i are the BC matrices, b the BC vector and $\alpha_1 < \cdots < \alpha_{m+1}$ or $\alpha_1 > \cdots > \alpha_{m+1}$ the switching points.

****
Parameters
****

FLIN SUBROUTINE, supplied by the user with specification:

SUBROUTINE FLIN(N, T, FL)
DOUBLE PRECISION T, FL(N,N)

where N is the order of the system. FLIN must evaluate for t = T the corresponding matrix $L_i(t)$ of the differential equation and place the result in the array FL(N,N). FLIN must be declared as EXTERNAL in the (sub)program from which MUTSDD is called.

FINH SUBROUTINE, supplied by the user, with specification:

SUBROUTINE FINH(N, T, FR)
DOUBLE PRECISION T, FR(N)

where N is the order of the system. FINH must evaluate for t = T the coresponding vector $r_i(t)$ of the differential equation and place the result in FR(1), FR(2), . . . , FR(N).
FINH must be declared as EXTERNAL in the (sub)program from which MUTSDD is called.
In the case that the system is homogeneous, i.e. all the $r_i = 0$, FINH is a dummy and one can use FLIN for FINH in the call to MUTSDD.

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER array of dimension (k), k ≥ NSP
IHOM(i) indicates whether the system is homogeneous or inhomogeneous on $[\alpha_i, \alpha_{i+1}]$, i = 1, . . . , NSP-1.
On entry:
IHOM(i) = 0 : the system is homogeneous on $[\alpha_i, \alpha_{i+1}]$,
IHOM(i) = 1 : the system is inhomogeneous on $[\alpha_i, \alpha_{i+1}]$.
On exit IHOM(i), i=1,...,NSP-1 is unchanged; IHOM(NSP) = 0, if the whole system is homogeneous, otherwise IHOM(NSP) = 1.

TSP DOUBLE PRECISION array of dimension (k), k ≥ NBP. On entry TSP must contain the switching points α_j, j = 1, . . . , NSP in monotone order, i.e.
TSP(j) = α_j, j = 1, . . . , NSP.
Unchanged on exit.

NSP INTEGER. NSP is the number of switching points.
Unchanged on exit.

BCM DOUBLE PRECISION array of dimension (N,N,k), k ≥ NSP.
On entry : BCM(. , . , j) must contain the BC matrix M_j, j = 1, . . . , NSP.

During computation the array BCM will be overwritten.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector.
During computation the array BCV will be overwritten.

ZM DOUBLE PRECISION array of dimension (N, N, k), k ≥ NSP.
On entry ZM(. , . , j) must contain the side condition matrix Z_j^-, j = 2, . . . , NSP-1.
During computation the array ZM will be overwritten.

ZP DOUBLE PRECISION array of dimension (N, N, k), k ≥ NSP.
On entry ZP(. , . , j) must contain the side condition matrix Z_j^+, j = 2, . . . , NSP-1.
During computation the array ZP will be overwritten.

BI DOUBLE PRECISION array of dimension (N, k), k ≥ NSP.
On entry BI(. , j) must contain the side condition vector B_j, j = 2, . . . , NSP-1.
During computation the array BI will be overwritten.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2 * ALI, a new output point is inserted.
If ALI ≤ 1 the defaults are:
If NRTI(1) = 0 : ALI := max(ER(1), ER(2)) / (2*ER(3)),
if NRTI(1) ≠ 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.
On exit ALI contains the actually used incremental factor.

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into
ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation.
On entry ER(3) must contain the machine constant EPS (see Remark 1.1).
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimate of the condition number of the BVP.
On exit ER(5) contains an estimate of the amplification factor.
On exit ER(6) contains an estimate of the amplification factor of the discrete multipoint BVP.

NRTI INTEGER array of dimension (k), k ≥ NBP
On entry NRTI is used to specify the required output points. There are three ways to specify the required output points:

1) NRTI(1) = 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.
2) NRTI(1) = 1, the output points are supplied by the user in the array TI.
3) NRTI(1) > 1, in this case the interval [TBP(j − 1) , TBP(j)], j = 2, . . . , NSP, are divided into NRTI(j) subintervals of equal length. The endpoints of these subintervals are the required output points.

Depending on the allowed incremental factor ALI, more output points may be inserted in the cases 2 and 3.
On exit: NRTI(1) contains the total number of output points.
For j = 2, . . . , NBP: if NRTI(j) < 0 then no change of dichotomy is detected on the succesive intervals [TBP(j − 1), TBP(j)] and [TBP(j) , TBP(j+1)]. If NRTI(j) > 0 then a change of dichotomy is dectected at TBP(j) and NRTI(j) contains the number of output points on the interval [TBP(i) , TBP(j)], where i < j, NRTI(i) > 0, NRTI(k) < 0, i < k < j, i.e. TBP(i) is the previous point where a change of dichotomy was detected.

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: α_1 = TI(1) < · · · < TI(k) = α_k or α_1 = TI(1) > · · · > TI(k) = α_k (k denotes the total number of required output points). The output points must include all switching points α_j, $j = 1, \ldots,$ NBP.
The routine split the switching points α_j, $j = 2, \ldots,$ NSP − 1 into two output points $\alpha_j^- := \alpha_j(1 - \mathrm{EPS})$ and $\alpha_j^+ := \alpha_j(1 + \mathrm{EPS})$.
On exit: TI(i), i = 1,2, . . . , NRTI(1), contains the output points.

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, D, PHI. When m(j) denotes the number of output points on the interval [TBP(j − 1) , TBP(j)], j = 2, . . . , NBP, and m the number of output points on the interval [TBP(1) , TBP(NBP)], i.e. m = m(2) + · · · + m(NBP), then
NTI ≥ m + 1 + max(4 − m(NBP) , 0).
If the routine was called with NRTI(1) > 1 and ALI ≤ 1 then m(j) = NRTI(j) + 1 , j = 2, . . . , NBP, so
NTI ≥ NBP + NRTI(2) + · · · + NRTI(NBP) + max(3 − NRTI(NBP),0).
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N,NTI).
On exit X(i,k) , i = 1,2, . . . , N contains the solution of the BVP at the output point TI(k), k = 1, . . . , NRTI(1).

U DOUBLE PRECISION array of dimension (NU,NTI).
On exit U(i , k) i = 1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NRTI(1) The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k), . . . , U(nj + j, k), where nj = (j − 1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to N * (N + 1) / 2.
Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N,N,NTI).
On exit Q(i , j, k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal matrix Q_k, k = 1, . . . , NRTI(1).

D DOUBLE PRECISION array of dimension (N,NTI).
If IHOM = 0 the array D has no real use and the user is recommended to use the same array for the X and the D.
If IHOM = 1 : on exit D(i, k) i = 1,2, . . . , N contains the inhomogeneous term d_k, k = 1,2, . . . , NRTI(1), of the multiple shooting recursion.

KPART INTEGER array of dimension (k), k ≥ NBP
On exit KPART(j) contains the global partitioning parameter on the interval [TBP(i_j) , TBP(i_{j+1})], j = 1, . . . , where the TBP(i_j) are the points where a change of dichotomy has been detected; $i_1 < i_2 < \cdots$ and NRTI(i_j) > 0.

PHI DOUBLE PRECISION array of dimension (NU,NTI).
On exit PHI contains a fundamental solution of the multiple shooting recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
LW ≥ N * (3 * N * N + 14 * N +15) / 2 + NSP * N * (3 * N + 5) / 2
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ (4 + NBP)*N + 4 * NBP .
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, CAMPF, CDI, CFUNRC, CKLREC, CNRHS, COPMAT, COPVEC, CONDW, CPSRC, CTIMI, CTIPL, CROUT, CUVRC, CWISB, DEFINC, DUR, FCBVP, FC2BVP, FQUS, FUNPAR, FUNRC, GKPMP, GTUR, GTUVRC, INPRO, INTCH, KPCH, LUDEC, MATVC, MTSDD, PSR, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SMBVP, SOLDE, SOLUPP, SORTD, SORTD0, SPLS2, SSDBVP, TAMVC, TUR, TUVRC, UPUP, UPVECP, UQDEC.

****
Remarks
****

MUTSMP is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****
Method
****

See chapter IV.

****
Example of the use of MUTSDD
****

Consider the ordinary differential equation

$$\frac{d}{dt}x(t) = L_i(t)x(t) + r_i(t)\ , \quad \begin{matrix} -3 \le t < 0\ , i = 1 \\ 0 \le t \le 3\ , i = 2 \end{matrix}\ ,$$

a jump condition at $t = 0$:

$$Z_2^- x(0^-) + Z_2^+ x(0^+) = b_2$$

and a boundary condition:

$$M_1 x(-3) + M_2 x(0^+) + M_3 x(3) = b\ ,$$

$$L_1(t) = \begin{bmatrix} 1\tfrac{1}{2} + \tfrac{1}{2}\cos(2t) & 1 - \tfrac{1}{2}\sin(2t) & 0 \\ -1 - \tfrac{1}{2}\sin(2t) & 1\tfrac{1}{2} - \tfrac{1}{2}\cos(2t) & 0 \\ 0 & 0 & -1 \end{bmatrix},\quad r_1(t) = \begin{bmatrix} -2\tfrac{1}{2} - \cos(2t) + \sin(2t) \\ -\tfrac{1}{2} + \cos(2t) + \sin(2t) \\ 1 \end{bmatrix},$$

$$L_2(t) = \begin{bmatrix} \tfrac{1}{2} + 3\cos(2t) & 1 - 3\sin(2t) & 0 \\ -1 - 3\sin(2t) & \tfrac{1}{2} - 3\cos(2t) & 0 \\ 0 & 0 & -1 \end{bmatrix},\quad r_2(t) = \begin{bmatrix} -1\tfrac{1}{2} - 3\,(\cos(2t) - \sin(2t)) \\ \tfrac{1}{2} + 3\,(\cos(2t) + \sin(2t)) \\ 1 \end{bmatrix},$$

$Z_2^- = I$, $Z_2^+ = -I$, $b_2 = (1, -2, 0)^T$,

$$M_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ M_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ M_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ b = \begin{bmatrix} 1 + \sin(3)\,e^{-3} \\ 2 \\ 1 \end{bmatrix}.$$

The solution of this problem is:

$$x(t) = (\,1 + \cos(t)\,e^{2t} - \sin(t)\,e^{t}\ ,\, 1 - \sin(t)\,e^{2t} - \cos(t)\,e^{t}\ ,\, 1)^T, -3 \le t < 0$$
$$x(t) = (\,1 + \sin(t)\,e^{-t}\ ,\, 1 + \cos(t)\,e^{-t}\ ,\, 1)^T, 0 \le t \le 3.$$

For $t < 0$ the ODE has fundamental solutions growing like $exp(2t)$, $exp(t)$ and $exp(-t)$; for $t \ge 0$ the ODE has fundamental solutions growing like $exp(2t)$ and $ep(-t)$, so there is a change of dichotomy at $t = 0$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION TSP(3),BCM(3,3,3),BCV(3),ZM(3,3,3),ZP(3,3,3),BI(3,3),
     1  ER(6),TI(13),X(3,13),U(6,13),Q(3,3,13),D(3,13),PHI(6,13),W(189)
      INTEGER IHOM(3),KP(3),NRTI(3),IW(33)
      EXTERNAL FLIN,FINH
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 3
      NSP = 3
      IHOM(1) = 1
      IHOM(2) = 1
      TSP(1) = -3.D0
      TSP(2) = 0.D0
      TSP(3) = 3.D0
      ER(1) = 1.D-11
      ER(2) = 1.D-6
      CALL EPSMAC(ER(3))
      ALI = 0.D0
      NTI = 13
      NU = 6
      LW = 189
      LIW = 33
      NRTI(1) = 2
      NRTI(2) = 5
      NRTI(3) = 5
C
C     SETTING THE BC MATRICES BCM, THE BC VECTOR BCV AND THE SIDE
C     CONDITION MATRICES ZM, ZP AND VECTOR BI.
```

```
C
      DO 1200 L = 1 , NSP
       DO 1100 I = 1 , N
       DO 1100 J = 1 , N
        IF (I.EQ.J) THEN
         ZM(I,I,L) = 1.D0
         ZP(I,I,L) = -1.D0
        ELSE
         ZM(I,J,L) = 0.D0
         ZP(I,J,L) = 0.D0
        ENDIF
        BCM(I,J,L) = 0.D0
1100   CONTINUE
       BI(1,L) = 1.D0
       BI(2,L) = -2.D0
       BI(3,L) = 0.D0
1200  CONTINUE
      BCM(3,3,1) = 1.D0
      BCM(2,2,2) = 1.D0
      BCM(1,1,3) = 1.D0
      BCV(1) = 1.D0 + DSIN(TSP(3)) * DEXP(-TSP(3))
      BCV(2) = 2.D0
      BCV(3) = 1.D0
C
C     CALL MUTSDD
C
      CALL MUTSDD(FLIN,FINH,N,IHOM,TSP,NSP,BCM,BCV,ZM,ZP,BI,ALI,ER,
     1      NRTI,TI,NTI,X,U,NU,Q,D,KP,PHI,W,LW,IW,LIW,IERROR)
      IF ((IERROR.NE.0).AND.(IERROR.NE.200).AND.(IERROR.NE.213).AND.
     1 (IERROR.NE.300)) THEN
        WRITE(*,300) IERROR
        STOP
      ENDIF
      CALL OUTPUT(N,ER,TI,X,NTI,NRTI,NSP)
      STOP
  300FORMAT(' TERMINAL ERROR IN MUTSDD : IERROR = ',I3)
      END
C
      SUBROUTINE FLIN(N,T,FL)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION FL(N,N)
C
      T2 = 2.D0 * T
      C = DCOS(T2) / 2.D0
      S = DSIN(T2) / 2.D0
```

```
      IF (T.LT.0.D0) THEN
       FL(1,1) = 1.5D0 +  C
       FL(1,2) = 1.D0 - S
       FL(1,3) = 0.D0
       FL(2,1) = -1.D0 - S
       FL(2,2) = 1.5D0 - C
       FL(2,3) = 0.D0
       FL(3,1) = 0.D0
       FL(3,2) = 0.D0
       FL(3,3) = -1.D0
      ELSE
       FL(1,1) = 0.5D0 + 3.D0*C
       FL(1,2) = 1.D0 - 3.D0*S
       FL(1,3) = 0.D0
       FL(2,1) = -1.D0 - 3.D0*S
       FL(2,2) = 0.5D0 - 3.D0*C
       FL(2,3) = 0.D0
       FL(3,1) = 0.D0
       FL(3,2) = 0.D0
       FL(3,3) = -1.D0
      ENDIF
      RETURN
C     END OF FLIN
      END
      SUBROUTINE FINH(N,T,FR)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION FR(N)
C
      T2 = 2.D0*T
      C = DCOS(T2) / 2.D0
      S = DSIN(T2) / 2.D0
      IF (T.LT.0.D0) THEN
       FR(1) = -2.5D0 - C + S
       FR(2) = -0.5D0 +C + S
       FR(3) = 1.D0
      ELSE
       FR(1) = -1.5D0  - 3.D0*(C - S)
       FR(2) = 0.5D0 + 3.D0*(C + S)
       FR(3) = 1.D0
      ENDIF
      RETURN
C     END OF FINH
      END
      SUBROUTINE OUTPUT(N,ER,TI,X,NTI,NRTI,NSP)
C
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION TI(NTI),X(N,NTI),ER(6)
      INTEGER NRTI(NSP)
C
C     PRINTING OF THE CONDITION NUMBER AND THE AMPLIFICATION FACTOR.
C
      WRITE(NOUT,200)
      WRITE(NOUT,245) ER(4),ER(5),ER(6)
      WRITE(NOUT,200)
C
C     COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND WRITING
C     OF THE SOLUTION AT THE OUTPUTPOINTS
C
      WRITE(NOUT,200)
      WRITE(NOUT,250)
      WRITE(NOUT,200)
      DO 2100 K = 1 , NRTI(1)
        C = DCOS(TI(K))
        S = DSIN(TI(K))
        E2T = DEXP(2.D0*TI(K))
        ET = DEXP(TI(K))
        EMT = DEXP(-TI(K))
        IF (TI(K).LT.0.D0) THEN
          EXSOL1 = 1.D0 + C*E2T - S*ET
          EXSOL2 = 1.D0 - S * E2T - C * ET
        ELSE
          EXSOL1 = 1.D0 + S * EMT
          EXSOL2 = 1.D0 + C * EMT
        ENDIF
        AE = EXSOL1 - X(1,K)
        WRITE(NOUT,260) K,TI(K),X(1,K),EXSOL1,AE
        AE = EXSOL2 - X(2,K)
        WRITE(NOUT,270) X(2,K),EXSOL2,AE
        EXSOL3 = 1.D0
        AE = EXSOL3 - X(3,K)
        WRITE(NOUT,270) X(3,K),EXSOL3,AE
 2100 CONTINUE
      RETURN
 200  FORMAT(' ')
 245  FORMAT(' CONDITION NUMBER      = ',D10.3,/,
     1       ' AMPLIFICATION FACTORS = ',D10.3,3X,D10.3)
 250  FORMAT('  I ',6X,'T',8X,'APPROX. SOL.',9X,'EXACT SOL.',8X,
     1  'ABS. ERROR')
 260  FORMAT(' ',I3,3X,F7.4,3(3X,D16.9))
 270  FORMAT(' ',13X,3(3X,D16.9))
      RETURN
```

END

CONDITION NUMBER = .101D+01
AMPLIFICATION FACTORS = .216D+01 .200D+01

I	T	APPROX. SOL.	EXACT SOL.	ABS. ERROR
1	-3.0000	.100457200D+01	.100457201D+01	.606644157D-08
		.104963862D+01	.104963863D+01	.382206178D-08
		.100000000D+01	.100000000D+01	.000000000D+00
2	-2.4000	.105520807D+01	.105520807D+01	.757860996D-08
		.107245374D+01	.107245374D+01	.308154124D-08
		.100000000D+01	.100000000D+01	.000000000D+00
3	-1.8000	.115476792D+01	.115476792D+01	.812305756D-08
		.106416538D+01	.106416540D+01	.111951213D-07
		.100000000D+01	.100000000D+01	.000000000D+00
4	-1.2000	.131359711D+01	.131359713D+01	.238768358D-07
		.975412599D+00	.975412620D+00	.208038813D-07
		.100000000D+01	.100000000D+01	.222044605D-15
5	-.6000	.155846863D+01	.155846867D+01	.369013986D-07
		.717113253D+00	.717113256D+00	.262743050D-08
		.100000000D+01	.100000000D+01	.222044605D-15
6	.0000	.200000034D+01	.200000000D+01	-.339824049D-06
		.144075222D-15	.221020034D-14	.206612512D-14
		.100000000D+01	.100000000D+01	.111022302D-15
7	.0000	.100000034D+01	.100000000D+01	-.339824046D-06
		.200000000D+01	.200000000D+01	-.111022302D-14
		.100000000D+01	.100000000D+01	.111022302D-15
8	.6000	.130988245D+01	.130988236D+01	-.883196336D-07
		.145295373D+01	.145295379D+01	.632408355D-07
		.100000000D+01	.100000000D+01	.000000000D+00
9	1.2000	.128072479D+01	.128072478D+01	-.757742802D-08
		.110914003D+01	.110914006D+01	.288902136D-07
		.100000000D+01	.100000000D+01	.444089210D-15
10	1.8000	.116097592D+01	.116097593D+01	.604394401D-08
		.962443732D+00	.962443746D+00	.143329353D-07
		.100000000D+01	.100000000D+01	.111022302D-15
11	2.4000	.106127663D+01	.106127664D+01	.711346293D-08
		.933105147D+00	.933105151D+00	.400970723D-08
		.100000000D+01	.100000000D+01	.111022302D-15
12	3.0000	.100702595D+01	.100702595D+01	.000000000D+00
		.950711177D+00	.950711176D+00	-.126165733D-08
		.100000000D+01	.100000000D+01	.000000000D+00

10. Subroutine MUTSEI

****
SPECIFICATION
****

```
      SUBROUTINE MUTSEI(FLINE, N, A, B, EIG, MA, MB, ALI, ER, NRTI, TI,
     1    NTI, X, NRSOL, U, NU, Q, KPART, PHI, W, LW, IW, LIW, IERROR)
C     INTEGER N, NRTI, NTI, NRSOL, NU, LW, IW(LIW), LIW, IERROR
C     DOUBLE PRECISION A, B, EIG(2), MA(N,N), MB(N,N), ALI, ER(5), TI(NTI),
C    1    X(N,NTI,N), U(NU,NTI), Q(N,N,NTI), PHI(NU,NTI), W(LW)
C     EXTERNAL FLIN
```

****
Purpose
****

MUTSEI solves the eigenvalue problem:

$$\frac{d}{dt}x(t,\lambda)=L(t,\lambda)x(t,\lambda)\,, \quad A\le t\le B \text{ or } B\le t\le A\,,$$

with BC:

$$M_A\,x(A,\lambda)+M_B\,x(B,\lambda)=0\,,$$

where λ is the parameter, $L(t,\lambda)$ an N×N matrix function, M_A and M_B are N×N matrices.

****
Parameters
****

FLINE SUBROUTINE, supplied by the user with specification:

```
SUBROUTINE FLINE(N, T, FL, ALAM)
DOUBLE PRECISION T, FL(N,N), ALAM
```

where N is the order of the system. FLINE must evaluate the matrix $L(t,\lambda)$ of the differential equation for t = T, λ = ALAM and place the result in the array FL(N,N).
FLINE must be declared as EXTERNAL in the (sub)program from which MUTSGE is called.

N INTEGER, the order of the system.
Unchanged on exit.

A,B DOUBLE PRECISION, the two boundary points.
Unchanged on exit.

EIG DOUBLE PRECISION array of dimension (2)
On entry EIG(1) and EIG(2) must contain the endpoints of an interval in which the required eigenvalue lies.
On exit EIG(1) and EIG(2) contains the endpoints of the interval in which an eigenvalue is found, where | EIG(1) - EIG(2) | < ER(2) + EIG(1) * ER(1).
EIG(1) is taken as an approximate for the eigenvalue.

MA,MB DOUBLE PRECISION array of dimension (N, N).
On entry : MA and MB must contain the matrices in the BC:
$M_A\, x(A, \lambda) + M_B\, x(B, \lambda) = 0.$
Unchanged on exit.

ALI DOUBLE PRECISION.
On entry ALI must contain the allowed incremental factor of the homogeneous solutions between two successive output points. If the increment of a homogeneous solution between two successive output points becomes greater than 2 * ALI, a new output point is inserted.
If ALI ≤ 1 the defaults are:
If NRTI = 0 : ALI := max(ER(1), ER(2)) / (2*ER(3)),
if NRTI ≠ 0 : ALI := SQRT(RMAX), where RMAX is the largest positive real number which can be represented on the computer used.
On exit ALI contains the actually used incremental factor.

ER DOUBLE PRECISION array of dimension (5).
On entry ER(1) must contain a relative tolerance for solving the differential equation and computing the eigenvalue. If the relative tolerance is smaller then 1.0 d-12 the subroutine will change ER(1) into
ER(1) := 1.d-12 + 2 * ER(3).
On entry ER(2) must contain an absolute tolerance for solving the differential equation and computing the eigenvalue.
On entry ER(3) must contain the machine constant EPS (see Remark 1.1).
On exit ER(2) and ER(3) are unchanged.
On exit ER(4) contains an estimate of the condition number of the BVP. If on exit ER(4) = -1, then NRSOL = N.
On exit ER(5) contains an estimate of the amplification factor.

NRTI INTEGER.
On entry NRTI is used to specify the required output points. There are three ways to specify the required output points:
1) NRTI = 0, the subroutine automatically determines the output points using the allowed incremental factor ALI.

2) NRTI = 1, the output points are supplied by the user in the array TI.
3) NRTI > 1, the subroutine computes the (NRTI+1) output points TI(k) by:

TI(k) = A + (k−1) * (B − A) / NRTI ;

so TI(1) = A and TI(NRTI+1) = B .

Depending on the allowed incremental factor ALI, more output points may be inserted in the cases 2 and 3. On exit NRTI contains the total number of output points.

TI DOUBLE PRECISION array of dimension (NTI).
On entry: if NRTI = 1 , TI must contain the required output points in strict monotone order: A = TI(1) < ··· < TI(k) = B or A = TI(1) > ··· > TI(k) = B
(k denotes the total number of required output points).
On exit: TI(i), i = 1,2, . . . , NRTI, contains the output points.

NTI INTEGER.
NTI is the dimension of TI and one of the dimensions of the arrays X, U, Q, D, PHI. Let m be the total number of output points then NTI ≥ max(5, m + 1).
If the routine was called with NRTI > 1 and ALI ≤ 1 the total number of required output points is NRTI + 1, so NTI ≥ max(5, NRTI + 2).
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NTI, N).
On exit X(i,k,l) , i = 1,2, . . . , N, l = 1 , . . . , NRSOL, contains the eigensolutions, at the output points TI(k), k = 1, . . . , NRTI, corresponding with the computed eigenvalue EIG(1).

NRSOL INTEGER.
On exit NRSOL contains the number of independent eigensolutions.

U DOUBLE PRECISION array of dimension (NU,NTI).
On exit U(i,k) i = 1,2, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NRTI. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k) , . . . , U(nj + j, k) where nj = (j−1) * j / 2.

NU INTEGER.
NU is one of the dimensions of U and PHI.
NU must be at least equal to N * (N+1) / 2.
Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N, N, NTI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal matrix Q_k, k = 1, . . . , NRTI.

KPART INTEGER.
On exit KPART contains the global k-partition of the upper triangular matrices U_k.

PHI DOUBLE PRECISION array of dimension (NU, NTI).
On exit PHI contains a fundamental solution of the multiple shooting recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W. LW ≥ 8*N + 7*N*N.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ 4*N.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, BCMAV, CDI, CNRHS, COPMAT, COPVEC, CONDW, CRHOL, CROUT, CWISB, DEFINC, DUR, FCBVP, FCEBVP, FQUS, FUNPAR, FUNRC, INPRO, INTCH, KPCH, LUDEC, MATVC, MTSE, QEVAK, QEVAL, QUDEC, RKF1S, RKFSM, SOLDE, SOLUPP, SORTD, TAMVC, UPUP, UPVECP.

****
Remarks
****

MUTSEI is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****

Method

****

See chapter VII.

****

Example of the use of MUTSEI

****

Consider the ordinary differential equation

$$\frac{d}{dt}x(t,\lambda)=\begin{bmatrix}0 & 1\\ -\lambda & 0\end{bmatrix}x(t,\lambda), \quad 0\le t\le 1$$

and a boundary condition $x(0)=0$ and $x(1)=0$.

This problem has an eigenvalue $\lambda_e=\pi^2$ and an eigensolution $x(t,\lambda_e)=(\frac{\sin(\pi t)}{\pi}, \cos(\pi))^T$.

In the next program this eigenvalue and eigensolution is computed, starting with an initial interval for λ : [9 , 11].
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION EIG(2),BMA(2,2),BMB(2,2),ER(5),TI(12),X(2,12,2),
     1          U(3,12),Q(2,2,12),PHI(3,12),W(44)
      INTEGER IW(8)
      EXTERNAL FLINE
C
C     SET INPUT PARAMETERS
C
      N = 2
      NU = 3
      NTI = 12
      NRTI = 10
      LW = 44
      LIW = 8
      A = 0.D0
      B = 1.D0
      AMP = 0.D0
      ER(1) = 1.1D-12
      ER(2) = 1.0D-6
      CALL EPSMAC(ER(3))
      DO 1100 I = 1 , N
```

```
         DO 1100 J = 1 , N
          BMA(I,J) = 0.D0
          BMB(I,J) = 0.D0
1100     CONTINUE
         BMA(1,1) = 1.D0
         BMB(2,1) = 1.D0
         EIG(1) = 9.D0
         EIG(2) = 11.0D0
C
C        CALL MUTSEI
C
         CALL MUTSEI(FLINE,N,A,B,EIG,BMA,BMB,AMP,ER,NRTI,TI,NTI,
     1          X,NRSOL,U,NU,Q,KPART,PHI,W,LW,IW,LIW,IERROR)
         IF ((IERROR.NE.0).AND.(IERROR.NE.200).AND.(IERROR.NE.213).AND.
     1    (IERROR.NE.300)) GOTO 5000
C
C        COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND WRITING
C        OF THE EIGENVALUE END EIGENSOLUTION
C
         WRITE(*,200) ER(4),ER(5)
         PI = 4.D0 * DATAN(1.D0)
         EXLAM = PI * PI
         ERR = EXLAM - EIG(1)
         WRITE(*,210) EXLAM,EIG(1),ERR
         WRITE(*,220)
         DO 1500 K = 1 , NRTI
          T = PI * TI(K)
          XEX = DSIN(T) / PI
          ERR = XEX - X(1,K,1)
          WRITE(*,230) K,TI(K),X(1,K,1),XEX,ERR
          XEX = DCOS(T)
          ERR = XEX - X(2,K,1)
          WRITE(*,240) X(2,K,1),XEX,ERR
      1500CONTINUE
         STOP
C
200      FORMAT(' CONDITION NUMBER     = ',D12.5,/,
     1    ' AMPLIFICATION FACTOR = ',D12.5,/)
210      FORMAT(' EXACT LAMBDA = ',D20.13,/,' COMP. LAMBDA = ',D20.13,/,
     1    ' ERROR        = ',D20.13,/)
220      FORMAT(' ',/,9X,'T',6X,'APPROX. EIGENSOL.'3X,'EXACT EIGENSOL.',
     1    8X,'ERROR',/)
230      FORMAT(' ',I2,2X,F8.5,3X,3(D16.9,3X))
240      FORMAT(' ',15X,3(D16.9,3X))
300      FORMAT(' TERMINAL ERROR IN MUTSEI: IERROR = ',I4)
         STOP
```

```
5000    WRITE(*,300) IERROR
        END
C
        SUBROUTINE FLINE(N,T,FL,PARM)
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        DIMENSION FL(N,N)
C
        FL(1,1) = 0.D0
        FL(1,2) = 1.D0
        FL(2,1) = -PARM
        FL(2,2) = 0.D0
        RETURN
        END
```

```
CONDITION NUMBER        = .70711D+00
AMPLIFICATION FACTOR    = .23117D+02

EXACT LAMBDA   = .9869604401089D+01
COMP. LAMBDA   = .9869604559034D+01
ERROR          = -.1579442141519D-06
```

	T	APPROX. EIGENSOL.	EXACT EIGENSOL.	ERROR
1	.00000	-.222615390D-10	.000000000D+00	.222615390D-10
		.100000000D+01	.100000000D+01	.000000000D+00
2	.10000	.983631646D-01	.983631643D-01	-.291091706D-09
		.951056520D+00	.951056516D+00	-.394011335D-08
3	.20000	.187097863D+00	.187097857D+00	-.595535582D-08
		.809017027D+00	.809016994D+00	-.327274168D-07
4	.30000	.257518123D+00	.257518107D+00	-.152225198D-07
		.587785293D+00	.587785252D+00	-.409873473D-07
5	.40000	.302730718D+00	.302730691D+00	-.261617198D-07
		.309017026D+00	.309016994D+00	-.313559396D-07
6	.50000	.318309923D+00	.318309886D+00	-.363406942D-07
		.289859515D-08	.612574227D-16	-.289859508D-08
7	.60000	.302730735D+00	.302730691D+00	-.431250540D-07
		-.309017037D+00	-.309016994D+00	.425759964D-07
8	.70000	.257518145D+00	.257518107D+00	-.379209960D-07
		-.587785342D+00	-.587785252D+00	.892135977D-07
9	.80000	.187097885D+00	.187097857D+00	-.281338394D-07
		-.809017123D+00	-.809016994D+00	.128821307D-06
10	.90000	.983631789D-01	.983631643D-01	-.146072169D-07
		-.951056673D+00	-.951056516D+00	.156757917D-06
11	1.00000	-.222615346D-10	.389976865D-16	.222615736D-10

-.100000020D+01	-.100000000D+01	.199901559D-06

11. Subroutine SPLS1

****
SPECIFICATION
****

```
      SUBROUTINE SPLS1(N, IHOM, A, B, G, NRI, M1, MN, BCV, NREC, X, Q,
     1     U, V, NU, PHI, D, KP, EPS, COND, AF, W, LW, IW, LIW, IERROR)
C     INTEGER N, IHOM, NRI, NREC, NU, KP, LW, IW(LIW), LIW, IERROR
C     DOUBLE PRECISION A(N,N,NRI), B(N,N,NRI), G(N,NRI), M1(N,N),
C    1     MN(N,N), BCV(N), X(N,NRI), Q(N,N,NRI), U(NU,NRI),
C    2     V(NU,NRI), PHI(NU,NRI), D(N,NRI), EPS, COND, AF, W(LW)
```

****
Purpose
****

SPLS1 solves the discrete two-point BVP:

$$A_i\, x_i + B_{i+1}\, x_{i+1} = g_{i+1}\ , \quad i = 1, \ldots, NREC - 1.$$

with BC:

$$M_1 x_1 + M_{NREC}\, x_{NREC} = BCV$$

where A_i, B_{i+1}, M_1, M_{NREC} are N×N matrices, x_i, g_{i+1} and BCV are N-vectors.

****
Parameters
****

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

A DOUBLE PRECISION array of dimension (N, N, NRI).
On entry A(. , . , i) must containt the matrix A_i, i = 1, . . . , NREC−1,
Unchanged on exit.

B DOUBLE PRECISION array of dimension (N, N, NRI).
On entry B(. , . , i) must contain the matrix B_i, i = 2, . . . , NREC,

On exit: if in the call to SPLS1 the same array is used for B and Q, B will contain the Qs; otherwise B is unchanged.

G DOUBLE PRECISION array of dimension (N, NRI).
If IHOM = 0, the array G has no real use and the user is recommended to use the same array for the X and the G.
If IHOM = 1, then on entry G(. , i) must contain the vector g_i, i = 2, . . . , NREC.
On exit: if in the call to SPLS1 the same array is used for the G and D, the G will contain the values for the D; otherwise the G is unchanged.

NRI INTEGER.
NRI is one of the dimension of A, B, G, X, Q, U, V, PHI and D. NRI ≥ NREC + 1.
Unchanged on exit.

M1,MN DOUBLE PRECISION array of dimension (N, N).
On entry : M1 must contain the matrix M_1 and MN must contain the matrix M_{NREC} of the BC:
$M_1 x_1 + M_{NREC}\, x_{NREC} = BCV$.
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector.
Unchanged on exit.

NREC INTEGER.
On entry NREC must contain the total number of the x_i of the recursion.
Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NRI).
On exit X(i,k) , i = 1, . . . , N contains the solution x_k, k=1, . . . , NREC.

Q DOUBLE PRECISION array of dimension (N, N, NRI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal transformation matrix Q_k, k = 1, . . . , NREC.

U DOUBLE PRECISION array of dimension (NU, NRI).
On exit U(i,k) i = 1, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NREC, of the transformed upper triangular recursion. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k) , . . . , U(nj + j, k) where nj = (j − 1) * j / 2.

V DOUBLE PRECISION array of dimension (NU, NRI).
On eit V(i,k) i = 1, . . . , NU contains the relevant elements of the upper triangular matrix V_k, k = 1, . . . , NREC, of the transformed upper triangular recursion. The elements are stored in the same way as the U_k.

NU INTEGER.
NU is one of the dimensions of U, V and PHI.
NU must be at least equal to N * (N + 1) / 2.
Unchanged on exit.

PHI DOUBLE PRECISION array of dimension (NU, NRI).
On exit PHI contains a fundamental solution of the transformed upper triangular recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

D DOUBLE PRECISION array of dimension (N, NRI).
If IHOM = 0 the array D has no real use and the user is recommended to use the same array for the X and the D.
If IHOM = 1 : on exit D(i,k) i = 1,, . . . , N contains the inhomogeneous term d_k, k = 2, . . . , NREC, of the transformed recursion.
It is possible to use the same array for the G and D in the call to SPLS1. If this is the case, this array will contain the values of the D on exit.

KP INTEGER.
On exit KP contains the global k-partition of the transformed upper triangular recursion.

EPS DOUBLE PRECISION.
On entry EPS must contain the machine constant EPS (see Remark 1.1).
Unchanged on exit.

COND DOUBLE PRECISION.
On exit COND contains an estimate of the condition number.

AF DOUBLE PRECISION.
On exit AF contains an estimate of the amplification factor.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
LW ≥ 3 * N + 2 * N * N.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ 4 * N.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, BCMAV, CAMPF, CFUNRC, COPMAT, COPVEC, CONDW, CPSRC, CROUT, CUVRC, FQUS, GTUVRC, INPRO, INTCH, LUDEC, MATVC, QEVAK, QEVAL, QUDEC, SBVP, SOLDE, SOLUPP, SORTD, SORTD0, TAMVC, TUVRC, UPUP, UPVECP.

****
Remarks
****

SPLS1 is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****
Method
****

See chapter VIII.

****
Example of the use of SPLS1
****

Consider the recursion:

$$A_i x_i + B_{i+1} x_{i+1} = g_{i+1}, \quad i = 1, \ldots, 10,$$

with BC:

$$M_1 x_1 + M_{11} x_{11} = b,$$

where

$$A_i = \begin{bmatrix} 1 & -6 & 6 \\ -4 & 2 & -10 \\ -2 & 7 & -12 \end{bmatrix}, \quad B_{i+1} = \begin{bmatrix} -2 & 7 & -3 \\ 8 & 3 & 5 \\ 4 & 1 & 6 \end{bmatrix},$$

$$g_{i+1} = (-2, 19, 24)^T$$

$$M_1 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad M_{11} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$b = (-2\,,\,3+2^{-10}\,,\,2)^T.$$

The solution of this problem is: $x(i) = (\,1+2^{1-i}\;,2\;,-1-2^{i-11}\,)^T$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(3,3,12),B(3,3,12),G(3,12),BM1(3,3),BMN(3,3),BCV(3),
     1     X(3,12),U(6,12),V(6,12),PHI(6,12),W(27)
      INTEGER IW(12),IB(12)
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 3
      IHOM = 1
      NU = 6
      NRI = 12
      LW = 27
      LIW = 12
      NREC = 11
      CALL EPSMAC(EPS)
C
C     SETTING OF THE RECURSION AND BC
C
      DO 1100 I = 1 , NREC-1
      A(1,1,I) =  1.D0
      A(1,2,I) = -6.D0
      A(1,3,I) =  6.D0
      A(2,1,I) = -4.D0
      A(2,2,I) =  2.D0
      A(2,3,I) = -10.D0
      A(3,1,I) = -2.D0
      A(3,2,I) =  7.D0
      A(3,3,I) = -12.D0
 1100 CONTINUE
      DO 1200 I = 2 , NREC
        B(1,1,I) = -2.D0
        B(1,2,I) =  7.D0
        B(1,3,I) = -3.D0
        B(2,1,I) =  8.D0
        B(2,2,I) =  3.D0
        B(2,3,I) =  5.D0
```

```
         B(3,1,I) =  4.D0
         B(3,2,I) =  1.D0
         B(3,3,I) =  6.D0
         G(1,I) = -2.D0
         G(2,I) = 19.D0
         G(3,I) = 24.D0
1200    CONTINUE
        DO 1300 I = 1 , N
        DO 1300 J = 1 , N
         BM1(I,J) = 0.D0
         BMN(I,J) = 0.D0
1300    CONTINUE
        BM1(2,1) = 1.D0
        BM1(3,2) = 1.D0
        BMN(1,3) = 1.D0
        BMN(2,1) = 1.D0
        BCV(1) = -2.D0
        BCV(2) =  3.D0 + 2.D0 ** (-10)
        BCV(3) =  2.D0
        IERROR = 0
C
C       CALL TO SPLS1
C
        CALL SPLS1(N,IHOM,A,B,G,NRI,BM1,BMN,BCV,NREC,X,B,U,V,NU,PHI,G,
     1       KP,EPS,COND,AF,W,LW,IW,LIW,IERROR)
        IF ((IERROR.NE.0).AND.(IERROR.NE.710)) GOTO 3000
C
C       WRITING OF THE SOLUTION AND THE ABSOLUTE ERROR
C
        CALL OUTSOL(COND,AF,KP,X,N,NRI,NREC)
        STOP
3000    WRITE(*,100) IERROR
        STOP
C
100     FORMAT(' TERMINAL ERRROR IN SPLS1 : IERROR = ',I4,/)
        END
C
        SUBROUTINE OUTSOL(COND,AF,KP,X,N,NRI,NREC)
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        DIMENSION X(N,NRI)
C
        WRITE(IO,200) COND,AF,KP
        WRITE(IO,100)
        DO 1100 I = 1 , NREC
         I1 = 1 - I
```

```
      N1 = I - NREC
      S1 = 1.D0 + 2.D0 ** I1
      S2 = 2.D0
      S3 = -1.D0 - 2.D0 ** N1
      WRITE(IO,110) I,X(1,I),S1,S1-X(1,I)
      WRITE(IO,120) X(2,I),S2,S2-X(2,I)
      WRITE(IO,120) X(3,I),S3,S3-X(3,I)
1100  CONTINUE
C
100   FORMAT(' ',/,' I',7X,'X APPROX',11X,'X EXACT',14X,'ERROR',/)
110   FORMAT(' ',I2,3X,3(D16.9,3X))
120   FORMAT(' ',5X,3(D16.9,3X))
200   FORMAT(' ',/,' CONDITION NUMBER    = ',D12.5,/,
     1  ' AMPLIFICATION FACTOR = ',D12.5,/)
      RETURN
      END
```

```
CONDITION NUMBER       =  .10038D+01
AMPLIFICATION FACTOR   =  .31591D+01
```

I	X APPROX	X EXACT	ERROR
1	.200000000D+01	.200000000D+01	-.888178420D-15
	.200000000D+01	.200000000D+01	.000000000D+00
	-.100097656D+01	-.100097656D+01	.222044605D-15
2	.150000000D+01	.150000000D+01	.222044605D-15
	.200000000D+01	.200000000D+01	-.133226763D-14
	-.100195313D+01	-.100195313D+01	-.444089210D-15
3	.125000000D+01	.125000000D+01	-.222044605D-15
	.200000000D+01	.200000000D+01	.222044605D-15
	-.100390625D+01	-.100390625D+01	-.444089210D-15
4	.112500000D+01	.112500000D+01	-.133226763D-14
	.200000000D+01	.200000000D+01	.666133815D-15
	-.100781250D+01	-.100781250D+01	.222044605D-15
5	.106250000D+01	.106250000D+01	-.666133815D-15
	.200000000D+01	.200000000D+01	.444089210D-15
	-.101562500D+01	-.101562500D+01	.222044605D-15
6	.103125000D+01	.103125000D+01	-.222044605D-15
	.200000000D+01	.200000000D+01	-.444089210D-15
	-.103125000D+01	-.103125000D+01	-.222044605D-15
7	.101562500D+01	.101562500D+01	.000000000D+00

	.200000000D+01	.200000000D+01	-.444089210D-15
	-.106250000D+01	-.106250000D+01	-.222044605D-15
8	.100781250D+01	.100781250D+01	-.177635684D-14
	.200000000D+01	.200000000D+01	.222044605D-15
	-.112500000D+01	-.112500000D+01	.155431223D-14
9	.100390625D+01	.100390625D+01	-.133226763D-14
	.200000000D+01	.200000000D+01	.888178420D-15
	-.125000000D+01	-.125000000D+01	.133226763D-14
10	.100195312D+01	.100195313D+01	.666133815D-15
	.200000000D+01	.200000000D+01	.000000000D+00
	-.150000000D+01	-.150000000D+01	.222044605D-15
11	.100097656D+01	.100097656D+01	.888178420D-15
	.200000000D+01	.200000000D+01	-.444089210D-15
	-.200000000D+01	-.200000000D+01	.000000000D+00

12. Subroutine SPLS2

****
SPECIFICATION
****

```
      SUBROUTINE SPLS2(N, IHOM, A, B, G, NRI, IJ, MI, KMI, BCV, NREC, X, Q,
     1       U, V, NU, PHI, D, KP, EPS, COND, AF, W, LW, IW, LIW, IERROR)
C     INTEGER N, IHOM, NRI, IJ(KMI), NREC(KMI), NU, KP(KMI),
C    1       LW, IW(LIW), LIW, IERROR
C     DOUBLE PRECISION A(N,N,NRI), B(N,N,NRI), G(N,NRI), MI(N,N,KMI),
C    1       BCV(N), X(N,NRI), Q(N,N,NRI), U(NU,NRI), V(NU,NRI),
C    2       PHI(NU,NRI), D(N,NRI), EPS, COND, AF, W(LW)
```

****
Purpose
****

SPLS2 solves the discrete two-point BVP:

$$A_i\, x_i + B_{i+1}\, x_{i+1} = g_{i+1} \ , \quad i = 1, \ldots , m-1.$$

with BC:

$$\sum_{j=1}^{k} M_j\, x_{i_j} = b$$

where A_i, B_{i+1}, M_j are N×N matrices, x_i, g_{i+1} and b are N-vectors and
$1 = i_1 < i_2 < \cdots < i_k = m$.
The subindices i_j are the so called "*switching points*"

****
Parameters
****

N INTEGER, the order of the system.
Unchanged on exit.

IHOM INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

A DOUBLE PRECISION array of dimension (N, N, NRI).
On entry A(. , . , i) must containt the matrix A_i, i = 1, . . . , $m-1$,
Unchanged on exit.

B DOUBLE PRECISION array of dimension (N, N, NRI).
On entry B(. , . , i) must contain the matrix B_i, i = 2, . . . , m.
On exit: if in the call to SPLS2 the same array is used for B and Q, B will contain the Qs; otherwise B is unchanged.

G DOUBLE PRECISION array of dimension (N, NRI).
If IHOM = 0, the array G has no real use and the user is recommended to use the same array for the X and the G.
If IHOM = 1, then on entry G(. , i) must contain the vector g_i, i = 2, . . . , m.
On exit: if in the call to SPLS2 the same array is used for the G and D, the G will contain the values for the D; otherwise the G is unchanged.

NRI INTEGER.
NRI is one of the dimension of A, B, G, X, Q, U, V, PHI and D. NRI $\geq m+1$.
Unchanged on exit.

IJ INTEGER array of dimension (KMI).
On entry IJ(j), j = 1, . . . , k must containt the subindex i_j of the x_{i_j} in the multipoint BC.
Unchanged on exit.

MI DOUBLE PRECISION array of dimension (N, N, KMI).
On entry : MI(. , . , j), j = 1, . . . , k must contain the matrix M_j of the multipoint BC.
Unchanged on exit.

KMI INTEGER.
KMI is one of the dimension of IJ, MI, NREC and KP.
On entry KMI must have the value of k, i.e. the total number of the BC matrices M_j.
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector b.
Unchanged on exit.

NREC INTEGER array of dimension (KMI).
On entry NREC(1) must contain the total number of the x_i of the recursion, i.e. NREC(1) = m.
On exit: NREC(1) is unchanged.
For j = 2, . . . , KMI: if NREC(j) < 0 then no change of dichotomy is detected in the recursion between the "switching points" IJ(j − 1) and IJ(j+1). If NREC(j) > 0 then a change of dichotomy is detected at IJ(j) and NREC(j) = IJ(j)−IJ(i)+1, where i < j,

NREC(i) > 0, NREC(l) < 0, i < l < j, i.e. IJ(i) is the previous "switching point" where a change of dichtomy was detected.

X DOUBLE PRECISION array of dimension (N, NRI).
On exit X(i,k) , i = 1, . . . , N contains the solution x_k, k=1, . . . , NREC(1).

Q DOUBLE PRECISION array of dimension (N, N, NRI).
On exit Q(i,j,k) i = 1,2, . . . , N, j = 1,2, . . . , N contains the N columns of the orthogonal transformation matrix Q_k, k = 1, . . . , NREC(1).

U DOUBLE PRECISION array of dimension (NU, NRI).
On exit U(i,k) i = 1, . . . , NU contains the relevant elements of the upper triangular matrix U_k, k = 2, . . . , NREC(1), of the transformed upper triangular recursion. The elements are stored column wise, the jth column of U_k is stored in U(nj + 1, k), U(nj + 2, k) , . . . , U(nj + j, k) where nj = (j − 1) * j / 2.

V DOUBLE PRECISION array of dimension (NU, NRI).
On eit V(i,k) i = 1, . . . , NU contains the relevant elements of the upper triangular matrix V_k, k = 1, . . . , NREC(1), of the transformed upper triangular recursion. The elements are stored in the same way as the U_k.

NU INTEGER.
NU is one of the dimensions of U, V and PHI.
NU must be at least equal to N * (N + 1) / 2.
Unchanged on exit.

PHI DOUBLE PRECISION array of dimension (NU, NRI).
On exit PHI contains a fundamental solution of the transformed upper triangular recursion. The fundamental solution is upper triangular and is stored in the same way as the U_k.

D DOUBLE PRECISION array of dimension (N, NTI).
If IHOM = 0 the array D has no real use and the user is recommended to use the same array for the X and the D.
If IHOM = 1 : on exit D(i,k) i = 1,, . . . , N contains the inhomogeneous term d_k, k = 2, . . . , NREC(1), of the transformed recursion.
It is possible to use the same array for the G and D in the call to SPLS2. If this is the case, this array will contain the values of the D on exit.

KP INTEGER.
On exit KP contains the global k-partition of the transformed upper triangular recursion.

EPS DOUBLE PRECISION.
On entry EPS must contain the machine constant EPS (see Remark 1.1).

Unchanged on exit.

COND DOUBLE PRECISION.
On exit COND contains an estimate of the condition number.

AF DOUBLE PRECISION.
On exit AF contains an estimate of the amplification factor.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
$LW \geq 3 * N + 2 * N * N$.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. $LIW \geq 4 * N + (N + 1) * KMI$.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See §14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, CAMPF, CFUNRC, COPMAT, COPVEC, CPSRC, CROUT, CUVRC, FQUS, GKPMP, GTUVRC, INPRO, INTCH, LUDEC, MATVC, QEVAK, QEVAL, QUDEC, SMBVP, SOLDE, SOLUPP, SORTD, SORTD0, TAMVC, TUVRC, UPUP, UPVECP.

****
Remarks
****

SPLS2 is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****

Method

****

See chapter VIII.

****

Example of the use of SPLS2

****

Consider the recursion:

$$A_i\, x_i + B_{i+1} x_{i+1} = g_{i+1} \quad i = 1, \ldots, 10.$$

and a multipoint boundary condition: $M_1 x_1 + M_2 x_6 + M_3 x_{11} = b$,
where

$$A_i = \begin{bmatrix} -\frac{1}{2} & 2 & 2 \\ -1\frac{1}{2} & 0 & 2 \\ 2 & 1 & 2 \end{bmatrix}, i = 1, \ldots, 5, \quad A_i = \begin{bmatrix} -\frac{1}{2} & 2 & \frac{1}{2} \\ -1\frac{1}{2} & 0 & \frac{1}{2} \\ 2 & 1 & \frac{1}{2} \end{bmatrix}, i = 6, \ldots, 10,$$

$$B_i = \begin{bmatrix} -1 & 1\frac{1}{2} & 1 \\ -5 & \frac{1}{2} & 1 \\ 8 & 1 & 1 \end{bmatrix}, i = 2, \ldots, 11,$$

$$g_i = (2\tfrac{1}{2}, -8\tfrac{1}{2}, 11)^T, i = 2, \ldots, 6,$$
$$g_i = (4, -7, 12\tfrac{1}{2})^T, i = 7, \ldots, 11,$$

$$M_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad M_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$b = (2, -1, 1)^T.$$

The solution of this problem is: $x_i = (1, 2, -1)^T$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(3,3,12),B(3,3,12),G(3,12),BMI(3,3,3),BCV(3),
     1          X(3,12),U(6,12),V(6,12),PHI(6,12),W(126)
      INTEGER IJ(3),NREC(3),KP(3),IW(24)
C
      N = 3
      IHOM = 1
      NU = 6
      NRI = 12
      KMI = 3
      LW = 126
      LIW = 24
      CALL EPSMAC(EPS)
      NREC(1) = 11
      IJ(1) = 1
      IJ(2) = 6
      IJ(3) = NREC(1)
C
C     SETTING OF THE RECURSION AND BC
C
      DO 1100 I = 1 , 10
        I1 = I + 1
        A(1,1,I) = -0.5D0
        A(2,1,I) = -1.5D0
        A(3,1,I) =  2.0D0
        A(1,2,I) =  2.0D0
        A(2,2,I) =  0.0D0
        A(3,2,I) =  1.0D0
        IF (I.LT.IJ(2)) THEN
          A(1,3,I) =  2.0D0
          A(2,3,I) =  2.0D0
          A(3,3,I) =  2.0D0
          G(1,I1) =  2.5D0
          G(2,I1) = -8.5D0
          G(3,I1) = 11.0D0
        ELSE
          A(1,3,I) = 0.5D0
          A(2,3,I) = 0.5D0
          A(3,3,I) = 0.5D0
          G(1,I1) =  4.0D0
          G(2,I1) = -7.0D0
          G(3,I1) = 12.5D0
        ENDIF
        B(1,1,I1) = -1.0D0
        B(2,1,I1) = -5.0D0
        B(3,1,I1) =  8.0D0
```

```
      B(1,2,I1) =  1.5D0
      B(2,2,I1) =  0.5D0
      B(3,2,I1) =  1.0D0
      B(1,3,I1) =  1.0D0
      B(2,3,I1) =  1.0D0
      B(3,3,I1) =  1.0D0
1100  CONTINUE
      DO 1200 L = 1 , KMI
      DO 1200 I = 1 , N
      DO 1200 J = 1 , N
       BMI(I,J,L) = 0.D0
1200  CONTINUE
      BMI(3,1,1) = 1.D0
      BMI(2,3,2) = 1.D0
      BMI(1,2,3) = 1.D0
      BCV(1) = 2.D0
      BCV(2) =-1.D0
      BCV(3) = 1.D0
      IERROR = 0
C
C     CALL TO SPLS2
C
      CALL SPLS2(N,IHOM,A,B,G,NRI,IJ,BMI,KMI,BCV,NREC,X,B,U,V,NU,PHI,
     1      G,KP,EPS,COND,AF,W,LW,IW,LIW,IERROR)
      IF ((IERROR.NE.0).AND.(IERROR.NE.710)) GOTO 3000
      CALL OUTSOL(IJ,COND,AF,KP,X,N,NRI,NREC(1))
      STOP
3000  WRITE(*,100) IERROR
      STOP
100   FORMAT(' TERMINAL ERRROR IN SPLS2 : IERROR = ',I4,/)
      END
C
      SUBROUTINE OUTSOL(IJ,COND,AF,KP,X,N,NRI,NREC)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(N,NRI)
      INTEGER IJ(3),KP(3)
C
      WRITE(*,190) (IJ(I),I=1,3)
      WRITE(*,200) COND,AF,(KP(J),J=1,2)
      E1 = 1.D0
      E2 = 2.D0
      E3 = -1.D0
      WRITE(*,100)
      DO 1100 I = 1 , NREC
       I1 = 1 - I
```

```
        WRITE(*,110) I,X(1,I),E1,E1-X(1,I)
        WRITE(*,120) X(2,I),E2,E2-X(2,I)
        WRITE(*,120) X(3,I),E3,E3-X(3,I)
1100   CONTINUE
C
100    FORMAT(' ',/,' I',7X,'X APPROX',11X,'X EXACT',14X,'ERROR',/)
110    FORMAT(' ',I2,3X,3(D16.9,3X))
120    FORMAT(' ',5X,3(D16.9,3X))
190    FORMAT(' "SWITCHING POINTS" IJ = ',3(I2,3X))
200    FORMAT(' ',/,' CONDITION NUMBER     = ',D12.5,/,
     1        ' AMPLIFICATION FACTOR = ',D12.5,/,
     2        ' K-PARTITIONINGS      = ',2(I2,2X),/)
300    FORMAT(' ')
310    FORMAT(' D(',I2,') = ',3(D16.9,3X))
       RETURN
       END
```

"SWITCHING POINTS" IJ = 1 6 11

CONDITION NUMBER = .12305D+01
AMPLIFICATION FACTOR = .49403D+01
K-PARTITIONINGS = 2 1

I	X APPROX	X EXACT	ERROR
1	.100000000D+01	.100000000D+01	.000000000D+00
	.200000000D+01	.200000000D+01	.000000000D+00
	-.100000000D+01	-.100000000D+01	-.999200722D-15
2	.100000000D+01	.100000000D+01	.555111512D-15
	.200000000D+01	.200000000D+01	.000000000D+00
	-.100000000D+01	-.100000000D+01	.888178420D-15
3	.100000000D+01	.100000000D+01	-.666133815D-15
	.200000000D+01	.200000000D+01	.222044605D-15
	-.100000000D+01	-.100000000D+01	-.111022302D-14
4	.100000000D+01	.100000000D+01	.000000000D+00
	.200000000D+01	.200000000D+01	.111022302D-14
	-.100000000D+01	-.100000000D+01	.000000000D+00
5	.100000000D+01	.100000000D+01	-.222044605D-15
	.200000000D+01	.200000000D+01	.222044605D-15
	-.100000000D+01	-.100000000D+01	-.555111512D-15
6	.100000000D+01	.100000000D+01	.000000000D+00

	.200000000D+01	.200000000D+01	.000000000D+00
	-.100000000D+01	-.100000000D+01	-.222044605D-15
7	.100000000D+01	.100000000D+01	-.444089210D-15
	.200000000D+01	.200000000D+01	.111022302D-14
	-.100000000D+01	-.100000000D+01	-.111022302D-14
8	.100000000D+01	.100000000D+01	.222044605D-15
	.200000000D+01	.200000000D+01	.222044605D-15
	-.100000000D+01	-.100000000D+01	-.122124533D-14
9	.100000000D+01	.100000000D+01	.222044605D-15
	.200000000D+01	.200000000D+01	.222044605D-15
	-.100000000D+01	-.100000000D+01	.444089210D-15
10	.100000000D+01	.100000000D+01	-.222044605D-15
	.200000000D+01	.200000000D+01	.133226763D-14
	-.100000000D+01	-.100000000D+01	-.888178420D-15
11	.100000000D+01	.100000000D+01	-.222044605D-15
	.200000000D+01	.200000000D+01	.222044605D-15
	-.100000000D+01	-.100000000D+01	-.888178420D-15

-

13. Subroutine SPLS3

****

SPECIFICATION

****

```
      SUBROUTINE SPLS1(N, IHOM, A, B, C, G, L, NREC, M1, MN, MZ, BCV,
     1      NPL, EPS, X, NX, Z, COND, AF, W, LW, IW, LIW, IERROR)
C     INTEGER N, IHOM, L, NREC, NPL, NX, LW, IW(LIW), LIW, IERROR
C     DOUBLE PRECISION A(N,N,NREC), B(N,N,NREC), C(N,L,NREC), G(N,NREC),
C    1      M1(NPL,N), MN(NPL,N), MZ(N,L), BCV(NPL), EPS,
C    2      X(N,NX), Z(L), COND, AF, W(LW)
```

****

Purpose

****

SPLS3 solves the discrete two-point BVP WITH PARAMETERS:

$$A_i\, x_i + B_i\, x_{i+1} + C_i\, z = g_i \quad , \quad i = 1, \ldots , NREC .$$

with BC:

$$M_1 x_1 + M_{NREC}\, x_{NREC+1} + M_z\, z = b$$

where A_i, B_{i+1} are N×N matrices, C_i an N×L matrix, g_i an N-vector, M_1, M_{NREC} are (N+L)×N matrices, M_z an (N+L)×L matrix and b an (N+L)-vector.
The vector z contains the L parameters.

****

Parameters

****

N — INTEGER, the order of the system.
Unchanged on exit.

IHOM — INTEGER.
IHOM indicates whether the system is homogeneous or inhomogeneous.
IHOM = 0 : the system is homogeneous,
IHOM = 1 : the system is inhomogeneous.
Unchanged on exit.

A — DOUBLE PRECISION array of dimension (N, N, NREC).
On entry A(. , . , i) must contain the matrix A_i, i = 1, . . . , NREC.

Unchanged on exit.

B DOUBLE PRECISION array of dimension (N, N, NREC).
On entry B(. , . , i) must contain the matrix B_i, i = 1, . . . , NREC.
Unchanged on exit.

C DOUBLE PRECISION array of dimension (N, L, NREC).
On entry C(. , . , i) must contain the matrix C_i, i = 1, . . . , NREC.
Unchanged on exit.

G DOUBLE PRECISION array of dimension (N, NREC).
If IHOM = 0, the array G has no real use and the user is recommended to use the same array for the X and the G.
If IHOM = 1, then on entry G(. , i) must contain the vector g_i, i = 1, . . . , NREC.
Unchanged on exit.

L INTEGER, the number of parameters.
Unchanged on exit.

NREC INTEGER.
NREC is one of the dimension of A, B, C and G. On entry NREC must contain the total number of recursions.
Unchanged on exit.

M1,MN DOUBLE PRECISION arrays of dimension (NPL, N).
On entry : M1 must contain the matrix M_1 and MN must contain the matrix M_{NREC} of the BC:
$M_1 x_1 + M_{NREC}\, x_{NREC+1} + M_z\, z = b$.
Unchanged on exit.

MZ DOUBLE PRECISION array of dimension (NPL, L).
On entry MZ must contain the matrix M_z of the BC.
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector b.
Unchanged on exit.

NPL INTEGER.
NPL is one of the dimension of M1, MN, MZ and BCV. On entry NPL must be equal to N + L !
Unchanged on exit.

EPS DOUBLE PRECISION.
On entry EPS must contain the machine constant EPS (see Remark 1.1).

Unchanged on exit.

X DOUBLE PRECISION array of dimension (N, NX).
On exit X(i,k) , i = 1, . . . , N contains the solution x_k, k=1, . . . , NREC + 1.

NX INTEGER.
NX is one of the dimension of X. On entry NX ≥ NREC + 1.
Unchanged on exit.

Z DOUBLE PRECISION array of dimension (L)
On exit Z(i), i = 1, . . . , L contains the solution for the parameters.

COND DOUBLE PRECISION.
On exit COND contains an estimate of the condition number.

AF DOUBLE PRECISION.
On exit AF contains an estimate of the amplification factor.

W DOUBLE PRECISION array of dimension (LW).
Used as work space.

LW INTEGER
LW is the dimension of W.
If IHOM = 0 : LW ≥ NPL*NPL*(7*NREC/2 + 11) + NPL*(5*NREC/2 + 8) + 1.
If IHOM = 1 : LW ≥ NPL*NPL*(7*NREC/2 + 11) + NPL*(7*NREC/2 + 10) + 1.
Unchanged on exit.

IW INTEGER array of dimension (LIW)
Used as work space.

LIW INTEGER
LIW is the dimension of IW. LIW ≥ 4 * NPL.
Unchanged on exit.

IERROR INTEGER
Error indicator; if IERROR = 0 then there are no errors detected.
See § 14 for the other errors.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines AMTES, APLB, BCMAV, CAMPF, CAPARC, CFUNRC, COPMAT, COPVEC, CONDW, CPSRC, CROUT, CUVRC, FQUS, GTUVRC, INPRO, INTCH, LUDEC, MATVC, QEVAK, QEVAL, QUDEC, SBVP, SOLDE, SOLUPP, SORTD, SORTD0, SPLS1, TAMVC, TUVRC, UPUP, UPVECP.

****

Remarks

****

SPLS3 is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: november 1991.

****

Method

****

See chapter VIII.

****

Example of the use of SPLS1

****

Consider the recursion:

$$A_i\, x_i + B_{i+1}\, x_{i+1} + C_i\, z = g_{i+1} \quad i = 1, \ldots, 10.$$

and a boundary condition :

$$M_1 x_1 + M_2 x_{11} + M_z\, z = b \ ,$$

where

$$A_i = \begin{bmatrix} 3 & -5 \\ 3 & -1 \end{bmatrix}, \quad C_i = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, i = 1, \ldots, 10,$$

$$B_i = \begin{bmatrix} 1 & -1 \\ 1 & 5 \end{bmatrix}, i = 1, \ldots, 5, B_i = \begin{bmatrix} 1 & -1 \\ 1 & 3 \end{bmatrix}, i = 6, \ldots, 10,$$

$$g_i = (15\tfrac{1}{2}, 5\tfrac{1}{2})^T, i = 1, \ldots, 5, \quad g_i = (15\tfrac{1}{2}, 7\tfrac{1}{2})^T, i = 6, \ldots, 10,$$

$$M_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad M_z = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix},$$

$$b = (3\tfrac{1}{2}, 1, \tfrac{1}{2})^T.$$

The solution of this problem is: $x_i = (2, -1)^T$, $z = 1\tfrac{1}{2}$.

In the next program the solution is computed and compared to the exact solution.
This program has been run on a OLIVETTI M24 personal computer (see Remark 1.2).

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(2,2,10),B(2,2,10),C(2,1,10),G(2,10),BM1(3,2),
     1    BMN(3,2),BMZ(3,1),BCV(3),X(2,11),Z(1),W(550)
       INTEGER IW(12)
C
C     SETTING OF THE PARAMETERS
C
      N = 2
      L = 1
      NPL = 3
      IHOM = 1
      NX = 11
      NREC = 10
      LW = 550
      LIW = 12
      CALL EPSMAC(EPS)
C
C     SETTING OF THE RECURSION AND BC
C
      DO 1100 I = 1 , 10
       A(1,1,I) =  3.D0
       A(1,2,I) = -5.D0
       A(2,1,I) =  3.D0
       A(2,2,I) = -1.D0
       C(1,1,I) = 1.D0
       C(2,1,I) = 1.D0
1100  CONTINUE
      DO 1200 I = 1 , 5
       B(1,1,I) =  1.D0
       B(1,2,I) = -1.D0
       B(2,1,I) =  1.D0
       B(2,2,I) =  5.D0
       G(1,I) = 15.5D0
       G(2,I) =  5.5D0
1200  CONTINUE
      DO 1300 I = 6 , 10
       B(1,1,I) =  1.D0
       B(1,2,I) = -1.D0
       B(2,1,I) =  1.D0
       B(2,2,I) =  3.D0
       G(1,I) = 15.5D0
       G(2,I) =  7.5D0
1300  CONTINUE
      BM1(1,1) = 0.D0
```

```
      BM1(2,1) = 1.D0
      BM1(3,1) = 0.D0
      BM1(1,2) = 0.D0
      BM1(2,2) = 0.D0
      BM1(3,2) = 1.D0
      BMN(1,1) = 1.D0
      BMN(2,1) = 0.D0
      BMN(3,1) = 0.D0
      BMN(1,2) = 0.D0
      BMN(2,2) = 1.D0
      BMN(3,2) = 0.D0
      BMZ(1,1) = 1.D0
      BMZ(2,1) = 0.D0
      BMZ(3,1) = 1.D0
      BCV(1) = 3.5D0
      BCV(2) = 1.0D0
      BCV(3) = 0.5D0
      CALL SPLS3(N,IHOM,A,B,C,G,L,NREC,BM1,BMN,BMZ,BCV,NPL,EPS,
     1           X,NX,Z,COND,AF,W,LW,IW,LIW,IERROR)
      IF ((IERROR.NE.0).AND.(IERROR.NE.710)) GOTO 3000
C
C     WRITING  OF THE SOLUTION
C
      CALL OUTSOL(COND,AF,X,N,NX,NREC,Z,L)
      STOP
3000  WRITE(*,100) IERROR
      STOP
100   FORMAT(' TERMINAL ERRROR IN SPLS3 : IERROR = ',I4,/)
      END
      SUBROUTINE OUTSOL(COND,AF,X,N,NX,NREC,Z,L)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(N,NX),Z(L)
C
      WRITE(*,200) COND,AF
      E1 = 1.5D0
      E2 = 2.D0
      E3 = -1.D0
      WRITE(*,210)
      DO 1100 J = 1 , L
        WRITE(*,220) Z(J),E1,E1-Z(J)
1100  CONTINUE
      WRITE(*,300)
      WRITE(*,100)
      DO 1200 I = 1 , NREC+1
        WRITE(*,110) I,X(1,I),E2,E2-X(1,I)
```

```
      WRITE(*,120) X(2,I),E3,E3-X(2,I)
1200  CONTINUE
C
100   FORMAT(' ',/,' I',7X,'X APPROX',11X,'X EXACT',14X,'ERROR',/)
110   FORMAT(' ',I2,3X,3(D16.9,3X))
120   FORMAT(' ',5X,3(D16.9,3X))
200   FORMAT(' ',/,' CONDITION NUMBER      = ',D12.5,/,
     1  ' AMPLIFACATION FACTOR = ',D12.5,/,/)
210   FORMAT(' ',4X,'Z APPROX',11X,'Z EXACT',14X,'ERROR',/)
220   FORMAT(' ',3(D16.9,3X))
300   FORMAT(' ')
310   FORMAT(' D(',I2,') = ',3(D16.9,3X))
      RETURN
      END
```

```
CONDITION NUMBER       =  .18883D+01
AMPLIFACATION FACTOR   =  .11000D+02


    Z APPROX          Z EXACT           ERROR

 .150000000D+01   .150000000D+01   -.399680289D-14


 I     X APPROX          X EXACT           ERROR

 1   .200000000D+01   .200000000D+01   -.133226763D-14
    -.100000000D+01  -.100000000D+01   -.166533454D-14
 2   .200000000D+01   .200000000D+01    .133226763D-14
    -.100000000D+01  -.100000000D+01    .111022302D-14
 3   .200000000D+01   .200000000D+01    .444089210D-15
    -.100000000D+01  -.100000000D+01   -.222044605D-15
 4   .200000000D+01   .200000000D+01    .444089210D-15
    -.100000000D+01  -.100000000D+01    .000000000D+00
 5   .200000000D+01   .200000000D+01    .444089210D-15
    -.100000000D+01  -.100000000D+01    .444089210D-15
 6   .200000000D+01   .200000000D+01    .444089210D-15
    -.100000000D+01  -.100000000D+01    .444089210D-15
 7   .200000000D+01   .200000000D+01   -.177635684D-14
    -.100000000D+01  -.100000000D+01    .000000000D+00
 8   .200000000D+01   .200000000D+01   -.444089210D-15
    -.100000000D+01  -.100000000D+01    .111022302D-14
```

9	.200000000D+01	.200000000D+01	-.888178420D-15
	-.100000000D+01	-.100000000D+01	.222044605D-15
10	.200000000D+01	.200000000D+01	.222044605D-15
	-.100000000D+01	-.100000000D+01	-.111022302D-15
11	.200000000D+01	.200000000D+01	-.177635684D-14
	-.100000000D+01	-.100000000D+01	.888178420D-15

14. Error messages

When an error is detected by one of the routines of BOUNDPAK, a terminal or warning error message with an error number IERROR is given. Three groups of error numbers can be distinguished:

i) $100 \le \text{IERROR} < 200$
These errors are INPUT errors and are detected before the actual computation starts. They are TERMINAL errors and occur when one or more parameters in the actual call to a BOUNDPAK routine have a wrong value.

ii) $200 \le \text{IERROR} < 300$
These errors are detected during the computation of the upper triangular recursion. Some are WARNING errors, but most are TERMINAL errors.

iii) $300 \le \text{IERROR} < 400$
These errors are detected during the computation of the solution of the linear multiple shooting system. These errors indicate that there is something wrong with your problem. Some are WARNING errors, others are TERMINAL errors.

Remark 14.1

BOUNDPAK contains a lot of subroutines. In most computer systems BOUNDPAK will be available via a BOUNDPAK library, which contains the object code of the subroutines. Therefore the most common way to use subroutines from BOUNDPAK is to write a program, in which calls are made to subroutines from BOUNDPAK, compile it and then link it with the BOUNDPAK library to obtain an execution code. The advantage is evident; instead compiling the program together with the BOUNDPAK package, only the program has to be compiled. However there is a disadvantage, namely, some programming errors are not detected, which would have been detected if the program together with the BOUNDPAK package was compiled as one large program. These undetected programming errors may cause an error message when the program is run. Therefore, if an error message occurs and according to your program it should not occur, check for the following mistakes in your program:

- Wrong number of parameters in a call to a subroutine.
- Parameters not in the right position in a call to a subroutine.
- Wrong type of parameter, e.g. integer parameter declared as real or real parameter declared as integer, etc.
- External subroutine not declared as external.

14.1 Errors detected by the subroutines:

INPUT errors.

100 $N < 1$
TERMINAL ERROR.

101 $IHOM \neq 0$ and $IHOM \neq 1$
TERMINAL ERROR

102 A=B or NRTI < 0.
TERMINAL ERROR

103 Either ER(1) or ER(2) or ER(3) is negative.
TERMINAL ERROR.

104 Value of NTI too small
TERMINAL ERROR

105 Value of NU is too small.
TERMINAL ERROR.

106 Either the value of LW or LIW is too small.
TERMINAL ERROR

107 Either $KSP < 1$ or $KSP \geq N$ or $NQD < KSP$.
TERMINAL ERROR.

108 IHOM = 0 and BCV = 0, so the solution will be zero.
TERMINAL ERROR

109 Either $A < B$ and $C \leq B$ or $A > B$ and $C \geq B$.
TERMINAL ERROR.

110 Subroutine is called with IEXT = 1, but the given value for C is wrong. It should be greater (less) than the actual used value for γ in the previous call to the subroutine (stored in TI(KEXT)) if A is less (greater) than B.
TERMINAL ERROR.

111 Value of NSP is too small.
TERMINAL ERROR.

112 NRTI(1) < 0.
TERMINAL ERROR.

113 $l < 1$.
TERMINAL ERROR.

114 NPL $\neq$ N+L.
TERMINAL ERROR.

115 IHOM(i) $\neq 0$ and IHOM(i) $\neq 1$ for $i=1, \ldots,$ NSP -1.

120 The routine was called with NRTI = 1, but the given output points in the array TI are not in strict monotone order.
TERMINAL ERROR.

121 The routine was called with NRTI = 1, but the first given output-point or the last output-point is not equal to A or B.
TERMINAL ERROR.

122 The switching points are not given in strict monotone order.
TERMINAL ERROR.

123 The routine was called with NRTI(1) = 1 , but the given output points in the array TI do not include all switching points.
TERMINAL ERROR.

Errors detected during computation.

200 This indicates that there is a minor shooting interval on which the incremental growth is greater than the AMP. The cause of this error lies in the used method for computing the fundamental solution.
WARNING ERROR.

201 This indicates that there is a minor shooting interval on which $\| M_j(i) \|$ is greater than max(ER(1) , ER(2)) / ER(3), i.e. TOL / EPS.
WARNING ERROR.

213 This indicates that the relative tolerance was too small. The subroutine has changed it into a suitable value.
WARNING ERROR.

215 This indicates that during integration the particular solution or a homogeneous solution has vanished, making a pure relative error test impossible. Must use non-zero absolute tolerance to continue.
TERMINAL ERROR.

216 This indicates that during integration the requested accuracy could not be achieved. User must increase error tolerance.
TERMINAL ERROR.

218 This indicates that the input parameter $N \leq 0$, or that either the relative tolerance or the absolute tolerance is negative.
TERMINAL ERROR.

222 This indicates that the increment of a fundamental solution has become greater than the allowed incremental factor ALI, so a new output point has to be inserted. However the current value of NTI is too small to insert a new output point. Output value is an estimate for NTI, taking into account possible not yet detected new output points, which have to be inserted when the increment of a fundamental solution becomes greater than ALI.
When changing the value of NTI, do not forget to change the arrays for which NTI is one of the dimensions.
TERMINAL ERROR

223 This indicates that the value of NTI is too small to compute the next necessary upertriangular matrix in the extension interval. Increase the value of NTI.
When changing the value of NTI, do not forget to change the arrays for which NTI is one of the dimensions.
TERMINAL ERROR.

224 This indicates that to avoid unnecessary overflow a new point has to be inserted, but the current value of NTI is too smal to insert new points. Output value is an estimate for NTI, taking into account possible not yet detected new points, which has to be inserted to avoid unnecessary overflow.
When changing the value of NTI, do not forget to change the arrays for which NTI is one of the dimensions.
TERMINAL ERROR

225 This indicates that a switching point is detected and has to be inserted in the output points. However, the current value of NTI is too small to insert a new output point. Output value is an estimate for NTI, taking into account the possible number of switching points, which are not detected at this stage.
When changing the value of NTI, do not forget to change the arrays for which NTI is one of the dimensions.
TERMINAL ERROR.

226 This indicates that $\| M(i) \|$ has become greater then max(ER(1) , ER(2)) / ER(3) (TOL / EPS) and a new output point has to be inserted. However the current value of NTI is too small to insert a new output point. Output value is an estimate for NTI, taking into account possible not yet dectected new output points, which have to be inserted if $\| M(i) \|$ becomes greater than TOL / EPS.
When changing the value of NTI, do not forget to change the arrays for which NTI is one of the dimensions.
TERMINAL ERROR.

250 This indicates that it was not possible to compute an SVD within 30 iterations.
TERMINAL ERROR.

300 This indicates that the global error is probably larger than the error tolerance due to instabilities in the system. Most likely the problem is ill-conditioned. Output value is the estimated amplification factor.
WARNING ERROR.

305 This indicates that the global error is probably larger than the error tolerance due to instabilities in the discrete multipoint BVP, derived from the side conditions and BC. Most likely the problem is ill-conditioned. Output value is an estimate for the amplification factor.
WARNING ERROR.

310 This indicates that one of the U_k is singular.
TERMINAL ERROR.

315 This indicates that the discrete multipoint BVP, derived from the side conditions and BC is singular.
TERMINAL ERROR.

320 This indicates that the problem is probably too ill-conditioned with respect to the BC.
TERMINAL ERROR.

325 This indicates that the problem is probably too ill-conditioned with respect to the BC.
TERMINAL ERROR.

330 The computed value for γ_{max} is larger than the given maximum value for γ in C. Output value is the estimated value for γ. The given value for γ_{max} is used for further computations.
WARNING ERROR

331 The computed number of unbounded growing modes on the interval $[\alpha, \beta]$ differs from the computed number of growing modes on the interval $[\alpha, \gamma]$. This might be caused by a very slowly increasing mode, or the problem is not dichotomic.
WARNING ERROR.

335 The number of exponentially growing modes is not the same as the number of unbounded modes. Probably the problem has non exponentially growing modes. It is also possible that the problem is not dichotomic, so check the value of ER(5).
WARNING ERROR.

340 This indicates that the BC is inconsistent with respect to the BC-vector. If also error 335 has occurred, then most probably both erros occured for the same reason. Otherwise, most probably the used value for B has been too small, so a larger value for B will solve this problem.

WARNING ERROR.

345 This indicates that the problem is ill-conditioned. A basis for a meaningful manifold will be computed.
WARNING ERROR.

350 This indicates that $\rho(EIG(1)) * \rho(EIG(2)) \geq 0$. Output values are the $\rho(EIG(1))$ and $\rho(EIG(2))$.
TERMINAL ERROR.

355 This indicates that no eigenvalue was found in the given interval. Output values are the boundary points of the given interval.
TERMINAL ERROR.

Errors of the special linear solvers.

600 N < 1.
TERMINAL ERROR.

601 IHOM ≠ 0 and IHOM ≠ 1.
TERMINAL ERROR.

602 NREC < 2.
TERMINAL ERROR.

603 Value of NRI is too small.
TERMINAL ERROR.

605 Value of NU is too small.
TERMINAL ERROR.

606 Either the value of LW or LIW is too small.
TERMINAL ERROR.

611 KMI < 2.
TERMINAL ERROR.

612 NREC(1) < 3.
TERMINAL ERROR.

613 L < 1.
TERMINAL ERROR.

614 Either NREC < 2 or NX < NREC + 1 or NPL ≠ N + L.
TERMINAL ERROR.

621 Either IJ(1) $\neq$ 1 or IJ(KMI) $\neq$ NREC(1).
TERMINAL ERROR.

622 The switching points are not given in strict monotonic order.
TERMINAL ERROR.

700 This indicates that the global error is probably larger than 1 / EPS, due to instabilities in the system. Most likely the problem is ill-conditioned. Output value is the estimated amplification factor.
WARNING ERROR.

710 This indicates that one of the A_i or B_i is singular in such a way that the linear system is singular.
TERMINAL ERROR.

720 This indicates that the problem is probably too ill-conditioned with respect to the BC.
TERMINAL ERROR.

15. Names of subroutines in BOUNDPAK.

In the next table all the names of the BOUNDPAK subroutines are given.

AMTES	ANORM1	APLB		
BCMAV				
CAMPF	CAPARC	CCI		CDI
CFUNRC	CHDIAU	CKLREC	CKPSW	CNRHS
CONDW	COPMAT	COPVEC	CPABC	CPARC
CPRDIA	CPSRC	CQIZI	CRHOL	CROUT
CSPABV	CTIMI	CTIPL	CUVRC	CWISB
DEFINC	DETSWP	DUR	DURIN	DURPA
EPSMAC				
FC2BVP	FCBVP	FCEBVP	FCIBVP	FCPBVP
FQUS	FUNPAR	FUNRC		
GKPMP	GKPPA	GOPBC	GTUR	GTURI
GTUVRC				
INPRO	INTCH			
KPCH				
LUDEC				
MATUP	MATVC	MTSDD	MTSE	MTSI
MTSMP	MTSP	MTSS	MUTSDD	MUTSEI
MUTSGE	MUTSIN	MUTSMI	MUTSMP	MUTSPA
MUTSPS	MUTSSE			
PSR				
QEVAK	QEVAL	QUDEC		
RKF1S	RKFSM			
SBVP	SMBVP	SOLDE	SOLUPP	SORTD
SORTD0	SPARC	SPLS1	SPLS2	SPLS3
SSDBVP	SVD			
TAMVC	TUR	TUVRC		
UPUP	UPVECP	UQDEC		