BOUNDPAK user's manual : chapter 3

Report no. 86-02

BOUNDPAK

User's Manual (Chapter III)

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

may 1986

project 830901

BOUNDPAK

A Package for Solving Boundary Value Problems

User's Manual

Chapter III

Multipoint Boundary Value Problems

R.M.M. Mattheij

G.W.M. Staarink

Mathematisch Instituut / Wiskundige Dienstverlening
Katholieke Universiteit
Toernooiveld
6525 ED Nijmegen
The Netherlands

CHAPTER III MULTIPOINT BOUNDARY VALUE PROBLEM

§ 1. Introduction

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

$$(1.1a) \quad \frac{dx}{dt} = L(t)x(t) + r(t) \ , \ \alpha \leq t \leq \beta,$$

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

$$(1.1b) \quad M_1 x(\alpha_1) + M_2 x(\alpha_2) + \ldots + M_{m+1} x(\alpha_{m+1}) = b,$$

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 $\alpha = \alpha_1 < \alpha_2 < \ldots < \alpha_{m+1} = \beta$, are the so called <u>switching points</u> .
Because of the linearity of (1.1a) we may write the solution $x(t)$ as:

$$(1.2) \quad x(t) = F(\alpha_i,t)c_i + w(\alpha_i,t), \ \alpha_i \leq t \leq \alpha_{i+1},$$

where $F(\alpha_i,t)$ is a fundamental solution on $[\alpha_i,\alpha_{i+1}]$ and $w(\alpha_i,t)$ a particular solution of (1.1a) on $[\alpha_i,\alpha_{i+1}]$. In priciple we may identify $F(\alpha_i,t)$ with $F(\alpha_j,t)$ for $i\neq j$, thus reducing (1.2) 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). 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:

$$(1.3) \quad F(\alpha_i,\alpha_{i+1})c_i = F(\alpha_{i+1},\alpha_{i+1})c_{i+1} + w(\alpha_{i+1},\alpha_{i+1}) - w(\alpha_i,\alpha_{i+1})$$

and the BC

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

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

The algorithm on which MUTSM is based now uses multiple shooting on each interval $[\alpha_i,\alpha_{i+1}]$. In this way we obtain a discrete analogue of (1.3) and (1.4) which constitutes a linear system **A** of order $m\times n$. The conditioning of the problem can be measured by $\|\mathbf{A}^{-1}\|$ 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 on consecutive intervals $[\alpha_i,\alpha_{i+1}],\ldots,[\alpha_{i+k},\alpha_{i+k+1}]$ say, the dichotomy does not change, 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 (1.3) and (1.4) change into

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

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

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

$$(1.7)\qquad M_1F(\alpha_1,\alpha_1)c_1 +\ldots+ \Big[\sum_{l=i}^{i+k} M_lF(\alpha_i,\alpha_l)\Big]c_i + M_{i+k+1}F(\alpha_{i+k+1},\alpha_{i+k+1}) + \ldots+ \Big[\sum_{l=m}^{m+1} M_lF(\alpha_m,\alpha_l)\Big]c_m = \hat{b} ,$$

$$\hat{b} = b - M_1w_1(\alpha_1,\alpha_1) -\ldots- \sum_{l=i}^{i+k} M_lw(\alpha_i,\alpha_l) -\ldots- \sum_{l=m}^{m+1} M_lw(\alpha_m,\alpha_l) .$$

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

§ 2. Global description of the algorithm

In this section we outline the actual computations performed.

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 will 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 $F_j(\alpha_i,.)$ and particular solutions $w_j(\alpha_i,.)$ 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 $Q_j(i)$ are orthogonal and the $U_j(i)$ upper triangular.
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:

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

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) (cf. [3,(3.4)]) and $\{z_j(i)\}_{j=1}^{N_i}$ some particular solution. Then for some vector c_i we should have:

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

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:

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

and

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

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

$$(2.7)\quad M_1Q_1(1)\Phi_1(1)c_1 + \ldots + [M_mQ_1(m)\Phi_1(m) + M_{m+1}Q_{N_m}(m)\Phi_{N_m}(m)]c_m = \tilde{b} \ ,$$

$$\tilde{b} = b - \sum_{i=1}^{m} M_iQ_1(i)z_1(i) - M_{m+1}Q_{N_m}(m)z_{N_m}(m)$$

$$- \sum_{i=1}^{m} M_iw_1(\alpha_i,\alpha_i) - M_{m+1}w_{N_m}(\alpha_m,\alpha_{m+1}) \ .$$

Denoting:

$$(2.8a)\quad \tilde{M}_i = M_iQ_1(i)\Phi_1(i) \qquad i=1,\ldots,m-1,$$

$$(2.8b)\quad \tilde{M}_m = M_mQ_1(m)\Phi_1(m) - M_{m+1}Q_{N_m}(m)\Phi_{N_m}(m) \ ,$$

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

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

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

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

then we obtain the linear system:

$$(2.9a)\quad \mathbf{Ac} = \mathbf{q} \ ,$$

where

$$(2.9b)\quad \mathbf{A} = \begin{bmatrix} \Psi_1 & -\Omega_2 & & & \\ & \ddots & \ddots & & \\ & & & \Psi_{m-1} & -\Omega_m \\ \tilde{M}_1 & \tilde{M}_2 & \cdots & \tilde{M}_{m-1} & \tilde{M}_m \end{bmatrix}, \ \mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_{m-1} \\ c_m \end{bmatrix} \ , \ \mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_{m-1} \\ \tilde{b} \end{bmatrix} .$$

Remark 2.10

In the case that the ODE (1.1a) 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 $\tilde{b}$ in (2.7) equals b and the vector **q** in (2.9a) becomes:

$$\mathbf{q} = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ 0 \\ b \end{bmatrix} .$$

§ 3. Special features of the method

The actual computation of the solutions $F(\alpha_i,.)$ and $w(\alpha_i,.)$ on each interval is basically the same as decribed in [3,§§3.1,3.2], i.e. the algorithm uses the adaptivity feature for the integration for the particular mode only. Also it 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 $[\alpha_i,\alpha_{i+1}]$. As a consequence we basically should reckon with a different partitioning integer k_p (cf. [§I.3.2]), indicating the dimension of the increasing solutionspace, on each such interval. If we denote this integer at the i^{th} interval by k(i), we then know from [1] that for well conditioned multipoint boundary value problems, k(i) is a non-increasing set, i.e. $k(1)\geq k(2)\geq\ldots\geq k(m)$. The fundamental solution $\{\Phi_j(i)\}_{j=1}^{N_i}$ cf.(2.3) on the i^{th} interval is then computed using the BC:

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

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

§ 3.2 Choosing $F(\alpha_i,\alpha_i)$ and $w(\alpha_i,\alpha_i)$

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 $Q_1(1) = F(\alpha_1,\alpha_1) = I$ and checks the ordering of the diagonal elements of the first upper triangular matrices $U_j(i)$, computed after reaching the endpoint of a minor shooting interval. If this ordering is found to be improper it performs permutation of columns like in [§I.3.3]. Arriving at $t=\alpha_2$ we have a complete freedom to choose $F(\alpha_2,\alpha_2)$. A very useful choice is:

$$(3.2)\qquad F(\alpha_2,\alpha_2) = Q_{N_1}(1) \;.$$

Indeed, if the dichotomy is invariant on $[\alpha_1,\alpha_3]$ then 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) $\quad a_j(2) =: a_{j+N_1}(1)$,

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.

§ 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)×n only, being of the form:

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

where

$$(3.4b)\quad \hat{\mathbf{A}} = \begin{bmatrix} \hat{\Psi}_1 & -\hat{\Omega}_3 & & & & \\ & \Psi_3 & -\Omega_4 & & & \\ & & & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & \cdot \\ \hat{M}_1 & \tilde{M}_3 & \tilde{M}_4 & \cdot & \cdot & \tilde{M}_m \end{bmatrix}, \quad \hat{\mathbf{c}} = \begin{bmatrix} c_1 \\ c_3 \\ \cdot \\ \cdot \\ \cdot \\ c_m \end{bmatrix}, \quad \hat{\mathbf{q}} = \begin{bmatrix} q_1 \\ q_3 \\ \cdot \\ \cdot \\ q_{m-1} \\ \tilde{b} \end{bmatrix}$$

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

$$(3.4c)\quad \hat{\Psi}_1 = \Phi_L(1) \; ; \quad \hat{\Omega}_3 = Q_L^{-1}Q_1(3)\Phi_1(3) \; ,$$

$$(3.4d)\quad \hat{M}_1 = M_1Q_1(1)\Psi_1(1) + M_2Q_{N_1}(1)\Psi_{N_1}(1) \; ,$$

$$(3.4e)\quad \hat{q}_1 = Q_L^{-1}(1)[w(\alpha_3,\alpha_3) - w_L(\alpha_1,\alpha_3)] + Q_L^{-1}(1)Q_1(3)z_1(3) - z_L(1) \; .$$

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.

§ 3.4 Special solution of the algebraic system (2.9)

Instead of solving the system (2.9) (or its condensed variant (3.4)) by LU-decomposition, we do the following: Rewrite the matrix **A** for simplicity as:

$$(3.5)\qquad \mathbf{A} = \begin{bmatrix} S_1 & -R_2 & & & & & \\ & S_2 & -R_3 & & & & \\ & & \cdot & \cdot & & & \\ & & & \cdot & \cdot & & \\ & & & & \cdot & \cdot & \\ & & & & & S_{N-1} & -R_N \\ T_1 & T_2 & T_3 & \cdot & \cdot & T_{N-1} & T_N \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ q_N \end{bmatrix}$$

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

$$(3.6)\qquad k_1 \geq k_2 \geq \cdots \geq k_{N-1} \geq k_N.$$

In the recursion (cf. (2.9) and (3.5))

$$(3.7)\qquad R_{i+1}c_{i+1} = S_i c_i - q_i ,$$

we have

$$(3.8a)\qquad R_{i+1} = \begin{bmatrix} R_{i+1}^{11} & R_{i+1}^{12} \\ \emptyset & I \end{bmatrix},$$

where R_{i+1}^{11} is $k_i \times k_i$ and the identity matrix is of order $n-k_{i+1}$, and

$$(3.8b)\qquad S_i = \begin{bmatrix} I & \emptyset \\ \emptyset & S_i^{22} \end{bmatrix},$$

where S_i is $(n-k_i)\times(n-k_i)$ and the identity matrix is of order k_i.

We now like to solve (3.7) 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.

$$(3.9) \qquad \Psi_i = \begin{bmatrix} \Psi_i^{11} & \Psi_i^{12} \\ \emptyset & \Psi_i^{22} \end{bmatrix}, \ \Psi_i^{11} \text{ of order } k_i \ .$$

At i=1 we define:

$$(3.10) \qquad [\Psi_1^{21} | \Psi_1^{22}] = [\ \emptyset \ | I_{n-k_1}] \ ,$$

and compute

$$(3.11a) \qquad \tilde{\Psi}_2^{22} = S_1^{22}\Psi_1^{22} \ ,$$

(For S_1^{22}, the right lower block of S_1, see (3.8b)), where $\tilde{\Psi}_2^{22}$ has the same order as S_1^{22} and $\tilde{\Psi}_1^{22}$.
Now compute Ψ_2^{22} as follows:

$$(3.11b) \qquad \Psi_2^{22} = \begin{bmatrix} I_{k_1-k_2} & \emptyset \\ \emptyset & \tilde{\Psi}_2^{22} \end{bmatrix}, \qquad \text{if } k_1 > k_2 \text{ and } \Psi_2^{22} = \tilde{\Psi}_2^{22} \text{ otherwise.}$$

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

$$(3.12a) \qquad \tilde{\Psi}_{i+1}^{22} = S_i^{22}\Psi_i^{22} \ ,$$

$$(3.12b) \qquad \Psi_{i+1}^{22} = \begin{bmatrix} I_{k_i-k_{i+1}} & \emptyset \\ \emptyset & \tilde{\Psi}_{i+1}^{22} \end{bmatrix}, \qquad \text{if } k_i > k_{i+1} \text{ and } \Psi_{i+1}^{22} = \tilde{\Psi}_{i+1}^{22} \text{ otherwise.}$$

At i=N we set

$$(3.13) \qquad [\Psi_N^{11} | \Psi_N^{12}] = [I_{k_N} | \emptyset] \ .$$

Then we have

$$(3.14) \qquad \Psi_N = \begin{bmatrix} \Psi_N^{11} & \emptyset \\ \emptyset & \Psi_N^{22} \end{bmatrix} = \begin{bmatrix} \tilde{\Psi}_N^{11} & \Psi_N^{12} \\ \emptyset & \tilde{\Psi}_N^{22} \end{bmatrix} ,$$

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

$$(3.15a) \qquad \Psi_{N-1}^{12} = R_N^{11}\Psi_N^{12} + R_N^{12}\tilde{\Psi}_N^{22} \ ,$$

$$(3.15b) \qquad \Psi_{N-1}^{11} = R_N^{11}\tilde{\Psi}_N^{11} \ .$$

And in general:

$$(3.16)\qquad \Psi_i = \begin{bmatrix} \Psi_i^{11} & \Psi_i^{12} \\ \emptyset & \Psi_i^{22} \end{bmatrix} = \begin{bmatrix} \tilde{\Psi}_i^{11} & \Psi_i^{12} \\ \emptyset & \Psi_i^{22} \end{bmatrix},$$

where Ψ_i^{11} is of order k_i and $\tilde{\Psi}_i^{11}$ is of order k_{i-1}.
Then:

$$(3.17a)\qquad \Psi_{i-1}^{12} = R_i^{11}\Psi_i^{12} + R_i^{12}\Psi_i^{22},$$

$$(3.16b)\qquad \Psi_{i-1}^{11} = R_i^{11}\tilde{\Psi}_i^{11},$$

Note that this scheme to compute $\{\Psi_i\}$ is a generalization of the dichotomic case dealt with in Ch. I.

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

$$(3.18a)\qquad p_1^2 = 0 \quad , \quad p_N^1 = 0$$

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

$$(3.18b)\qquad \tilde{p}_{i+1}^2 := S_i^{22}p_i^2 - q_i^2 \ ;$$

$$(3.18c)\qquad p_{i+1}^2 = \tilde{p}_{i+1}^2 , \quad \text{if } k_i = k_{i+1},$$

$$p_{i+1}^2 = \begin{bmatrix} \emptyset \\ \tilde{p}_{i+1} \end{bmatrix}, \quad \text{if } k_i > k_{i+1},$$

i.e. the first $k_i - k_{i+1}$ elements of p_{i+1}^2 are 0.

At the backward sweep we typically compute

$$(3.18d)\qquad p_i = \begin{bmatrix} p_i^1 \\ p_i^2 \end{bmatrix} = \begin{bmatrix} \tilde{p}_i^1 \\ \tilde{p}_i^2 \end{bmatrix},$$

where $\tilde{p}_i^1$ is a vector of order k_{i-1}.

$$(3.18e)\qquad p_{i-1}^1 = R_i^{11}\tilde{p}_i^1 + R_i^{12}\tilde{p}_i^2 + q_{i-1}^1,$$

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:

$$(3.19)\qquad c_i = \Psi_i v + p_i \ ,$$

where the vector v can be found from:

$$(3.20)\qquad \Big[\sum_{j=1}^{N} T_i\Psi_i\Big]v = \tilde{b} - \sum_{j=1}^{N} T_i p_i$$

§ 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 polychomic , 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 Ch. I,§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 (3.20), that is of the matrix W defined by

$$(3.21)\qquad W := \sum_{j=1}^{m} \tilde{M}_i\Psi_i \ .$$

One can show that in general

$$(3.22)\qquad \|W^{-1}\| \leq (m+1)CN,$$

where

$$(3.23)\qquad CN := \max_{t\in[\alpha,\beta]} \|F(t)\Big[\sum_{j=1}^{m+1} M_j F(\alpha_j)\Big]^{-1}\| \ ,$$

where F is any fundamental solution. Note that (3.23) is a straightforward generalization of I.(3.12) and is a measure for amplifications of perturbation in the BC. For stability with respect to perturbations in the ODE as such we may monitor appropiate blocks of the upper triangular matrices.

§ 4. Computational aspects

The routine MUTSM 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 MUTSG for two-point BVP (see Ch.I). Only the choice of the $Q_1(i)$, i=2,...,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 Ch.I).

§ 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$(\prod_{j=1}^{N_1} U_j(1))$ is ordered.

As a first choice for $Q_i(1)$, i=2,...,m we take (see §3.2)

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

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 satisfies $k_{i-1}\geq k_i$) 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 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 does not turn out 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,...,m, a partitioning parameter k_i, where k_i is the number of elements of diagonal$(\prod_{j=1}^{N_i} U_j(i))$, 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 $k_l = k_{l+1} = \ldots = k_p \neq k_{p+1}$.

step 5: Compute a global partitioning parameter $\tilde{k}_l$ 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

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

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 polychotomic also this procedure may not be satisfactory, naturally, and a large amplification factor may result (as is to be expected of course).

§ 4.3 **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).
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.

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, On the Conditioning of Multipoint Boundary Value Problems, Report CMA ..... , Canberra Australia (1986).

[2] F.R. de Hoog, R.M.M. Mattheij, An Algorithm for Solving Multipoint Boundary Value Problems, Report CMA-R31-85, Canberra Australia (1985).

[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), 745-763.

****
SPECIFICATION (FORTRAN IV)
****

```
      SUBROUTINE DMUTSM(FLIN,FDIF,N,IHOM,TSP,NSP,BCM,BCV,AMP,ER,NRTI,TI,
     1                  NTI,X,U,NU,Q,D,KPART,PHIREC,W,LW,IW,LIW,IERROR)
C     INTEGER N,IHOM,NSP,NRTI(NSP),NU,KPART(NSP),LW,IW(LIW),LIW,IERROR
C     DOUBLE PRECISION TSP(NSP),BCM(N,N,NSP),BCV(N),AMP,ER(5),TI(NTI),
C    1                 X(N,NTI),U(NU,NTI),Q(N,N,NTI),D(N,NTI),
C    2                 PHIREC(NU,NTI),W(LW)
C     EXTERNAL FLIN,FDIF
```

****
Purpose
****

DMUTSM solves the multi-point BVP:

$$\frac{dy(t)}{dt} = L(t)y(t) + r(t) \quad , A_1 \leq t \leq A_k \text{ or } A_k \leq t \leq A_1 \ ,$$

with BC:

$$M_{A_1} y(A_1) + M_{A_2} y(A_2) + \cdots + M_{A_k} y(A_k) = BCV \ , \quad k > 1 \ ,$$

where M_{A_j} , $j = 1,\ldots,k$ are the NxN BC matrices, BCV an N BC vector and $A_1 < A_2 <\ldots< A_k$ or $A_1 > A_2 >\ldots> A_k$ the switching points.

****
Parameters
****

FLIN SUBROUTINE, supplied by the user with specification:

```
SUBROUTINE FLIN(T,Y,F)
DOUBLE PRECISION T,Y(N),F(N)
```

where N is the order of the system. FLIN must evaluate the homogeneous part of the differential equation, L(t)y(t), for t=T and y(t)=Y and place the result in F(1),F(2),...,F(N).

FLIN must be declared as EXTERNAL in the (sub)program from which DMUTSM is called.

FDIF SUBROUTINE, supplied by the user, with specification:

```
SUBROUTINE FDIF(T,Y,F)
DOUBLE PRECISION T,Y(N),F(N)
```

where N is the order of the system. FDIF must evaluate the righthand-side of the inhomogeneous differential equation, L(t)y(t) + r(t), for t=T and y(t)=Y and place the result in F(1),F(2),...,F(N).

FDIF must be declared as EXTERNAL in the (sub)program from which DMUTSM is called.

In the case that the system is homogeneous FDIF is the same as FLIN.

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.

TSP DOUBLE PRECISION array of dimension (m), m ≥ NSP.
On entry TSP must contain the switching points A_j, j=1,...,NSP in monotone order, i.e. TSP(j) = A_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,m), m ≥ NSP.
On entry : BCM(.,.,j) must contain the BC matrix M_{A_j}, j=1,...,NSP.
Unchanged on exit.

BCV DOUBLE PRECISION array of dimension (N).
On entry BCV must contain the BC vector.
Unchanged on exit.

AMP DOUBLE PRECISION.
On entry AMP must contain the allowed incremental factor of the homogeneous solutions.
AMP should be greater than 1, if not the subroutine will change AMP into max(ER(1),ER(2)) / ER(3).
If NRTI(1) > 0, AMP is a dummy parameter.

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 e-12 the subroutine will change ER(1) into 1.E-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.
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 array of dimension (m), m ≥ NSP.
On entry:
NRTI(1) = 0, in this case the subroutine determine automatically the output-points using AMP.
NRTI(1) = 1, in this case the output-points are supplied by the user in the array TI. The output-points must be given in strict monotone order and must include the switching points.
NRTI(1) > 1, in this case the NRTI(j), j=2,...,NSP must contain the number of output-points - 1 , which are required on the interval [TSP(j-1),TSP(j)], j=2,...,NSP. The output-points are then computed by:

TI(1) = TSP(1),
TI((j-1)*NRTI(j+1)+1+m)=TSP(j)+m*(TSP(j+1)-TSP(j))/NRTI(j+1),
m = 1,...,NRTI(j+1).
If NRTI(i) < 2, i=2,...,NSP, NRTI(i)=1 is used in the above formular.
Note that the switching points will always be output-points.
On exit NRTI(1) contains the total number of output-points.

TI DOUBLE PRECISION array of dimension (NTI).
On entry; if NRTI(1) = 1, TI must contain the required output-points in strict monotone order: A_1=TI(1)<...<TI(1)=A_k or A_1=TI(1)>...>TI(1)=A_k (l denotes the total number of required output-points). The output-points must include all switching points A_j,j=1,...,k.
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, PHIREC. NTI must be greater then the total number of output-points + 4 .
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 uppertriangular 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 PHIREC.
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 recommanded 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 >NSP.
On exit KPART(j) contains the global partitioning parameter of the interval [TSP(j-1),TSP(j)], j = 2,...,NSP.

PHIREC DOUBLE PRECISION array of dimension (NU,NTI).
On exit PHIREC contains a fundamental solution of the multiple shooting recursion. The fundamental solution is uppertriangular 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 the work array W.
NW ≥ 3*N*N + 8*N + NSP*(1.5*N*N + 2.5*N)

IW INTEGER array of dimension (NIW)
Used as work space.

NIW INTEGER.
NIW is the dimension of the INTEGER array IW.
NIW ≥ 3*N + (N+1)*NSP.

IERROR INTEGER.
Error indicator; IERROR = 0 then there are no errors detected.

****
Error indicators
****

Errors detected by the subroutine

IERROR = 0 No errors detected .

IERROR = 100 INPUT ERROR: either N < 2 or IHOM < 0 or NSP < 2 or NRTI(1) < 0 or NTI < NSP + 4 or NU < N*(N+1)/2.
TERMINAL ERROR.

IERROR = 101 INPUT ERROR: either ER(1) or ER(2) or ER(3) is negative.
TERMINAL ERROR.

IERROR = 103 INPUT ERROR: either LW < 3*N*N + 8*N + NSP*(1.5*N + 2.5) or LIW < 3*N + NSP*(N+1).
TERMINAL ERROR.

IERROR = 120 INPUT ERROR: the routine was called with NRTI = 1, but the given output-points in the array TI are not in strict monotone order.
TERMINAL ERROR.

IERROR = 121 INPUT ERROR: 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.

IERROR = 122 INPUT ERROR: the value of NTI is too small; the number of output-points is greater than NTI - 4.
TERMINAL ERROR.

IERROR = 130 INPUT ERROR: the switching points are not given in strict monotone order.
TERMINAL ERROR.

IERROR = 131 INPUT ERROR: 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.

IERROR = 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.

IERROR = 213 This indicates that the relative tolerance was too small. The subroutine has changed it into a suitable value.
WARNING ERROR.

IERROR = 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.

IERROR = 216 This indicates that during integration the requested accuracy could not be achieved. User must increase error tolerance.
TERMINAL ERROR.

IERROR = 218 This indicates that the input parameter N <= 0, or that either the relative tolerance or the absolute tolerance is negative.
TERMINAL ERROR.

IERROR = 240 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 error amplification factor.
WARNING ERROR.

IERROR = 250 This indicates that one of the U_k is singular.
TERMINAL ERROR.

IERROR = 260 This indicates that the problem is probably too ill-conditioned with respect to the BC.
TERMINAL ERROR.

****
Auxiliary Routines
****

This routine calls the BOUNDPAK library routines DMTSMP.

****
Remarks
****

DMUTSM is written by G.W.M. Staarink and R.M.M. Mattheij.
Last update: 01-04-1986.

****
Method
****

See chapter III of BOUNDPAK User's Manual

****
Example of the use of DMUTSM
****

Consider the ordinary differential equation

$$\frac{dx(t)}{dt} = L(t)\,x(t) + r(t)\,, \quad -1 \le t \le 1$$

and a BC

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x(-1) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e \\ 1+e^{-1} \end{bmatrix},$$

where

$$L(t) = \begin{bmatrix} -t+\frac{1}{2}-(t+\frac{1}{2})\cos 2t & 1+(t+\frac{1}{2})\sin 2t \\ -1+(t+\frac{1}{2})\sin 2t & -t+\frac{1}{2}+(t+\frac{1}{2})\cos 2t \end{bmatrix},$$

$$r(t) = \begin{bmatrix} (-3+\cos t(\cos t-\sin t)(2t+1))e^{-t} \\ (-1+\sin t(\sin t-\cos t)(2t+1))e^{-t} \end{bmatrix}.$$

The solution of this problem is $x=(e^{-1},e^{-t})^T$. 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 AS9000 VM/CMS computer.

```
      DOUBLE PRECISION TSP(3),BCM(2,2,3),BCV(2),AMP,ER(5),TI(16),
     1 X(2,16),U(3,16),Q(2,2,16),D(2,16),PHIREC(3,16),W(61),XEX,AE
      INTEGER KPART(3),NRTI(3),IW(15)
      EXTERNAL FLIN,FDIF
C
C     SETTING OF THE INPUT PARAMETERS
C
      N = 2
      IHOM = 1
      NSP = 3
      NTI = 16
      NU = 3
      LW = 61
      LIW = 15
      TSP(1) = -1.D0
      TSP(2) = 0.D0
      TSP(3) = 1.D0
      ER(1) = 1.1D-12
      ER(2) = 1.D-6
      ER(3) = 1.1D-15
      NRTI(1) = 2
```

```
      NRTI(2) = 4
      NRTI(3) = 4
      DO 1100 I = 1 , NSP
      DO 1100 J = 1 , N
      DO 1100 L = 1 , N
        BCM(J,L,I) = 0.DO
 1100 CONTINUE
      BCM(1,1,1) = 1.DO
      BCM(2,1,2) = 1.DO
      BCM(2,2,3) = 1.DO
      BCV(1) = DEXP(1.DO)
      BCV(2) = 1.DO + DEXP(-1.DO)
C
C     CALL DMUTSP
C
      CALL DMUTSM(FLIN,FDIF,N,IHOM,TSP,NSP,BCM,BCV,AMP,ER,NRTI,TI,NTI,
     1            X,U,NU,Q,D,KPART,PHIREC,W,LW,IW,LIW,IERROR)
C
C     PRINTING OF THE SWITCHING POINTS, CONDITION NUMBER AND
C     AMPLIFICATION FACTOR
      WRITE(*,100) (TSP(I),I=1,NSP)
      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(*,120)
      DO 1200 I = 1 , NRTI(1)
        XEX = DEXP(-TI(I))
        AE = XEX - X(1,I)
        WRITE(*,130) I,TI(I),X(1,I),XEX,AE
        AE = XEX - X(2,I)
        WRITE(*,140) X(2,I),XEX,AE
 1200 CONTINUE
      STOP
  100 FORMAT(' SWITCHING POINTS: ',3(F5.2,3X),/)
  110 FORMAT(' CONDITION NUMBER     = ',D12.5,/,
     1       ' AMPLIFICATION FACTOR = ',D12.5,/)
  120 FORMAT('   I',6X,'T',8X,'APPROX. SOL.',7X,'EXACT SOL.',9X,
     1 'ABS. ERROR',/)
  130 FORMAT(' ',I3,3X,F7.3,3(3X,D16.9))
  140 FORMAT(' ',13X,3(3X,D16.9))
      END
      SUBROUTINE FLIN(T,Y,F)
C     ----------------------
C
      DOUBLE PRECISION T,Y(2),F(2),TI,SI,CO
C
      TI = 2.DO * T
      SI = DSIN(TI) * (T + 0.5DO)
      CO = DCOS(TI) * (T + 0.5DO)
      TI = 0.5DO - T
      F(1) = (-CO + TI) * Y(1) + (1.DO + SI) * Y(2)
      F(2) = (-1.DO + SI) * Y(1) + (CO + TI) * Y(2)
      RETURN
      END
      SUBROUTINE FDIF(T,Y,F)
C     ----------------------
C
```

```
      DOUBLE PRECISION T,Y(2),F(2),TI,SI,CO
C
      CALL FLIN(T,Y,F)
      TI = 2.DO * T + 1.DO
      SI = DSIN(T)
      CO = DCOS(T)
      F(1) = F(1) + (CO * (CO - SI) * TI - 3.DO) * DEXP(-T)
      F(2) = F(2) + (SI * (SI - CO) * TI - 1.DO) * DEXP(-T)
      RETURN
      END
```

```
SWITCHING POINTS: -1.00    0.00    1.00

CONDITION NUMBER      =  0.61297D+01
AMPLIFICATION FACTOR =  0.43276D+01
```

I	T	APPROX. SOL.	EXACT SOL.	ABS. ERROR
1	-1.000	0.271828183D+01	0.271828183D+01	0.000000000D+00
		0.271828175D+01	0.271828183D+01	0.735283456D-07
2	-0.750	0.211699998D+01	0.211700002D+01	0.392049353D-07
		0.211699991D+01	0.211700002D+01	0.108340285D-06
3	-0.500	0.164872118D+01	0.164872127D+01	0.933285598D-07
		0.164872114D+01	0.164872127D+01	0.128283103D-06
4	-0.250	0.128402527D+01	0.128402542D+01	0.150341585D-06
		0.128402529D+01	0.128402542D+01	0.127439107D-06
5	0.000	0.999999808D+00	0.100000000D+01	0.191680902D-06
		0.999999891D+00	0.100000000D+01	0.109373627D-06
6	0.250	0.778800571D+00	0.778800783D+00	0.211765101D-06
		0.778800694D+00	0.778800783D+00	0.886011105D-07
7	0.500	0.606530374D+00	0.606530660D+00	0.285309543D-06
		0.606530718D+00	0.606530660D+00	-0.580605573D-07
8	0.750	0.472366284D+00	0.472366553D+00	0.268479159D-06
		0.472366790D+00	0.472366553D+00	-0.237313370D-06
9	1.000	0.367879306D+00	0.367879441D+00	0.134962726D-06
		0.367879633D+00	0.367879441D+00	-0.191680902D-06