report no. WD 85-06

BOUNDPAK

User's Manual (Chapter II)

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

december 1985

project 830901
BOUNDPAK
A Package for Solving Boundary Value Problems
User's Manual
Chapter II
Linear non-stiff Boundary Value Problems

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

Mathematisch Instituut / Wiskundige Dienstverlening
Katholieke Universiteit
Toernooiveld
6525 ED Nijmegen
The Netherlands
CHAPTER II LINEAR BVP ON INFINITE INTERVALS

§ 1. Introduction

If for an ODE

\[ \frac{dx}{dt} = L(t)x(t) + r(t), \quad t > a, \]

a BC is given

\[ M_\alpha x(\alpha) + M_\infty x(\infty) = b, \]

then it can be shown that \( x \) can be written as

\[ x(t) = P^2(t)c^2 + w(t), \]

where \( w(t) \) is a bounded particular solution and \( P^2(t) \) is a matrix solution \( (P^2(t) = nxk_0 \text{ say}) \) of bounded homogeneous solutions (see [1]). If only the complementary part of homogeneous solutions, \( P^1(t) \) say, consists of exponentially increasing solution, the algorithm finds a way to obtain an \( O(\text{TOL}) \) accurate solution on a prescribed interval \([\alpha, \beta]\). For this the integrations are automatically extended over some interval \([\beta, \gamma]\). For more general classes of problem explicit knowledge of some of the quantities computed in the program may be helpful. If \( \text{IERROR} = 265 \) occurs one needs to study the options described in §§3.3 and 3.4.

Remark 1.4. Although the algorithm computes \( c^2 \) from the (usually) singular system \([M_\alpha P^2(\alpha) + M_\infty P^2(\infty)]c^2 = \bar{b} \) (where \( \bar{b} \) is derived from the BC in a least square sense) we can still determine a quantity like 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

\[ \frac{dx(t)}{dt} = L(t)x(t) + r(t), \quad \alpha < t < \infty, \]

where \( L(t) \) is an \( nxn \)-matrix and \( r(t), x(t) \) are \( n \)-vectors for all \( t \). Let the BC be given by
(2.2) \( M_x(a) + M_\omega x(\omega) = b. \)

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

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

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

\[
(2.4) \quad x(t) = F^2(t)c^2 + w(t).
\]

Upon substituting (2.4) in (2.2) we find

\[
(2.5) \quad [M_x F^2(a) + M_\omega F^2(\omega)]c^2 = b - M_x w(a) - M_\omega w(\omega).
\]

Note that in case \( F^2(t) \), \( w(t) \rightarrow 0, t \rightarrow \infty \), the condition above reduces to an initial value condition (though rank deficient!). Because of the requirements on \( F^2 \) 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 nonincreasing ("stable") manifold.

With some adaptations this can be done along the lines of the method described in Ch I. Suppose we like to have output values for \( x \) on the interval \([a, \beta]\) within an accuracy \( TOL \). Let us assume that \( F^1 \) consists of exponentially increasing solutions only. Then there certainly exists a point \( \gamma \), such that

\[
(2.6) \quad \|P(\gamma)PP(\beta)^{-1}\| > TOL^{-1},
\]

where \( P = \begin{bmatrix} I_{k_u} & 0 \\ 0 & 0 \end{bmatrix} \); 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 §I.2.1, with at least \( \alpha = t_0, \beta = t_M \) and \( \gamma = t_N \) as output points, resulting in an upper triangular recursion

\[
(2.7) \quad a_{i+1} = U_{i+1}a_i + d_{i+1}, \quad i = 0, 1, \ldots, N-1,
\]

(cf. I.(2.1)), with

\[
(2.8) \quad x(t_i) = Q_1 a_i + w_i(t_i).
\]

-- 2 --
Then compute a particular solution \( \{ z_1 \} \) of (2.7), satisfying

\[
(2.9) \quad z_0^2 = 0 , \quad z_N^1 = 0
\]

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

\[
(2.10) \quad \psi_0^2 = I ; \quad \psi_N^1 = \varphi ,
\]

like in I.(3.2).

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

\[
(2.11) \quad a_i = \psi_i c^2 + z_i .
\]

From (2.8), (2.11) and (2.22) we thus derive the following relation for \( c^2 \)

\[
(2.12) \quad [ M_{\alpha} \psi_0 + M_{\alpha}^2 \psi_0 ] c^2 = b - M_{\alpha} \psi_0 (a) - M_{\alpha} \psi_M (\beta) - M_{\alpha} c_0 - M_{\alpha}^2 c_M .
\]

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

§ 3. Special features

The previously outlined algorithm is implemented as MUTSI. For the computation of the multiple shooting recursion on the interval \([\alpha, \gamma]\) the same strategy is used as in §§1.3.1-1.3.3 for BVP with general BC.

§ 3.1. Errors introduced by finite choice of \( \gamma \)

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

\[
(3.1) \quad U_i = \begin{bmatrix} E_i & C_i \\ 0 & E_i \end{bmatrix}
\]

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

\[
(3.2) \quad \| [ \Pi_{i=1}^M B_i ]^{-1} \| < \kappa e^{\lambda(\gamma - \beta)}
\]
Since we do not know the bounded (and non increasing) solutions at $t_N$ exactly we choose their component in span $F^i(t_N)$ to be zero, cf. (2.9) and (2.10). Hence we introduce a truncation error $T^{(N)}_i$ cf [2], which satisfies the homogeneous part of I (3.2b):

$$T^{(N)}_i = B^{(N)}_i.$$

Because of the boundedness of those solutions we have

$$\|T^{(N)}_i\| = O(1),$$

whence,

$$\|T^{(N)}_i\| = O(e^{\lambda(y-t_i)}).$$

Hence if $e^{\lambda(y-t)} < TOL$ (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 §I.3.4) has to be redefined here. Since we virtually rule out the increasing components we may define the subcondition numbers cf. [2]:

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

where $I-P = \begin{bmatrix} 0 & 0 \\ 0 & I_k \end{bmatrix}$ and + denotes a pseudo-inverse. By making use of the approximate $\{Q_i P_i\}$ instead of $F^2(t_i)$, we can estimate $CN_p(\beta)$ by (cf. I.(3.13))

$$\kappa_p(\beta) = \|\left[M_\alpha Q_{\infty} P_{\infty} + M_\infty Q_{\alpha} P_{\alpha}\right]^+\|.$$

§ 3.3. Problems with polynomials increasing modes

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

\begin{equation}
F^1(t) \equiv [G_1(t) | G^2(t)],
\end{equation}

where $G^2$ is an $n \times k_q$ matrix representing the polynomially increasing modes, $G^1$ 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 $w$ and $G^2$ have asymptotically constant directions. If we partition the truncation error $T^{(N)}_i$ in two components, viz.

\begin{equation}
T^{(N)}_i = \begin{bmatrix} [T^{(N)}_i]_1 \\ [T^{(N)}_i]_2 \end{bmatrix},
\end{equation}

where $[T^{(N)}_i]_1$ has $k$ components, then it makes sense to try some asymptotical expansion for $[T^{(N)}_i]_2$, e.g.

\begin{equation}
[T^{(N)}_i]_2 = v_0 + t^{-\omega} v_1 + t^{-2\omega} v_2 + \cdots,
\end{equation}

where $\omega > 0$ and $v_0, v_1, \ldots$ are independent of $t_N$; obviously the user should provide the model for this.

If we apply this idea we see that the point $\gamma$ is mainly determined by the exponential behaviour of $G^1 (c.f. (3.3b))$. 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 $w$ we may try an asymptotical expansion like

\begin{equation}
x(t) = u_0 + t^{-\omega} u_1 + t^{-2\omega} u_2 + \cdots,
\end{equation}

where $\omega$ and $u_0, u_1, \ldots$ are independent of $t$ (we assume $t-\alpha$ large enough); again the user should provide the proper model. If we choose $\gamma$ large enough, so that exponentially increasing modes have been damped out within TOL on $[a, b]$, 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 $\hat{x}$:
\[ (3.10a) \dot{x}(t_i) := x(t_i) + e(t_i), \]

with

\[ (3.10b) \dot{x}(t_i) = Q_i(y_i^2+z_i) \]

\[ (3.10c) e(t_i) = Q_i \hat{\varphi} \hat{c}, \]

where \( \hat{\varphi} \) is an \( n \times k \)-matrix, representing the polynomially increasing modes and \( \hat{c} \) a constant \( k \)-vector, only depending on the choice of \( \gamma \). Now one should realize that \( \{ y_i \} \) can be computed in much the same way as \( \{ \varphi_i \} \). 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 \( u_0, u_1, \ldots \), from monitoring \( \dot{x}(t_i) \) for various values of \( t_i \). Note that we only need \( k \) points \( t_i \) to find \( \hat{c} \) in case \( x \) is a constant vector.

§ 4. Computational aspects

The code MUTSI is based on the computational framework as outlined in I. Some special aspects are considered below.

§ 4.1 Determination of \( \gamma \) and bounded solutions

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

\[ (4.1) \lambda := (\ln m)/(\beta-\alpha), \]

where \( m \) is the absolutely smallest diagonal element of \( H B_i \). From this a value of \( \gamma \) is computed as

\[ (4.2) \gamma := \beta - \frac{\ln \text{TOL}}{\lambda}, \]

Arriving at \( t = \gamma \) it is checked whether the increment is large enough indeed, and if necessary a new (larger) \( \gamma \) is computed, using an updated \( \lambda \). If the latter value of \( \gamma \) is still insufficient to give large enough increments, a warning error IERROR = 265 occurs. It may happen that \( \gamma \) as defined by (4.2) is already quite large (due to a pessimistic choice of the partitioning parameter \( k \)). Therefore the user should provide a maximum value of \( \gamma \), \( \gamma_{\text{max}} \) say. If \( \gamma \) becomes larger then \( \gamma_{\text{max}} \), \( \gamma_{\text{max}} \) is taken as the value for \( \gamma \) and a warning error IERROR = 263 occurs.

-- 6 --
§ 4.2. Use of BC and determination of conditioning constants

System (2.12) can be written as

\[(4.3) \quad [M_\alpha^0 \varphi | \varphi_{0}] + M_\infty^r [\varphi | \varphi_r] [c_2]^0 = \delta.\]

where \(\delta = b - M_\alpha^0 \xi_0 - M_\infty^r \xi_r.\) 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 \(\Sigma\), such that

\[(4.4) \quad M_\alpha^0 [\varphi | \varphi_0] + M_\infty^r [\varphi | \varphi_r] = \Sigma V^T,\]

where \(\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n),\) with \(\sigma_1 \geq \ldots \sigma_{k_b} \geq 0,\) \(\sigma_{k_b+1} = \ldots = \sigma_n = 0,\) and

\[\begin{bmatrix} 0 \\ c_2 \end{bmatrix} = V\xi, \text{Then (4.3) can be rewritten as}\]

\[(4.5) \quad \Sigma \xi = U^T\delta.\]

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

\[(4.6) \quad \sigma_i = 0 \Rightarrow \xi_i = 0, \quad i = 1, \ldots, n.\]

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

\[(4.7) \quad \sigma_i < TOL \Rightarrow \xi_i < TOL, \quad i = 1, \ldots, n\]

is true or false. If (4.7) is false a warning error \(\text{IERROR} = 270\) is given. It is possible that \(\text{IERROR} = 270\) occurs after the warning error \(\text{IERROR} = 265.\) In that case \(\text{IERROR} = 265\) is likely to cause \(\text{IERROR} = 270\) too.

If we write \(\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_i, 0, \ldots, 0)\) \((1 \leq k_b),\) we can define its pseudo inverse as \(\Sigma^+ = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_i^{-1}, 0, \ldots, 0)\) and hence solve (4.5) in a straight forward 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 \(\text{IERROR} = 260\) is given. In such a case, and more generally if \(\sigma_{i+1}, \ldots, \sigma_{k_b}\) are smaller than or equal to \(TOL,\) we choose
(4.9) \( \kappa = \sigma_{\perp}^{-1} \),

unless \( \lambda_{\perp} = 1 \). 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 \( \sigma_{\perp i} \), \( i = 1, \ldots, k_b \). Let us write

(4.10) \( V = (v_{\perp 1} \cdots v_{\perp k_b}) \),

then these basis solutions are defined by

(4.11) \( \{Q_{i,i} v_{\perp j} \}_{i=0}^{N}, j = 1, \ldots, k_b \).

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 MUTSI for problems with slowly increasing modes

For problems without an exponential dichotomy MUTSI may fail to compute a bounded solution accurate up to TOL. If the warning error IERROR = 265 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 MUTSI 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 \( \gamma \). A new call to MUTSI results in the computation of a new solution using the new value of \( \gamma \). This means that one can use approximate solutions for various \( \gamma \) and hence utilize asymptotics. Because of the variety of possible expansions the user should write himself a program that calls MUTSI and then uses Richardson extrapolation (for instance). Obviously, denoting the approximate value of \( x(\alpha) \) obtained from using \( \gamma \) as a terminal point by \( x_{\gamma}(\alpha) \), it follows from an assumption like (3.8) that also \( x_{\gamma}(\alpha) \) satisfies an expansion in \( \gamma^{-\omega} \).

In case (ii) of § 3.3 the fundamental solutions \( \Psi \) and \( \Psi \) are stored in the array PHIREC. Then not only an approximate \( x_{\gamma}(\alpha) \) is given but also the values of the not exponentially increasing solutions at the output points.

When applying the previous idea, one should realize that all computations are exact within \( O(TOL) \). 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 \( \gamma \) as small as possible when slowly increasing modes (that should not influence its choice !) are detected. If \( \gamma \) happens to be equal to \( \gamma_{\max} \) the actual found partitioning integer \( k_{\max} \) is based on the criterion that exponentially growing modes should at least correspond to a \( \lambda \) (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


**SPECIFICATION**

SUBROUTINE DMUTSI(FLIN,FDIF,N,IHOM,A,B,C,MA,MINF,BCV,AMP,ER,NRTI,
1       TI,NTI,IEXT,X,NRSOL,U,NU,Q,D,KU,KE,KPART,PHIREC,
2       w,LW,LIW,IERI,IERROR)

C INTEGER N,IHOM,NRTI,NTI,IEXT,NRSOL,NU,KU,KE,KPART,LW,LIW,IERI
C 1 IERROR
C DOUBLE PRECISION A,B,C,MA(N,N),MINF(N,N),BCV(N),AMP,ER(5),TI(NTI),
C 1 X(N,NTI,N),U(NU,NTI),Q(N,N,NTI),D(N,NTI),
C 2 PHIREC(NU,NTI),W(LW)
C EXTERNAL FLIN,FDIF

Purpose

DMUTSI solves the two-point BVP defined on an infinite interval:

\[
\frac{dx(t)}{dt} = L(t)x(t) + r(t) , \quad t > A ,
\]

with BC:

\[
M_A x(A) + M_\infty x(\infty) = BCV
\]

where \(M_A\) and \(M_\infty\) are the BC matrices and \(BCV\) the BC vector.

It gives output on a subinterval \([A,B]\) specified by the user.

Parameters

FLIN SUBROUTINE, supplied by the user with specification:

SUBROUTINE FLIN(T,X,F)
DOUBLE PRECISION T,X(N),F(N)

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

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

FDIF SUBROUTINE, supplied by the user, with specification:

SUBROUTINE FDIF(T,X,F)
DOUBLE PRECISION T,X(N),F(N)

where \(N\) is the order of the system. FDIF must evaluate the righthand side of the inhomogeneous differential equation, \(L(t)x(t) + r(t)\), for \(t=T\) and \(x(t)=X\) and place the result in \(F(1), F(2), \ldots, F(N)\).
FDIF must be declared as EXTERNAL in the (sub)program from which DMUTSG 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.

A, B DOUBLE PRECISION.
A, B denotes the interval \([a, \beta]\) (see §2.). If \(M \neq 0\) B should be taken sufficiently large.
Unchanged on exit.

C DOUBLE PRECISION.
When IEXT=0 C must contain the value for \(\gamma_{max}\) (see §4). The actual used value for \(\gamma\) is stored in TI(NRTI+KE).
When IEXT=0, the routine computes the solution using the given value in C as the value for \(\gamma\). When TI(1)<TI(NRTI+KE), C must be greater than TI(NRTI+KE) and C must be smaller than TI(NRTI+KE) otherwise. Note that on subsequent calls to DMUTSI with IEXT=0, the value of KE may change.
Unchanged on exit.

MA, M DOUBLE PRECISION array of dimension (N,N).
On entry : MA and M sub inf must contain the matrices in the BC:
\(M_A x(A) + M_\infty x(\infty) = BCV\).
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 > 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 than 1.0 e-12 the subroutine will change ER(1) into 1.0 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.
On entry:
NRTI = 0, in this case the subroutine determine automatically the output-points using AMP.
NRTI = 1, in this case the output-points are supplied by the user in the array TI.
NRTI > 1, in this case the subroutine computes the output-points TI(k) by:
\[
TI(k) = A + (k-1) \times (B - A) / NRTI,
\]
so \( TI(1) = A \) and \( TI(NRTI+1) = B \).
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: \( A = TI(1) < \ldots < TI(1) = B \) or \( A = TI(1) > \ldots > TI(1) = B \) (1 denotes the total number of required output-points).
On exit: \( TI(1), i = 1, 2, \ldots, NRTI, \) contains the output-points and \( TI(NRTI+KE) \) contains the actual value used for \( \gamma \).

**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 than the total number of output-points + 3.
Unchanged on exit.

**IEXT**

INTEGER.
When calling DMUTSI for the first time IEXT must be zero.
When the extension interval \([B,C]\) is too small, a new call to DMUTSI with IEXT=1, and a new value for C results in the computation of a new solution with the new value of C.
Unchanged on exit.

**X**

DOUBLE PRECISION array of dimension \((N,NTI,N)\).
On exit \( X(i,k,j) \), \( i = 1, 2, \ldots, N \) contains the solution of the BVP at the output-point \( TI(k) \), \( k = 1, \ldots, NRTI \). If there is no unique solution the base of the manifold is given in \( X(i,k,j) \), \( j = 2, \ldots, NRSOL \).

**NRSOL**

INTEGER.
On exit NRSOL contains the information concerning the uniqueness of the solution. If NRSOL=1 then the solution is unique, else the solution of the problem is a manifold for which the base is given in \( X(i,k,j) \), \( j = 2, \ldots, NRSOL \).

**U**

DOUBLE PRECISION array of dimension \((NU,NTI)\).
On exit \( U(i,k) \), \( i = 1, 2, \ldots, NU \) contains the relevant elements of the upper triangular matrix \( U_k \), \( k = 2, \ldots, NRTI \). The elements are stored column wise, the \( j \)th column of \( U_k \) is stored in \( U(nj+1,k), U(nj+2,k), \ldots, U(nj+j,k) \), where \( nj = (j-1) \times j / 2 \).

**NU**

INTEGER.
NU is one of the dimensions of U and PHIREC.
DMUTSI

NU must be at least equal to N * (N+1) / 2. Unchanged on exit.

Q DOUBLE PRECISION array of dimension (N,W,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 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, of the multiple shooting recursion.

KU INTEGER.
On exit KU is the number of detected unbounded growing modes on the interval [a,y]. Growing modes with an increment greater than 2 are considered to be unbounded growing modes.

KE INTEGER.
On exit KE contains the detected number of exponentially growing modes on the interval [β,γ]. Growing modes are con­sidered to be exponentially increasing when there increment on the interval [β,γ] is greater than 1./ER(2).

KPART INTEGER.
On exit KPART contains the global k-partition of the upper triangular matrices U_k.

PHIREC DOUBLE PRECISION array of dimension (NU,NTI).
On exit PHIREC 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. LW > 8*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 > 3*N. Unchanged on exit.

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 NRTI < 0 or NTI < 5 or NU < N * (N+1) / 2 or A = B.  
  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 < 8*N + 2*N*N or LIW < 3*N.  
  TERMINAL ERROR.

IERROR = 108  
  INPUT ERROR: either A > B and C < B, or A < B and C > B.  
  TERMINAL ERROR.

IERROR = 109  
  INPUT ERROR: routine 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 last call to the routine (stored in TI(NRTI+KE)) if A is less (greater) than B.  
  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-N-3.  
  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 = 201  
  This indicates that it was not possible to compute an ordered upper triangular matrix U₂. Most probably the homogeneous part of the ODE has solutions which neither increase nor decrease. Check the amplification factor.  
  WARNING ERROR.

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

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

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

IERRO = 230 This indicates that it was impossible to order the product of the \( U_k \) (\( k = 2, \ldots, \text{NETI} \)). Most probably the homogeneous part of the ODE has solutions which neither increase nor decrease. Check the amplification factor.
WARNING ERROR.

IERRO = 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.

IERRO = 250 This indicates that one of the \( U_k \) is singular.
TERMINAL ERROR.

IERRO = 260 This indicates that the problem is ill-conditioned with respect to the BC (see §4.2.).
WARNING ERROR.

IERRO = 263 The computed value for \( \gamma \) is larger than the given maximum value for \( \gamma \) in C. Output value is the estimated value for \( \gamma \).
The given value for \( \gamma_{\text{max}} \) is used for further computations.
WARNING ERROR.

IERRO = 264 The computed number of unbounded growing modes on the interval \([\alpha, \beta]\) differs from the compute number on the interval \([\alpha, \gamma]\). This might be caused by a very slowly increasing mode, or the problem is not dichotomic.
WARNING ERROR.

IERRO = 265 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.

IERRO = 270 The problem is inconsistent. This may be caused by non exponentially increasing modes, or by the BC. The problem is inconsistent. If also error 265 has occurred then most probably this error is caused by error 265. It is also possible that the BC are inconsistent, however a larger value of B may solve this problem.

-- 15 --
WARNING ERROR.

IERROR = 280  It is not possible to compute the SVD within 30 iterations.
TERMINAL ERROR.

***************
Auxiliary Routines
***************

This routine calls the BOUNDPAK library routines DDURI, DDURT, DGTR, DCEXIN,
DKPCH, DFUNRC, DPSR, DBCMAV, DSV.

***************
Remarks
***************

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

***************
Method
***************

See chapter II of BOUNDPAK User's Manual

***************
Example of the use of DMUTSI
***************

Consider the ordinary differential equation
\[ \frac{dx(t)}{dt} = \begin{bmatrix} 1 & -1-0.2t \\ 0 & -0.2t \end{bmatrix} + \begin{bmatrix} 0.2t \\ 0.2t \end{bmatrix}, \quad t > 0. \]

and a boundary condition
\[ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(0) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x(\infty) = \begin{bmatrix} 2 \\ 1 \end{bmatrix}. \]

The solution of this problem is: \[ x(t) = \left(1 + \exp(-0.1t^2), 1 + \exp(-0.1t^2)\right)^T \]

In the next program the solution is computed, with \( B=5 \) and \( C=15 \), and compared to the exact solution. The exact solution. This program has been run on a AS9140 VM/CMS computer.
EXAMPLE PROGRAM FOR DMUTSI.

DOUBLE PRECISION A,B,C,MA(2,2),MINF(2,2),BCV(2),AMP,ER(5),TI(16),
1 X(2,16,2),U(3,16),Q(2,2,16),D(2,16),PHIREC(3,16),
2 W(24),XEX,ERR

INTEGER IW(6)
EXTERNAL FLIN,FDIF

SETTING OF THE INPUT PARAMETERS

N = 2
IHOM = 1
A = 0.DO
B = 10.DO
C = 20.DO
ER(1) = 1.1D-12
ER(2) = 1.D-4
ER(3) = 2.2D-15
NRTI = 10
NTI = 16
IEXT = 0
NU = 3
LW = 24
LIW = 6

SETTING THE BC MATRICES MA AND MINF AND THE BC VECTOR BCV

MA(1,1) = 0.DO
MA(1,2) = 1.DO
MA(2,1) = 0.DO
MA(2,2) = 0.DO
MINF(1,1) = 0.DO
MINF(1,2) = 0.DO
MINF(2,1) = 1.DO
MINF(2,2) = 0.DO
BCV(1) = 2.DO
BCV(2) = 1.DO

CALL TO DMUTSI

CALL DMUTSI(FLIN,FDIF,N,IHOM,A,B,C,MA,MINF,BCV,AMP,ER,NRTI,NI,NT1,
1 IEXT,X,NRSOL,U,NU,Q,D,KU,KE,KPART,PHIREC,W,LW,IW,LIW,
2 IERROR)

IF (((IERROR.EQ.0).OR.((IERROR.EQ.200).AND.(IERROR.LE.213))).OR.
2 (IERROR.LE.270))))) THEN

PRINTING A, B, THE ACTUAL USED VALUE FOR GAMMA, TOLERANCE,
CONDITION NUMBER AND AMPLIFICATION FACTOR.

WRITE(6,100) A,B,TI(NRTI+KE),ER(2),ER(4),ER(5)

COMPUTATION OF THE ABSOLUTE ERROR IN THE SOLUTION AND PRINTING
THE SOLUTION AT THE OUTPUT POINTS.
C
WRITE(6,110)
DO 1100 K = 1 , NRTI
  XEX = 1.DO + DEXP(-0.1DO*TI(K)*TI(K))
  ERR = XEX - X(1,K,1)
WRITE(6,120) TI(K),X(1,K,1),XEX,ERR
  ERR = XEX - X(2,K,1)
WRITE(6,130) X(2,K,1),XEX,ERR
1100 CONTINUE
IF (NRSOL.GT.1) THEN
WRITE(6,140)
DO 1200 K = 1 , NRTI
WRITE(6,150) TI(K),X(1,K,2)
WRITE(6,160) X(2,K,2)
1200 CONTINUE
END IF
ENDIF NRSOL
ELSE
WRITE(6,300) IERROR
ENDIF IERROR
C
100 FORMAT(' A = ',D12.5,2X,'B = ',D12.5,2X,'C = ',D12.5,/, ' TOL = ',
     1 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(' ',F7.3,3(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 DMUTSI: IERROR = ',I3)
C
STOP
END
SUBROUTINE FLIN(T,Y,F)
----------------------
C
DOUBLE PRECISION T,Y(2),F(2)
----------------------
C
F(1) = 1.DO * Y(1) - (1.DO + 0.2DO*T) * Y(2)
F(2) = -0.2DO*T * Y(2)
RETURN
END
SUBROUTINE FDIF(T,Y,F)
----------------------
C
DOUBLE PRECISION T,Y(2),F(2)
----------------------
C
CALL FLIN(T,Y,F)
F(1) = F(1) + 0.2DO * T
F(2) = F(2) + 0.2DO * T
RETURN
END
\[
A = 0.000000D+00 \quad B = 0.10000D+02 \quad C = 0.19306D+02 \\
TOL = 0.10000D-05 \quad COND = 0.10000D+01 \quad AMPLI = 0.24624D+01
\]

<table>
<thead>
<tr>
<th>T</th>
<th>X APPROX</th>
<th>X EXACT</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.200000000D+01</td>
<td>0.200000000D+01</td>
<td>0.206463469D-08</td>
</tr>
<tr>
<td>1.000</td>
<td>0.190483284D+01</td>
<td>0.190483742D+01</td>
<td>0.457679277D-05</td>
</tr>
<tr>
<td>2.000</td>
<td>0.167031885D+01</td>
<td>0.167032005D+01</td>
<td>0.119904925D-05</td>
</tr>
<tr>
<td>3.000</td>
<td>0.140657177D+01</td>
<td>0.140656966D+01</td>
<td>-0.211390106D-05</td>
</tr>
<tr>
<td>4.000</td>
<td>0.120188937D+01</td>
<td>0.120189652D+01</td>
<td>0.714315006D-05</td>
</tr>
<tr>
<td>5.000</td>
<td>0.108208088D+01</td>
<td>0.108208500D+01</td>
<td>0.411750430D-05</td>
</tr>
<tr>
<td>6.000</td>
<td>0.096188149D+01</td>
<td>0.096189500D+01</td>
<td>0.350542931D-05</td>
</tr>
<tr>
<td>7.000</td>
<td>0.084020011D+01</td>
<td>0.084021372D+01</td>
<td>0.20437011D-05</td>
</tr>
<tr>
<td>8.000</td>
<td>0.071764951D+01</td>
<td>0.071766658D+01</td>
<td>0.126991370D-05</td>
</tr>
<tr>
<td>9.000</td>
<td>0.059318829D+01</td>
<td>0.059319652D+01</td>
<td>0.126667628D-04</td>
</tr>
<tr>
<td>10.00</td>
<td>0.046772703D+01</td>
<td>0.046774372D+01</td>
<td>0.373188500D-06</td>
</tr>
<tr>
<td></td>
<td>0.099995454D+00</td>
<td>0.100004540D+00</td>
<td>0.905871349D-04</td>
</tr>
<tr>
<td></td>
<td>0.100005438D+01</td>
<td>0.100025440D+01</td>
<td>0.189498361D-07</td>
</tr>
</tbody>
</table>