Numerical methods for the solution of ill conditioned linear boundary value problems

H.W Locksley

"NUMERICAL METHODS FOR THE SOLUTION OF ILL CONDITIONED LINEAR BOUNDARY VALUE PROBLEMS"

H.W.Locksley

This thesis deals with the numerical solution of linear boundary value problems in ordinary differential equations, and it concentrates particularly on the practical numerical difficulties encountered in the solution of ill conditioned such problems.

The chief methods available for the numerical solution of well conditioned problems are described followed by a discussion of the nature of ill conditioning. The main section of the thesis is then concerned with the techniques proposed for tackling ill conditioned problems. These methods are illustrated by the numerical solution of chosen test problems by means of microcomputer programs, written especially by the author for that purpose, and the final section contains his conclusions based on this numerical experience.

CONTENTS

INTRODUCTION

In recent years, the numerical solution of initial value problems has been extensively researched, but the corresponding treatment of boundary value problems, particularly as regards practical implementation, has to date been comparatively sparse. In this thesis, we concentrate on the numerical solution of one class of boundary value problems viz. those which are linear and for which the system differential equations are ordinary, and we focus in particular on the numerical difficulties created by ill conditioned such problems.

The contents are divided into five sections (I-V), followed by an Appendix (VI) and a reference list. In I we outline the chief methods applicable for the numerical solution of well conditioned problems, followed in II by a discussion of the nature of ill conditioning as it applies to such problems. Then in III, we develop the techniques proposed to deal with ill conditioned problems, these methods being sophistications of those described in I for well conditioned problems. Section IV is devoted to practical numerical experience in solving selected test problems by means of programs written for and run on a microcomputer. Obviously, the degree of difficulty of the problems tackled here and the accuracy of the numerical results obtained were limited by the numerical accuracy of the computer itself. However, the practical difficulties encountered would be similar to those which would have to be faced if attempting to solve more ill conditioned problems using a much more accurate main-frame computer. In this respect, therefore, it is hoped that the practical work of this section, and the conclusions drawn in the following section V, will be useful in a wider field.

No attempt has been made in the text to distinguish typographically between matrices and vectors and scalars, but whenever a matrix or vector is introduced its dimensions are given. Also, references such as (6) refer to the reference list at the end, whereas ones like (3.2) refer to the relevant subsection of the Appendix (VI). Finally, equations are numbered from (i) in each subsection. An equation reference number thus refers to the one in the current subsection, unless otherwise indicated.

I. STATEMENT OF THE PROBLEM

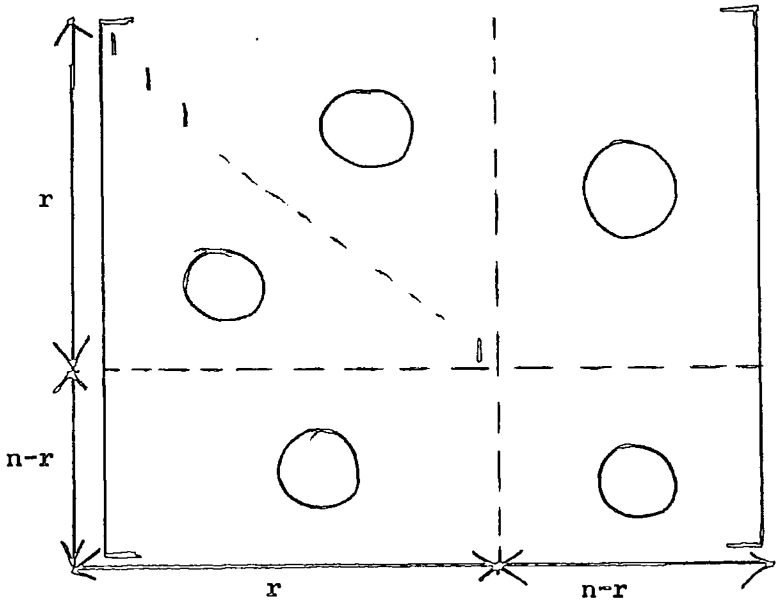
We restrict ourselves throughout to the consideration of linear two point boundary value problems for which the system of ordinary differential equations can be written in the form $\dot{y} = A(t). y + f(t)$, $0 \leqslant t \leqslant 1$, where t is the independent variable, A is $(n x n)$ and y and f are both $(n \times 1)$. (There is no loss of generality in stipulating therange $[o,1]$ because any system can be transformed to this interval by a change of variable.) Also we shall seek a solution vector y(t) of this system which satisfies linear "separated" boundary conditions i.e. where r components are known at t = 0 and n-r components are known at t = 1, but where some components may be known at both ends and some at neither. For convenience, we shall number the components known at t = 0 as $y_1, y_2, \ldots. y_r$ so the boundary conditions could be written

$$y_i(o) = \alpha_i \qquad (i = 1, \ldots\ldots r)$$

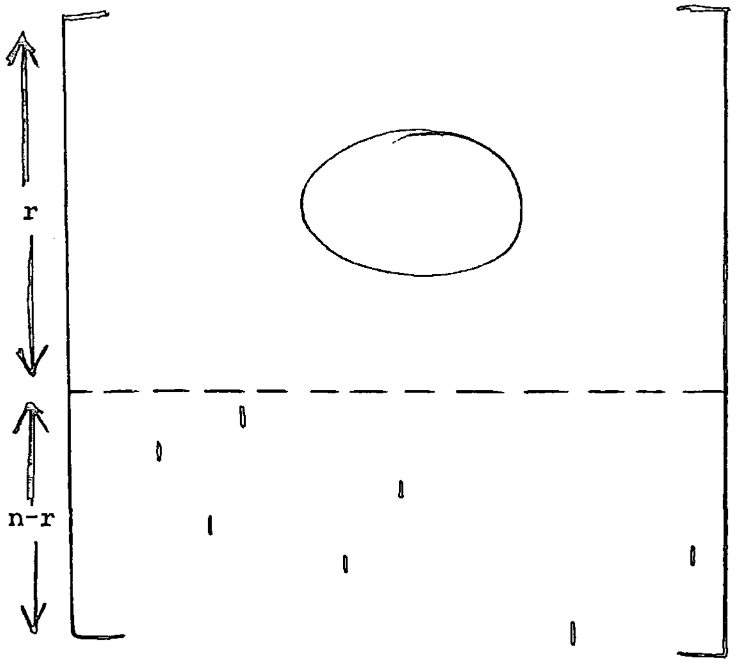
$$y_{j-p}(1) = \beta_j \qquad (j = r + 1, \ldots\ldots n)$$

where p takes $(n-r)$ values each in the range $0 \leq p \leq r$, and $n-r \leq r$. These could be more concisely expressed as : $L_0 y(0) + L_1 y(1) = C$, where L_0 and L_1 are each $(n \times n)$, C is $(n \times 1)$ and $C^T = (\alpha_1, \ldots\ldots \alpha_r, \beta_{r+1} \ldots. \beta_n)$.
The matrices L_0 and L_1 would then have the forms:

L_0 =



$L_1 =$



Thus our linear two point boundary value problem can be stated as:

$$\left.\begin{array}{lr} \dot{y} = A(t)y + f(t) & \text{(i)} \\ L_o\, y(o) + L_1 y(1) = C & \text{(ii)} \end{array}\right\} \text{I} \qquad o \leq t \leq 1$$

and we will assume throughout that this problem possesses a unique solution vector y(t). (This assumption is essential here since any given boundary value problem may possess no solution or an infinity of solutions.) Note that system I(i) above includes as a special case any nth order linear differential equation, since such an equation can always be expressed as an equivalent system of n simultaneous linear first order differential equationS(1.2). Finally, we may point out that although we are restricting ourselves to the solution of linear problems, the methods which follow are applicable to non-linear problems in so far as any such problem can be reduced to a sequence of linear problems by adopting the process of quasi-linearisation (1.3).

Applicable methods of solution

The chief methods that are available for the numerical solution of our linear two point boundary value problem I can be divided into three categories viz.

A) the method of linear adjoints, otherwise known as the Goodman-Lance method.

B) the superposition methods

C) The Riccati Transformation method or the method of invariant imbedding.

We now briefly outline the application of each of these methods:

A) Linear adjoints method

Corresponding to the system I(i) : $\dot{y} = Ay + f$, we define the adjoint system $\dot{x} = -A^T x$ (iii), where x is $(n \times 1)$. It can now be shown (1.4) that these two systems are connected by the fundamental adjoint identity:

$$x^T(1) \cdot y(1) - x^T(o) \cdot y(o) = \int_o^1 x^T \cdot f \cdot dt \qquad \text{(iv)},$$

which is satisfied by any consistent y(o), y(l) pair and any consistent x(o), x(l) pair. We now proceed to find the n-r missing initial values (at t = o) viz. $y_{r+1}(o), \ldots\ldots y_n(o)$, necessary for the solution of the problem, as follows.

Integrate the adjoint system (iii) backwards from t = 1 to t = o, n-r times, taking the starting vector x(l) each time to be a Kronecker delta vector of the form $(000100\ldots\ldots 0)^T$, where the position of the 1 corresponds successively to the position of the known end values viz. $\beta_{r+1}, \beta_{r+2} \ldots\ldots \beta_n$, so that the vector product $x^T(1) \cdot y(1)$ in (iv) successively takes the values $\beta_{r+1}, \beta_{r+2} \ldots\ldots\ldots \beta_n$. For each integration, the vector x(t) is stored at discrete points over the range $[o,1]$, so that in each case the value of the integral term $\int_o^1 x^T(t) \cdot f(t) \cdot dt$ in (iv) can be obtained approximately by numerical integration by using, say, Simpson's Rule. Also each time the term $x^T(o) \cdot y(o)$ in (iv) can be split into two terms thus:

$$[x_1(o) \ldots\ldots x_r(o)] \cdot \begin{bmatrix} y_1(o) \\ \vdots \\ y_r(o) \end{bmatrix} + [x_{r+1}(o) \ldots\ldots\ldots x_n(o)] \cdot \begin{bmatrix} y_{r+1}(o) \\ \vdots \\ y_n(o) \end{bmatrix}$$

in which the only unknowns will be $y_{r+1}(o)\ldots\ldots y_n(o)$. Therefore, from each backward integration of the adjoint set (iii), we can obtain from the fundamental identity (iv), an equation of the form:

$$\left[x_{r+1}(o)\ \ldots\ldots\ x_n(o)\right] \circ \begin{bmatrix} y_{r+1}(o) \\ | \\ \vdots \\ y_n(o) \end{bmatrix} = \mathcal{D}_K \quad \text{, where}$$

$\mathcal{D}_K$ (K = 1, ..., n-r) is known i.e. one linear equation involving the n-r unknowns $y_{r+1}(o)\ldots\ldots y_n(o)$. Thus, in total, the n-r backward integrations will supply us with n-r such linear equations which can be written in matrix form as : $B.Z = \mathcal{D}$, where B is (n-r) x (n-r), $\mathcal{D}$ is (n-r) x 1 and both are known, and $Z = (y_{r+1}(o)\ \ldots\ldots\ y_n(o))^T$.

Hence Z can be found provided matrix B is non singular, which will be so (in theory) if we assume that our problem I possesses a unique solution. The vector y(o) is now known and so theoretically it can be used to integrate the system $\mathring{y} = Ay + f$ (I i) forwards from t = o to t = 1, so obtaining the solution vector y(t) of our problem over the complete range, as required.

In practice, however, if problem I is "ill conditioned", in the manner described later in section II, then this last mentioned integration may not be possible over $[o,1]$ and alternative techniques may have to be adopted. Another practical numerical difficulty which may arise in such cases is that the resulting solution matrix B may be ill conditioned i.e. although theoretically non-singular, the value of the normalised determinant of B may be so nearly zero as to make the accurate computation of B^{-1} (and therefore of Z) very difficult. The fact that the (n-r) initial Kronecker delta vectors $x^T(1)$ are orthogonal and therefore linearly independent at t = 1 guarantees, in theory, that the resulting vectors $x^T(t)$, obtained by backward integration, will be independent for any other value of t, and in particular that the set $x^T(o)$ will be independent. This in turn should guarantee the independence of the row vectors of matrix B. But, in practice, due to round off errors in the calculation process, this may not be so and B may turn out to be almost singular. In sections II and III we therefore discuss in detail the possible numerical difficulties which may be encountered in practice and some of the techniques we can adopt to overcome them.

Finally, we may note that in the adjoint method any set of linearly independent vectors x(1) could be used to start the backward integrations of system (iii), but this would require n integrations resulting in n linear equations to be solved for n unknowns : the (n-r) missing initial values at t = o plus also the r unknown values at t = 1. By taking the x(1) vectors to be the Kronecker delta vectors we reduce the number of equations to be solved to n-r, a considerable saving in computational effort if n is large.

B) Superposition methods

The underlying idea of all of these methods is that we seek to express the general solution vector y(t) of system $\dot{y} = Ay + f$ (I i) in the form:

$$y(t) = \sum_{K=1}^{n} b_K \, y^K(t) \quad , \text{ for } o \leqslant t \leqslant 1,$$

where the $\{y_K(t)\}$ are a constructed set of n linearly independent vectors. The n constants $\{b_K\}$ are then determined by applying the given boundary conditions:

$L_o \circ y(o) + L_1 \circ y(1) = C$, so as to determine the required solution vector y(t) of our problem over $[o,1]$. Several alternatives exist for generating the set $\{y_K(t)\}$ but we shall concentrate on two of the most popular methods viz.

(a) the variation of parameters method

and (b) the complementary function method,

the details of each of which now follow.

a) Variation of parameters method

Here we express the general solution vector y(t) of the system $\dot{y} = Ay + f$ as a linear combination of n linearly independent solution vectors of the corresponding homogeneous system $\dot{y} = Ay$ plus one particular solution vector of the given inhomogeneous system $\dot{y} = Ay + f$. First we integrate the system $\dot{y} = Ay$ forwards over $[o,1]$ n times starting each time at t = o from a different Kronecker delta vector, and storing the resulting solution vectors $u_i(t)$, $i = 1 \dots n$, in the columns of the fundamental matrix N(t), for $o \leqslant t \leqslant 1$, so that N(o) = I, the $(n \times n)$ identity matrix. Then we integrate the inhomogeneous system forwards over $[o,1]$ once starting at t = o from the null vector, and storing the resulting solution in vector W(t), $(n \times 1)$, where W(o) = o. Hence the general solution y(t) of system $\dot{y} = Ay + f$ can be written:

$$y(t) = N(t) . B + W(t), \quad o \leqslant t \leqslant 1 \qquad \text{(i), where}$$

N(o) = I, W(o) = o and B(nxl) is the vector of combination constants to be determined so as to satisfy the given boundary conditions of the problem viz. $L_o . y(o) + L_1 . y(1) = C$ (ii). Note that putting t = o in (i) gives us $B = y(o)$, so that (i) becomes : $y(t) = N(t). y(o) + W(t)$. Also putting t = 1 in this last equation now gives: $y(1) = N(1).y(o) + W(1)$, and substituting this into (ii) we obtain:

$$L_o . y(o) + L_1 \left[N(1) . y(o) + W(1) \right] = C$$

$$\Rightarrow \quad \left[L_o + L_1 \ N(1) \right] . y(o) = C - L_1 . W(1)$$

or $M.B = T$ where $M = L_o + L_1 \ N(1)$, (nxn)

$T = C - L_1 \ W(1)$, $(nx1)$

and $B = y(o)$.

Since M and T are now known, vector B can be found from $B = M^{-1} T$ and this can then be substituted back into (i) to give us the solution vector y(t), required for our problem I, at each stored point in [o,1].

In practice, however, if the solution matrix M is ill conditioned, as is likely to be the case if our original problem I was sensitive (see II), then the accurate solution of the set of linear equations $M\beta = T$ may prove difficult, and so to improve the condition number of matrix M we may have to resort to multiple shooting (see III), based on this variation of parameters method. Finally, we may note that this method requires, in total, n + 1 integrations of the homogeneous and inhomogeneous systems of equations.

b) Complementary function method

This is essentially a simplification of (a) whereby the total number of integrations required is reduced to (n-r+1), a considerable saving if n is large. We first integrate the homogeneous system $\dot{y} = Ay$ forwards over [o,1], n-r times starting each time at t = o from a different special Kronecker delta vector, these being vectors whose components are all zero except for a 1 in the $(r+1)^{th}$, $(r+2)^{th}$....... n^{th}, position successively e.g. for n = 6, r = 3 these initial vectors would be:

$$\begin{bmatrix} o \\ o \\ o \\ 1 \\ o \\ o \end{bmatrix}, \quad \begin{bmatrix} o \\ o \\ o \\ o \\ 1 \\ o \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} o \\ o \\ o \\ o \\ o \\ 1 \end{bmatrix}.$$

The calculated values of these homogeneous solution vectors are stored at discrete points over [o,1] in the last (n-r) columns of the fundamental matrix N(t), which is (n x n) of which the first r columns are zero for all

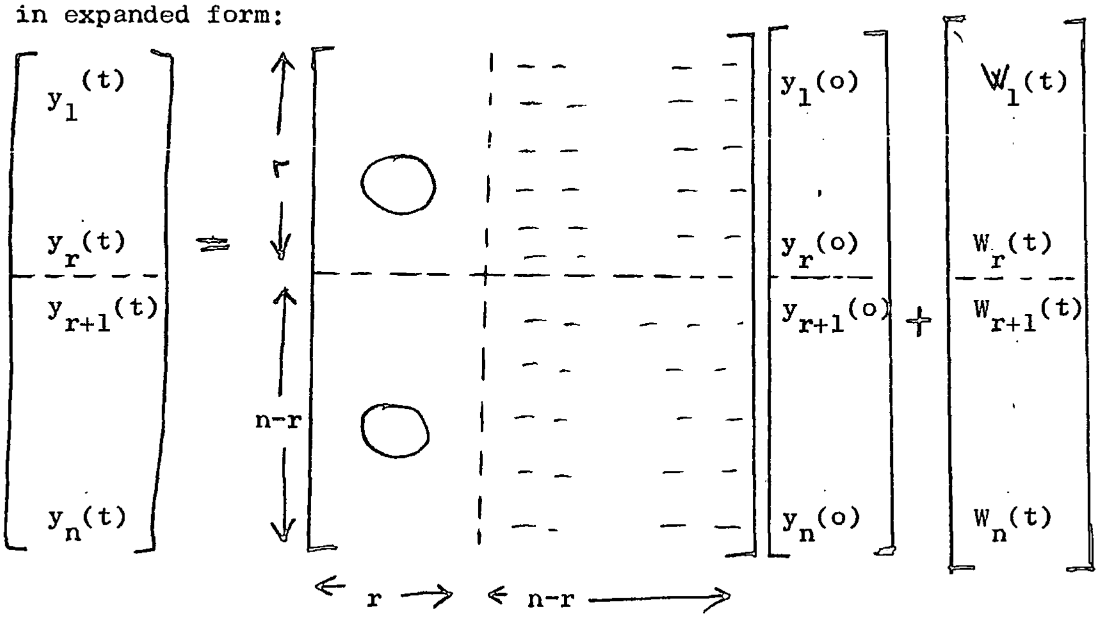
t. The inhomogeneous system is integrated forwards once over $[o,1]$ starting at t = o from vector W_o where

$$W_o = \left[\alpha_1, \alpha_2, \ldots\ldots \alpha_r, \quad 0, 0 \ldots\ldots\ldots\ldots 0\right]^T,$$

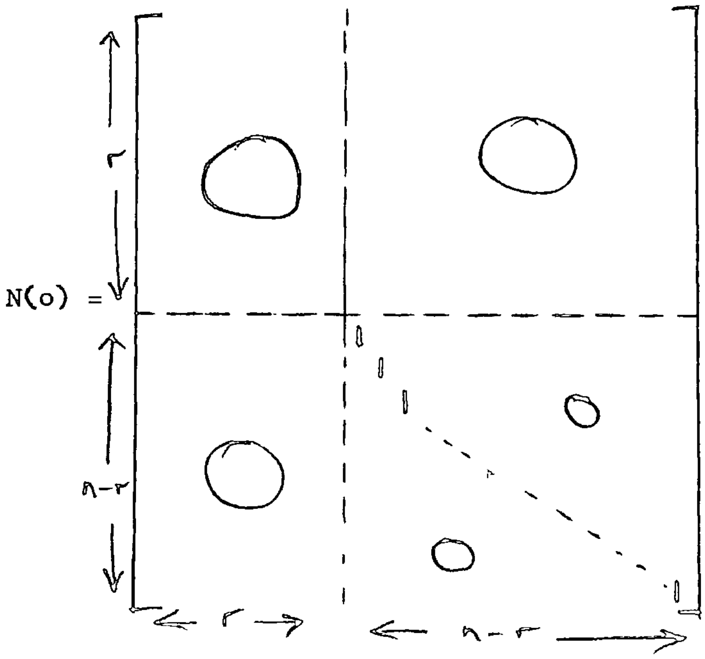
where α_i, i = 1.....r, are the given initial values. This inhomogeneous solution is stored in vector W(t), $0 \leq t \leq 1$, so that $W(o) = W_o$. Now, as in (a) above, the general solution vector y(t) of the system $\dot{y} = Ay + f$ can be written:

$$y(t) = N(t).\ y(o) + W(t), \quad o \leq t \leq 1,$$

or in expanded form:



where



and

$$W(o) = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_r \\ \hline 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

As in (a) we now determine the combination vector y(o) by applying the boundary conditions given in the problem, leading us again to the solution of a matrix equation of the form MB = T, for B = y(o). The working is identical to that in (a) except that now we have different definitions for our fundamental matrix N(t) and the inhomogeneous initial vector W(o). Also analagous to (a), if the resulting solution matrix M is ill conditioned then we can adopt a more sophisticated technique, based on this complementary function method, known as Conte's Re-orthonormalisation method (see III).

Note that in both superposition method (a) and (b) the final calculation of y(o) depends only on the terminal values N(1) and W(1) of the homogeneous matrix and the inhomogeneous vector respectively. Thus, the required memory capacity could be reduced by storing the values of the homogeneous and inhomogeneous vectors of integration only at the terminal point t = 1, and then obtaining the solution vector y(t) of the problem by integrating the system $\dot{y} = Ay + f$ forwards over $[o,1]$, starting at t = o from the calculated vector B = y(o). The drawback of this approach is that for sensitive problems this last mentioned integration is unlikely to be possible over the full interval, in which case we are compelled to save the values of the calculated homogeneous and inhomogeneous vectors at discrete points throughout $[o,1]$ and then use these to construct y(t) from y(t) = N(t).y(o) + W(t), at these storage points.

Finally, we may note that these superposition methods are closely related to the adjoint method (A). In fact, starting from the variation of parameters formula for the general solution of the system $\dot{y} = Ay + f$, it is possible (1.5) to derive the fundamental adjoint identity by an alternative proof to that given in (1.4).

c) The Riccati Transformation method

This method is also known as the method of invariant imbedding because we imbed our given problem I in a family of related problems. In order to do this, we must first re-write our given system of equations $\dot{y} = Ay + f$

(I i) in what is known as characteristic form, as follows.

Consider the n^{th} order system of differential equations:

$$\dot{y}\ (t) = E(t).\ y + h(t) \qquad \text{(i)}$$

where E is $(n \times n)$ and y and h are $(n \times 1)$. This system can now be expressed as an $(m + p)^{th}$ order system by arbitarily splitting the components of y into two characteristic vectors of dimensions $(m \times 1)$ and $(p \times 1)$ where $m + p = n$.

Thus, for example, if we choose

$u_i = y_i$ ($i = 1 \ldots m$) and

$x_i = y_{m+i}$ $(i = 1 \ldots p)$ then:

$$\begin{bmatrix} y_1 \ldots y_n \end{bmatrix}^T = \begin{bmatrix} u_1 \ldots u_m\ o \ldots o \end{bmatrix}^T + \begin{bmatrix} o \ldots o\ x_1 \ldots x_p \end{bmatrix}^T,$$

where $u = \begin{bmatrix} u_1 \ldots u_m \end{bmatrix}^T$ is then called the surface characteristic vector and $x = \begin{bmatrix} x_1 \ldots x_p \end{bmatrix}^T$ is the base characteristic vector. System (i) above can now be written in characteristic form as:

$$\left.\begin{aligned} \dot{u}\ (t) &= A(t)u + B(t)x + F(t) \\ \dot{x}\ (t) &= C(t)u + D(t)x + G(t) \end{aligned}\right\} \qquad \begin{aligned} &\text{(ii)} \quad (m \times 1) \\ &\text{(iii)} \quad (p \times 1), \end{aligned}$$

where A $(m \times m)$, B $(m \times p)$, C $(p \times m)$ and D $(p \times p)$ are the characteristic matrices, and F $(m \times 1)$ and G $(p \times 1)$ are the source vectors, all of which will depend on the original choice of surface and base vectors. (Do not confuse characteristic matrix A with the problem matrix A $(n \times n)$ of our original problem I.). An example of this re-casting of a system of equationS into characteristic form is to be found in the solved test problems in section IV, example 3. More concisely, equations (ii) and (iii) could be expressed in giant matrix form as:

$$\begin{bmatrix} \mathring{u} \\ \mathring{x} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} . \begin{bmatrix} u \\ x \end{bmatrix} + \begin{bmatrix} F \\ G \end{bmatrix} .$$

We can now show (1.6) that, for any $t \geqslant o$, the characteristic vectors $u(t)$ and $x(t)$ are connected by the transformation:

$$u(t) = U(t).\, x(t) + v(t) \qquad \text{(iv)},$$

where the matrix U $(m \times p)$ satisfies the matrix Riccati equation:

$$\mathring{U}\ (t) = B(t) + A(t)U - UD(t) - UC(t)U \qquad \text{(v)},$$

where $U(o) = o$, and the vector $v(m \times 1)$ satisfies the vector equation:

$$\mathring{v}(t) = \Big[A(t) - U(t).\ C(t)\Big].\, v(t) - U(t)G(t) + F(t) \qquad \text{(vi)},$$

where $v(o) = u(o)$. Note that equation (v) is $(m \times p)$ and equation (vi) is $(m \times 1)$.

If we now wish to solve our problem I, let its order be n, where m values are known at $t = o$ and p values at $t = 1$, so that $m + p = n$. Let these initial and terminal known vectors be denoted by a $(m \times 1)$ and b $(p \times 1)$ respectively. Choose the m components of y which are known at $t = o$ to be the surface vector u so that the remaining p components will be the base vector x. We can now state our problem with the system equations written in characteristic form and with the boundary conditions separated:

$$\left.\begin{aligned} \mathring{u} &= A(t)u + B(t)x + F(t) \\ \mathring{x} &= C(t)u + D(t)x + G(t) \end{aligned}\right\} \qquad \text{(vii)} \qquad \text{where}$$

$u(o) = \alpha$ and $g_1\ u(1) + g_2\ x(1) = b$, where g_1 is $(p \times m)$ and $g_2\ (p \times p)$, and this latter condition is equivalent to $L_1\, y(1) = [0, ---- 0, \beta_{r+1} ---- \beta_n]^T$, from our original boundary condition $L_o . y(o) + L_1\, y(1) = C$.
To solve the problem now requires three integrations over $[o, 1]$, two forwards and one backwards. First we integrate the matrix Riccati equation (v) forwards starting at $t = o$ from $U(o) = o$, and store the values of $U(t)$ at discrete points. We then use these stored values to integrate forwards the $\mathring{v}$ equation (vi) starting from $v(o) = a$, and again store the values of $v(t)$. Now putting $t = 1$ in the transformation equation (iv) gives:

$u(1) = U(1)\,x(1) + v(1)$, and by substituting this into the terminal boundary equation $g_1\,u(1) + g_2\,x(1) = b$ we obtain:

$$g_1 \cdot \left[U(1)\,x(1) + v(1) \right] + g_2 \cdot x(1) = b \quad \text{and}$$

hence: $$\left[g_1\,U(1) + g_2 \right] \circ x(1) = b - g_1\,v(1)$$

or $$x(1) = \left[g_1\,U(1) + g_2 \right]^{-1} \cdot \left[b - g_1\,v(1) \right] \qquad \text{(viii).}$$

This is a $(\rho \times 1)$ equation in which the only unknown is $x(1)$, which therefore can be found. If we now substitute equation (iv):

$u(t) = U(t)\,x(t) + v(t)$ into the base equation (vii):

$\dot{x} = C(t)u + D(t)x + G(t)$ we obtain:

$$\dot{x} = \left[C(t).U(t) + D(t) \right] \cdot x(t) + C(t)\,v(t) + G(t) \qquad \text{(ix)}$$

which is $(\rho \times 1)$.

The stored values of $U(t)$ and $v(t)$ are now used to integrate this equation (ix) <u>backwards</u> over $[o,1]$, starting at $t = 1$ from the vector $x(1)$ just obtained, and the values of $x(t)$ are also stored at the discrete points. We thus now have obtained the base vector values $x(t)$ over the complete interval $[o,1]$, from which we can compute the corresponding surface vector values $u(t)$ by employing the transformation equation (iv):

$u(t) = U(t)\,x(t) + v(t)$, using the stored values of $U(t)$, $x(t)$ and $v(t)$. Finally, the solution vector $y(t)$ of our given problem is now obtained at any stored value t in $[o,1]$ by simply compounding the corresponding surface and base vector components.

Note that for well conditioned problems it may be possible to reduce the amount of memory storage capacity required by adopting an alternative strategy. After finding vector $x(1)$ from equation (viii), instead of integrating equation (ix) we could integrate the pair of characteristic equations (vii) backwards simultaneously over [o,1], starting at t = 1 from the vectors x(1) and u(1), where u(1) = U(1), x(1) + v(1), and so obtain the surface and base vectors, u(t) and x(t), at once. However, as in the case of the superposition methods, if the given problem is at all sensitive, then this simultaneous integration of (vii) may not be numerically possible over the complete interval [o,1]. Indeed, for such a problem, the forward integration of the Riccati equation (v) is also unlikely to be possible and this is the chief drawback of the Riccati method. In III, we discuss a refinement of this method, known as the Riccati Inverse (or continuation) method, which can be used for ill conditioned problems.

Finally, we may note that the Riccati equation (v) does not depend on the boundary conditions but only on the characteristic matrices A,B,C and D, and so, assuming that the equation can be integrated forwards over [o,1], this one integration will serve to solve several related problems all possessing the same system matrix but subject to varying boundary conditions.

Numerical integration

In all three of the above methods (A,B and C) for solving our problem I, it is necessary to integrate a system of linear simultaneous first order differential equations over [o,1] from a given starting vector. Indeed, each method requires several such integrations, either forwards or backwards. Many integration schemes exist to solve these initial value problems numerically but we can briefly categorise them under the headings of:

(i) linear multi-step methods, either implicit or explicit

(ii) predictor-corrector pairs

and(iii) Runge-Kutta methods

A detailed discussion of these is not relevant here as we are concentrating on the solution of boundary value problems (but see (1.8)).

Suffice to say that in section IV, where we have solved some test problems, the integration schemes used in the programs were:

a) a Runge-Kutta six stage method of order 5 (Lawson's method) and

b) a linear explicit 4 step method of order 4, for which the starter values were provided by the Runge-Kutta method in (a).

II <u>NUMERICAL DIFFICULTIES ENCOUNTERED IN THE SOLUTION OF ILL CONDITIONED PROBLEMS</u>

The success of all of the methods outlined in I for solving our problem I depends very much on our being able to integrate either the inhomogeneous system $\dot{y} = Ay + f$, or the homogeneous system $\dot{y} = Ay$, or the adjoint system $\dot{x} = -A^T . \chi$ or the Riccati equation, either backwards or forwards over the complete interval $[o,1]$ starting from various initial vectors. In all of these integrations the system matrix is either the problem matrix A(t) itself or is directly dependent on A, as in the case of the adjoint system and the Riccati equation. If the problem matrix A is what is known as "ill conditioned", "unstable" or "sensitive" (see below) then there is a danger that the method will fail for either of the following reasons:

(a) '<u>blow up</u>' : one of the integrations may blow up before the end point of the interval is reached i.e. the numbers involved in the calculation will for some value of $t < 1$ become so large as to be beyond the capacity of the computer to handle causing a breakdown in the calculation process. Most present microcomputers, for example, can store numbers only up to the order of 10^{38} approximately. All three methods outlined in I (A,B and C) are prone to this risk of 'blow up'.

(b) '<u>poor conditioning</u>' : in the case of the superposition methods (but not the Riccati method), although all of the integrations may be achieved without blow up occurring, the resulting fundamental matrix N(1) of homogeneous vectors at t = 1 may itself be badly conditioned which in turn can mean that the solution matrix $M = L_o + L_1N(1)$ is also so badly conditioned that it is impossible to obtain a reasonably accurate solution vector B from the linear simultaneous set of equations MB = T, as is required (see IB).

As stated earlier, any matrix A is said to be ill conditioned if it is almost singular i.e. if the value of its normalised determinant is almost zero. Ill conditioning is a consequence of the matrix having two or more row (or column) vectors which are almost parallel i.e. linearly dependent, and this in turn stems from the fact that the eigenvalues of the matrix are widely separated in real part. The degree of ill conditioning of a matrix can be measured by its condition number: the greater the condition number the more ill conditioned is the matrix and vice versa. Two of the most frequently used condition numbers are:

(i) $P(A) = \left| \frac{\lambda_{max}}{\lambda_{min}} \right|$ where $|\lambda_{max}|$ and $|\lambda_{min}|$

are the greatest and least eigenvalues respectively of A in mod value. If $P(A) \approx 1$ then the matrix A is well conditioned since the eigenvalues are in this case not widely spread. (Used only where all eigenvalues have the Same sign.)

(ii) $\alpha(A) = \|A\| \, . \, \|A^{-1}\|$ where

$$\|A\| = \max_{j} \sum_{i=1}^{n} |a_{ij}| \quad \text{or} \quad \max_{i} \sum_{j=1}^{n} |a_{ij}| \; .$$

From a practical point of view, if we are trying to solve the matrix equation MB = T for B, then the fact that M is ill conditioned means that relatively small errors in the calculated values of M and/or T may result in a large error in the computed value of vector B. This is because the relationship between the percentage error in B corresponding to those in M and T is given approximately by: $\text{error}_B \leq \alpha(M) \, . \left[\text{error}_M + \text{error}_T \right]$, provided the actual error in M is small (2.1). Thus if $\alpha(M)$ is high (i.e. if M is badly conditioned) then the upper bound for the error in B will be high also. To return now to problem (b) of poor conditioning, in order to see how the condition of the fundamental matrix N(1) can influence the condition of the solution matrix $M = L_o + L_1 N(1)$ we can employ a result of Gunderson (2.2) viz. that under certain conditions:

$\alpha(M) \leq K . \alpha(N(1))$ where we may usually assume that K is small compared to $\alpha(N(1))$. This indicates that if N(1) is poorly conditioned then M may also be or, to be more definite, only by ensuring that N(1) is well conditioned can we be sure that M will also be.

Although the two problems (a) and (b) of blow up and poor conditioning both stem from a poorly conditioned problem matrix A, we can draw a distinction between them. The danger of blow up is inherent in the problem itself i.e. it is really entirely due to the bad condition of matrix A and will occur sooner or later for a sufficiently large value of t, regardless of how accurate is the calculating process. However, if, for any given problem, blowup is to be avoided before t = 1 is reached, then the step length h must lie in a certain range determined by the accuracy of the computer and of the integration scheme.

The poor condition of matrix N(1), on the other hand, is basically due to 'round off' error at the various stages of the calculation. In the superposition methods, the initial vectors for the forward integrations of $\dot{y} = Ay$ are either the Kronecker delta vectors or (for the complementary function method) the special Kronecker delta vectors. In both cases, these are orthogenal and therefore linearly independent sets. Thus, <u>in theory</u>, these vectors should remain independent for all t. But in practice as the integration process proceeds from t = o, due to round off error, there is a danger that some of the vectors may gradually become almost parallel by the time t = 1 is reached. Although this problem is basically due to the lack of accuracy of the computer, it is still more liable to become serious in a problem where the problem matrix A is badly conditioned.

We may note here one advantage which the Riccati method has over the superposition methods viz. the Riccati method is not prone to poor matrix conditioning in the sense described above since the solution vector y(t) is obtained directly by combining the surface and base vectors without the need to invert a solution matrix.

We now consider precisely why an ill conditioned problem matrix A can cause blowup or make the danger of loss of independence more likely. The condition of a matrix is determined by the nature and distribution of its eigenvalues i.e. we say that A(t) is ill conditioned if for any value of t in $[o,1]$ it possesses either:

(i) an eigenvalue with a positive real part, particularly if this is large, or

(ii) an eigenvalue which greatly exceeds all the other eigenvalues in real part i.e. one which is well separated from the remainder.

For the moment let us assume that A is a constant matrix with n distinct eigenvalues.

It can be shown (2.3) that the solution vector y(t) of the initial value problem (I.V.P) : $\dot{y} = Ay + f$, $y(t_o) = \alpha$ can be written:

$$y(t) = \sum_{s=1}^{n} K_s \cdot e^{\lambda_s (t-to)} C_s + \emptyset(t)$$

where λ_s (s = 1 → n) are the eigenvalues of the system matrix A, C_s(s = 1 → n) are the corresponding eigenvectors, $\emptyset(t)$ is a particular solution vector of $\dot{y} = Ay + f$ (i.e. $\dot{\emptyset} = A\emptyset+f$ for all t), and the constants K_s (s = 1 → n) are uniquely determined by:

$$\alpha - \emptyset(t_o) = \sum_{s=1}^{n} K_s C_s \quad .$$

Likewise for the homogeneous I.V.P.

$\dot{y} = Ay$, $y(t_o) = \alpha$ the solution vector is

$$y(t) = \sum_{s=1}^{n} K_s \; e^{\lambda_s \cdot (t-to)} C_s \qquad \text{where}$$

$$\alpha = \sum_{s=1}^{n} K_s \cdot C_s \; , \quad \text{because now } \emptyset(t) \equiv o.$$

In order to solve our problem I by either the superposition or Riccati method, we must first solve several I.V.P.^{s} such as those above starting at t_o (=o) or t_f (= 1) from various initial vectors. If A is ill conditioned, then in any one of these forward or backward integrations blow up may occur, or in any pair of homogeneous integrations the solution vectors may gradually lose their independence for the following reasons:

(i) blow up: If A possesses an eigenvalue (say λ_i) with a (large) positive real part and if the corresponding constant K_i is non zero so that the term $K_i\, e^{\lambda_i(t-to)} C_i$ appears in the solution y(t), then as the homogeneous or inhomogeneous integration proceeds i.e. as t increases from o, a value of t will eventually be reached at which the components of this vector term, and therefore those of the solution y(t), will become unmanageably large. Whether this blow up occurs before the end point t = 1 is reached will depend on just how large λ_i is.

(ii) loss of independence: Suppose instead that matrix A possesses an eigenvalue (λ_i) which dominates all the other eigenvalues in the sense that $\text{Re}(\lambda_i)$ is much greater than the real part of any of the other eigenvalues. Consider two forward integrations of the homogeneous system from t_o (=o) starting respectively with initial vectors $y(t_o) = \alpha_1$, and $y(t_o) = \alpha_2$ where α_1 and α_2 are uniquely determined by $\alpha_1 = \sum_{s=1}^{n} K_s C_s$ and $\alpha_2 = \sum_{s=1}^{n} \ell_s C_s$.

Denote the respective solution vectors obtained by $y_1(t)$ and $y_2(t)$. Now if the constants K_i and ℓ_i, corresponding to λ_i, are both non zero then the term containing $e^{\lambda_i(t-to)}$ will appear in both solution vectors $y_1(t)$ and $y_2(t)$, and as the integration progresses ($t > o$), this term will gradually dominate both solutions. In theory, the other terms should ensure that $y_1(t)$ & $y_2(t)$ remain independent for all t. But in practice, due to round off error, as t increases: $y_1(t) \rightarrow K_i\, e^{\lambda_i(t-to)} C_i$ and $y_2(t) \rightarrow \ell_i\, e^{\lambda_i(t-to)} C_i$ as these terms increasingly dominate each solution. Thus the vectors $y_1(t)$ $y_2(t)$ will both gradually approach the direction of eigenvector C_i. How quickly the angle between $y_1(t)$ and $y_2(t)$ approaches zero will depend on how dominant is the eigenvalue λ_i i.e. on how badly conditioned is matrix A. Note that this difficulty can occur, (for $t > o$), even if all the eigenvalues have negative real parts e.g. suppose $\text{Re}(\lambda_i) = -0.1$ and $\text{Re}(\lambda_j) < -50$, ($j = 1 \rightarrow n$, $j \neq i$), then λ_i would still dominate the solution vectors for forward integration, but more slowly.

Although both these numerical difficulties (i) and (ii) stem from the poor condition of matrix A, they will be more evident for some choices of initial vector α than for others. If the choice of $y(o) = \alpha$ is such that the constant K_i, corresponding to the offending eigenvalue λ_i, is zero then the term containing λ_i will not appear in y(t) and so, in theory, will cause no trouble. Similarly, if K_i is sufficiently small the rate of increase of the offending term may be so reduced as to prevent it from blowing up before t = 1 is reached. For example, if A is a real symmetric matrix than its n eigenvectors C_i ($i = 1 \rightarrow n$) will be orthogonal. Thus if α is so chosen that $\alpha - \emptyset(o)$ is orthogonal (or nearly orthogonal) to eigenvector C_i, corresponding to the offending eigenvalue λ_i, i.e. if $\left[\alpha - \emptyset(o)\right] . C_i \approx 0$, then $\left[\sum_{s=1}^{n} K_s C_s\right] . C_i \approx 0 \Rightarrow K_i \approx 0$, so that the offending vector term in y(t) will be (almost) eliminated. However, in practice, even if $K_i = 0$ for the choice of the initial vector α of integration, because of 'round off' error in the calculating process, as the integration proceeds the term causing the blow up is liable to be gradually re-introduced. The more dominant is λ_i and the more inaccurate is the computer the more quickly this will happen, but its effect can be delayed by adopting an integration scheme of higher order of local accuracy or by varying the step length h so as to minimise the upper bound on the total error e, in y_c, at any value of t.

Since the superposition methods require several integrations starting from different initial vectors, then if A is ill conditioned there is a good chance that some of the integrations will run into one of these difficulties and it needs only one integration to blow up to cause the method to fail. Indeed, if the problem matrix A is very sensitive then although the system may be integrable over $\{o,1\}$ without blow up for, say, $y(o) = \alpha$, even the smallest variation in α can cause blow up. This explains why although $\dot{y} = Ay + f$ may in theory be integrable from the <u>exact</u> initial vector y(o) without blow up, this may occur if we integrate from y(o) using the <u>calculated</u> values of the missing initial values, even though these may have been found to a high degree of accuracy. Hence in the superposition methods, as mentioned earlier, it is unlikely that we will be able to save memory storage space by finding y(t) by forward integration of the system $\dot{y} = Ay + f$. Instead we will have to store values at intermediate points and re-construct y(t) from these using $y(t) = N(t).\, y(o) + W(t)$.

To demonstrate this last point about sensitivity with respect to choice of initial vector of integration, let us consider the I.V.P. : $\ddot{y} = K^2 y$, $y(o) = 1$, $\dot{y}(o) = S$ where $K = 50$ and S is to be chosen. For any choice of S, the solution curve for y is given by: $y(t) = \frac{1}{2K}\left[(K+S)e^{Kt} + (K-S)e^{-Kt}\right]$ for $t \geqslant o$, as obtained by analytical solution. If we choose $S = -50$ i.e. if we integrate forwards from $t = o$ starting from $[1,\ -50]^T$, then y(t) simply reduces to $y(t) = \frac{1}{2K}(K-S)e^{-Kt} = e^{-50t}$ and there is no difficulty, because $y(t) \approx o$ for all $t > o$. But now suppose that instead we try to integrate numerically forwards from $[1, -49.5]^T$, choosing $S = -49.5$. Now $y(t) = \frac{1}{100}\left[o.5e^{50t} + 99.5e^{-50t}\right]$ in which the first term completely dominates the second for $t > o$, so that in effect $y \approx \frac{e^{50t}}{200}$. Thus when $t = 2$, $y \approx 1.34 \times 10^{41}$, a number beyond the range of most microcomputers, so the calculation would breakdown before $t = 2$ was reached. Hence in this example only a 1% error in the estimation of one of the components of vector y(o) causes an otherwise straightforward integration to blow up. This is because here the system matrix A is $\begin{bmatrix} 0 & 1 \\ K^2 & o \end{bmatrix}$ for which the eigenvalues are $\pm K$

i.e. ± 50, so that A is very badly conditioned.

In all of the preceding, we have assumed that the problem matrix A is constant. If A(t) varies then the situation is more complicated because now the eigenvalues λ_i $(i = 1 \rightarrow n)$, the eigenvectors $C_i (i = 1 \rightarrow n)$ and the constants $K_i (i = 1 \rightarrow n)$ will all also vary with t, and the solution vector y(t), as given by $y(t) = \sum_{s=1}^{n} K_s e^{\lambda_s (t-t_0)} C_s$, will only be approximately true over a small interval Δt in which A(t) may be assumed constant. Whether blow up occurs will now depend not only on the condition of the initial matrix A(o) but also on how rapidly the condition deteriorates or improves as t increases from t = o i.e. on the sensitivity of the eigenroots of A to changes in the values of the elements of A, which depend on t.

In section III we consider the techniques which have been proposed to modify the superposition methods and the Riccati method so as to attempt to overcome the numerical difficulties explained above which can occur in practice when dealing with sensitive problems. We will also discuss, in connection with the re-orthonormalisation method, another difficulty which may cause loss of accuracy in the solution vector y(t) viz. 'build up' error or 'loss of significance', and how this method helps to minimise this error.

III PROPOSED TECHNIQUES FOR DEALING WITH SENSITIVE PROBLEMS

a) Reversal : We can reverse the statement of our problem I by assuming that t = 1 is the 'initial' point and t = o is the 'terminal' point, the direction of the integration now being from t = 1 to t = o and $C^T = \left[\beta_1 --- \beta_{n-r}, \alpha_1, --- \alpha_r\right]$ with the boundary matrices L_o and L_1 altered accordingly. We can thus use either of the superposition methods outlined in IB to solve the reverse problem, the 'initial' missing conditions now being those at t = 1. For well conditioned problems there is little advantage to be gained from doing this, the computed solution vector y(t) being virtually identical to that obtained by forward solution. But for a sensitive problem in which one of the forward integrations blows up at $t \doteq t_1$, where $t_1 < 1$, it may be possible to avoid this by reversing the problem.

Since the solution vector y(t) of the I.V.P. $\dot{y} = Ay + f$, $y(T) = \alpha$ can be written

$$y(t) = \sum_{s=1}^{n} K_s e^{\lambda_s(t-T)} C_s + \phi(t) \text{ for any } t,$$

if all of the eigenvalues λ_s ($s = 1 \rightarrow n$) have (large) positive real parts and we take T = 0 and integrate forwards (i.e. $t > o$) then the computed solution vector y(t) will, due to round off error, stray very rapidly from the exact solution because as the integration proceeds the cumulative errors in the exponential terms will build up, until blow up occurs. Indeed this is still likely to occur with forward integration even if there is just one large positive dominant eigenvalue. But if in such a case we reverse the direction of integration i.e. start at t = 1, then, putting T = 1 :

$$y(t) = \sum_{s=1}^{n} K_s e^{\lambda_s(t-1)} C_s + \phi(t) \text{ where } t < 1, \text{ and the previously}$$

large positive exponent(s) will now be negative and so these term(s) will be very small and there will be no danger of blow up. But this device will really only be effective in a case where the problem system matrix A is such that all its negative eigenvalues are small since these will become positive for reverse integration. In general, matrix A is liable to have a mixture of positive and negative eigenvalues with a dominant positive eigenvalue for forward integration and an equally dominant negative eigenvalue which will become positive when integration is reversed, so that reversal will be of no avail. Thus we cannot regard reversing the direction of integration as a general technique for dealing with sensitive problems.

b) Re-orthonormalisation (using the Gram-Schmidt process)

This technique was first proposed by Godunov[15] and then followed up by Conte[13] and others. It attempts to overcome both the numerical difficulties encountered with sensitive problems viz. blow up and loss of independence of calculated vectors obtained from the homogeneous and inhomogeneous integrations. Repeated use is made of the Gram-Schmidt process which converts any given set Y of N linearly independent vectors (each n x 1) into a corresponding set Z of N orthonormal (n x 1) vectors i.e. mutually orthogonal vectors each of unit length. The transformation (as described in (3.1))is effected by the orthonormalisation matrix P which is lower triangular whose elements p_{ij} depend on the input set Y. The output set Z of orthonormal vectors is thus given by:

$Z = YP^T$ where P^T is upper triangular (N x N)

and Z is 1 x N & Y is 1 x N, both being vectors with vector components.

We now describe Conte's method[13] of re-orthonormalisation based on the complementary function method (see IBb) and utilising the Gram-Schmidt process above. We first divide the total range of integration $[o,1]$ into m subintervals with nodes at $t_o(= o)$, t_1, t_2 ..... $t_f(= 1)$ determined according to one of the re-orthonormalisation tests outlined later. In the first subinterval $[t_o, t_1]$ we integrate forwards the (n - r) homogeneous u vectors, starting each time at $t = t_o$ from one of the special Kronecker delta vectors, and the inhomogeneous V vector starting from $\left[\alpha_1 -- \alpha_r, o --- o\right]^T$, (as in IBb), + we store these vector values at intervals of λh, where h is the step length employed, and λ is as small a multiple as the memory capacity will allow. At $t = t_1$, we use the Gram Schmidt process to convert the (n - r) linearly independent homogeneous u vectors $\left(u^i_{old} \quad (t_1) \quad , i = 1 \rightarrow n - r \right)$ into the corresponding set of orthonormal vectors $\left(u^i_{new} \quad (t_1) \quad , i = 1 \rightarrow n - r\right)$ by using the orthonormal matrix P_1, obtained from $u^i_{old} \ (t_1)$, i.e. in matrix form:

$$U_{new} = U_{old} \cdot P_1^T \quad \text{where } P_1^T \text{ is } (n-r) \times (n-r) \ ,$$

U_{new} and U_{old} are $1 \times (n-r)$ and U_{new} is the orthonormal set of vectors $u^i_{new}(t_1)$. The elements of the matrix P_1^T are stored. Also at $t = t_1$, we convert the inhomogeneous vector $V(t_1)$ obtained by forward integration (ie V_{old}) into its orthogonal complement ($V_{new}(t_1)$) by subtracting from V_{old} a certain linear combination of the orthonormal $u^i_{new} \ (t_1)$ vectors i.e

$$V_{new} = V_{old} - U_{new} \cdot W_1, \text{ where}$$

the <u>components</u> of the projection vector $W_1(n\text{-}r \times 1)$ are given by:

$$W_j = V_{old} \cdot u^j_{new} \quad , \quad j = 1 \longrightarrow n\text{-}r \ .$$

Note that this ensures that $V_{new} \cdot u^j_{new} = o$ for $j = 1 \rightarrow n\text{-}r$ i.e. V_{new} is orthogonal to each of the new orthornormal vectors. The components of vector W_1 are also saved.

In the second sub interval $\left[t_1, t_2\right]$ we now integrate forwards the (n-r) homogeneous U vectors starting at $t = t_1$ from the $u^i_{new} \ (t_1)$ vector values just found by the Gram Schmidt process. Also we integrate forwards the inhomogeneous system starting from $V_{new} \ (t_1)$ and store this vector and the homogeneous vectors at the intermediate points. At $t = t_2$, the re-orthonormalisation process is repeated i.e. we convert the old independent u^i vectors into the new orthonormal set $u^i_{new}(t_2)$ by using the orthonormalisation matrix P_2 (obtained from the $u^i_{old}(t_2)$). Likewise the inhomogeneous vector $V_{old}(t_2)$ is converted into its orthagonal complement $V_{new}(t_2)$, and the projection vector W_2 and matrix P_2 are both again stored. Then in the next sub interval we integrate forwards over $[t_2, t_3]$ starting the homogeneous integrations at t_2 from $u^i_{new}(t_2)$ and the inhomogeneous from $V_{new}(t_2)$ and store the resulting vectors.

The process is repeated, re-orthonormalising the u and v vectors at the end of each sub interval and then using these vectors each time as the starting vectors for the homogeneous and inhomogeneous integrations in the next interval and storing the orthonormalisation matrix P and the projection vector W used at each node and the u and V vectors obtained for each interval. When $t = t_f$ is reached the final re-orthonormalisation gives us $u^i_{new}(t_f)$ and $V_{new}(t_f)$ together with the values of W_{t_f} and P_{t_f} used.

As in IBb we now solve a matrix equation of the form MB = T at the terminal point t_f, by selecting from the final $u^i_{new}(t_f)$ and $V_{new}(t_f)$ vectors the (n - r) components which correspond to the position of <u>known</u> terminal values β_j (j = 1 ⟶ n-r) in the given problem, and then using them to solve for $\gamma_f = \left[\gamma_1 \cdots\cdots \gamma_{n-r}\right]^T$ the matrix equation:

$$\begin{bmatrix} u^1_1 & u^2_1 & \cdots\cdots & u^{n-r}_1 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ u^1_{n-r} & u^2_{n-r} & \cdots\cdots & u^{n-r}_{n-r} \end{bmatrix}^{-1} \begin{bmatrix} \beta_1 - V_1 \\ \vdots \\ \vdots \\ \beta_{n-r} - V_{n-r} \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \vdots \\ \vdots \\ \gamma_{n-r} \end{bmatrix}$$

where all vector values are at t = t_f. Unlike IBb, however, in this case the vector γ_f is not $y^i(o)$,(i = r+1 --> n), the vector of missing initial values for the problem. In order to find this, we must now work *backwards* through the sub intervals from t_f to t_o, using the stored projection vectors W_i and orthonormalisation matrices P_i (i = 1 → m) at each node to obtain the γ vector corresponding to *each* sub interval, from the iteration:

$$\gamma_i = P^T_{i+1} \cdot \left[\gamma_{i+1} - W_{i+1} \right] , \quad i = (m - 1) \rightarrow o$$

where $\gamma_m = \gamma_f$, $P_m^T = P^T_{t_f}$ & $W_m = W_{t_f}$.

This finally gives us γ_o, the vector corresponding to the *first* sub interval $[t_o, t_1]$, and this is in fact the vector of missing initial values for the problem.

We now construct the solution vector y(t), $o \leq t \leq 1$, for the given problem in a piece-wise continuous fashion, interval by interval. In each interval separately, the solution vector y(t) is equal to the inhomogeneous vector $V(t)$ plus a certain linear combination of the homogeneous vectors $u^i(t)$, these being the calculated vectors that were stored for that interval. The linear combination vector required is the γ vector corresponding to *that* interval i.e. for *each* interval:

$$y(t) = V(t) + U(t).\gamma \quad \text{where} \quad U(t)$$

is the homogeneous set of vectors (n x n-r).

Note that only in the first sub interval $[t_o, t_1]$ do the calculated homogeneous and inhomogeneous vectors correspond to the special initial vectors of IBb. In all of the other intervals, they are obtained by forward integration starting from the orthonormal u^i_{new} vectors and the orthogonal complement V_{new} obtained by the Gram Schmidt process. Yet it can be shown (3.2) that the piecewise solution vector y(t), defined above for each interval separately, does provide a continuous solution vector which is the true solution of the given problem over the complete interval $\left[t_o, t_f\right]$.

Memory storage space can be saved by storing only the W vector and P^T matrix at the end of each interval and the final u^i and V vectors at $t = t_f$. These will be sufficient to obtain γ_f and hence $\gamma_o = \{y_{r+1}(o) \ldots y_n(o)\}$. The solution vector y(t) for the problem could now theoretically be found by integrating the set $\dot{y} = Ay + f$ forwards from t = o since all the initial values are known. In practice, however, if the given problem is sensitive in that it cannot be solved by "single" shooting, it is unlikely that this integration can be performed without blow-up either.

The success of this method is very much dependent on the choice of the re-orthonormalisation nodes i.e. on the partition of the total interval $\{o,1\}$ into sub intervals. Various tests have been proposed to determine when re-orthonormalisation should take place so as to obtain the optimum overall computed solution vector y(t) i.e. the solution vector for which the norm of the overall error vector is least for a given step length h of integration. The common objective of all of the tests is to check at frequent intervals on the norms and directions of the computed homogeneous vectors $u^i(t)$ and to re-orthonormalise whenever blow up or loss of independence is imminent. The big drawback of most of the tests, however, is that although they are straightforward in theory, they create practical difficulties in programming and are very time consuming and therefore expensive. Some of the chief tests for optimum re-orthonormalisation are:

(i) Godunov(15) originally proposed that the eigenvalues of the problem system matrix A(t) should be frequently calculated (every few steps) and the difference between the real parts of λ_{max} and λ_{min} computed each time.

If it is noted that this difference has increased since the previous calculation then the length of the following sub interval is proportionately decreased as compared with the present sub interval, and vice versa. Thus re-orthonormalisation will occur most frequently during periods when A(t) is most ill conditioned. This test is only applicable when A(t) is variable and in any case is highly impractical, even for a small system, on account of the formidable amount of calculation required.

(ii) Conte[13] proposes that re-orthonormalisation should occur whenever the norm of any of the calculated homogeneous vectors u^i or that of the inhomogeneous vector V exceeds a pre-assigned value M. This is comparatively simple to operate but the difficulty lies in being able to fix a suitable value for M for any given problem. One possibility is to run the program with several different values of M which avoid blow up and then to average the solution vectors obtained.

(iii)Another suggestion of Conte[13] is to compute the angle between each pair of homogeneous $u^i(t)$ vectors at regular intervals. Re-orthonormalisation then takes place whenever the computed angle is noted to be less than a pre-assigned value α^o.

Again the main difficulty is to fix α for a given problem and so we must resort to averaging over several trial values of α. Also for a large system the amount of calculation involved here would be considerable and in any case the test does not guard against the possibility of more than two vectors becoming dependent.

(iv) Another alternative is to incorporate tests (ii) and (iii) i.e. to re-orthonormalise if either test fails i.e. if

$$\|u_i\| > M \quad \text{for any } i \qquad \text{or} \quad \|V\| > M$$

or if

$$\text{arcos} \left\{ \frac{u_i \circ u_j}{\|u_i\| \circ \|u_j\|} \right\} < \alpha \qquad \text{for any pair } i, j \;, \; i \neq j.$$

This would require several runs with different choices for the values of M and α each time and there is no guarantee that the average computed solution vector so obtained would be better than that obtained by using either (ii) or (iii) separately.

(v) At regular intervals, calculate the value of D, the determinant of the normalised matrix formed by the calculated homogeneous vectors $u^i(t)$. Re-orthonormalisation takes place if $/D/ < K$ where K is a pre-assigned small positive fraction. Again we would have to run with several different values of K and average the solutions, and for a large system the frequent operations of matrix normalisation and determinant calculation would be very time consuming.

Generally speaking, the best results will be obtained when the re-orthonormalisations are distributed throughout the whole interval. Obviously the more ill conditioned is the problem matrix A, the more re-orthonormalisations will be needed, but too frequent re-orthonormalisation will be self defeating because the 'round off' error introduced by the extensive matrix and vector multiplication required, in order to work backwards through the sub intervals to find γ_o from γ_f, is liable to cancel out the benefits achieved by re-orthonormalisation.

In the programs of solution of the test problems in IV we have confined ourselves throughout to re-orthonormalisation with <u>equal</u> sub intervals and have investigated the variation in the accuracy of the computed solution vector $y_c(t)$ with changes in m, the number of re-orthonormalisations, and in h, the step length employed.

As mentioned in II, the re-orthonormalisation method possesses another advantage which may make it desirable when using a superposition method viz. reduction in 'build up' error or 'loss of significance'. Suppose that the "single shooting" complementary function method (IBb) is being used and that all integrations have been achieved over $[0,1]$ without blow up occurring. Suppose also that the final matrix M is sufficiently well conditioned to solve fairly accurately for $\gamma_o = \left[\gamma_1 \ldots \gamma_{n-r}\right]^T$, the combination vector of the homogeneous vectors $u_i(t)$, $i = 1 \rightarrow n-r$, in the solution $y_c(t)$. There still remains a further difficulty which may render the computed solution $y_c(t)$ hopelessly inaccurate viz. the difficulty of recombination. The solution $y_c(t)$ is given by:

$$y_c(t) = \gamma_1 u_1(t) + \gamma_2 u_2(t) + \ldots\ldots + \gamma_{n-r} \; u_{n-r}(t) + V(t) ,$$

for $o \leq t \leq 1$. Now if the norms of the calculated $u_i(t)$ vectors are very large relative to $//y_{exact}(t)//$, for any value of t, so that the values $\gamma_1 \ldots\ldots \gamma_{n-r}$ are very small, then even a small error in the calculation of these components will be grossly magnified when multiplied by the components of the $u_i(t)$ vectors, with a consequent error in the value of $y_c(t)$. If further, the majority of the errors in the calculated values of $\gamma_1 \ldots\ldots \gamma_{n-r}$ are all in the same direction (eg all rounded down) then the cumulative effect on the calculated components of $y_c(t)$ could be very large indeed. Thus, ideally, it is desirable, if possible, to ensure that the components of the $u_i(t)$ vectors are of approximately the same order as those of $y_{exact}(t)$, for any value of t. To this end, re-orthonormalisation can be employed, the re-orthonormalisations being performed before the norms of the $u_i(t)$ vectors become too large, thereby reducing the danger of loss of significance as described above. We must, however, be careful not to go to the opposite extreme which can occur if too many re-orthonormalisations are employed, resulting in the norms of the $u_i(t)$ vectors being very small compared to that of $y_{exact}(t)$ in some sub intervals.

c) Multiple (or parallel) shooting

This method is similar to re-orthonormalisation in that we divide the total range $[o,1]$ of the problem into sub intervals according to some criterion (see later) and then in each sub interval separately we use "single" shooting to find the general solution vector which satisfies the system $\dot{y} = Ay + f$ of our problem I, for all t in that sub interval. These vectors must then be "matched up" at each of the internal nodes and with the given initial and terminal values at $t = o$ and $t = 1$ respectively, so as to achieve an overall solution vector $y(t)$ which is continuous over $[o,1]$ and which satisfies the system $\dot{y} = Ay + f$ subject to the given boundary conditions $L_o y(o) + L_1 y(1) = C$, as is required. Any superposition method can be used for the "single" shootings in the sub intervals but we will assume that method IBa is adopted so that the description which follows refers to multiple shooting based on the variation of parameters method.

Suppose that the total interval $[t_o, t_f]$ is divided into N sub intervals with nodes at $t_o(=0)$, $t_1, t_2 \ldots t_f(=1)$. As described in IBa, in each of these sub intervals separately we must integrate the homogeneous system $\dot{y} = Ay$ forwards n times starting from the Kronecker delta vectors and the inhomogeneous system $\dot{y} = Ay + f$ forwards once starting from the null vector. This gives us in each sub interval n homogeneous vectors and one inhomogeneous vector (requiring in total $N(n+1)$ integrations) from which we obtain the general solution vector for each sub interval. Now in practice this could be done by integrating the *same* set of differential equations ($\dot{y} = Ay$ for the homogeneous and $\dot{y} = Ay + f$ for the inhomogeneous) successively over *different* sub intervals viz, $[t_o, t_1]$, $[t_1, t_2], \ldots [t_{N-1}, t_f]$. However, it is more convenient for our explanation if we adopt the equivalent notation (due to Keller(11)) in which instead we integrate successively *different* sets of differential equations over the *same* interval $[o,1]$ each time. To achieve this we define a new independent variable s on each sub interval by $s = \dfrac{t - t_{i-1}}{\Delta_i}$ where $\Delta_i = t_i - t_{i-1}$

$(i = 1 \longrightarrow N)$ so that $t_{i-1} \leq t \leq t_i \Rightarrow 0 \leq s \leq 1$. We can now show (3.3) that under this change of variable the system of differential equations for each sub interval is given by:

$$y'(s) = \left[\Delta_i \, A(s)\right] .y(s) + \left[\Delta_i \, f(s)\right] \quad , \text{ where}$$

$o \leq s \leq 1$ and $i = 1 \longrightarrow N$, and $'$ denotes $\frac{d}{ds}$.

Keller's notation(11) reduces the given problem to a sequence of similar problems all defined on the same interval $[o,1]$. The advantage of this approach is that the final matrix equation to be solved can be written in a form directly analagous to that for "single" shooting given in IBa, the only difference being that now the vectors are giant vectors (i.e. with vector components) and the matrices are giant matrices (i.e. with matrix elements), as we see later. Each interval separately ($i = 1$ --- N) now has its own system matrix $A_i = \Delta_i A$, its own forcing vector $f_i = \Delta_i f$, its own homogeneous fundamental matrix N_i, its own particular solution vector W_i and hence its own general solution vector y_i. For each interval separately: from A_i we obtain $N_i(s)$ by integrating $y' = A_i y$ forwards n times from the Kronecker delta initial vectors from $s = o$ to $s = 1$. Then from A_i and f_i we obtain $W_i(s)$ by integrating the inhomogeneous set $y' = A_i y + f_i$ forwards once from $W_i(o) = o$, $o \leq s \leq 1$. Finally from $N_i(s)$ and $W_i(s)$ we obtain $y_i(s)$ from: $y_i(s) = N_i(s) . B_i + W_i(s)$, $i = 1$ --- N, $o \leq s \leq 1$, where $N_i(o) = I$ and $W_i(o) = o$, and $B_i = y_i(o)$ are the combination vectors to be found by matching at the internal nodes and at the end points. This last set of general solution vectors is analogous to $y(t) = N(t)y(o) + W(t)$ $o \leq t \leq 1$, obtained in IBa for "single" shooting.

We can now show (3.4) that, as mentioned earlier, the matrix equation to be solved in order to match up these interval general solution vectors at the internal nodes $t_1 --- t_{N-1}$ so as to obtain an overall solution vector y(t) which is continuous over $[t_o, t_f]$ and also satisfies the given initial and final conditions, can be written in a form analogous to that found in IBa for single shooting viz.

$$\bar{M} \,.\, \bar{B} = \bar{T} \quad \text{where } \bar{M} = \bar{L}_o + \bar{L}_1 \,.\, \bar{N}\,(1)$$

and $\bar{T} = \bar{C} - \bar{L}_1 \bar{W}\,(1)$.

Here, as shown in (3.4), the boundary condition matrices $\bar{L}_o$ and $\bar{L}_1$ are now giant matrices of size N x N where each element is n x n. The matrix $\bar{N}(1)$ is the overall homogeneous fundamental matrix and is also a giant (Nn x Nn) matrix. $\bar{C}$ and $\bar{W}$ (1) are giant (Nn x 1) vectors as also is $\bar{B} = [B_1 --- B_N]^T$, the combination vector of initial values for which we must solve. Having found the combination vectors $B_i (i = 1 --- N)$ from $\bar{B} = (\bar{M})^{-1} \,.\, \bar{T}$, these can then be used to compute the solution vector y(t) for each sub interval from:

$$y_i(s) = N_i(s).\, B_i + W_i(s) \quad , \qquad o \leq s \leq 1.$$

These piecewise interval solution vectors, which can be written in giant vector form as $\bar{Y}$ (s) = $(y_1(s), y_2(s) - - - - - , y_N(s))$, will now be continuous at the nodes and will satisfy the given initial and terminal boundary conditions. Therefore $\bar{Y}(s)$ is the overall required solution vector of our problem.

The success of the multiple shooting method depends very largely on how well conditioned is the matrix equation $\bar{M} \,.\, \bar{B} = \bar{T}$ i.e. on the condition number of matrix $\bar{M}$. As for re-orthonormalisation, this in turn depends on the choice of partition i.e. on the positions of the internal nodes. We mentioned in II that, for "single" shooting, only by ensuring that the homogeneous fundamental matrix N(1) is well conditioned can we be sure that the solution matrix M will be. To this end, we can employ any of the re-orthonormalisation tests described in (b) to check on the condition of N(t) at regular intervals and to insert a node whenever it is found to be deteriorating. This, in theory, would give us the optimum partition and hence the optimum solution vector y(t) overall, for any given step length h. However, Gunderson (18) has shown (3.5) that any partition for multiple shooting should always produce a better conditioned solution matrix $\bar{M}$ than the solution matrix M employed in single shooting, which implies that the solution vector y(t) obtained by any multiple shooting partition should be more accurate overall than the solution vector obtained by single shooting. But this theoretical result takes no account of the round off error inevitably encountered in practice and so, as for re-orthonormalisation, if too many sub intervals are taken then the benefits of multiple shooting are liable to be cancelled out.

Keller (11) explained the theoretical advantage of multiple shooting over single shooting by comparing the respective bounds on the error vectors. He showed that for single shooting:

$//y_c(1) - y_e(1)// \leqslant h^p . M_1 . \exp\ (K_1)$ where $y_c(1)$ and $y_e(1)$ are respectively the computed and exact solution vectors at $t = 1$, h is the step length, p the order of the integration scheme, and M_1 and K_1 constants, assuming $t_o = 0$ and $t_f = 1$. This means that the bound on the error vector at the terminal point $t = 1$ is proportional to exp (K_1). By comparison, for multiple shooting with N-sub intervals he obtained the corresponding result:

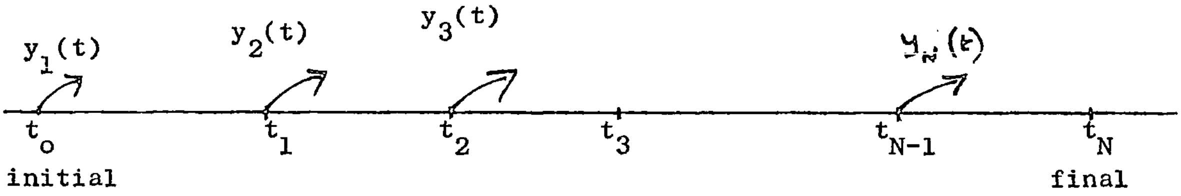
$//\bar{Y}_c(1) - \bar{Y}_e(1)// \leqslant h^p . M_2 . \exp \left(\frac{K_1}{N}\right)$ where

M_2 is constant and $\bar{Y}_c(1)$ and $\bar{Y}_e(1)$ are respectively the overall computed and exact solution vectors at the end of each sub interval ie $\bar{Y}(1) = \{y_1(1), y_2(1) \ldots \ldots y_N(1)\}$. Thus the bound on the overall error vector is proportional to $\exp\left(\frac{K_1}{N}\right)$ which means that, in theory, the accuracy of the computed solution vector, taken overall, will increase with N, which implies that the more subdivisions taken the better. But again, in practice, this is only true up to a point, because the above result takes no account of round off error. In the multiple shooting programs in IV we have confined ourselves, as in the case of re-orthonormalisation, to equal sub intervals and obtained a comparison between the two methods for the same number of sub intervals and step length for a given problem.

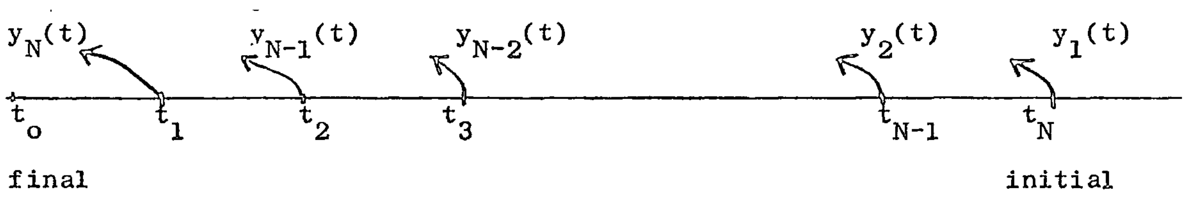
Multiple shooting is very similar to re-orthonormalisation in the way in which the total interval [o,l] is partitioned into sub intervals and the overall solution vector y(t) is formed by piecing together the separate solution vectors in the sub intervals so as to obtain a continuous solution vector which satisfies the given end conditions. But they differ in two essential respects. In multiple shooting we integrate forwards in each sub interval from the same initial vectors each time to obtain the homogeneous and inhomogeneous vectors, whereas in re-orthonormalisation we obtain at the beginning of each sub interval a different set of initial orthonormal vectors by using the Gram-Schmidt process to convert the linearly independent vectors at the end of the previous interval. Also, in re-orthonormalisation the combination vectors γ_i, required in each sub interval to obtain the corresponding solution vectors, are found iteratively by working backwards through the sub intervals and solving a sequence of matrix equations each only of size (n-r) and involving the stored values of the re-orthonormalisation matrices $\bar{P}_i$ and the projection vectors W_i. But in multiple shooting the sub interval combination vectors B_i are all found at once by the solution of one giant matrix equation $\bar{M} . \bar{B} = \bar{T}$ of size Nn. For this reason, if the problem size n is large or if a large number of sub intervals N is required, re-orthonormalisation may be preferable to multiple shooting as less memory capacity will be required and fewer calculations will be needed so that the program running time will be shorter. This was verified by the practical experience in IV in which the running times for the re-orthonormalisation programs were invariably found to be much shorter than for multiple shooting with the same number of sub intervals.

There are several methods available in practice for the numerical solution of a large set of simultaneous linear equations such as $\bar{M} \cdot \bar{B} = \bar{T}$ (see (9)), but one which does not entail the finding of the inverse $(\bar{M})^{-1}$ is preferable as this is not directly required. In the multiple shooting programs in IV we employed the method of Gaussian elimination.

The multiple shooting method described above is the 'forward shooting' type i.e. we shoot forwards over each sub interval from the initial node:

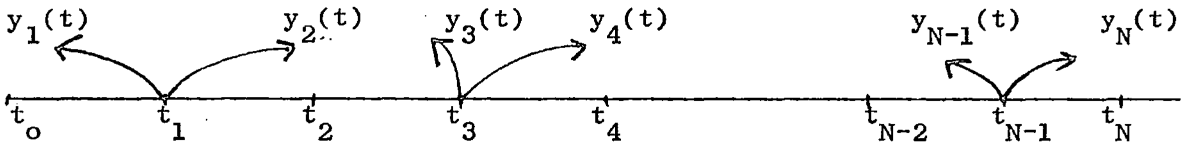


But, as explained in IIIa, if the problem matrix A possesses several large positive dominant eignvalues but no negative ones (or at any rate, ones with only very small negative real parts) it may be advantageous to reverse the direction of shooting in each sub interval, treating the given 'initial' conditions at t = o as the terminal conditions and those at t = 1 as the initial conditions. This variation is known as backward shooting:



By reversing such a problem the cumulative errors in the exponential terms due to round off will be greatly reduced and this means that it may be possible to obtain a more accurate overall solution vector with fewer sub intervals.

If, as is more likely, the real parts of the eigenvalues of the problem matrix are mixed positive and negative of roughly equal size so that the error growth is approximately equal in both directions of integration, then another variation that can be employed is to use an even number N of sub intervals and to shoot backwards and forwards from each odd numbered node thus:



In this case the continuity equations at the internal nodes become:

$$y_r(t_r) = y_{r+1}(t_r) \Rightarrow B_r = B_{r+1} \quad \text{for } r = 1 ---- (N-1) \atop \text{(odd nos)}$$

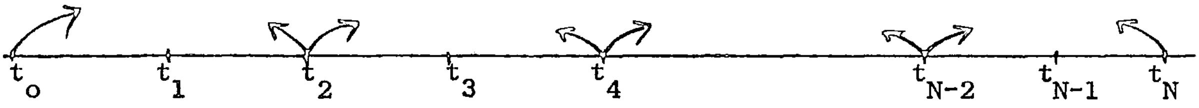
$$\text{and} \quad y_s(t_s) = y_{s+1}(t_s) \Rightarrow N_s(t_s).\,B_s + W_s(t_s)$$

$$= N_{s+1}(t_s)\,.\,B_{s+1} + W_{s+1}(t_s)$$

$$\text{for} \quad s = 2 ---- (N-2) \atop \text{(even nos)}$$

making n(N-1) linear equations. In addition we have r equations at t_o and (n-r) equations at t_N giving nN equations in total.

Another possible variation is (again for N even) to shoot in both directions from each even node, forwards from t_o and backwards from t_f thus:



For the case N = 2, this method reduces to what is known as "matching in the middle".

Instead of calculating the eigenvalues of the problem matrix A in order to decide which variation to employ, it is possibly better to solve the problem by several variations and then to take the average of the solution vectors obtained. If A(t) is variable, then averaging is certainly preferable.

d) The Riccati Inverse (or continuation) method

With reference to the Riccati Transformation method described in IC, consider the case where the transformation matrix U is square $\rho \times \rho$ i.e. where both base and surface vectors are $\rho \times 1$. As explained in (1.6), if we integrate the set of characteristic equations:

$$\begin{pmatrix} \dot{u} \\ \dot{x} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} u \\ x \end{pmatrix} + \begin{pmatrix} F \\ G \end{pmatrix}$$

forwards from u(o) = a (fixed) and x(o) = s (arbitrary) then we obtain the surface vector u(t,x) in terms of the base vector x(t) from the transformation:

$$u(t) = U(t)\, x(t) + v(t) \quad \text{for } t \geq o,$$

where the matrix U(t) satisfies the Riccati equation (v) and vector v(t) satisfies equation (vi), both of IC. Now consider the system obtained by reversing the definition of surface and base vectors i.e. by interchanging x and u:

$$\begin{pmatrix} \dot{x} \\ \dot{u} \end{pmatrix} = \begin{pmatrix} D & C \\ B & A \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} + \begin{pmatrix} G \\ F \end{pmatrix}$$

If now we integrate this system forwards from x(o) = a (fixed) and u(o) = s (arbitrary) then we obtain the surface vector x(t,u) in terms of the base vector u(t) from the inverse transformation:

$$x(t) = W(t)\,u(t) + \mathcal{Z}(t) \quad \text{for } t \geqslant o,$$

where $W(t) = U^{-1}(t)$, provided it exists.

Now: $u(t) = U(t)\,x(t) + v(t)$

$$\Rightarrow \quad U^{-1}(t)\,u(t) = x(t) + U^{-1}(t)\,v(t)$$

$$\Rightarrow \quad x(t) = U^{-1}(t)\,u(t) - U^{-1}(t)\,v(t)$$

Hence $\mathcal{Z}(t) = -U^{-1}(t).\ v(t)$.

Thus, if over any interval both $U(t)$ and $U^{-1}(t)$ exist and are bounded then we have the option of obtaining the solution vector $y(t)$ either by using the normal transformation:

$$u(t) = U(t).\ x(t) + v(t) \text{ (as described in IC)}$$

or instead by employing the inverse transformation:

$x(t) = U^{-1}(t)\,u(t) + \mathcal{Z}(t)$, where now u is the base vector and x the surface vector. If using the inverse method of solution we must make the following adjustments to the normal imbedding equations (ii)→(vi) of IC:

interchange matrices $A \Leftrightarrow D$, $B \Leftrightarrow C$, $F \Leftrightarrow G$

replace U by $W(=U^{-1})$

interchange $u \Leftrightarrow x$

replace v by $\mathcal{Z}$.

Thus if we were solving a problem over sub interval $[t_1, t_2]$, in $[o,1]$ using U^{-1} instead of U throughout, then the inverse imbedding equations required would be as follows, where u now denotes the base vector and x the surface vector:

(i) $\overset{\circ}{W} = C + DW - WA - WBW$ (Riccati). Integrate this equation forwards starting from $W(t_1) = U^{-1}(t_1)$. This is the equivalent of equation IC(v) for the normal transformation.

(ii) $\dot{z} = (D - WB)\, z - WF + G$. Integrate this equation forwards starting from $z(t_1) = -U^{-1}(t_1).\ v(t_1)$.

This corresponds to equation IC(vi).

(iii) Solve for $u(t_2)$ the matrix equation:

$$\left[g_1 W(t_2) + g_2\right] u(t_2) = b - g_1 .\ z(t_2).$$

This corresponds to equation IC(viii).

(iv) $\dot{u} = (BW + A)u + Bz + F$. Integrate this equation backwards starting from $u(t_2)$. This corresponds to equation IC(ix).

(v) Obtain the corresponding surface vector $x(t)$ from the base vector $u(t)$ at any value of t in $\left[t_1, t_2\right]$ from $x(t) = U^{-1}(t).\ u(t) + z(t)$.

This corresponds to equation IC(iv).

For any given problem there exists a critical length $\hat{t}$ at which the solution $U(t)$ of the Riccati equation will become unbounded. In fact, it can be shown that:

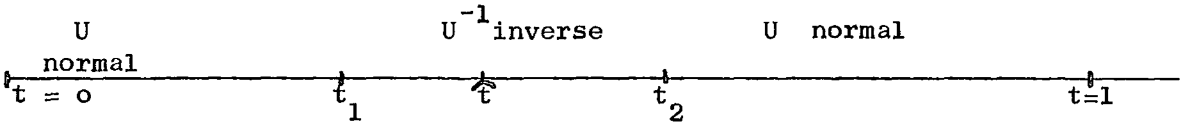
$$\hat{t} = \frac{1}{K+d} \cdot \ln \left[1 + \frac{K+d}{c} \right] \quad \text{where} \quad //A(t)// \leqslant a,$$

$//B(t)// \leqslant b$, $//C(t)// \leqslant c$, $//D(t)// \leqslant d$ and

$K = \max \left\{ a + b\ ,\ c + d \right\}$ over the interval $\left[0,1\right]$, (see 3.7). This difficulty is chiefly, but not solely, due to the quadratic term $U.C(t).U$ in this equation. Thus as $t \rightarrow \hat{t}$ the numbers involved in the calculation will eventually become unmanageably large and will cause blow up before $\hat{t}$ is reached. Therefore if $\hat{t}$ is near the terminal point 1 (or if $\hat{t} < 1$) we must adopt one of the following approaches in order to overcome the critical point:

1. <u>inverse method</u>: if U is a <u>square</u> matrix i.e. if we are given the same number of initial and final values in the problem, then we may be able to make use of the inverse transformation $U^{-1}(t)$ to get by $\hat{t}$. Although $U(t)$ may be becoming unbounded as $t \rightarrow \hat{t}$, $U^{-1}(t)$ may exist and be bounded in the neighbourhood of $\hat{t}$. If this is so, then we can switch the solution

method from $U(t)$ to $U^{-1}(t)$, the switch over point being some point $t = t_1$ before $\hat{t}$, at which $U(t_1)$ is still manageable, and continuing with $U^{-1}(t)$ to some point t_2 beyond $\hat{t}$. At $t = t_2$ we may then invert back and continue the solution with $U(t)$ again, thus:



This procedure can be repeated each time a critical point of $U(t)$ or $U^{-1}(t)$ is encountered. If a switch is made from U transformation to U^{-1} at t_1 then, as shown in (3.6), the U^{-1} integration of the inverse Riccati equation will start from $U^{-1}(t_1)$, obtained simply by inverting $U(t_1)$. Likewise at t_2, when we switch back from U^{-1} to U, the U integration will restart from $U(t_2)$ i.e. from $\left(U^{-1}(t_2)\right)^{-1}$. But since $z(t) = -U^{-1}(t).v(t)$, the starting vector for the integration of the $\overset{o}{z}$ equation will be $-U^{-1}(t_1)_{\circ}v(t_1)$ at t_1, and $-U(t_2).\,z(t_2)$ at t_2 for the $\overset{o}{v}$ equation.

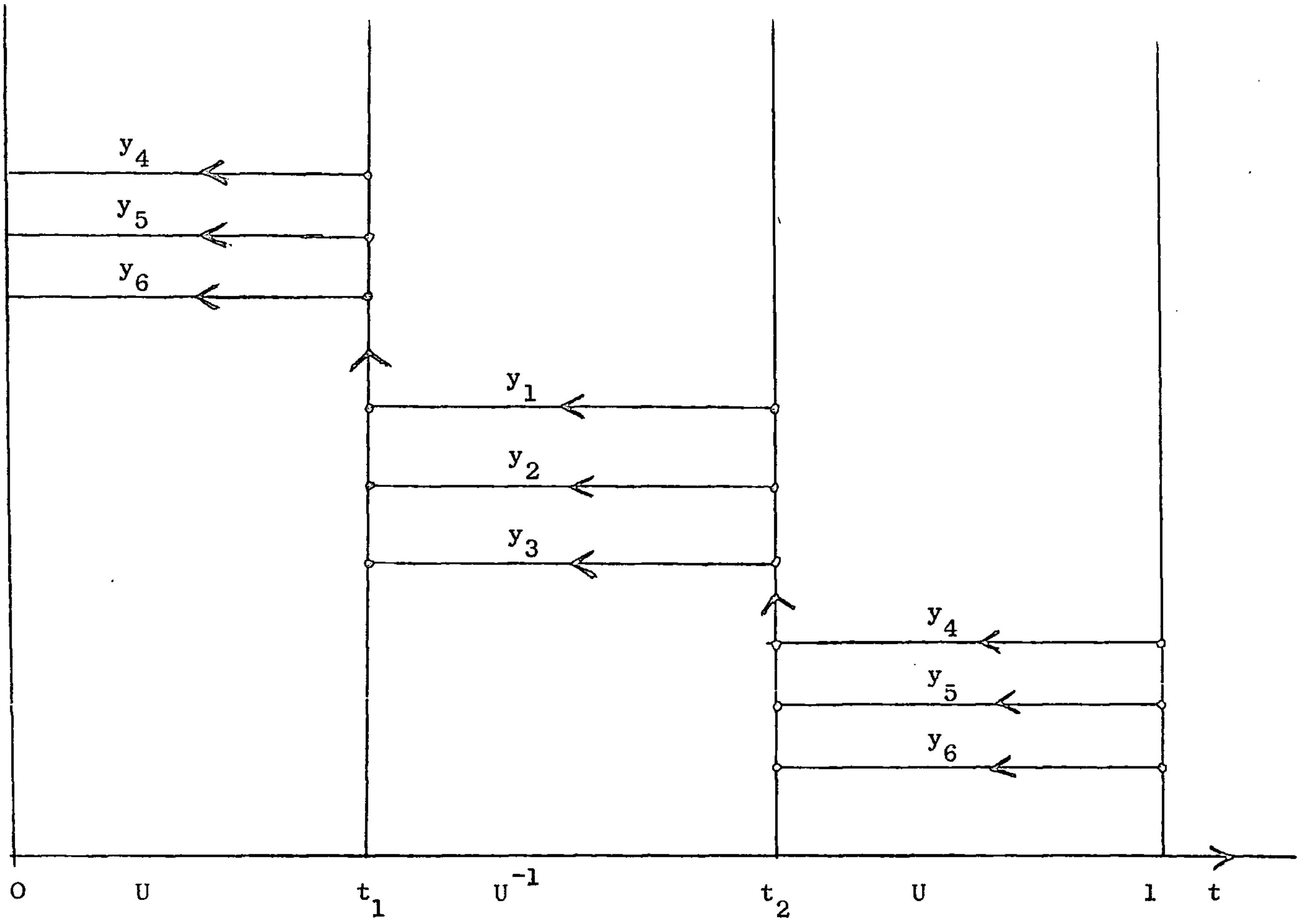
To illustrate how the inverse method is used in practice, suppose that in our problem we are given the values of y_1, y_2 and y_3 at $t = o$ and, say, those of y_1, y_2 and y_6 at $t = 1$. Then for the normal U transformation $(y_1\ y_2 . y_3)^T$ is the surface vector u and $(y_4 y_5\ y_6\ )^T$ is the base vector x, but for U^{-1} these are reversed, i.e. base $u = (y_1\ y_2\ y_3)^T$ & surface $x = (y_4\ y_5\ y_6)^T$. Suppose also that we decide to switch twice, from U to U^{-1} at t_1 and then back from U^{-1} to U at t_2, as shown above. We first integrate the normal Riccati and $\overset{o}{v}$ equation forwards from $t = o$ as in IC. At t_1 we switch to the inverse imbedding equations (i) and (ii) above, restarting the Riccati equation from $U^{-1}(t_1)$ and the $\overset{o}{z}$ equation from $-U^{-1}(t_1)_{\circ}v(t_1)$, and integrating both forwards to t_2. At t_2, we switch back to the normal equations again restarting at $U(t_2)$ for the Riccati and at $-U(t_2)_{\circ}\,z(t_2)$ for the $\overset{o}{v}$ equation and continue these integrations forward to the end point $t = 1$. Thus we now have the values of $U(1)$ and $v(1)$ from which we can obtain $x(1)$, the normal base vector, by using IC(viii). Also as in IC, we can now integrate equation (ix) backwards over $[t_2,1]$ to obtain the normal base vector $x(t)$ in this sub interval. At t_2, we use the transformation:

$u(t_2) = U(t_2)\, x(t_2) + v(t_2)$ to obtain the normal surface vector $u(t_2)$ which will be the base vector for the U^{-1} transformation. We must therefore now integrate backwards the inverse base equation (iv), given above, starting from $u(t_2)$ over the range $[t_1,t_2]$. This gives us the inverse base vector $(y_1 y_2 y_3)^T$ in this sub interval. At t_1 we use the inverse transformation (v) above:

$x(t_1) = U^{-1}(t_1).\, u(t_1) + Z(t_1)$ to obtain $x(t_1)$ which is the inverse surface vector i.e. the base vector for U. This is now the starting vector for the backward integration of equation (ix) of IC over $[o,t_1]$ which gives us the normal base vector for this sub interval. In the ranges $[o,t_1]$ and $[t_2,1]$ we now have the normal base vector $x = (y_4 y_5 y_6)^T$ from which we can find the corresponding surface vector $u = (y_1 y_2 y_3)^T$, at any storage point t, by using: $u(t) = U(t).x(t) + v(t)$. Similarly in the range $[t_1,t_2]$ we have the inverse base vector $u = (y_1 y_2 y_3)^T$ from which we can find the corresponding surface vector $x = (y_4 y_5 y_6)^T$ at any storage point t from:

$$x(t) = U^{-1}(t)\, u(t) + Z(t).$$

Thus we have found the surface and base vectors at each storage point t and hence the solution vector $y(t)$ of the problem over the complete range $[o,1]$. The procedure can be summarised thus:



Obviously, the above procedure can be extended to cater for any number of switching points, the underlying idea being that we always work backwards through the subintervals starting from t = 1, finding the base vector in each sub interval and then obtaining the surface vector at the switching point which will be the starting base vector for the backward integration in the next sub interval. In this respect of working backwards through the sub intervals, the Riccati Inverse method closely resembles the re-orthonormalisation method.

However, if matrix U is not square then obviously the above method is not applicable, since U^{-1} is not defined, and so to overcome a critical point we resort to the Riccati reverse method:

2) <u>Reversal</u>: As mentioned in connection with the multiple shooting method in III(a) and (c), sometimes 'blow up' can be avoided by reversing the direction of integration of an equation. In this case, however, if we reverse the direction of integration of the Riccati equation IC(v) this will also necessitate the reversal of the $\mathring{v}$ equation (vi) and the $\mathring{x}$ equation (ix). Moreover, we must interchange the dimensions of u and x so that $m \leftrightarrow \rho$ throughout and this causes alterations in the characteristic matrices A, B, C, D and in the vectors F and G. To be specific we would proceed as follows:

Define the surface vector u to be those components of y which are known at t =1 and the base vector x to be the remaining components. Thus the boundary conditions now become:

$$u(1) = b \text{ and } g_1\, u(o) + g_2\, x(o) = a \qquad \text{(i)}$$

where g_1 is $(m \times \rho)$ and g_2 is $(m \times m)$, assuming that m values are known at t = o and ρ values at t = 1. The corresponding characteristic equations will now be:

$$\begin{pmatrix} \mathring{u} \\ \dot{x} \end{pmatrix} = \begin{pmatrix} a(t) & b(t) \\ c(t) & d(t) \end{pmatrix} \begin{pmatrix} u \\ x \end{pmatrix} + \begin{pmatrix} f(t) \\ g(t) \end{pmatrix} \quad \text{where}$$

a(t), b(t) etc. will be different from A(t), B(t) etc. The transformation $u(t) = U(t)\, x(t) + v(t)$ (ii) still holds and so U(1) = o corresponds to v(1) = u(1) = b. We now integrate <u>backwards</u> the Riccati equation:

$$\dot{U} = b(t) + a(t)\, U - Ud(t) - Uc(t)U$$ from t = 1 to t = o starting from U(1) = o and save the values.

Also integrate <u>backwards</u> over $[o,1]$ the equation: $\mathring{y} = \left[a(t) - U(t)\, c(t)\right].V - U(t)g(t) + f(t)$ starting from v(1) = b and save the values.

Now from (ii): $u(o) = U(o)\, x(o) + v(o)$.

Substituting in (i) we get:

$$g_1 . \left[U(o)x(o) + v(o)\right] + g_2\, x(o) = a$$

$$\Rightarrow \left[g_1\, U(o) + g_2\right].\, x(o) = a - g_1\, v(o),$$

where U(o) and v(o) are known, so that this matrix equation can be solved for x(o).

Now integrate forwards the equation:

$$\dot{x} = \left[c(t)U(t) + d(t)\right] \circ x + c(t)\ v(t) + g(t),$$

starting from x(o), over $\{o,1\}$, and save the values. Finally, find the surface vector u(t) at each saved point t by using:

$u(t) = U(t)\ x(t) + v(t)$, and hence obtain the solution vector y(t) from x(t) and u(t).

IV PRACTICAL NUMERICAL EXPERIENCE

All the programs used in this section to solve the test problems were written in Spectrum Basic programming language and are intended to be run on a SpectrumZX microcomputer with 48K Ram. In this machine, numbers are stored in floating point binary arithmetic, to an accuracy of 9 or 10 digits, in the form mx 2^{e-128} where m, the mantissa, gives us the digits in the number and e, the exponent, fixes the position of the decimal point, where $\frac{1}{2} \leqslant m \leqslant 1$ and $1 \leqslant e \leqslant 255$. Thus the largest number stored is about 10^{38} ($\approx 2^{127}$) and the smallest positive number about 4×10^{-39} ($\approx \frac{1}{2} \cdot 2^{-127}$), but the largest integer which can be held completely accurately is only $2^{32} - 1$ ($\approx 4 \times 10^{9}$). This means that the results of any calculation process involving numbers greater than, say, 10^{10} must be suspect and if numbers of the order 10^{15} are present then the results may be so inaccurate as to be virtually useless. For this reason we have restricted ourselves to the solution of problems for which the mod of the exact solution components do not exceed 10^{9}, so that some reliance can be placed on the accuracy of the results achieved.

We have concentrated mainly on the solution of two test problems, A and B. For each, the problem matrix is of size n = 6 and contains a variable element L which can be used to alter the condition of the matrix so that in effect each problem is really a family of problems. For a chosen value of L, the eigenvalues λ_s and corresponding eigenvectors C_s (s = 1 - - 6) of the problem matrix were calculated. A particular solution vector $\emptyset(t)$ was then assumed and the constants K_s (s = 1 - - 6) defined so that the constant K_i, corresponding to the maximum positive eigenvalue λ_i, was zero. Thus the exact solution vector $y_e(t)$, $0 \leqslant t \leqslant 1$, was given by:

$$y_e(t) = \sum_{s=1}^{6} K_s e^{\lambda_s \circ t} C_s + \emptyset(t), \text{ where,}$$

theoretically, the term involving the eigenvalue λ_i was completely excluded. However, in practice, as explained in section II, if λ_i is sufficiently large, then as the integration proceeds forwards from t = 0 to t = 1 this term will gradually be re-introduced into the calculated solution vector due to round off error, with a consequent loss in accuracy which will "snowball" the further the integration goes.

In example 3, we also solved a third problem, C, details of which are given later.

In order to obtain a boundary value problem, in each case we assumed that only three components (y_1, y_2 and y_3) were known at t = 0 and three at t = 1 (y_3, y_4 y_5 for problem A and y_1, y_2, y_6 for problems B and C). From these we attempted to obtain a computed solution vector $y_c(t)$, stored at intermediate points in [o,1] , which could then be compared with the exact solution $y_e(t)$ at the corresponding t points. The accuracy of $y_c(t)$, at any stored point t, was measured by computing the L_2 norm of the error vector viz: $\left\{\sum_{i=1}^{6}\left(y_e^i(t) - y_c^i(t)\right)^2\right\}^{\frac{1}{2}}$. An overall measure of the accuracy of $y_c(t)$, taken over the full interval [o,1] could then be found by computing the percentage error norms at a number of chosen test points:

t = 0, t = 0.5, t = 1, <u>or</u> t = 0, t = 0.25, t = 0.5, t = 0.75, t = 1.

(The percentage error norm at any point t is the ratio of the norm of the error vector to the norm of the exact solution vector at that point). In this way, by comparing the percentage error values obtained, we were able to measure, for any given problem, the relative effectiveness of either:

(a) different solution methods employing the same step length h each time, or

(b) the same solution method with different values of h, or

(c) the re-orthonormalisation method (or multiple shooting method) with varying number of subintervals (m) and varying values of h.

As stated earlier, in the case of the re-orthonormalisation method and multiple shooting method we confined ourselves to equal subintervals only. Also in examples 1 to 4 all of the integrations of the system equations in all of the programs were performed by employing a Runge Kutta method. In example 5, we obtained a comparison of the accuracy of this integration scheme with that of a linear multistep method (employing the above Runge Kutta as starter), by comparing the accuracy of the results obtained in solving a given problem using a given step length h and given method of solution, but employing first one integration method and then the other. The details of these integration schemes are as follows:

Runge Kutta Method: This was a six stage method with a local order of accuracy p = 5, known as Lawson's method, defined by:

$$y_{n+1} - y_n = \frac{h}{90}\left[7K_1 + 32K_3 + 12K_4 + 32K_5 + 7K_6\right]$$

where:

$$K_1 = f(x_n, y_n)$$
$$K_2 = f(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hK_1)$$
$$K_3 = f\left(x_n + \tfrac{1}{4}h, y_n + \tfrac{1}{8}h(K_1 + K_2)\right)$$
$$K_4 = f(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hK_3)$$
$$K_5 = f(x_n + \tfrac{3}{4}h, y_n + \frac{3h}{16}.(-K_2 + 2K_3 + 3K_4))$$
$$K_6 = f(x_n + h, y_n + \frac{h}{7}(K_1 + 4K_2 + 6K_3 - 12K_4 + 8K_5))$$

Linear multistep method: This was an explicit 4 step Adams-Bashforth method of local order of accuracy p = 4, defined by:

$$\sum_{j=0}^{K} \alpha_j y_{n+j} = h \sum_{j=0}^{K} \beta_j f_{n+j} \quad \text{where}$$

$K = 4, \alpha_4 = 1, \alpha_3 = -1, \alpha_2 = \alpha_1 = \alpha_0 = 0$.

$$\beta_4 = 0, \quad \beta_3 = \frac{55}{24}, \quad \beta_2 = \frac{-59}{24}, \quad \beta_1 = \frac{37}{24}, \quad \beta_0 = \frac{-9}{24}$$

The details of the solved test problems, each with system equations $\dot{y} = Ay + f$ and over the interval $[0,1]$, were as follows:

Problem A: $n = 6$

$$A = \begin{bmatrix} L & 1 & 0 & 0 & 0 & 0 \\ 0 & 10 & 1 & 0 & 0 & 0 \\ 0 & 0 & 5 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The eigenvalues of A are : L, 10, 5, 3, 2.6180339, 0.3819661, and the corresponding K constants were defined as: 0, 1,1, 1, 1, 1. The particular solution vector $\phi(t)$ was taken as: $\phi(t) = [t, 0, 0, 0, 0, 0]^T$ which meant that vector f(t) was given by:

$$f(t) = [1 - Lt, 0, 0, 0, 0, 0]^T \text{ since}$$

$$f = \overset{\circ}{\phi} - A\,\phi \,.$$

L was now chosen to be a large positive value (e.g. 85) and the corresponding eigenvectors and the exact solution vector obtained, from which we assumed to be given: $y_1(0)$, $y_2(0)$, $y_3(0)$ and $y_3(1)$, $y_4(1)$, $y_5(1)$.

Problem B: $n = 6$

$$A = \begin{bmatrix} 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 10 & 1 & 0 & 0 & 0 \\ 0 & 0 & 5 & 1 & 0 & 0 \\ 0 & 0 & 0 & L & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The eigenvalues of A are: 3,10,5,L, 2.6180339, 0.3819661 and the corresponding K constants were defined as: 1, 1, 1, 0, 1, 1. The particular solution vector was taken to be : $Ø(t) = (0,0,0,t,0,0)^T$ which meant that vector f(t) was given by:

$$f(t) = (0,0, -t, 1 -Lt, 0, 0)^T$$

As for problem A, corresponding to each chosen large positive value of L the eigenvectors and exact solution vector were obtained, from which it was assumed that we were given $y_1(\Theta)$, $y_2(\Theta)$, $y_3(\Theta)$ and $y_1(1)$, $y_2(1)$, $y_6(1)$.

<u>Problem C</u> : n = 6

$$A = \begin{bmatrix} 9.11 & 5.32 & 1.97 & 2.12 & 1.44 & 7.65 \\ 5.32 & 8.11 & -4.24 & 3.21 & 2.34 & 1.46 \\ 1.97 & -4.24 & 7.64 & 1.03 & 5.02 & -4.58 \\ 2.12 & 3.21 & 1.03 & 9.33 & 3.72 & 1.26 \\ 1.44 & 2.34 & 5.02 & 3.72 & 9.98 & -5.04 \\ 7.65 & 1.46 & -4.58 & 1.26 & -5.04 & 8.33 \end{bmatrix}$$

The eigenvalues of A are -3.2676, 1.1056, 6·2342, 9.3012, 18.358, 20·7686.

The particular solution vector Ø(t) was taken to be: $Ø(t) = \left[\cos t, \Theta, t, \Theta, t^2, \Theta\right]^T$ and the K constants this time were chosen so that the initial vector $y(\Theta) = 0$, and the exact solution vector y(t), $\Theta \leq t \leq 1$, was then obtained. The exact value of vector y(1) was:

$$\begin{bmatrix} -3.0699764e8 \\ -2.3760237e8 \\ 1.7249322e8 \\ -1.1540849e8 \\ 98125236 \\ -3.3069843e8 \end{bmatrix}$$

We assumed that we were given $y_1(\Theta)$, y_2 (Θ), y_3 (Θ) and y_1 (1), y_2 (1) y_6 (1).

The programs of the solution methods used to tackle the above problems were as follows:

Program 1: Multiple shooting based on the variation of parameters method (IIIc) including single shooting.

2: Reverse single and multiple shooting (IIIa and c).

3: Conte's Reorthonormalisation Method (IIIb)

4: Riccati Transformation Method (Ic).

In examples 1 to 4, all of these programs employed the Runge Kutta integration scheme. In example 5 we also used program 3 with the linear4step integrator.

Results: All numerical results are given to an accuracy of two decimal places, except where otherwise stated.

Example 1 : Reorthonormalisation (program 3):

Here we concentrated on the solution of variations of Problem B (obtained by assigning different values to L) using varying numbers of equal subintervals, m.

First we attempted a solution of the problem for which L = 85, using a step length h = 1/48. With m =2, the calculation failed due to a blow up in the reothonormalisation subroutine. When m was increased to 4, a solution was obtained but was discarded as being totally meaningless since all the calculated components at t = 1 exceeded 10^{20} in mod value. With m = 8, however, a solution was obtained for which the norms of the error vectors at t = 0 ($||e_0||$), t = 0.25 ($||e_{1/4}||$) , t = 0.5 ($||e_{1/2}||$), t = 0.75 ($||e_{3/4}||$) and t = 1 ($||e_1||$) were computed together with the percentage errors atthese points. The results were as follows:

Problem B : L = 85 : h = 1/48 : m = 8

		% error
$\| e_0 \|$	191.46	1.02
$\| e_{\frac{1}{4}} \|$	216.86	1.05
$\| e_{\frac{1}{2}} \|$	258.36	1.13
$\| e_{\frac{3}{4}} \|$	342.88	1.21
$\| e_1 \|$	148,364.00	93.48

The above solution is acceptably accurate over the range t = 0 to t = 0.75 but between t = 0.75 and t=1 the term involving the eigenvalue λ = L = 85 is introduced into the calculated solution by the round off error, resulting in a massive error in the fourth component y_4 at t = 1, this being the component chiefly affected by L. However, this example does demonstrate the effectiveness of increasing the number m of reorthonormalisations when tackling a problem with a large positive dominant eigenvalue. A further increase in m was tried but the calculated solutions for m = 16, m = 32 and m = 48 proved to be virtually identical to that given above for m = 8, the steplength h being kept at 1/48. When, however, the value of h was reduced to 1/192 with m = 8 the percentage error norm at t = 1 was reduced to 9.92%, but again further increases in m produced identical solutions to that obtain for m = 8. (see also example 5).

In an attempt to assess the effectiveness of the reorthonormalisation method in solving progressively more ill conditioned problems, we now solved Problem B taking L = 15, 25, 35, 45, 55, 70 and 85 successively, using a steplength of h = 1/48 and m = 8 each time. For each computed solution we obtained the error norms and percentage error norms at t = 0, t = 0.5, t = 1 and these results are recorded below:

Problem B : L = 1/48 : m = 8

	t = 0		t = 0.5		t = 1	
L	$\|\| e_0 \|\|$	%	$\|\| e_{\frac{1}{2}} \|\|$	%	$\|\| e_1 \|\|$	%
15	33.15	1.02	44.95	1.10	2845.00	1.82
25	55.37	1.02	74.82	1.11	10098.00	6.46
35	77.84	1.02	98.31	1.04	21882.00	13.98
45	100.58	1.02	135.84	1.13	38166.00	24.34
55	123.35	1.02	166.64	1.13	58952.00	37.50
70	157.07	1.02	212.40	1.13	98599.00	62.46
85	191.46	1.02	258.36	1.13	148,364.00	93.48

We see that the percentage errors at t = 0 and t = 0.5 remain virtually constant as L is increased i.e. as the problem becomes more ill conditioned, indicating that the accuracy of the calculated solutions over this range is not much affected by the increasingly dominant large positive eigenvalue L. But the extent to which the term containing L is introduced into the calculated solution is obvious from the percentage errors at t = 1. Only for L = 15 is the calculated solution acceptably accurate over the full range [0,1].

In our problem B the components known at t = 1 are y_1, y_2 and y_6 whereas the component most affected by the eigenvalue L is y_4. Thus the calculated value of γ_f would be fairly accurate because this is found (see section IIIb) by using the components of the final homogeneous and inhomogenous vectors at t = 1 which correspond to the positions of the known terminal values i.e. the first, the second and the sixth.

This in turn meant that the accuracy of the calculated values of the other γ vectors would not be seriously affected since these are derived by backward iteration from γ_f. This explains the acceptable accuracy of the solutions over the range t = 0 to t = 0.5. The inaccuracy of the solution towards t = 1 arises mainly from the error in the fourth component of the calculated homogeneous and inhomogeneous vectors as they are integrated forwards and the term involving eigenvalue L is introduced. By increasing the number m of reorthonormalisations from 2 to 8 we did reduce the rate of growth of this error sufficiently to give (for the case L = 85) an acceptable solution as far as t= 0.75.

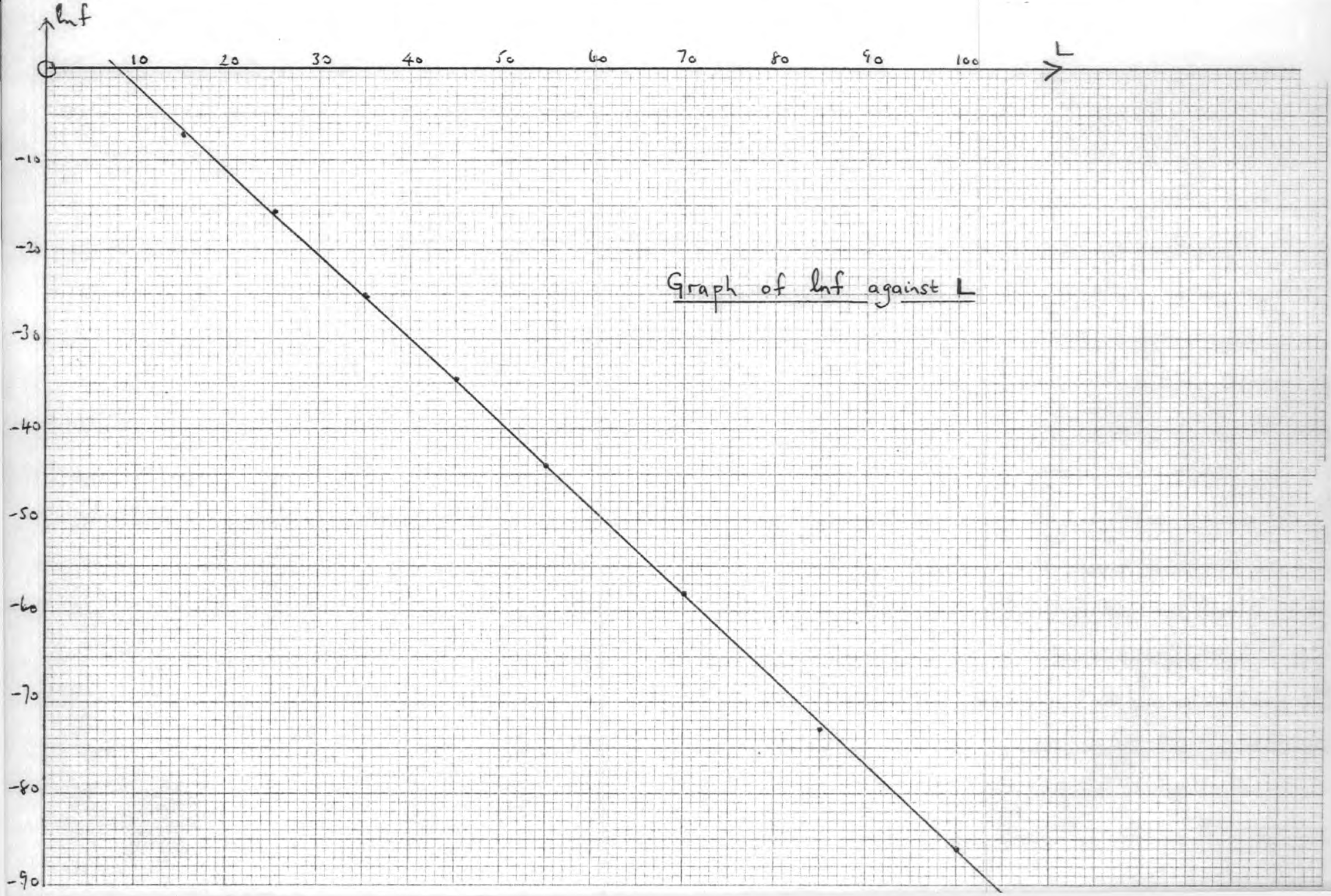
Since the error at t = 1 is due to the introduction into the calculated solution of a term containing e^{L}, if we denote $||e_1||$ by E we may write: $E = f.e^{L}$ where E and f both depend on L. This gives us:

$\ln E = \ln f + L$ or $\ln f = \ln E - L$. From our results above for L = 15, 25, 35, 45, 55, 70 and 85 we obtained a linear graph of lnf against L as shown below. From the graph we see, for example, that when L = 100, lnf = -86 so that lnE = 14 i.e. $E = e^{14} \approx 1,202,000$. This is the value of $||e_1||$ we would expect for the solution of Problem B with L = 100. In fact when we solved this problem using h = 1/48 and m = 8 the error $||e_1||$ turned out to be 1,110,000, approximately. If we restrict our consideration to the range $35 \leqslant L \leqslant 70$ then the error E at t = 1 can be written $E = f.e^{L}$ where $f \approx \exp\left[-0.956L + 8.48\right]$ so that $E \approx \exp\left[0.044L + 8.48\right]$.

Example 2 : Forward and reverse shooting (Programs 1 and 2)

In this example we showed the advantage of reverse shooting over forward shooting in the solution of a problem possessing positive eigenvalues particularly where one is large and well separated from the others i.e. where the problem matrix is very ill-conditioned.

A solution of Problem A with L = 100 was attempted using the multiple shooting method with a steplength of h = 1/24. Forward shooting with



Graph of lnf against L

$m = 1$, $m = 2$ and $m = 4$ all failed to produce a solution due to a blow up either in one of the homogeneous integrations or in the matrix solution subroutine. But when the direction of shooting was reversed a solution was obtained each time. These three solutions were virtually identical and acceptably accurate as the percentage error values at $t = 0$, $t = 0.5$ and $t = 1$ show below:

Problem A : $L = 100 : h = 1/24 : m = 1, 2, 4$

Reverse shooting

	% error
$t = 0$	0.32
$t = 0.5$	0.50
$t = 1$	2.28

Moreover, further attempts to obtain a solution by forward shooting using $m = 5$ and $h = 1/50$ and then $h = 1/100$ also failed due to blowup in the matrix calculation.

As mentioned in section III, the explanation for this is the re-introduction into the calculated solution of the term containing the large positive eigenvalue $L = 100$. When the integration starts at $t = 0$ the K constants have been chosen so as to exclude this term completely. If the calculation process were 100% accurate then this would still be the case at $t = 1$. But in practice as the integration proceeds forwards from $t = 0$ the calculated solution (y_c) strays from the exact solution (y_e) due to round off error at each step so that by the time $t = t_1$ (say) is reached $y_c(t_1) = y_e(t_1) + \varepsilon$. If we now imagine the integration to be stopped and then restarted at $t = t_1$ the starting vector will be $y_c(t_1)$ and not $y_e(t_1)$ as it should be. Now the starting vector determines the K constants which means these must be different if $y_c(t_1)$ is used instead of $y_e(t_1)$. With $y_e(t_1)$ the K constants would be such that the K_i corresponding to the term containing L would be exactly zero. But with $y_c(t_1)$ this will not be so and K_i will now have a small non-zero value which means that the term $K_i e^{Lt} C_i$ has been introduced into the solution $y_c(t)$ for $t > t_1$. Once this term has gained a foothold in the solution, no matter how small, its value rapidly snowballs as t increases because of the very rapid growth of the factor e^{Lt} $(t > 0)$ and will eventually cause $y_c(t)$ to blowup if L is large enough and if the integration proceeds long enough. For reverse integration, on the other hand, the effect of introducing this term is negligible since in this case it contains a factor $e^{L(t-1)}$ where $t < 1$. Thus the greater is the positive value L the more marked will be the difference in accuracy between forward and reverse shooting and if L is sufficiently large (100 in our example) quite accurate solutions may be obtained by reverse shooting whilst forward shooting may result in blowup.

Example 3 : Comparison of Riccati method with single shooting (Programs 4 and 1 respectively)

First we solved Problem C, Problem A with L = 6, Problem B with L = 15 and Problem B with L = 20 by both the forward single shooting variation of parameters method (program 1) and the Riccati method (program 4) with a step length of h = 1/100 each time. For the solutions of Problem C we calculated the error norms $||e_0||$, $||e_{1/2}||$ and $||e_1||$ together with the overall error norm $||e\ ||$ and these are listed below:

Problem C : h = 1/100

	Single shooting	Riccati
$\|\|e_o\|\|$	1.11	1.05
$\|\|e_{1/2}\|\|$	86.66	95.18
$\|\|e_1\|\|$	4450.34	4452.14
$\|\| e\|\|$	4451.18	4453.16

For the other solved problems we calculated the percentage error norms in the respective solution vectors and the results were as follows:

Problem A : L = 6 : h = 1/100

% error at	Single shooting	Riccati
t = 0	0.04	0.06
t = 0.5	0.04	0.06
t = 1	0.09	0.13

Problem B : L = 15 : h = 1/100

% error at	Single shooting	Riccati
t = 0	0.24	0.25
t = 0.5	0.26	0.27
t = 1	0.49	0.51

Problem B : L = 20 : h = 1/100

% error at	Single shooting	Riccati
t = 0	0.23	0.24
t = 0.5	0.28	0.28
t = 1	1.05	1.05

We see that for the above problems the solutions obtained by the Riccati method are very nearly as accurate overall as those obtained by single shooting. However, a difference in accuracy between the two methods was revealed when we solved cases of Problem A. For example, the table below shows the percentage error results for the solutions of Problem A with L = 15 using a steplength of h = 1/100:

Problem A ; $L = 15$: $h = \frac{1}{100}$

% error at	Single shooting	Riccati
t = 0	0.01	0.01
t = 0.5	0.02	0.20
t = 1	0.11	17.10

In this case, the solution obtained by single shooting is overall very accurate, but the Riccati solution, although acceptable at t = 0 and t = 0.5 is highly inaccurate at t = 1. This latter fact is entirely due to the error in the first calculated component $y_1(1)$: the other missing components at t = 1 ($y_2(1)$ and $y_6(1)$) were found almost exactly. In fact the calculated value of $y_1(1)$ was 41,859 as compared to the exact value of 22,211. Now for the Riccati program the surface vector was $u_1 = y_1$, $u_2 = y_2$, $u_3 = y_3$ since for our problem A these are the three components given at t = 0, and the base vector x was taken to be $x_1 = y_4$, $x_2 = y_5$, $x_3 = y_6$. The initial boundary condition was:

$u(o) = a = \left[y_1(0), y_2(0), y_3(0) \right]^T$ whilst at t = 1 we had:

$g_1\ u(1) + g_2\ x(1) = b$ and since we are given $y_3(1)$, $y_4(1)$ and $y_5(1)$ this became:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1(1) \\ y_2(1) \\ y_3(1) \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_4(1) \\ y_5(1) \\ y_6(1) \end{bmatrix} = \begin{bmatrix} y_4(1) \\ y_5(1) \\ y_3(1) \end{bmatrix}$$

The characteristic equations were:

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} L & 1 & 0 \\ 0 & 10 & 1 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_4 \\ y_5 \\ y_6 \end{bmatrix} + \begin{bmatrix} 1 - 15t \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \dot{y}_4 \\ \dot{y}_5 \\ \dot{y}_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} + \begin{bmatrix} 3 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_4 \\ y_5 \\ y_6 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

For Problem A, y_1 is the component affected by the largest eigenvalue L = 15 and since this element is in characteristic matrix A (see above) it appears in both of the forward integrations ($\mathring{U}$ and $\mathring{v}$) of the Riccati method (see section Ic). The surface vector u(t), at any storage point t, is then obtained from the equation.

u(t) = U(t). x(t) + v(t), which is dependent on the saved values of U(t) and v(t) from these forward integrations, so that any inaccuracies in these calculated values will obviously be reflected in the computed value of u(t). If we evaluate the above equation at t = 1 , then the calculated value of $y_1(1)$ i.e. of the first surface component $u_1(1)$ is given by:

$$u_1 = U_{11}x_1 + U_{12}x_2 + U_{13}x_3 + v_1 \qquad (1)$$

where all values are at t = 1 and $v = \begin{bmatrix} v_1, & v_2, & v_3 \end{bmatrix}^T$. Now to determine precisely how the calculated value of y_1 (1) was arrived at we obtained from the program the calculated values of U_{11}, U_{12}, U_{13} and

v_1 at $t = 1$ and these were as follows:

Problem A : $L = 15$: $h = 1/100$: Calculated Values at $t = 1$:
(correct to 5 decimal places)

U_{11}	=	265.56232
U_{12}	=	-500.50947
U_{13}	=	336.34299
v_1	=	4,919,309.90000

The exact values of x_1, x_2 and x_3 for this problem were:

$x_1 = -7310.3319 \qquad x_2 = 3630.5784$

$x_3 = -3325.1946$

We now see from equation (1) above just how critical is the accuracy of the calculated values of U_{11}, U_{12}, U_{13} and v_1 in the final calculation of u_1. For example, an error of only 1 in 265 ($\approx$ 0.4%) in the calculated value of U_{11} would cause a corresponding error of more than 7000 in the calculated value of u_1. Similarly if the calculated value of v_1 is in error by only 0.5% this causes a resultant error in u_1 of almost 25,000. In fact, we re-calculated the values of U_{11}, U_{12}, U_{13} and v_1 for this problem by using the reduced step length of $h = 1/300$ to perform the integrations and we obtained the following slightly different values:

Problem A ; $L = 15$: $h = 1/300$: Calculated Values at $t = 1$
(correct to 5 decimal places)

U_{11}	=	265.11648
U_{12}	=	-499.13561
U_{13}	=	335.17387
v_1	=	4,892,857.60000

Substituting these values into equation (1) above, and again assuming the exact values for x_1, x_2 and x_3 gives the value $u_1 \approx 28,098$ which is a considerable improvement on our calculated value of 41,859 for $y_1(1)$ but still nearly 6000 in error compared to the exact value of 22,211.

To discover why this combination error that we have just discussed did not affect the accuracy of our Riccati solutions to the <u>other</u> problems solved in this section, we obtained the calculated values of all the elements of matrix U and vector v at t = 1 for Problem C (which has a maximum positive eigenvalue of 20.7), Problem B with L = 15 and Problem A with L = 15, each solved by the Riccati method with h = 1/100. The results are summarised below for comparison:

<u>Riccati method : h 1/100 : Calculated Values at t = 1</u>
<u>(correct to 5 decimal places)</u>

	Problem C	Problem B(L=15)	Problem A (L=15)
U_{11}	-0.50459	0.00162	265.56232
U_{12}	1.19204	-1.15484	-500.50947
U_{13}	1.45812	1.09883	336.34299
U_{21}	1.45284	0.01973	30.67662
U_{22}	-0.58717	-8.32769	- 52.11025
U_{23}	0.03725	7.89736	33.39030
U_{31}	-1.57826	0.09999	3.19456
U_{32}	1.81358	-0.64403	- 3.71904
U_{33}	0.56730	0.51605	2.07486
v_1	-34.49757	30082.28300	4919309.90000
v_2	247.00414	211148.6000	412500.90000
v_3	-246.77351	2671.69000	54326.60000

The difference between the nature of Problems C and B and that of A, from the point of view of the Riccati method, is evident from the above table. For Problems C and B, all the elements of the U matrix are very small (Some in modulus less than 1) so even a large percentage error in the calculation of these values would not have a serious affect on the calculation of the surface vector u. Moreover the components of vector v are much less in the case of Problems C and B than for A.

Thus we would conclude that for the solution of some problems the accuracy of the solution calculated by the Riccati method may be impaired due to the effect of the combination (or loss of significance) error described above. This difficulty is inherent in the problem itself and it is not immediately obvious as to whether a given problem is liable to be susceptible to this source of error as our considerations of Problems A and B (both with a maximum positive eigenvalue of 15) have shown. It would be advisable, therefore, when using the Riccati method, to first check on the calculated values of the U matrix at t = 1. If any of these is large in mod value then the accuracy of the calculated components of the surface vector might be suspect.

Since for both our Problems A and B above the characteristic matrix C was zero this meant that the quadratic term UCU was missing from the Riccati equation and so we were able to express this equation as a set of nine linear simultaneous differential equations of the form $\overset{\circ}{U} = H\,U + E$ where now U is (9 x1) and H (9 x 9). We then calculated the eigenvalues of system matrix H for each of our problems A and B with the following results:

Eigenvalues of matrix H (correct to 3-decimal places)

Problem A (L = 15)	Problem B (L= 15)
4.618	4.618
2.382	2.382
9.618	9.618
7.382	7.382
2.000	-10.000
7.000	- 5.000
12.000	-12.000
14.618	2.618
12.382	0.382

The first four eigenvaluesare common to the two cases but examination of the remaining five explains why the components of the U matrix increase much more rapidly in mod value in Problem A compared to those in Problem B, for forward integration.

Finally, as a further check on the accuracy of our Riccati equation integration we obtained the eigenvectors corresponding to the above eigenvalues, for case A, and then, after finding a particular solution vector we calculated the K constants and so determined the exact solution of this initial value problem. The exact values of U_{11}, U_{12}, U_{13} at t = 1 are given below along with our previously calculated values (obtained with a step length of h = 1/300) to 5 decimal places accuracy:

	Exact	Calculated (k = 1/300)
U_{11}	265.05947	265.11648
U_{12}	-497.82992	-499.13561
U_{13}	332.91292	335.17387

N.B. It must be remembered that the calculated U values from the Riccati integration are used in the forward integration of the $\dot{v}(t)$ equation so that any errors in the former will be transmitted to the latter, at each step, and therefore affect the accuracy of the computed v(t) values, and, in particular, that of $v_1(1)$ which appears in equation (1) above for the calculation of $u_1(1)$ $(=y_1(1))$.

Example 4: Comparison of forward multiple shooting (program 1) with reothonormalisation (program 3)

These two solution methods are very similar in that in both we obtain the solution vector $y(t)$ of the problem by "piecing together" the individual subinterval solution vectors. In reothonormalisation the subinterval combination vectors (γ_i) are found by backward iteration from γ_f (see section IIIb) whereas in multiple shooting (see IIIc) the equivalent vectors (B_i) are all found simultaneously by the solution of one matrix equation.

In order to assess the relative effectiveness of these two methods we used each in turn to solve Problem B with L = 20, taking the number of subintervals (m) to be 4 and using a steplength of first h = 1/100 and then h = 1/200. It was found that for each value of h the two calculated solution vectors obtained by the different methods were virtually identical at all stored points in [0,1]. The percentage errors at t = 0, t = 0.5 and t = 1 are given below:

Problem B : L = 20 : Multiple shooting and reothornomalisation:m = 4

	% errors	
	h = 1/100	h = 1/200
t = 0	0.24	0.06
t = 0.5	0.27	0.07
t = 1	1.02	0.38

The solution provided by the methods to this problem is very accurate at h = 1/200.

As a further check, we also solved Problem A with L = 20 by both methods, again taking m = 4 and using a steplength of h = 1/100 in each case. The error norms at t = 0, t = 0.5 and t = 1 are listed below:

Problem A ; L = 20 : m = 4 : h = 1/100

	Multiple Shooting	Reorthonormalisation
$\| e_o \|$	0.69	0.69
$\| e_{\frac{1}{2}} \|$	1.39	1.39
$\| e_1 \|$	12626.00	13391.00

In this case the multiple shooting method is marginally more accurate than the reorthonormalisation at t = 1. For both methods, $\| e_1 \|$ was entirely due to the inaccuracy in the calculated value of $y_1(1)$, the values of $y_2(1)$ and $y_6(1)$ being found almost exactly.

Finally, we solved Problem A with L=22 using h = 1/100 and m = 4. The two calculated solutions were virtually identical this time with the following percentage errors:

Problem A ; L = 22 : h = 1/100 : m = 4

Multiple Shooting and reorthonormalisation

	% error
t = 0	0.02
t = 0.5	0.03
t = 1	28.81

The error at t = 1 was again entirely due to the inaccuracy in the calculated value of component $y_1(1)$.

We would conclude from the above results that there is no significant difference between the accuracy of the multiple shooting and the re-othonormalisation method for the solution of problems of this type. However, although we could not distinguish between the two methods as regards accuracy, it was found that the program running time for the multiple shooting method was about 20% longer than for the reorthonormalisation method with the same step length. Part of this time difference might be attributed to the lack of sophistication of the programs themselves operating to the disadvantage of the multiple shooting, but it seems clear that reorthonormalisation is the quicker of the two methods.

Example 5 : Comparison of integration schemes

In examples 1 to 4, all of the integrations of the system equations in all of the programs were performed by the Lawson Runge Kutta method. We now compared the relative effectiveness of this integrator with the linear 4 step method, details of both of which were given earlier in this section. In order to do this, we again solved Problem B with L = 20 by the reorthonormalisation method (program 3) with m = 4 and h = 1/100, but now employing the linear integrator instead of the Runge Kutta to perform the homogeneous and inhomogeneous integrations. The accuracy of the calculated solution vector obtained was then compared with those from example 4, as shown below:

Problem B : L = 20 : m = 4 : Percentage errors: Reorthonormalisation

	Runge Kutta		Linear 4 step
	h = 1/100	h = 1/200	h = 1/100
t = 0	0.24	0.06	0.01
t = 0.5	0.27	0.07	0.02
t = 1.0	1.02	0.38	0.11

We see that for this problem the linear 4 step method produces a solution vector with h = 1/100 which is overall more accurate than that obtained by using the Runge Kutta method with h = 1/200.

As a follow-up to example 1, we re-solved Problem B with L = 85 by the reorthonormalisation method employing the linear 4 step integrator with $h = 1/192$ and $m = 8$. Comparison of this calculated solution vector with the corresponding one obtained using the Runge Kutta integrator gave the following percentage error results at $t = 0$, $t = 0.5$, $t = 1$:

Problem B : L = 85 : h = 1/192 : m = 8.
Percentage errors : Reorthonormalisation

	Runge Kutta	Linear 4 step
t = 0	0.07	0.01
t = 0.5	0.08	0.01
t = 1	9.92	4.67

Again the solution vector obtained by employing the linear 4 step integrator is overall the more accurate of the two.

The Lawson Runge Kutta has got a higher local order of accuracy than the linear 4 step method (5 as compared to 4) but here we are concerned with the total accumulated truncation error incurred by each method. As regards stability, both methods have intervals of absolute stability which are negative : (-5.7, 0) for the Lawson Runge Kutta and (-0.3,0) for the Adams-Bashforth explicit 4 step method (see Lambert (7)). But for the problems being considered here, all the eigenvalues are positive, i.e. $\bar{h} = h\lambda > 0$ for all λ, which means that both methods are absolutely unstable.

However, the linear 4 step method is relatively stable for any $\bar{h} \geqslant -0.214$ (see Stroud 12(a)) so that the integrations performed by this method were relatively stable. This meant that although the modulus of the accumulated truncation error of the method did not decay as the integration proceeded forward from t = 0, its rate of growth was limited to a rate similar to the rate of increase of the modulus of the exact solution. For the Runge Kutta method, however, there is no comparable definition of relative stability applicable as this is a one-step method (see Lambert (7), Chapter 4). This explains the marginally superior accuracy of the solution obtained by using the linear 4 step integrator over that obtained by the Runge Kutta method, in the solution of the above problems.

V CONCLUSIONS

The conclusions which can be drawn from the practical numerical experience, gained in solving the problems of the previous section, can be summarised as follows:

(1) The results show that microcomputer programs can be used to implement the theoretical methods described in the literature for the numerical solution of both well and ill-conditioned linear boundary value problems. With extra add-on memory in the form of micro-drives (up to 700K) now available, storage capacity will no longer be a serious problem but the chief drawbacks still remaining will be:-

(a) limited numerical accuracy. All present microcomputers store numbers to an accuracy of only 9 or 10 significant digits

(b) speed of calculation. If using a small step length h, program running times can be several hours. This is particularly true of the Spectrum.

Both of the above limit the range of problems that can be tackled.

(2) The choice of integration method to be used is obviously important since all the methods for solving linear boundary value problems involve several integrations of the system equations, so that any errors incurred here will affect the accuracy of the calculated solution vector obtained. The type of boundary value problem that we have considered throughout is one where at least one of the eigenvalues of the problem matrix will be positive in real part, whereas all intervals of absolute stability for both linear multi-step and Runge Kutta methods are strictly negative. Therefore, assuming the programs possess no facility for step control, when choosing an integration method we should ignore absolute stability characteristics and instead opt for a method which has:

(a) as large as possible local order of accuracy

(b) as large as possible interval of relative stability extending on both sides of t = 0, so as to cater for both positive and negative eigenvalues.

(3) No significant difference was found between the multiple shooting and reorthonormalisation methods as regards accuracy of solutions obtained, but the reorthonormalisation method certainly proved to be the quicker of the two in terms of program running time and this latter fact could be important commercially where cost has to be taken into consideration.

For the solution of an ill conditioned problem, with a large positive dominant eigenvalue, reverse multiple shooting was found to be particularly effective, though some success was also achieved with the reorthonormalisation method by increasing the number of reorthonormalisations employed.

The Riccati Transformation method produced acceptably accurate solutions to some of the test problems but it was found that for other problems the method compared unfavourably with single shooting due to a loss of accuracy caused by combination errors incurred in obtaining the surface vector u(t) from the equation:

$u(t) = U(t).\ x(t) + v(t)$, using the stored values of U(t) and v(t) from the forward integrations and those of x(t) from the backward integration. The greater the absolute rate of increase of the elements of matrix U(t) the more serious was the affect of this source of error on the computed solution.

(4) Although our results show that reasonably accurate solutions can sometimes be obtained from the multiple shooting and re-orthonormalisation methods by employing <u>equal</u> subintervals, it would obviously be preferable to be able to vary the lengths of the subintervals. Indeed, for the solution of more ill conditioned problems this would be essential. It

would certainly be possible to incorporate some of the re-orthonormalisation tests into the programs used in IV, so as to determine a better partition of the range, but this would inevitably greatly extend running times and so for the Spectrum, at any rate, would be impractical. Looking to the near future, however, when operation speeds of all microcomputers will no doubt be greatly increased, this would then certainly be a practical possibility and be worth investigating.

(5) As regards the longer term future solution of linear boundary value problems, one possible way in which solution times could be greatly reduced would be to employ parallel (as opposed to serial) computers, so enabling us to carry out several parts of a program simultaneously and then combine the results from each.

The multiple shooting method (in which, for N subintervals, the N similar initial value problems could be solved simultaneously) and the Riccati Transformation method (in which the Riccati equation and the associated $\dot{v}$ equation could be integrated forwards simultaneously) lend themselves ideally to this concept of parallel computation. Parallel computers are already available in mainframe form (e.g. the Control Data Cyber 205). If and when this concept can be extended to micro-computers, solution times for linear boundary value problems will be greatly reduced, so that even with present operation speeds more sophisticated programs, such as those utilising re-orthonormalisation tests and variable step lengths, will become feasible.

(6) Finally, if designing a 'package' to be used as an automatic solver of linear boundary value problems it would be inadvisable to rely solely on one solution method. Ideally, the program should incorporate two methods such as a variant of multiple shooting and re-orthonormalisation, each method to be used to solve the problem with several different step lengths and the vector of initial missing values computed by each method

for each step length. Only when these two vectors showed acceptable agreement could the user be satisfied that the average of the corresponding calculated solution vectors, obtained from the two methods, would provide an acceptably accurate solution to the given problem, at least over part of the range. However, if the problem is very ill-conditioned then the accuracy of the calculated solution vector towards the end of the range might still be suspect.

VI APPENDIX

1.2 Consider the n^{th} order differential equation

$y^n = F(t, y, \dot{y}, \ddot{y}, \ldots\ldots, y^{n-1})$ where F is linear in $y, \dot{y}, \ddot{y}, \ldots y^{n-1}$.

Let $y_1 = y$, $y_2 = \dot{y}$, $y_3 = \ddot{y}$, $\ldots$ $y_n = y^{n-1}$. Then

$$\dot{y}_1 = \dot{y} = y_2$$

$$\dot{y}_2 = \ddot{y} = y_3 \quad \text{etc}$$

$\dot{y}_n = y^n = F$, which is linear in $y_1, y_2, \ldots y_n$. Hence the given n^{th} order differential equation can be written in the form:

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \vdots \\ \dot{y}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & 0 \ldots & 0 \\ & & & & \ddots & \\ 0 & 0 & 0 & \ldots & \ldots 0 & 1 \\ K_1 & K_2 & K_3 & \ldots & \ldots & K_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ g(t) \end{bmatrix}$$

where K_i $(i = 1 \ldots n)$ are constants,

i.e. it can be written in the form: $\dot{y} = Ay + f$ of our problem I(i).

1.3 See Roberts and Shipman (6), Chapter 5.

1.4 We have: $\dot{y} = Ay + f$ (Ii) and $\dot{x} = -A^T x$ (iii)

Now: $\frac{d}{dt}(x^T . y) = x^T . \dot{y} + \dot{x}^T . y$

$$= x^T(Ay + f) - (A^T x)^T . y$$

$$= x^T Ay + x^T f - x^T Ay$$

$$= x^T f$$

$$\therefore \int_{t=0}^{t=1} \frac{d}{dt}\{x^T . y\} . dt = \int_{t=0}^{t=1} x^T . f\, dt$$

$$\Rightarrow \int_{0}^{1} d\,(x^{T}.y). \quad = \quad \int_{0}^{1} X.^{T}.f.dt$$

$$\Rightarrow \left[x^{T}\circ\, y\right]_{0}^{1} \quad = \quad \int_{0}^{1} x^{T}\circ f\ dt$$

$$\Rightarrow \quad x^{T}(1).\ y(1) \ - \ x^{T}(o).\ y(o) \quad = \quad \int_{o}^{1} x^{T}.f.dt$$

as required.

1.5 Consider the inhomogeneous system of equations $\dot{y} = Ay + f$. The general solution vector y(t) of the system is given by the variation of parameters formula as:

$$y(t) \ = \ N(t).y(t_{o}) \ + \ N(t)\int_{to}^{t} N^{-1}(s).f(s)\ ds \qquad (1)$$

where $t = t_{o}$ is the initial time. Likewise, the general solution vector of the corresponding homogeneous system, for which $f(t) \equiv o$ is given by : $y(t) = N(t).y(t_{o})$. N(t), the fundamental matrix of homogeneous vectors, satisfies the matrix equation: $\overset{\circ}{N} = AN$, $N(t_{o}) = I$, since N(t) is obtained by integrating the system $\dot{y} = Ay$ forwards starting from the Kronecker delta vectors at $t \doteq t_{o}$. From this we can deduce (see Lemma) that the general solution vector of the adjoint system $\mathring{x} = -A^{T}x$ is given by: $x(t) = N(t)^{-T}.\ x(t_{o})$ (2), where $N^{-T} = (N^{-1})^{T}$ or $(N^{T})^{-1}$ and $\left[N(t_{o})\right]^{-T} = I$.

Now multiply (1) by $x(t)^{T}$:

$$x^{T}(t).y(t) = x^{T}(t)N(t)\ y(t_{o}) \ + \ \int_{to}^{t} x^{T}(t).N(t)\ N^{-1}(s)f(s)ds \qquad (3)$$

From (2): $x^{T}(t) \doteq x^{T}(t_{o}).\ N^{-1}(t)$

$$\Rightarrow \quad x^{T}(t).\ N(t) \ = \ x^{T}(t_{o}) \qquad (4)$$

Substitute in (3):

$$x^T(t).y(t) = x^T(t_o)y(t_o) + \int_{to}^{t} x^T(t)\ N(t).\ N^{-1}(s).f(s).ds \qquad (5)$$

From (2), putting t = s:

$$x(s) = N(s)^{-T}.\ x(t_o)$$

But from (4) : $x(t_o) = N^T(t).\ x(t).$

$$\therefore \quad x(s) = N^{-T}(s).\ N^T(t).x(t)$$

$$\Rightarrow \quad x^T(s) = x^T(t).\ N(t).\ N^{-1}(s)$$

Substitute in (5):

$$x^T(t).y(t) = x^T(t_o).y(t_o) + \int_{t_o}^{t} x^T(s).f(s)\ ds$$

i.e. $$x^T(t).y(t) - x^T(t_o).y(t_o) = \int_{t_o}^{t} x^T(s).f(s)\ ds$$

which reduces to the basic adjoint identity if we let $t = t_f$.

Lemma: Let the general solution vector of the adjoint homogeneous system $\dot{x} = -A^T x$ be given by $x(t) = M(t).\ x(t_o)$ where $M(t)$ satisfies the equation: $\dot{M} = -A^T M$ (6), where $M(t_o) = I$.

Now $M = (N^{-1})^T \Rightarrow M^T = N^{-1} \Rightarrow \dot{M}^T = \frac{d}{dt}(N^{-1})$

$$\Rightarrow \quad \dot{M}^T = -N^{-1}.\dot{N}.N^{-1}, \text{ using } N.N^{-1} = I$$

$$\Rightarrow \quad \dot{M}^T = -N^{-1}.\ A \quad (\because \ \dot{N} = AN)$$

$$\Rightarrow \quad \dot{M} = -A^T (N^{-1})^T$$

Thus $M = (N^{-1})^T$ satisfies equation (6).

Also $N(t_o) = I \Rightarrow N^{-T}(t_o) = I \Rightarrow M(t_o) \equiv I$

$\therefore$ The general solution vector of the adjoint system can be written $x(t) = N^{-T}(t).\, x(t_0)$ where $N^{-T}(t_o) = I.$

1.6 Consider the system of differential characteristic equations:

$$\left.\begin{aligned} \dot{u}(t) &= A(t)u + B(t)x + F(t) \\ \dot{x}(t) &= C(t).u + D(t)x + G(t) \end{aligned}\right\} \qquad \begin{aligned} (1) \\ (2) \end{aligned}$$

Suppose this system is integrated forwards from $t = o$ starting from $u(o) = a$ (fixed) and $x(o) = s$ (variable). For each choice of s, we obtain $u(t,s)$ and $x(t,s)$, for any $t > o$. By eliminating s between these two we can obtain, for any value of t, a connection between $u(t)$ and $x(t)$. Suppose this connection is given by:

$u(t) = U(t)\, x(t) + v(t)$ (3), for some matrix U and some vector v, of dimensions $(m \times p)$ and $(m \times 1)$ respectively.

Now from (3): $\dot{u} = \dot{U}x + U\dot{x} + \dot{v}$

Substitute from (1) and (2):

$Au + Bx + F = \dot{U}x + U\left[Cu + Dx + G\right] + \dot{v}$

Substitute from (3):

$A(Ux + v) + Bx + F = \dot{U}x + UC(Ux + v) + UDx + UG + \dot{v}$

$\Rightarrow \left[AU + B - \dot{U} - UCU - UD\right].\, x = UCv + UG + \dot{v} - Av - F$

This equation must be true for <u>all</u> $x(t)$.

$\therefore$ $AU + B - \dot{U} - UCU - UD = o$

<u>and</u> $UCv + UG + \dot{v} - Av - F = o$

i.e. $\dot{U} = B + AU - UD - UCU$ (4) (Riccati)

and $\dot{v} = (A-UC)v - UG + F$ (5)

From (3): $u(o) = U(o)x(o) + v(o)$ where $u(o) = a$

$\therefore$ $v(o) = a$ corresponds to $U(o) = o$.

Thus, corresponding values of $U(t)$ and $v(t)$ will be obtained by integrating equation (4) forwards from $U(o) = o$ and equation (5) forwards from $v(o) = a$.

1.8 For a detailed discussion of initial value problems consult J.D.Lambert (7).

2.1 For a proof of this result consult Noble (12).

2.2 If $M \equiv L_o + L_1 N(1)$ is non singular and if rank $(L_o, L_1) = n$ and rank $(L_o + L_1) = m \leqslant n$, where n is the dimension of the problem, then $\alpha[M] \leqslant K_o \alpha[N(1)]$ where in general we may assume that K is small compared with $\alpha[N(1)]$. The above result holds for $(L_o + L_1)$ singular or non singular and it may be found in George and Gunderson's paper (18).

2.3 For a discussion of this result consult Lambert (7), Chapter 1.

3.1 Gram Schmidt process: In order to convert the linearly independent vector set y^k $(k = 1 \ldots N)$ into the corresponding set z^k $(k = 1 \ldots N)$ we must define the auxiliary vector set α^k and a scalar set w_{kk}. In practice, the transformation is then effected by applying the following set of equations recursively $(k = 1 \ldots N)$:

$$\left.\begin{aligned} \alpha^k &= y^k - \sum_{s=1}^{k-1} (y^k \cdot z^s)\, z^s && \text{(i)} \\ w_{kk} &= \left(\alpha^k \cdot \alpha^k\right)^{\frac{1}{2}} && \text{(ii)} \\ z^k &= \frac{\alpha^k}{w_{kk}} && \text{(iii)} \end{aligned}\right\}$$

Thus from y^1 we obtain α^1 and w_{11} and hence z^1. Then from y^2 and z^1 we obtain α^2 and w_{22} and hence z^2 etc. Each z^k $(k = 1 \ldots N)$ depends on y^k and on $z^1 \ldots\ldots Z^{k-1}$, where each z^s $(s = 1 \ldots k-1)$ depends on y^s, so that in effect z^k depends on $y^1 \ldots\ldots y^k$. This is why the transformation matrix

P(N x N) which converts the set $Y = \{y^1 \ldots y^N\}$ into the set $\mathfrak{z} = [\mathfrak{z}^1 \ldots \mathfrak{z}^N]$ is lower triangular i.e.

$$\begin{bmatrix} \mathfrak{z}^1 \\ \vdots \\ \vdots \\ Z^N \end{bmatrix} = \begin{bmatrix} P_{11} & & & \bigcirc \\ P_{21} & P_{22} & & \\ P_{31} & P_{32} & P_{33} & \\ \vdots & & & \ddots \\ P_{N1} & P_{N2} & P_{N3} \cdots & P_{NN} \end{bmatrix} . \begin{bmatrix} y^1 \\ \vdots \\ \vdots \\ y^N \end{bmatrix}$$

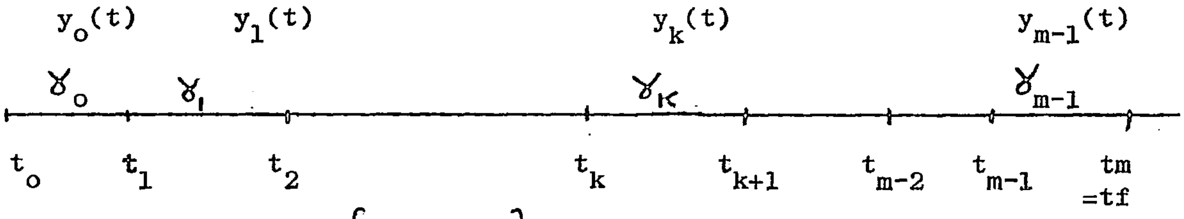
or $\mathfrak{z} = PY$ where $\mathfrak{z}$ and Y are each (N x 1) with components (n x 1).

For the operation of Conte's method of re-orthonormalisation it is more convenient to have the transpose of the above equation

i.e. $\mathfrak{z}^T = Y^T . P^T$ where now

P^T is <u>upper</u> triangular and $\mathfrak{z}^T$ and Y^T are both (1 x N)

3.2 Justification of Conte's Re-orthonormalisation method



In each sub interval $[t_k , t_{k+1}]$, k = o.... m-1, the solution vector $y_k(t)$ was given by:

$$y_k(t) = V_k(t) + U_k(t).\,\gamma_k \quad \text{for } t_k \le t \le t_{k+1}, \text{ where}$$

$V_k(t)$ and $U_k(t)$ were the inhomogeneous and homogeneous vectors respectively, obtained and stored for that sub interval.

Let us denote the solution vectors immediately before and after the final re-orthonormalisation at $t = t_m$ by $y(t_m-)$ and $y(t_m+)$ respectively.

Then $y(t_m+) = V_{new}(t_m) + U_{new}(t_m) \circ \gamma_m$ (1)

where $\gamma_m = \gamma_f$ was the vector whose components were determined by matching with the given terminal values of the problem, and the suffix 'new' denotes the vectors obtained after the re-orthonormalisation. Now from the re-orthonormalisation process we have:

$$V_{new}(t_m) = V_{old}(t_m) - U_{new}(t_m) \circ W_m \qquad (2)$$

and $U_{new}(t_m) = U_{old}(t_m) \cdot P_m^T$ (3)

where P_m^T and W_m were the orthonormalisation matrix and projection vector respectively used at $t = t_m$.

Substituting (2) into (1) gives us:

$$\begin{aligned} y(t_m+) &= V_{old}(t_m) - U_{new}(t_m) \cdot W_m + U_{new}(t_m) \circ \gamma_m \\ &= V_{old}(t_m) + U_{new}(t_m) \left[\gamma_m - W_m \right] . \end{aligned}$$

And substituting (3) into the above gives:

$$y(t_m+) = V_{old}(t_m) + U_{old}(t_m) \cdot P_m^T \cdot \left[\gamma_m - W_m \right] \qquad (4)$$

Now if we define γ_{m-1} by

$\gamma_{m-1} = P_m^T \cdot \left[\gamma_m - W_m \right]$ (4^1), then equation (4) becomes:

$$y(t_m+) = V_{old}(t_m) + U_{old}(t_m) \cdot \gamma_{m-1} \circ \qquad (5)$$

But for any t in $\left[t_{m-1}, t_m \right]$ we have:

$$y_{m-1}(t) = V_{m-1}(t) + U_{m-1}(t) \cdot \gamma_{m-1}$$

$$\Rightarrow \quad y(t_m-) = V_{old}(t_m) + U_{old}(t_m) \circ \gamma_{m-1} \qquad (6)$$

where 'old' denotes vectors before reorthonormalisation at $t = t_m$.

Thus by comparing equations (5) and (6) we see that the definition of γ_{m-1} in terms of γ_m, W_m and P_m^T, as given in equation (4^1), ensures that at $t = t_m$: $y(t_m+) = y(t_m-)$ i.e. that y(t) is continuous just before and just after re-orthonormalisation at $t = t_m$. Also since $y_{m-1}(t)$ is continuous in $\left[t_{m-1}, t_m\right]$ the combination vector γ_{m-1}, as found for continuity at t_m, must be constant throughout this sub interval so that:

$$y(t_{m-1}^{+}) = V_{new}(t_{m-1}) + U_{new}(t_{m-1}).\gamma_{m-1}$$,'where 'new' denotes vectors after reorthonormalisation at $t = t_{m-1}$'

The above argument can now be repeated at the nodes $t = t_{m-1}, t_{m-2}, \ldots, t_1$ successively, and we see that at each node the γ_i vector as obtained from the iteration equation: $\gamma_i = P_{i+1}^T\left[\gamma_{i+1} - W_{i+1}\right]$, $i = (m-2)\ldots\ldots o$, and as employed in Conté's method, ensures the continuity of the y(t) vector just before and just after each re-orthonormalisation, and γ_i must be constant throughout its sub interval. In particular, in the first sub interval $\{t_o, t_1\}$ the vector γ_o, as obtained from γ_1, W_1 and P_1^T at t_1, is applicable at t_o and so we have:

$$y_o(t_o) = V_o(t_o) + U_o(t_o).\gamma_o \text{ where}$$

$V_o(t_o) = \left[\alpha_1 \ldots \ldots \alpha_r \; o \ldots\ldots o\right]^T$ and $U_o(t_o)$ are the special Kronecker delta vectors i.e.

$$\left[\begin{array}{c} y_1 \\ \vdots \\ y_r \\ \hline y_{r+1} \\ \vdots \\ y_n \end{array}\right] = \left[\begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_r \\ \hline 0 \end{array}\right] + \left[\begin{array}{cccc} & 0 & & \\ \hline 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{array}\right]\left[\begin{array}{c} \gamma_1 \\ \vdots \\ \hline \vdots \\ \gamma_{n-r} \end{array}\right]$$

where $\gamma_o = \left[\gamma_1 \ldots\ldots \gamma_{n-r}\right]^T$.

Hence $\gamma_o = \left[y_{r+1} \;--\; y_n \right]^T$, the vector of missing initial values for the problem.

Thus, the fact that the original γ vector, γ_f, is found by matching with the given terminal values of the problem and that the subsequent γ_i vectors (obtained from γ_f by backward iteration using the stored P_i matrices and W_i vectors) ensure the continuity of the individual sub interval solution vectors $y^i(t)$ from $\left[t_{m-1}, t_m \right]$ through to $\left[t_o, t_1 \right]$, means that these solution vectors form a continuous solution vector y(t) which is the required solution of the problem over the complete interval $\left[t_o, t_f \right]$.

(A detailed discussion of the above may be found in Conte's paper (13)).

3.3 Let $s = \dfrac{t - t_{i-1}}{\Delta_i}$ (1) where $\Delta_i = t_i - t_{i-1}$

$= \text{length of } i^{th} \text{ sub interval}$

$(i = 1 \_\_ N)$

and let $\dot{y}(t) = A(t)y + f = F(t)$.

In <u>any</u> sub interval, from (1):

$$\frac{ds}{dt} \equiv \frac{1}{\Delta_i}.$$

Also for any function F(t) : $\dfrac{dF}{dt} = \dfrac{dF}{ds} \cdot \dfrac{ds}{dt}$

$$= \frac{1}{\Delta_i} \frac{dF}{ds}$$

$$\therefore \quad \frac{d}{dt} \equiv \frac{1}{\Delta_i} \cdot \frac{d}{ds}$$

In any sub interval: $\dfrac{dy(t)}{dt} = F(t)$

$$\Rightarrow \quad \frac{1}{\Delta_i} \frac{d}{ds} \cdot y(s) = F(s) \text{ where } t = \Delta_i s + t_{i-1}$$

$$\Rightarrow \quad y'(s) = \Delta_i F(s)$$

i.e. $y'(s) = \Delta_i \left[A(s)\, y(s) + f(s) \right]$ (2)

where $'$ denotes $\dfrac{d}{ds}$, and $o \leqslant s \leqslant 1$.

Thus to find the successive general solution vectors for the N sub intervals we can integrate the different sets of equations (2) ($i = 1 - N$) over the same interval $[o,1]$ each time. For example, in the first sub interval $[t_o, t_1]$, $i = 1$ so $t = \Delta_1 s + t_o$, where $\Delta_1 = t_1 - t_o$ is known. We transform the differential equations from t variable to s variable by using this substitution and then integrate forwards the set $y^{d}(s) = \Delta_i \cdot \left[A(s).y(s) + f(s)\right]$ over $[o,1]$.

3.4 For each sub interval separately ($i = 1 - N$) we have the fundamental matrix $N_i(s)$, the particular solution vector $W_i(s)$, the general solution vector $y_i(s)$, and the combination vector $B_i = y_i(o)$, $o \leqslant s \leqslant 1$.

The solution interval vectors to be matched up are given by:

$$y_i(S) = N_i(s)B_i + W_i(s) \qquad (1)$$

where $N_i(o) = I$, $W_i(o) = 0$ $\qquad (i = 1 - N)$
$\qquad (o \leqslant s \leqslant 1)$

At the internal nodes $t_1 - - - t_{N-1}$ we have for continuity:

$$y_i(1) = y_{i+1}(o) \qquad i = 1 - - - (N-1)$$

$$\Rightarrow \qquad N_i(1)B_i + W_i(1) = B_{i+1}$$

$$\text{or} \qquad W_i(1) = -N_i(1)B_i + B_{i+1} \quad (2), \; i = 1 - - - (N-1)$$

Also the given boundary conditions of our problem viz. $L_o y(o) + L_1 y(1) = C$ now become

$$L_o y_1(o) + L_1 y_N(1) = C$$

$$\Rightarrow \quad L_o B_1 + L_1 \left[N_N(1) B_N + W_N(1)\right] = C \;, \text{ from } (1) \;;$$

$$\Rightarrow \quad L_o B_1 + L_1 N_N(1) B_N = C - L_1 W_N(1) \qquad (3)$$

Equation (3) plus the set of equations (2) can be written together as:

$$\begin{bmatrix} L_o & & & & & L_1 N_N \\ -N_1 & I & & & & \\ & -N_2 & I & & & \\ & & -N_3 & I & & \\ & & & \ddots & \ddots & \\ O & & & & -N_{N-1} & I \end{bmatrix} \begin{bmatrix} B_1 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} C - L_1 W_N \\ W_1 \\ W_2 \\ \vdots \\ W_{N-1} \end{bmatrix} \qquad (4)$$

where all N_i matrices and W_i vectors are evaluated at the end point (s=1) of their respective sub intervals and N_i are (n x n) and W_i are (n x 1). Now if we denote the matrix on the L.H.S. of the above equation (4) by E(N x N) and we define the matrices $\bar{L}_o$ (N X N), $\bar{L}_1$ (N x N) and $\bar{N}(1)$ (N x N), as follows:

$$\bar{L}_o = \begin{bmatrix} L_o & & & & O \\ & I & & & \\ & & I & & \\ & & & I & \\ & & & & \ddots & \\ O & & & & & I \end{bmatrix}$$

each element (nxn)

$$\bar{L}_1 = \begin{bmatrix} 0 & & & & & L_1 \\ -I & 0 & & & O & \\ & -I & 0 & & & \\ & & -I & 0 & & \\ & & & \ddots & \ddots & \\ O & & & & -I & 0 \end{bmatrix}$$

each element (nxn)

$$\bar{N}\ (1) = \begin{bmatrix} N_1(1) & & & & \bigcirc \\ & N_2(1) & & & \\ & & N_3(1) & & \\ & & & \ddots & \\ \bigcirc & & & & N_N(1) \end{bmatrix} \quad \text{each element (nxn)}$$

and we also define the vectors $\bar{C}$ (N x 1), and $\bar{W}(1)$, (N x 1), as shown:

$$\bar{C} = \begin{bmatrix} C \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}, \quad \bar{W}\ (1) = \begin{bmatrix} W_1(1) \\ W_2(1) \\ \vdots \\ \vdots \\ W_N(1) \end{bmatrix}$$

where each element is (n x 1), then it can be seen that:

$$E = \bar{L}_o + \bar{L}_1 \circ \bar{N}\ (1).$$

Further, if we denote the vector on the R.H.S. of equation (4) by $\bar{T}$ then:

$$\bar{T} = \bar{C} - \bar{L}_1 . \bar{W}(1)$$

Therefore equation (4) becomes:

$$\left[\bar{L}_o + \bar{L}_1 \circ \bar{N}\ (1) \right] \circ \bar{B} = \bar{C} - \bar{L}_1 \circ \bar{W}\ (1) \qquad (5)$$

where $\bar{B} = \left[B_1, B_2 \ldots\ldots\ldots B_N \right]^T$ is (N x 1)

with each element (n x 1).

Finally (5) can be reduced to the form given in the text viz: $\bar{M} \circ \bar{B} = \bar{T}$

where $\bar{M} = \bar{L}_o + \bar{L}_1\ \bar{N}\ (1)$, $\bar{T} = \bar{C} - \bar{L}_1\ \bar{\bar{W}}\ (1)$.

Thus for multiple shooting, with N sub intervals, our linear boundary value problem (I) could be restated as follows:

Solve $\bar{Y}' = \bar{A}(s)\,\bar{Y}(s) + \bar{F}(s)$ over $[o,1]$

subject to $\bar{L}_o\,.\bar{Y}(o) + \bar{L}_1\,\bar{Y}(1) = \bar{C}$

where $\bar{L}_o$, $\bar{L}_1$ are the giant (Nn x Nn) matrices previously defined and

$$\bar{Y}(s) = \left[y_1(s),\ y_2(s),\ \ldots\ldots\ y_N(s)\right]^T \quad ,$$

$$\bar{F}(s) = \left[f_1(s),\ f_2(s),\ \ldots\ldots\ f_N(s)\right]^T$$

& $$\bar{A}(s) = \begin{bmatrix} A_1(s) & & \bigcirc \\ & A_2(s)\ \ddots & \\ \bigcirc & & A_N(s) \end{bmatrix}$$

where $\bar{Y}(o) = \bar{B}$.

3.5. Corresponding to the vector norms $//x//_1 = \sum_{i=1}^{n} /x_i/$ and $//x//_\infty =$ $= \max_i |x_i|$ where x is (n x 1) we have the associated matrix norms:

$$//A//_1 = \max_j \sum_{i=1}^{n} /a_{ij}/ \text{ and } //A//_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$$ where A is (n x n). We can now define the measure function $\mu[A]$ of matrix A corresponding to these norms as:

$$\mu_1[A] = \max_j \left[a_{jj} + \sum_{\substack{i \neq j \\ i=1}}^{n} |a_{ij}| \right]$$ and

$$\mu_\infty[A] = \max_{i} \left[a_{ii} + \sum_{\substack{j\neq i \\ j=1}}^{n} /a_{ij}/ \right]$$

In the following, either norm is applicable.

Gunderson (18), now obtains the results:

$$//N(1)// \leqslant \exp \int_0^1 \mu[A(s)] . ds \text{ and}$$

$$\left\| N^{-1}(1) \right\| \leqslant \exp \int_0^1 \mu\left[-A(s)\right]$$

ds, where N(t) is the fundamental matrix of homogeneous vectors in single shooting. Using the condition number definition

$$\alpha\left[A\right] = //A//. //A^{-1}// \qquad \text{we thus get:}$$

$$\alpha\left[N(1)\right] \leqslant \exp\left\{ \int_0^1 \left(\mu\left[A(s)\right] + \mu\left[-A(s)\right]\right) ds \right\} \quad (1) .$$

Now from (2.2):

$$\alpha\left[L_0 + L_1 N(1)\right] \leqslant K . \alpha\left[N(1)\right] \quad (2),$$

where K is assumed small compared to $\alpha\left[N(1)\right]$.

From (1) and (2) therefore:

$$\alpha\left[L_0 + L_1 N(1)\right] \leqslant K \exp\left\{ \int_0^1 \left(\mu\left[A(s)\right] + \mu\left[-A(s)\right]\right) ds \right\} \quad (3) .$$

Thus the exponential term on the R.H.S. of (3) provides an upper bound for $\alpha\left[L_0 + L_1 N(1)\right]$, which is the condition number of the solution matrix M in single shooting.

Now to consider multiple shooting with N sub intervals we have from (3.4):

$$\bar{N}(1) = \begin{bmatrix} N_1(1) & & & \bigcirc \\ & N_2(1) & & \\ & & \ddots & \\ \bigcirc & & & N_N(1) \end{bmatrix}$$

$$\bar{N}^{-1}(1) = \begin{bmatrix} N_1^{-1}(1) & & & \bigcirc \\ & N_2^{-1}(1) & & \\ & & \ddots & \\ \bigcirc & & & N_N^{-1}(1) \end{bmatrix}$$

Thus $//\bar{N}(1)// = \max_k //N_k(1)//, \; K = 1 \ldots N$

and $//\bar{N}^{-1}(1)// = \max_j //N_j^{-1}(1)// \;, \; j = 1 \ldots N$

$\therefore \alpha\left[\bar{N}(1)\right] = \max_k //N_k(1)// \; . \; \max_j //N_j^{-1}(1)//$

But $//N_k(1)// \leqslant \exp \int_{t_{k-1}}^{t_k} \mu\left[A(s)\right] . \, ds$ and

$//N_j^{-1}(1)// \leqslant \exp \int_{t_{j-1}}^{t_j} \mu\left[-A(s)\right] . \, ds$ since the sub interval of integration for $N_k(s)$ is $\left[t_{K-1}, t_k\right]$ and likewise for $N_j(s)$.

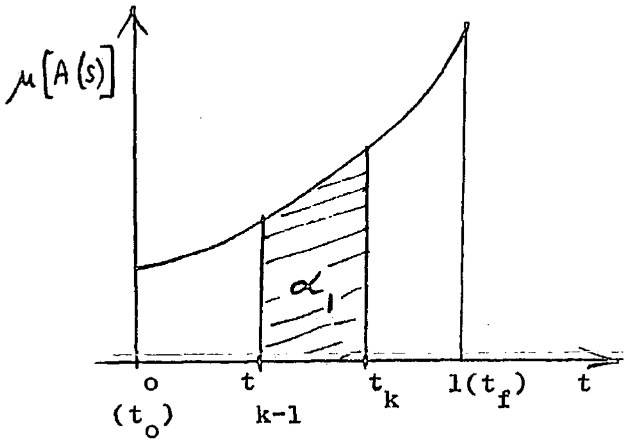
$$\therefore \alpha\left[\bar{N}(1)\right] \leqslant \exp \left\{ \max_k \int_{t_{k-1}}^{t_k} \mu\left[A(s)\right] . \, ds + \max_j \int_{t_{j-1}}^{t_j} \mu\left[-A(s)\right] . \, ds \right\} \quad (4)$$

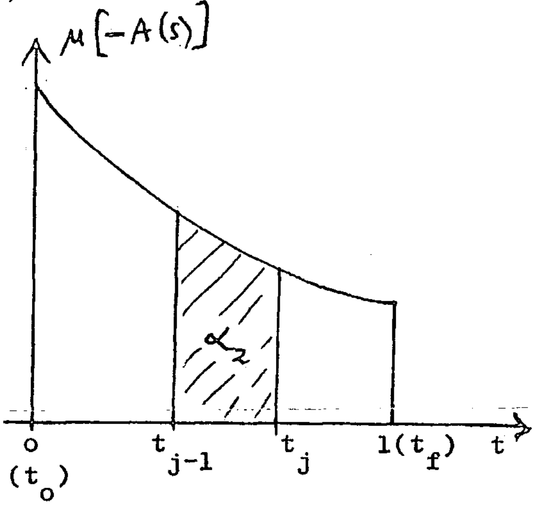
Also, analogous to result (2.2), we have for multiple shooting:

$\alpha\left[\bar{L}_o + \bar{L}_1 \, . \, \bar{N}(1)\right] \leqslant K_1 \, . \, \alpha\left[\bar{N}(1)\right]$ where K_1 is small compared to $\alpha\left[\bar{N}(1)\right]$.

Thus: $\alpha\left[\bar{L}_o + \bar{L}_1 \, . \, \bar{N}(1)\right] \leqslant K_1 \, E$, (5), where E, the exponential term on the R.H.S. of (4), provides an upper bound for the condition number of $\bar{L}_o + \bar{L}_1 \; \bar{N}(1)$ which is the solution matrix $\bar{M}$ for multiple shooting.

Now in considering the size of E, if either $\mu[A(s)]$ or $\mu[-A(s)]$ is negative over $[o,1]$ then it can be ignored since its contribution will be very small. Thus we need only consider the case where both measure functions are positive, as shown. The <u>total</u> areas under the graphs of $\mu[(A(s))]$ and $\mu[-A(s)]$, from $t = t_o$ to $t = t_f$, are denoted by α and β respectively.





Suppose that the maximum sub interval areas are α_1 and α_2 respectively as shown.

Then we have, from equation (3):

for single shooting $\alpha[M] \leq K. \exp(\alpha + \beta)$,

whereas for multiple shooting, from equation (5):

$$\alpha[\bar{M}] \leq K_1 \cdot \exp(\alpha_1 + \alpha_2).$$

Therefore, it is very likely that

$$\alpha[\bar{M}] < \alpha[M], \text{since } \alpha_1 + \alpha_2 < \alpha + \beta$$

for any partition.

This means that, theoretically, <u>any</u> partition for multiple shooting <u>should</u> produce a better conditioned solution matrix $\bar{M}$ than the solution matrix M used in single shooting. To ensure that $\alpha[\bar{M}]$ is the least possible for fixed N i.e. to obtain the best possible conditioning with a given number of sub intervals, we require the partition such that $(\alpha_1 + \alpha_2)$ is a minimum. Thus if it were known that $\mu[A(s)]$ and $\mu[-A(s)]$ both monotonically increased over the full interval $[t_o, t_f]$ then, in order to minimise $(\alpha_1 + \alpha_2)$, as t increased towards t_f the lengths of the sub intervals would have to be progressively decreased.

3.6 Suppose that we have started the solution of the problem by using the normal transformation U(t) over sub interval $[o, t_1]$, and at $t = t_1$ we wish to switch to $U^{-1}(t)$. Let $U(t_1) = K$, so that at $t = t_1$:

$$\dot{K} = B + AK - KD - KCK \quad (1)$$

, since K satisfies the normal Riccati equation (ICV). Now assume that $W(t_1) = J$, then:

$$\dot{J} = C + DJ - JA - JBJ \quad (2)$$

, since J will satisfy the inverse Riccati equation (IIIdi).

From (2):

$$J^{-1}\dot{J}J^{-1} = J^{-1}CJ^{-1} + J^{-1}D - AJ^{-1} - B$$

$$\Rightarrow \quad -J^{-1}\dot{J}J^{-1} = B + AJ^{-1} - J^{-1}D - J^{-1}CJ^{-1} \quad (3)$$

But $\frac{d}{dt}(J . J^{-1}) = \frac{d}{dt}(I) = 0$

$$\Rightarrow \quad J . (J^{-1})' + J' . J^{-1} = 0$$

, where $'$ denotes $\frac{d}{dt}$.

$$\Rightarrow \quad J . (J^{-1})' = -J' . J^{-1}$$

$$\Rightarrow \quad (J^{-1})' = -J^{-1} . J' . J^{-1}$$

$\therefore$ From (3):

$$(J^{-1})' = B + AJ^{-1} - J^{-1}D - J^{-1} . C . J^{-1}$$

By comparing this equation with equation (1) we see that $K = J^{-1}$ or $J = K^{-1}$

i.e. at $t = t_1$: $W(t_1) = U^{-1}(t_1)$.

Thus when we switch from U transformation to U^{-1} transformation at t_1 the integration of the inverse Riccati equation starts from $W = U^{-1}(t_1)$.

3.7 See Meyer (4) Chapter 1.3.

REFERENCES

Books

1. 'An introduction to numerical methods for differential equations" (Ortega and Poole)

2. 'A first look at numerical functional analysis' (W.W.Sawyer)

3. 'Computational Methods in Engineering Boundary Value Problems' (Na)

4. 'Initial Value Methods for Boundary Value Problems (Meyer)

5. 'Computational Techniques for Ordinary Differential Equations' (Gladwell and Sayer : section by Fox)

6. 'Two Point Boundary Value ProblemS: Shooting Methods' (Roberts and Shipman)

7. 'Computational Methods in Ordinary Differential Equations' (Lambert)

8. 'Collected problems in numerical methods' (Cherkasova)

9. 'A Handbook of Numerical Matrix Inversion and Solution of Linear Equations' (Westlake)

10. 'Numerical Solution of Differential Equations' (Fried)

11. 'Numerical Methods for Two Point Boundary Value Problems' (Keller)

12. 'Applied Linear Algebra (Noble)

12(a). 'Numerical Quadrature and Solution of Ordinary Differential Equations' (A.H.Stroud)

Papers

13. 'The Numerical Solution of Linear Boundary Value Problems' (Conte : Siam Vol 8, No 3, 1966)

14. 'Solution of Ill-Conditioned Linear Two-point Boundary Value Problems by the Riccati Transformation' (Mufti, Chow & Stock : Siam Vol . 11, No 4, 1969)

15. 'Computational Solution of Linear Two-Point Boundary Value Problems via Orthonormalisation' (Scott & Watts : Siam : Vol 14 : No.1 : 1977)

16. 'A Classification and Survey of Numerical Methods for Boundary Value Problems in Ordinary Differential Equations' (Aktas & Stetter : Int. Journal for Numerical Methods in Eng. Vol 11, 1977)

17. 'Comparing Routines for the Numerical Solution of Initial Value Problems of Ordinary Differential Equations in Multiple Shooting' (Diekhoff et al. Numer. Math. 27, 1977)

18. 'Conditioning of Linear Boundary Value Problems' (George & Gunderson, Bit 12 (1972))