Application of cubic B-splines to boundary value and optimal control problems

Deborah Ann Skidmore

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/854
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
APPLICATION OF CUBIC B-SPLINES TO
BOUNDARY VALUE AND OPTIMAL
CONTROL PROBLEMS

by

Deborah A. Skidmore

Bachelor of Science
University of Nevada, Las Vegas
1996

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in

Mathematics

Department of Mathematics
University of Nevada, Las Vegas
May 1998
The Thesis prepared by

Deborah A. Skidmore

Entitled

Application of Cubic B-Splines to Boundary Value
and Optimal Control Problems

is approved in partial fulfillment of the requirements for the degree of

Master of Science

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Application of Cubic B-Splines to Boundary Value and Optimal Control Problems

by

Deborah A. Skidmore

Dr. Rohan Dalpatadu, Examination Committee Chair
Associate Professor of Mathematics
University of Nevada, Las Vegas

In this thesis, a one-dimensional second order differential two-point boundary value problem will be approximated by a variational method using cubic B-splines. This method will then be extended to solving N-dimensional two-point boundary value problems. In the final part of the thesis, certain control problems will be converted into boundary value problems, which in turn, will be approximated by the previous variational method which uses cubic B-splines.
LIST OF FIGURES

Figure 1:  The normalized graph of the cubic B-spline, $\hat{\phi}_i$, centered at $t_i$ .................. 11
Figure 2:  The graph of $\phi_0$ .............................................................................................................. 15
Figure 3: The graph of $\phi_1$ ............................................................................................................... 15
Figure 4: The graph of $\phi_{n-1}$ ......................................................................................................... 15
Figure 5: The graph of $\phi_n$ ............................................................................................................. 16
TABLE OF CONTENTS

ABSTRACT ..................................................................................................................................... iii
LIST OF FIGURES ....................................................................................................................... iv
ACKNOWLEDGEMENTS ...........................................................................................................

CHAPTER 1 INTRODUCTION ................................................................................................ 1

CHAPTER 2 BEST APPROXIMATES OF BOUNDARY VALUE PROBLEMS............. 3
  Variational Methods ........................................................................................................ 3
  Rayleigh-Ritz Method ..................................................................................................... 4
  Existence and Uniqueness of Best Approximates ..................................................... 4
  The Inner Product Space ............................................................................................. 8
  Reformulation of the Boundary Value Problem ....................................................... 9
  Cubic B-splines............................................................................................................. 11

CHAPTER 3 EXTENSION OF THE RAYLEIGH-RITZ METHOD......................... 19
  Transformation of the General Boundary Value Problem ...................................... 19
  Examples for One-Dimensional Boundary Value Problems .................................. 21
  Systems of Equations ................................................................................................. 22

CHAPTER 4 APPLICATION TO OPTIMAL CONTROL PROBLEMS ...................... 27
  The Optimal Control Problem .................................................................................. 27
  The Case: B is Invertible ........................................................................................... 28
  Examples for Optimal Control Problems .................................................................. 29
  The Case: Rank of B is less than n ......................................................................... 33

APPENDIX I RESULTS ........................................................................................................... 39
  Table I - Example 3.1 ............................................................................................... 39
  Table II - Example 3.2 ............................................................................................. 40
  Table III - Example 3.3 ............................................................................................ 41
  Table IV - Example 4.1 ............................................................................................ 42
  Table V - Example 4.2 ............................................................................................. 43
  Table VI - Example 4.3 ............................................................................................ 44

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
APPENDIX II FORTRAN 77 PROGRAMS........................................................................ 45
Main Program............................................................................................................... 45
List of Functions............................................................................................................. 48
Subroutine - Bandsolver.............................................................................................. 55
Subroutine - Error.......................................................................................................... 56
Subroutine - Gaussj....................................................................................................... 57
Subroutine - Gseidel...................................................................................................... 59
Subroutine - L Ud ecomp............................................................................................... 60
Subroutine - MatrixE.................................................................................................... 62
Subroutine - Polint........................................................................................................ 64
Subroutine - Qromb...................................................................................................... 65
Subroutine - Transform............................................................................................... 66
Subroutine - Trapzd...................................................................................................... 67
Subroutine - VectorB.................................................................................................... 68

REFERENCES.................................................................................................................. 70

VITA........................................................................................................................................ 71
ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my undergraduate, graduate and thesis advisor, Dr. Rohan Dalpatadu, for all of the help and guidance that he has given me throughout my studies at UNLV. I also would like to thank both he and Dr. Bowman for getting me interested in attending graduate school.

I would also like to thank Dr. Douglas Burke for freely giving his time to helping me learn. Similarly, I would like to thank Dr. Miel, Dr. Bhatnager, and all of my other professors for devoting their time to teaching, for knowledge is truly the best gift that anyone could receive.

Also, I would like to thank Trevor Wilcox, Ross Iaci, and all of the Math Department secretaries for making graduate school not only interesting but also fun.

Last, but not least, I would like to thank my husband and parents for all of the love and support that they have given me throughout the years.
CHAPTER 1

INTRODUCTION

Much of the phenomenon that occurs in nature can be expressed in terms of differential equations, so it is not surprising that there is an interest in how to effectively approximate differential equations. For years, mathematicians and engineers have studied and devised various methods to approximate differential equations with two point boundary conditions. Some of the more common methods that they devised have problems that make the application of them difficult, such as: their lack of stability and their inability to produce a solution with a “tolerable” accuracy. Even the method that is currently used the most for applications, the Finite Difference method, has its problems. While this method is fairly stable, it is extremely tedious to implement. Much work is required to obtain a solution with a specified accuracy. Consequently, the purpose of this thesis is to consider a variational method that is not only “simple” to use but is also extremely stable and approximates solutions to boundary value problems with an accuracy better than most methods.

In this thesis, we will first consider a specific boundary value problem of the type:

\[ ( - p(t) x')' + q(t) x = r(t); \]

with an unique solution. We will approximate this boundary value problem by applying
cubic B-splines to it and using a variational principle. These procedures will then be extended to systems of equations and also to the case of piecewise smooth coefficient functions. Finally, certain optimal control problems will be numerically solved by first converting the control problem to a problem in the calculus of variations and then using the previous methods on the resulting boundary value problem.
CHAPTER 2

BEST APPROXIMATES OF BOUNDARY VALUE PROBLEMS

In this chapter, we will briefly explain the procedure that variational methods use to find the best approximation to the solution of a boundary value problem. In particular, we will be considering the Rayleigh-Ritz method for finding best approximates. Next, we will show that best approximates not only exist but are unique under certain conditions. Then, we will show that when we apply the Rayleigh-Ritz method to our boundary value problem, it satisfies the conditions under which best approximates exist and are unique. Finally, a brief description of the fortran program that solves for best approximates will be given.

Variational Methods

Given the equation:

\[ Ax = y, \]

where \( A : X \to Y \), and \( X, Y \) are normed linear spaces (i.e. normed vector spaces) with norms \( \| \cdot \|_X \) and \( \| \cdot \|_Y \) respectively. Let \( X_n \) be a finite dimensional subspace of \( X \) where \( X_n = \text{span}\{ \phi_1, \phi_2, \ldots, \phi_n \} \). A variational method is just a numerical algorithm for finding a function \( x_n \in X_n \) such that
Thus, variational methods simply transforms boundary value problems to become a minimization problem [10]. The variational method that we will be using to find the best approximate to the solution of our boundary value problem is the Rayleigh-Ritz method.

**Rayleigh-Ritz Method**

The Rayleigh-Ritz method reformulates a given boundary value problem by changing it to a problem of choosing, from the set of all sufficiently differentiable functions that satisfy certain boundary conditions, the function \( x_n \), that minimizes a certain integral. That set is then reduced in size resulting in an approximation to the solution to the minimization problem [3]. Consequently, an approximation to the solution to the boundary value problem is found. Before we can apply the Rayleigh-Ritz method to our boundary value problem to find the best approximate to the solution, we have to make sure that the approximate exists and is unique.

**Existence and Uniqueness of Best Approximates**

**Theorem 2.1 (Existence of Best Approximates)**

Let \( X \) be a normed linear space with norm \( \| \cdot \| \) and let \( X_n \) be a finite dimensional subspace of \( X \). Then each point \( x \in X \) has a best approximation \( x_n \in X_n \). Thus

\[
\| x - x_n \| = \min_{y \in X_n} \| x - y \|
\]

**Proof.** Let \( X \) be a normed linear space with norm \( \| \cdot \| \) and let \( X_n \) be a finite dimensional subspace of \( X \). Fix \( x \in X \). Let \( z \in X_n \) and let \( d = \| x - z \| \). Let

\[
\| Ax_n - y \|_Y + \| x_n - x \|_X \to 0.
\]
\[ K = \left\{ z \in X_n : \|x - z\| \leq d \right\}. \]

To show that \( K \) is closed we must show that \( K = \overline{K} \) where \( \overline{K} \) is the set of points of closure of \( K \). Let \( \hat{z} \in \overline{K} \); since \( \hat{z} \) is a point of closure, for every \( \delta > 0 \) there exists a point \( z_0 \in K \) such that \( \|\hat{z} - z_0\| < \delta \). Hence, we have

\[
\|x - \hat{z}\| = \|x - z_0 + z_0 - \hat{z}\| \leq \|x - z_0\| + \|z_0 - \hat{z}\| \leq d + \delta.
\]

Thus

\[
\|x - \hat{z}\| \leq d + \delta.
\]

Since this inequality holds for every \( \delta \), it follows that

\[
\|x - \hat{z}\| \leq d.
\]

Therefore, \( \hat{z} \in K \) and \( \overline{K} \subseteq K \). Clearly \( K \subseteq \overline{K} \), thus \( K = \overline{K} \) which implies that \( K \) is closed. Clearly \( K \) is bounded. Therefore, being a closed and bounded subset of a finite dimensional space, \( K \) is compact. Define \( g(z) = \|x - z\|, z \in K \). Let \( z_0 \in K \). Pick \( \epsilon > 0 \) and let \( \delta_{\epsilon} = \epsilon \). Let \( \tilde{z} \in K \) such that \( \|z_0 - \tilde{z}\| < \delta_{\epsilon} \), thus we have:

\[
|\|z_0 - x\| - \|x - \tilde{z}\| | \leq \|z_0 - x + x - \tilde{z}\| = \|z_0 - \tilde{z}\| < \delta_{\epsilon}.
\]

\[
|g(z_0) - g(\tilde{z})| = |\|z_0 - x\| - \|x - \tilde{z}\| | < \delta_{\epsilon} = \epsilon.
\]

\[
|g(z_0) - g(\tilde{z})| < \epsilon.
\]

Thus, \( |g(z_0) - g(\tilde{z})| < \epsilon \) whenever \( \|z_0 - \tilde{z}\| < \delta_{\epsilon} \), hence \( g \) is continuous at \( z_0 \). Clearly, \( g \) is continuous at any \( z \in K \), therefore \( g \) is continuous on \( K \). Since \( K \) is compact, \( g \) assumes its minimum at some point \( x_n \in K \). Thus

\[
\|x - x_n\| = \min_{y \in K} \|x - y\|.
\]
Now that we have shown that best approximates do, indeed, exist, we would like for them to be unique. The following theorem shows that best approximates are, in fact, unique.

**Theorem 2.2 (Uniqueness of Best Approximates)**

Best approximations from (closed) finite dimensional subspaces of an inner product space are unique.

**Proof.** Let $X$ be an inner product space and let $M$ be a closed subspace of $X$. Let $x \in X$, and suppose $x_o, y_o \in M$ are best approximates to $x$. Then $\|x - x_o\| = \|x - y_o\| = d$.

If $d = 0$, then $x_o = y_o = x$; so assume that $d > 0$. By the triangle inequality, we have

$$\|\frac{1}{2}(x_o + y_o) - x\| = \frac{1}{2}\|x_o - x\| + \frac{1}{2}\|y_o - x\| = d.$$ 

Since $\frac{1}{2}(x_o + y_o) \in M$, $\frac{1}{2}(x_o + y_o)$ is also a best approximation to $x$, thus we obtain

$$\|\frac{1}{2}(x_o + y_o) - x\| = d.$$ 

Hence,$$d = \|\frac{1}{2}(x_o + y_o) - x\| \leq \frac{1}{2}\|x_o - x\| + \frac{1}{2}\|y_o - x\| \leq d.$$ 

Thus,

$$\|\frac{1}{2}(x_o + y_o) - x\| = \frac{1}{2}\|x_o - x\| + \frac{1}{2}\|y_o - x\|.$$ 

Now consider the square of the left-hand side (LHS)

$$\left(\|\frac{1}{2}(x_o - x) + \frac{1}{2}(y_o - x)\|^2 = \frac{1}{2}\|x_o - x\|^2 + 2\frac{1}{2}(x_o - x) \cdot \frac{1}{2}(y_o - x) + \frac{1}{2}\|y_o - x\|^2\right).$$

$$= \frac{1}{4}\|x_o - x\|^2 + \frac{1}{2}\|(x_o - x) \cdot (y_o - x)\| + \frac{1}{4}\|y_o - x\|^2.$$ 

Also consider the square of the right-hand side (RHS)

$$\left(\frac{1}{2}\|x_o - x\| + \frac{1}{2}\|y_o - x\|\right)^2 = \frac{1}{4}\|x_o - x\|^2 + \frac{1}{2}\|x_o - x\| \cdot \|y_o - x\| + \frac{1}{4}\|y_o - x\|^2.$$
Therefore from
\[ \|\frac{1}{2}(x_0 + y_0) - x\|^2 = \left( \frac{1}{2}\|x_0 - x\| + \frac{1}{2}\|y_0 - x\| \right)^2. \]
we have
\[ ((x_0 - x).(y_0 - x)) = \|x_0 - x\| \|y_0 - x\|. \]

Now let \( z = (y_0 - x) - \lambda(x_0 - x) \), where \( \lambda = \|y_0 - x\|/\|x_0 - x\| \). Thus
\[
\|z\|^2 = \|(y_0 - x) - \lambda(x_0 - x)\|^2
\]
\[
= \|y_0 - x\|^2 - 2((y_0 - x).(\lambda(x_0 - x)) + \|\lambda(x_0 - x)\|^2
\]
\[
= \|y_0 - x\|^2 - 2\lambda((y_0 - x).(x_0 - x)) + \lambda^2\|(x_0 - x)\|^2
\]
\[
= \|y_0 - x\|^2 - 2\lambda\|y_0 - x\|\|x_0 - x\| + \lambda^2\|(x_0 - x)\|^2
\]
\[
= (\|y_0 - x\| - \lambda\|(x_0 - x)\|)^2.
\]

Since \( \lambda = \|y_0 - x\|/\|x_0 - x\| \), we have
\[
\|z\| = \|y_0 - x\| - (\|y_0 - x\|/\|x_0 - x\|)\|(x_0 - x)\|
\]
\[
= \|y_0 - x\| - \|y_0 - x\| = 0.
\]

Thus,
\[ z = (y_0 - x) - \lambda(x_0 - x) = 0. \]

and
\[ (y_0 - x) = \lambda(x_0 - x). \]

for some \( \lambda \). If \( \lambda = 1 \) then clearly we have \( x_0 = y_0 \), hence uniqueness. Suppose \( \lambda \neq 1 \).

then from
\[(y_o - x) = \lambda(x_o - x).\]

we have

\[x = \frac{y_o - \lambda x_o}{1 - \lambda}.\]

Clearly \(x \in M\), so \(x\) is a best approximate of \(x\) and \(\|x - x\| = 0 = d\). but \(d > 0\). which is a contradiction, so \(\lambda = 1\). ■

Now that we have shown that best approximates exist and are unique, we have to show that we will be working under the conditions for which the best approximates exist and are unique. Therefore, we must show that we will be working in a finite dimensional (closed) subspace of an inner product space.

**The Inner Product Space**

Let \(\mathcal{P}C[a,b]\) be the set of all functions piecewise continuous on the interval \([a,b]\).

and let \(f(t), g(t) \in \mathcal{P}C[a,b]\). Consider

\[(f, g) = \int_a^b f(t)g(t)dt. \quad (2.4.1)\]

It is easily shown that (2.4.1) is an inner product on \(\mathcal{P}C[a,b]\).

**Proof.** Let \(f(t), g(t), h(t) \in \mathcal{P}C[a,b]\) and let \(\alpha\) and \(\beta\) be scalars.

(i) \((f, g) = \int_a^b f(t)g(t)dt = \int_a^b g(t)f(t)dt = (g, f)\).

(ii) \((\alpha f + \beta g, h) = \int_a^b (\alpha f + \beta g)(t)h(t)dt\)

\[= \int_a^b (\alpha f(t)h(t) + \beta g(t)h(t))dt\]

\[= \alpha \int_a^b f(t)h(t)dt + \beta \int_a^b g(t)h(t)dt\]
\( = \alpha(f, h) + \beta(g, h) \).

(iii) \( (f, f) = \int_a^b (f(t))^2 \, dt \). Clearly \( (f(t))^2 \geq 0 \). \( \forall t \in [a, b] \), thus we have:

\( (f, f) = \int_a^b (f(t))^2 \, dt \geq 0 \). Suppose \( f \equiv 0 \), then \( (f, f) = \int_a^b 0^2 \, dt = 0 \). Now suppose \( (f, f) = 0 \). then \( (f, f) = \int_a^b (f(t))^2 \, dt = 0 \). which implies that \( f(t) = 0 \). \( \forall t \in [a, b] \). since \( f \) is a piecewise continuous function and \( f(t) = 0 \) over each of its domains. \( f \equiv 0 \). Thus, we have that \( (f, f) > 0 \) and \( (f, f) = 0 \) iff \( f \equiv 0 \). ■

Clearly, \( PC[a, b] \) is a vector space, thus we have that \( PC[a, b] \) is an inner product space with the inner product given by (2.4.1).

**Reformulation of the boundary value problem**

Consider the boundary value problem

\[
\begin{aligned}
( - p(t) x' )' + q(t) x &= r(t) ; \\
\end{aligned}
\]

where \( t \in [0, 1] \); and \( x(0) = 0 = x(1) \). Let \( q, r \in PC[0, 1] \); and \( p \in PC^1[0, 1] \), the set of all functions having one piecewise continuous derivative on \( [0, 1] \). If we also require that \( p(t) > 0 \) and \( q(t) \geq 0 \); for \( t \in [0, 1] \), it has been shown by Bailey, Shampine, and Waltman [2] that these conditions are sufficient to guarantee a unique solution to (2.5.1).

Now let \( A \) be the mapping from \( PC^2[0, 1] \), the set of all functions having two piecewise continuous derivatives on \( [0, 1] \), to \( PC[0, 1] \); defined as

\[
A x(t) = ( - p(t) x'(t) )' + q(t) x(t) \quad t \in [0, 1].
\]

Then \( A x(t) = r(t) \in PC[0, 1] \), whenever \( x(t) \in PC^2[0, 1] \).
Since $\mathcal{PC}^2[0,1] \subset \mathcal{PC}[0,1]$, it has been shown by Prenter [10] that the inner product (2.4.1) on the vector space $\mathcal{PC}[0,1]$ gives rise to a second inner product on $\mathcal{PC}^2[0,1]$, defined by

$$(x,y)_A = (Ax,y)$$  \hspace{1cm} (2.5.3)$$

for each $x, y \in \mathcal{PC}^2[0,1]$.

Let $x \in \mathcal{PC}^2[0,1]$ and let $y \in \mathcal{PC}^2[0,1]$ such that $y(0) = y(1) = 0$, then by (2.5.3), we obtain

$$(x,y)_A = (Ax,y) = \int_0^1 \left( \left( - p(t)x'(t) \right)' + q(t)x(t) \right)y(t) \, dt$$

$$= \int_0^1 (- p(t)x'(t))'y(t) \, dt + \int_0^1 q(t)x(t)y(t) \, dt$$

$$= - p(t)x'(t)y(t)|_0^1 + \int_0^1 p(t)x'(t)y'(t) \, dt + \int_0^1 q(t)x(t)y(t) \, dt .$$

Since $y(0) = y(1) = 0$, the equation reduces to

$$(x,y)_A = \int_0^1 p(t)x'(t)y'(t) \, dt + \int_0^1 q(t)x(t)y(t) \, dt . \hspace{1cm} (2.5.4)$$

Since $Ax(t) = r(t)$, we also have

$$(x,y)_A = (Ax,y) = (r,y) = \int_0^1 r(t)y(t) \, dt . \hspace{1cm} (2.5.5)$$

Thus,

$$\int_0^1 p(t)x'(t)y'(t) \, dt + \int_0^1 q(t)x(t)y(t) \, dt = \int_0^1 r(t)y(t) \, dt . \hspace{1cm} (2.5.6)$$

If $x$ is a solution to (2.5.1), then equation (2.5.6) holds for any $y$ with $y(0) = y(1) = 0$.

Therefore we can consider functions $x$ and $y$ from an $N$-dimensional subspace of $\mathcal{PC}^2[0,1]$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Cubic B-splines

Let \( h = 1/n \). for some \( n \in \mathbb{N} \). Consider the equally spaced partition of \([0,1]\)

\[
0 = t_0 < t_1 < \cdots < t_n = 1.
\]

where \( t_j - t_{j-1} = h \). for \( j = 1,2,\ldots,n \). The elements of the subspace of \( P^2C[0,1] \) that we want to construct come from the linearly independent set of B-splines.

\[
\{ \hat{\phi}_{-1}, \hat{\phi}_0, \hat{\phi}_1, \ldots, \hat{\phi}_n, \hat{\phi}_{n+1} \}, \text{ where each } \hat{\phi}_i \text{ is centered at } t_i. \text{ The graph of } \hat{\phi}_i \text{ for } i = -1,0,1,\ldots,n,n + 1 \text{ is illustrated in Figure 1.}
\]

![Figure 1: The normalized graph of the cubic B-spline, \( \hat{\phi}_i \), centered at \( t_i \).](image)

These cubic interpolatory splines for a function \( x. \) satisfy the following conditions. for \( i = -1,0,1,\ldots,n,n + 1 \) [3]

1.) \( \hat{\phi}_i \) is a cubic polynomial, denoted by \( \hat{\phi}_{i,k} \), on \( [t_k, t_{k+1}] \) for \( k = i - 2, i - 1, i, i + 1 \).

2.) \( \hat{\phi}_i(t_k) = x(t_k) \), for \( k = i - 2, i, i + 2 \).

3.) \( \hat{\phi}_{i,k+1}(t_{k+1}) = \hat{\phi}_{i,k}(t_{k+1}) \), for \( k = i - 2, i - 1, i \).

4.) \( \hat{\phi}_{i,k+1}'(t_{k+1}) = \hat{\phi}_{i,k}'(t_{k+1}) \), for \( k = i - 2, i - 1, i \).
5.) \( \hat{\phi}_{k+1}(t_{k+1}) = \hat{\phi}_{k}(t_{k+1}) \), for \( k = i - 2, i - 1, i \).

6.) \( \hat{\phi}_i''(t_{i-2}) = \hat{\phi}_i''(t_{i-2}) = 0 \).

7.) \( \hat{\phi}_i'(t_{i-2}) = x'(t_{i-2}) \) and \( \hat{\phi}_i(t_{i-2}) = x'(t_{i-2}) \).

Thus, the basis of B-splines is defined as:

\[
\hat{\phi}_i(t) = \begin{cases} 
0 & \text{if } t \notin [t_{i-2}, t_{i-2}] \\
(t - t_{i-2})^3 & \text{if } t \in [t_{i-2}, t_{i-1}] \\
h^3 + 3h^2(t - t_{i-1}) + 3h(t - t_{i-1})^2 - 3(t - t_{i-1})^3 & \text{if } t \in [t_{i-1}, t_{i}] \\
h^3 + 3h^2(t_{i-1} - t) + 3h(t_{i-1} - t)^2 - 3(t_{i-1} - t)^3 & \text{if } t \in [t_{i}, t_{i+1}] \\
(t_{i+2} - t)^3 & \text{if } t \in [t_{i+1}, t_{i+2}] 
\end{cases}
\]

for \( i = -1, 0, 1, \ldots, n, n + 1 \).

From the set of B-splines, we want to construct an \( N \)-dimensional (where \( N = n + 1 \)) set of B-splines, \( B_N \), which has elements that vanish at the end points of the interval \([0, 1]\). Since each \( \hat{\phi}_i \), for \( 2 \leq i \leq n - 2 \), vanishes at \( t = 0 \) and \( t = 1 \), these \( \hat{\phi}_i \)'s are elements of \( B_N \). Thus, \( \{ \hat{\phi}_2, \hat{\phi}_3, \ldots, \hat{\phi}_{n-2} \} \) forms part of the basis for \( B_N \). Now we need to find four more cubic splines, \( \{ \hat{\phi}_0, \hat{\phi}_1, \hat{\phi}_{n-1}, \hat{\phi}_n \} \), so that when added to the set \( \{ \hat{\phi}_2, \hat{\phi}_3, \ldots, \hat{\phi}_{n-2} \} \), the new set forms a basis for \( B_N \) [10]. There are many choices for determining these four cubic splines. One way is to use the six functions in \( \{ \hat{\phi}_{-1}, \hat{\phi}_0, \hat{\phi}_1, \ldots, \hat{\phi}_n, \hat{\phi}_{n+1} \} \), that do not vanish at \( t = 0 \) and \( t = 1 \), by determining linear combinations of these functions that satisfy certain conditions. In particular, we seek to
find the constants $a$, $b$, $c$, $d$, $e$, $f$, $g$, and $h$
\[
\phi_0(t) = a\phi_0(t) + b\phi_0(t)
\]
\[
\phi_1(t) = c\phi_0(t) + d\phi_1(t)
\]
\[
\phi_{n-1}(t) = e\phi_{n-1}(t) + f\phi_n(t)
\]
\[
\phi_n(t) = g\phi_n(t) + h\phi_{n-1}(t).
\]
so that $\phi_0, \phi_1, \phi_{n-1}$, and $\phi_n$ satisfy the following interpolation problem [6]
\[
\phi_0(t_0) = 0
\]
\[
\phi_0(t_1) = 1/4
\]
\[
\phi_1(t_0) = 0
\]
\[
\phi_1(t_{-1}) = 1/4
\]
\[
\phi_{n-1}(t_{n+1}) = 1/4
\]
\[
\phi_{n-1}(t_n) = 0
\]
\[
\phi_n(t_{n-1}) = 1/4
\]
\[
\phi_n(t_n) = 0.
\]
Solving for the constants: $a$, $b$, $c$, $d$, $e$, $f$, $g$, and $h$ we obtain:
\[
\phi_0(t) = \hat{\phi}_0(t) - 4\hat{\phi}_{-1}(t)
\]
\[
\phi_1(t) = \hat{\phi}_0(t) - 4\hat{\phi}_1(t)
\]
\[
\phi_{n-1}(t) = \hat{\phi}_n(t) - 4\hat{\phi}_{n-1}(t)
\]
\[
\phi_n(t) = \hat{\phi}_n(t) - 4\hat{\phi}_{n+1}(t).
\]
Thus, $\phi_0, \phi_1, \phi_{n-1}$, and $\phi_n$ are defined as

$$
\phi_0(t) = \begin{cases} 
0 & \text{if } t \not\in [t_0, t_2] \\
\frac{1}{4h^3} \left[ h^3 + 3h^2(t_1 - t) + 3h(t_1 - t)^2 - 7(t_1 - t)^3 \right] & \text{if } t \in [t_0, t_1] \\
(t_2 - t)^3 & \text{if } t \in [t_1, t_2] \\
0 & \text{if } t \in [t_2, t_3] \\
6h^2(t - t_0) - 2(t - t_0)^3 & \text{if } t \in [t_0, t_1] \\
6h^2( t - t_2) - 2(t - t_2)^3 & \text{if } t \in [t_1, t_2] \\
(t_3 - t)^3 & \text{if } t \in [t_2, t_3] \\
(t - t_{n-3})^3 & \text{if } t \in [t_{n-3}, t_{n-2}] \\
h^3 + 3h^2(t - t_{n-2}) + 3h(t - t_{n-2})^2 - 3(t_{n-2} - t)^3 & \text{if } t \in [t_{n-2}, t_{n-1}] \\
6h^2(t_n - t) - 2(t_n - t)^3 & \text{if } t \in [t_{n-1}, t_n] \\
0 & \text{if } t \not\in [t_{n-2}, t_n] \\
(t - t_{n-2})^3 & \text{if } t \in [t_{n-2}, t_{n-1}] \\
h^3 + 3h^2(t - t_{n-1}) + 3h(t - t_{n-1})^2 - 7(t - t_{n-1})^3 & \text{if } t \in [t_{n-1}, t_n] \\
0 & \text{if } t \not\in [t_{n-1}, t_n]
\end{cases}
$$

The following figures 2, 3, 4, and 5 show the graphs of the cubic B-splines

$\phi_0, \phi_1, \phi_{n-1}$, and $\phi_n$, respectively.
\[ \phi_0(t) \]

Figure 2: Graph of $\phi_0$

\[ \phi_1(t) \]

Figure 3: Graph of $\phi_1$

\[ \phi_{n-1}(t) \]

Figure 4: Graph of $\phi_{n-1}$
Figure 5: Graph of $\phi_n$

Now, let $\phi_i = \hat{\phi}_i$, for $2 \leq i \leq n - 2$. It can be easily shown that the set 
\{ $\phi_0, \phi_1, \ldots, \phi_n$ \}

is linearly independent. Also we have that each element in 
\{ $\phi_0, \phi_1, \ldots, \phi_n$ \}

vanishes at $t = 0$ and $t = 1$. Thus, the set \{ $\phi_0, \phi_1, \ldots, \phi_n$ \} forms a basis for the set $B_n$.

Therefore let $B_N = \text{span}\{\phi_0, \phi_1, \ldots, \phi_n\}$. It can also be easily shown that the set $B_N$ is a

subspace of $PC^2[0,1]$. Thus, we have found an $N$-dimensional subspace of $PC^2[0,1]$

from which we can choose the functions $x$ and $y$.

Let $x, y \in B_N$, such that

$$x(t) = \sum_{i=0}^{n} c_i \phi_i(t), \quad \text{for some scalars } \{c_0, c_1, \ldots, c_n\}; \quad \text{and} \quad (2.6.1)$$

$$y(t) = \sum_{j=0}^{n} d_j \phi_j(t), \quad \text{for some scalars } \{d_0, d_1, \ldots, d_n\}. \quad (2.6.2)$$

Substituting into equation (2.5.6), we obtain

$$\int_0^l p(t) \sum_i c_i \phi_i'(t) \sum_j d_j \phi_j'(t) \, dt + \int_0^l q(t) \sum_i c_i \phi_i(t) \sum_j d_j \phi_j(t) \, dt = \int_0^l r(t) \sum_j d_j \phi_j(t) \, dt.$$

Fix $k$ and take $d_j = 1$ if $j = k$, and $d_j = 0$ if $j \neq k$. Thus,
\[
\int_0^l p(t) \sum_i c_i \phi'_i(t) \phi_k(t) \, dt + \int_0^l q(t) \sum_i c_i \phi_i(t) \phi_k(t) \, dt = \int_0^l r(t) \phi_k(t) \, dt.
\]

Also, \( \phi_k(t) = 0 = \phi_k'(t) \) if \( t \notin [t_{k-2}, t_{k-1}] \). Thus the above equation becomes

\[
\sum_{i=k-3}^{k+3} \left( \int_{t_{i-1}}^{t_i} p(t) c_i \phi_i'(t) \phi_k(t) \, dt + \int_{t_{i-1}}^{t_i} q(t) c_i \phi_i(t) \phi_k(t) \, dt \right) = \int_{t_{k-3}}^{t_{k+3}} r(t) \phi_k(t) \, dt. \tag{2.6.3}
\]

Define \( b_k \) and \( e_{ki} \) as

\[
b_k = \int_{t_{k-3}}^{t_{k+3}} r(t) \phi_k(t) \, dt. \tag{2.6.4}
\]

\[
e_{ki} = \int_{t_{k-3}}^{t_{k+3}} (p(t) \phi_i'(t) \phi_k(t) + q(t) \phi_i(t) \phi_k(t)) \, dt. \tag{2.6.5}
\]

then equation (2.6.3) becomes

\[
c_{k-3} e_{k,k-3} + c_{k-2} e_{k,k-2} + c_{k-1} e_{k,k-1} + c_k e_{k,k} + c_{k+1} e_{k,k+1} + c_{k+2} e_{k,k+2} + c_{k+3} e_{k,k+3} = b_k.
\]

for \( k = 0,1,2,...,n \). Hence, there are \( (n+1) \) equations which can be simplified as

\[
Ec = b. \tag{2.6.6}
\]

where \( c \) and \( b \) are \( (n+1) \) column vectors, and \( E \) is a symmetric seven banded \( (n+1) \times (n+1) \) matrix. The vector \( c \) can be easily solved for on a computer, but before we can solve for \( c \), the entries of the matrix \( E \) and the vector \( b \) must be determined.

**Program CubicB**

The program that was written to solve the equation (2.6.6) can be found in the appendix. In this program, the entries for the matrix \( E \) and the vector \( b \) are calculated by a simple integration subroutine called qromb. Then depending on the size of the matrix \( E \), the vector \( c \) is either solved by a gaussian elimination subroutine or by a subroutine that takes advantage of \( E \) being banded. For \( n \leq 40 \), \( c \) is determined by the subroutine gaussj.
This subroutine simply determines \( c \) by gaussian elimination and then backsubstitution.

The subroutine \texttt{gaussj} is good to implement in that it determines the vector \( c \) with a high degree of accuracy. The subroutine, though, can not solve large matrices, which is a severe limitation. For this problem, when \( n > 40 \) the gaussian elimination subroutine is not reliable, so another subroutine, called \texttt{Bandsolver}, must be used to determine \( c \).

The subroutine \texttt{Bandsolver} is more complex in that the matrix \( E \) is first transformed from an \((n + 1) \times (n + 1)\) matrix to an \((n + 1) \times 7\) matrix by the subroutine. Transform. Once \( E \) is transformed, it is then decomposed into the product of lower and upper triangular matrices using the subroutine \texttt{LUdecomp}. Only after this is the subroutine \texttt{Bandsolver} used to determine the vector \( c \). The major advantage of this subroutine over \texttt{gaussj} is its ability to solve systems where \( n \) is as large as 800. It does not, though, solve for \( c \) with as high of degree of accuracy as \texttt{gaussj}, which is important since \( c \) is used to determine the approximate to the solution of the boundary value problem.

Once the vector \( c \) is determined, it is then substituted into equation (2.6.1) to give an approximate to the unique solution of the equation (2.5.1). The absolute error is then computed, and finally the rate of convergence is computed. It has been proven by Schultz [10] pgs 107-108. that cubic B-splines converge to the solution at a rate of \( O(h^4) \). when the second derivative of the solution is continuous everywhere.
CHAPTER 3

EXTENSION OF THE RAYLEIGH-RITZ METHOD

In this chapter, the techniques used to approximate the boundary value problem given in (2.5.1) will be extended to solve the more general boundary value problem

\[
(-p(t)x')' + q(t)x = r(t),
\]

where \( t \in [a, b] \); \( x(a) = \alpha \) and \( x(b) = \beta \). Then two one-dimensional examples will be given. One for when the solution to (3.0.1) has a continuous second derivative, and one for when the second derivative is discontinuous at a some point.

Transformation of the General Boundary Value Problem

The boundary value problem given by (3.0.1) can be transformed by the change of variables

\[
z = x - \beta t - (1 - t)\alpha
\]

into the form

\[
(-p((b-a)w + a)z')' + (b-a)^2 q((b-a)w + a)z = f(w);
\]

where \( w \in [0,1] \); and \( z(0) = 0 = z(1) \). Since \( a \leq t \leq b \), then

\[
0 \leq \frac{t-a}{b-a} \leq 1.
\]
Define \( w \) to be
\[
w = \frac{t - a}{b - a}. \tag{3.1.3}
\]
so \( 0 \leq w \leq 1 \). Thus we can define \( t \) in terms of \( w \).
\[
t = (b - a)w + a. \tag{3.1.4}
\]
From (3.1.4), we obtain:
\[
dt = (b - a)dw. \tag{3.1.5}
\]
Substituting (3.1.4) and (3.1.5) into (3.0.1), we obtain:
\[
\int \left( -p((b - a)w + a)z' \right) + q((b - a)w + a)x = r((b - a)w + a). \tag{3.1.6}
\]
Now, by rearranging (3.1.1) and substituting it into (3.1.6), we obtain:
\[
\frac{1}{(b - a)^2} \left( -p((b - a)w + a)z' \right) + (b - a)^2 q((b - a)w + a)x + \beta \left( z + \beta t + (1-t)\alpha \right) = (b - a)^2 r((b - a)w + a). \tag{3.1.7}
\]
Thus, we have the LHS of the above equation to be
\[
-p'((b - a)w + a)(z' + (b - a)(\beta - \alpha)) + (b - a)^2 q((b - a)w + a)(z + \beta t + (1-t)\alpha).
\]
Rearranging the LHS, we obtain
\[
(-p((b - a)w + a)z') + (b - a)^2 q((b - a)w + a)z + s(w). \tag{3.1.8}
\]
where \( s(w) = -(b - a)(\beta - \alpha)p'((b - a)w + a) + (b - a)^2 q((b - a)w + a)(\beta t + (1-t)\alpha) \).

Now, substitute (3.1.8) back into (3.1.7), we obtain
\[
(-p((b - a)w + a)z') + (b - a)^2 q((b - a)w + a)z + s(w) = (b - a)^2 r((b - a)w + a).
\]
If we let \( f(w) = (b - a)^2 r((b - a)w + a) - s(w) \), the above equation becomes
\[-p((b-a)w + a)z' + (b-a)^2 q((b-a)w + a)z = f(w),\]

where \( w \in [0.1] \) and \( z(0) = 0 = z(1) \). Thus, by using this transformation, we can also determine approximates of the solutions to boundary value problem's of the form given by (3.0.1).

**Examples for One-Dimensional Boundary Value Problems**

Example 3.1 Let \( p(t) = t^2 \), \( q(t) = 2 \), and \( r(t) = -4t^2 \); for \( t \in [0.1] \). Also, let \( x(0) = 0 = x(1) \). Clearly, these conditions are such that they satisfy the requirements for a unique solution to (3.0.1). When we substitute these values into (3.0.1), the boundary value problem becomes

\[
(-t^2x')' + 2x = -4t^2. \quad \text{for } t \in [0.1].
\]

It can be shown that the solution to this example is

\[ x(t) = t^2 - t. \]

The solution to (3.2.1) was approximated using the program CubicB found in Appendix II, with the step size \( h = 1/n \). The errors for \( h, h/2, h/4 \) are listed in Table I, and it can be easily seen that the approximation is converging to the solution at a rate of \( O(h^4) \) which supports the conclusion that this method produces approximates that converge to the actual solution at the rate of \( O(h^4) \) when the second derivative of the solution is continuous, not just piecewise continuous.
Example 3.2 Now, let us consider the case when $x$ has a piecewise continuous second derivative, and the second derivative of the function is discontinuous at some point. Thus, take the case when

$$p(t) = 1 = q(t), \text{ and } (3.2.2)$$

$$r(t) = t|t| + (1 + \pi^2)\sin(\pi t) + \begin{cases} 2 & \text{if } t \in [-1,0] \\ -2 & \text{if } t \in (0,1] \end{cases} \quad (3.2.3)$$

where $t \in [-1,1]$, $x(-1) = -1$ and $x(1) = 1$. Substituting (3.2.2) and (3.2.3) into (3.0.1) we obtain

$$-x'' + x = t|t| + (1 + \pi^2)\sin(\pi t) + \begin{cases} 2 & \text{if } t \in [-1,0] \\ -2 & \text{if } t \in (0,1] \end{cases} \quad (3.2.4)$$

It can be shown that the solution to equation (3.2.4) is

$$x(t) = t|t| + \sin(\pi t).$$

In this example, it can be seen from Table II that the overall rate of convergence for this problem is not $O(h^4)$ but, instead, is $O(h^2)$. This discrepancy is due to the discontinuity of the second derivative of $x$ at $t = 0$.

**Systems of Equations**

Consider the boundary value problem

$$(-P(t)x')' + (U(t)x)' - U^T(t)x' + Q(t)x = g(t), \quad (3.3.1)$$

where $t \in [0,1]$; $x(0) = \tilde{x} = x(1)$; and $P$, $U$, and $Q$ are $m \times m$ matrices, such that $P(t)$ is positive definite, $P$ and $U$ are piecewise smooth and continuous, and $Q$ is piecewise
continuous. Also, \( x \) and \( g \) are \( m \times 1 \) vectors where \( x \) is piecewise smooth and continuous and \( g \) is piecewise continuous \([9]\). Clearly, if \( m = 1 \) then

\[
(U(t)x)' - U^T(t)x' = 0
\]

and (3.3.1) becomes

\[
(-P(t)x') + Q(t)x = g(t)
\]

which is the boundary value problem given by (2.5.1). Thus, let us take the case when \( m > 1 \). By using the method outlined in Chapter 2, (3.3.1) can be reformulated to become a minimization problem with the variational form

\[
\int_0^1 \left[ y'^T P x' - y'^T U x - y'' U^T x' + y'^Q x \right] dt = \int_0^1 y'^T g dt. \tag{3.3.2}
\]

where \( y \) is a piecewise smooth and continuous \( m \times 1 \) vector, such that \( y(0) = 0 = y(1) \). From Chapter 2, we have that if \( x \) is a solution to (3.3.1) then (3.3.2) holds for any \( y \) with the condition that \( y(0) = 0 = y(1) \). Thus, we can choose the entries of \( x \) and \( y \) to be from \( B_N \), the set that we constructed in Chapter 2. Hence, \( x \) and \( y \) can be written as

\[
x = \sum_{i=0}^{n} \bar{c}_i \phi_i, \tag{3.3.3}
\]

\[
y = \sum_{j=0}^{n} \bar{d}_j \phi_j; \tag{3.3.4}
\]

where \( \bar{c}_i, \bar{d}_j \in \mathbb{R}^m \) (the set of all real-valued \( m \times 1 \) vectors) and \( \phi_i, \phi_j \in B_N \) for \( i, j = 0,1,...,n \). Substituting (3.3.3) and (3.3.4) into equation (3.3.2), we obtain

\[
\int_0^1 \left[ \sum_j \bar{d}_j^T \phi_j P \sum_i \bar{c}_i \phi'_i - \sum_j \bar{d}_j^T \phi_j U^T \sum_i \bar{c}_i \phi_i - \sum_j \bar{d}_j^T \phi_j U^T \sum_i \bar{c}_i \phi'_i + \sum_j \bar{d}_j^T \phi_j Q \sum_i \bar{c}_i \phi_i \right] dt = \int_0^1 \sum_j \bar{d}_j^T \phi_j g dt.
\]

From Chapter 2, for a fixed \( k \), the above equation becomes
\[
\int_0^1 \left[ P \sum_i \bar{c}_i \phi_i' \phi_i' - U \sum_i \bar{c}_i \phi_i \phi_i - U^T \sum_i \bar{c}_i \phi_i \phi_i' + Q \sum_i \bar{c}_i \phi_i \phi_i \right] dt = \int_0^1 \phi_k g dt
\] (3.3.5)

Now, let \( P(t) = P(t_k) = P_k \), \( U(t) = U(t_k) = U_k \), \( Q(t) = Q(t_k) = Q_k \), and

\[
g(t) = g(t_k) = g_k; \quad \text{for } t \in [t_{k-2}, t_{k-1}].
\]

Also, \( \phi_k = 0 = \phi'_k \) if \( t \not\in [t_{k-2}, t_{k-1}] \). Thus, substituting into (3.3.5), we have

\[
\sum_{i=k-3}^{k+3} \left[ P_k \int_{t_{i-1}}^{t_i} \bar{c}_i \phi_i \phi_i' dt - U_k \int_{t_{i-1}}^{t_i} \bar{c}_i \phi_i \phi_i' dt - U_k^T \int_{t_{i-1}}^{t_i} \bar{c}_i \phi_i \phi_i' dt + Q_k \int_{t_{i-1}}^{t_i} \bar{c}_i \phi_i \phi_i' dt \right] = g_k \int_{t_{i-1}}^{t_i} \phi_k dt.
\] (3.3.6)

Since \( \bar{c}_i \) is a vector with entries that are real numbers, we obtain

\[
\sum_{i=k-3}^{k+3} \left[ P_k \int_{t_{i-1}}^{t_i} \phi_i \phi_i' dt - U_k \int_{t_{i-1}}^{t_i} \phi_i \phi_i' dt - U_k^T \int_{t_{i-1}}^{t_i} \phi_i \phi_i' dt + Q_k \int_{t_{i-1}}^{t_i} \phi_i \phi_i' dt \right] \bar{c}_i = g_k \int_{t_{i-1}}^{t_i} \phi_k dt.
\]

Define \( \bar{e}_{k,i} \) and \( \bar{b}_k \) to be

\[
\bar{e}_{k,i} = P_k \int_{t_{i-1}}^{t_i} \phi_i \phi_i' dt - U_k \int_{t_{i-1}}^{t_i} \phi_i \phi_i' dt - U_k^T \int_{t_{i-1}}^{t_i} \phi_i \phi_i' dt + Q_k \int_{t_{i-1}}^{t_i} \phi_i \phi_i' dt
\]

for \( i = k - 3, k - 2, \ldots, k + 3 \). Substituting into (3.3.6), the equation becomes

\[
\bar{e}_{k,k-3} \bar{e}_{k-3} + \bar{e}_{k,k-2} \bar{e}_{k-2} + \bar{e}_{k,k-1} \bar{c}_k + \bar{e}_{k,k-1} \bar{c}_{k-1} + \bar{e}_{k,k+1} \bar{c}_k + \bar{e}_{k,k+1} \bar{c}_{k+1} + \bar{e}_{k,k+2} \bar{c}_{k+2} + \bar{e}_{k,k+3} \bar{c}_{k+3} = \bar{b}_k.
\]

for \( k = 0, 1, 2, \ldots, n \). Hence, we have

\[
\bar{E} \bar{e} = \bar{b}.
\] (3.3.7)

where \( \bar{e} \) and \( \bar{b} \) are \( m(n + 1) \) column vectors, and \( \bar{E} \) is a seven block banded \( m(n + 1) \times m(n + 1) \) matrix. If we view \( \bar{E} \) as a block matrix, then \( \bar{E} \) is symmetric. Thus, \( \bar{E} \) is an \( (n + 1) \times (n + 1) \) matrix with entries that are \( m \times m \) matrices. Similarly, \( \bar{e} \) and \( \bar{b} \) are \( (n + 1) \) column vectors with entries that are \( m \times 1 \) vectors. As in Chapter 2,
the system given by (3.3.7) is solved by the program found in Appendix II, which, in turn, gives rise to the approximate to the solution of (3.3.1).

Using the method that is found at the beginning of this chapter, it can be shown that a boundary value problem of the form

\[
(-P(t)x')' + (U(t)x) - U^T(t)x' + Q(t)x = g(t),
\]

\[a \leq t \leq b.\text{ and} \]

\[x(a) = x_a \text{ and } x(b) = x_b, \text{ where } x_a, x_b \in \mathbb{R}^m;
\]

can be transformed into the form given by

\[
(-P((b-a)w+a)x')' + (b-a)(U((b-a)w+a)x) - (b-a)U^T((b-a)w+a)x' + (b-a)^TQ((b-a)w+a)x = \tilde{g}(w).
\]

\[\tilde{g}(w) = \tilde{g}_1(w) + \tilde{g}_2(w).
\]

\[\tilde{g}_1(w) = (b-a)^T(g((b-a)w + a) - Q((b-a)w + a)(wx_b + (1-w)x_a)).
\]

\[\tilde{g}_2(w) = (b-a)(x_b - x_a)(U^T((b-a)w + a) - U((b-a)w + a)).
\]

\[0 \leq w \leq 1. \text{ and} \]

\[z(0) = \tilde{0} = z(1), \text{ where } \tilde{0} \in \mathbb{R}^m.
\]

Thus, by using this transformation, solutions to the more generalized boundary value problem given by (3.3.8) can also be approximated.

Example 3.3 Let \( P(t) = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}, Q(t) = \begin{bmatrix} 1 - e^{-t} & 1 \\ t & -e^t \end{bmatrix}, U(t) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \) and:

\[g(t) = \begin{bmatrix} t \\ 0 \end{bmatrix} \text{ for } t \in [0,1]. \text{ and let } x(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } x(1) = \begin{bmatrix} x_1(1) \\ x_2(1) \end{bmatrix} = \begin{bmatrix} e \\ 1 \end{bmatrix}.
\]

Substituting into (3.3.8), we obtain the boundary value problem
\[
\begin{bmatrix}
-1 & 0 \\
1 & -2
\end{bmatrix}
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix}
+ \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix}
- \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix}
+ \begin{bmatrix} 1 - e^{-t} & 1 \\ t & -e^t \end{bmatrix}
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix}
= \begin{bmatrix} t \\
0
\end{bmatrix}.
\]

for \( t \in [0, 1] \). It can be shown that the solution to this boundary value problem is

\[
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix}
= \begin{bmatrix} e^t \\
t
\end{bmatrix}.
\]

The absolute errors for \( h = 1/n, h/2, \) and \( h/4 \) can be found in Table III, for \( n = 8 \). As we saw in example 3.1, the rate of convergence is \( O(h^4) \), which supports the theoretical result that this method causes the approximate to converge to the solution at a rate of \( O(h^4) \).
CHAPTER 4

APPLICATION TO OPTIMAL CONTROL PROBLEMS

In this chapter, we will be converting certain optimal control problems into boundary value problems, so that we can apply the methods found in Chapter 2 and 3 to the boundary value problems. A couple of examples will then be given to support the extension of the Rayleigh-Ritz method to approximating optimal control problems.

The Optimal Control Problem

Let \( m, n \in \mathbb{N} \) and \( a, b \in \mathbb{R} \), where \( m \leq n \) and \( a < b \). Also let \( x_a, x_b \in \mathbb{R}^n \).

Define

- \( A \) \( n \times n \) matrix that is piecewise continuous and bounded;
- \( B \) \( n \times m \) matrix that is piecewise continuous and bounded;
- \( P \) \( n \times n \) matrix that is piecewise continuous, bounded, and symmetric;
- \( R \) \( m \times m \) matrix that is piecewise continuous, bounded, symmetric, and positive definite;
- \( x \) \( n \times 1 \) piecewise smooth and continuous vector; and
- \( u \) \( n \times 1 \) piecewise continuous vector.

It has been shown that the control problem, given by
\[ x' = Ax + Bu, \quad (4.1.1) \]

\[ x(a) = x_a. \]

\[ x(b) \in \mathcal{B}, \text{ where } \mathcal{B} = (x_b) \text{ or } \mathcal{B} = \mathbb{R}^n. \]

can be approximated by finding the pair \((\hat{x}, \hat{u})\) from all pairs \((x, u)\) that minimizes the following integral [5]

\[ J(x) = \int_a^b \frac{1}{2}(x^T P x + u^T R u) \, dt. \quad (4.1.2) \]

It has been shown by Athans & Falb [1] that this problem has a unique solution, and the conditions are such that an optimal solution may be found.

**Case 1: B is Invertible**

Let \(m = n\) and \(B\) be invertible, and define \(P_i, Q_i,\) and \(R_i\) to be:

\[ P_i = A'^T B'^T R B'^T A + P \quad (4.2.1) \]

\[ Q_i = B'^T R B'^T A \quad (4.2.2) \]

\[ R_i = B'^T R B'^T. \quad (4.2.3) \]

It has been shown by Dalpatadu [5] that if \(P\) is positive semi-definite, then we have the following conditions

\(P_i, Q_i,\) and \(R_i\) are piecewise continuous and bounded on \([a, b]\).

\(P_i(t)\) is symmetric and positive semi-definite for \(t \in [a, b]\).

\(R_i(t)\) is symmetric and positive definite for \(t \in [a, b]\).

Substituting (4.2.1) - (4.2.3) into (4.1.2) and rearranging, we obtain

\[ J_i(x) = \int_a^b \frac{1}{2}(x'^T P_i x - 2x'^T Q_i x + x'^T R_i x') \, dt. \quad (4.2.4) \]
Dalpatadu [5] has also shown that \((\hat{x}, \hat{u})\) is the unique solution to (4.1.1) iff \(\hat{x}\) is the unique solution that minimizes (4.2.4) and where \(\hat{u} = B^{-1}(\hat{x}' - Ax)\). It has also been shown that the Euler-Lagrange equation for (4.2.4) is [7]

\[
(-Q_i x + R_i x')' - P_i x + Q_i^T x' = 0 \quad \text{for} \ t \in [a, b].
\]

Rearranging, the equation becomes

\[
(-R_i x')' + (Q_i x)' - Q_i^T x' + P_i x = 0 \quad \text{for} \ t \in [a, b]. \tag{4.2.5}
\]

\[
x(a) = x_a.
\]

\[
x(b) \in \mathcal{B}, \text{ where } \mathcal{B} = (x_b) \text{ or } \mathcal{B} = \mathbb{R}^n.
\]

When \(x(b)\) is fixed, i.e. \(x(b) = x_b\), then equation (4.2.5) is the boundary value problem given by (3.3.8). Thus, (4.2.5) can be approximated by the methods found in Chapter 2 and 3. Also, we have that if \(\hat{x}\) is a solution to (4.2.5), then \(\hat{x}\) is a solution to (4.2.4) [5]. Hence, the pair \((\hat{x}, \hat{u})\) is the optimal solution to the control problem given by (4.1.1).

Therefore, the optimal solution to (4.1.1) may be determined by simply finding the best approximate to (4.2.5).

**Examples for Full Rank Problems**

**Example 4.1** Let \(m = n = 1; \ t \in [0, 1]; \ A(t) = -1, \ B(t) = 1, \ P(t) = 3, \text{ and } R(t) = 1.\) Also, let \(x(0) = 1\) and \(x(1) = e^{-2}.\) Then (4.1.1) becomes

\[
x'(t) = -x(t) + u(t). \tag{4.3.1}
\]

We also have

\[
P_1 = (-1)(1)(1)(-1) + 3 = 4.
\]
\[ Q_1 = (1)(1)(-1) = -1. \]
\[ R_1 = (1)(1) = 1. \]

Substituting these values into (4.2.5), we obtain
\[ -x'' - x' + x' + 4x = 0. \]
\[ -x'' + 4x = 0. \quad (4.3.2) \]

for \( t \in [0,1] \); and \( x(0) = 1 \) and \( x(1) = e^{-2} \). It can be shown that the unique solution to this boundary value problem is
\[ x(t) = e^{-2t} \quad \text{for} \quad t \in [0,1]. \]

From (4.3.1), we have
\[ u(t) = -e^{-2t} \quad \text{for} \quad t \in [0,1]. \]

Like the previous examples in Chapter 3, (4.3.2) is approximated by using the program found in Appendix II. Table IV lists the error for \( x(t) \) using the step sizes, \( h \):

\( \frac{1}{8}, \quad \frac{1}{16} \), and \( \frac{1}{32} \). From Table IV, we can see that the rate of convergence is \( O(h^4) \).

Example 4.2 Let \( m = n = 2; \ t \in [0,\sqrt{2}] \); \( P = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \); \( R = \begin{bmatrix} 1/3 & 0 \\ 0 & 1 \end{bmatrix} \); \( A = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \).

\( B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \); and \( x(0) = \begin{bmatrix} \sqrt{2} \\ 1 \end{bmatrix} \), \( x(\sqrt{2}) = \frac{1}{e^2} \begin{bmatrix} \sqrt{2} \\ 1 \end{bmatrix} \). Substituting into (4.1.1), we obtain the control problem
\[ x' = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u. \quad (4.3.3) \]

We also have

---

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ P_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1/3 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \]

\[ Q_1 = \begin{bmatrix} 1 & 0 & 1/3 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/3 \end{bmatrix}. \]

\[ R_1 = \begin{bmatrix} 1 & 0 & 1/3 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix}. \]

Substituting into (4.2.5), we obtain the boundary value problem
\[ \begin{bmatrix} -1/3 & 0 \\ 0 & -1 \end{bmatrix} x'' + \begin{bmatrix} 0 & 1/3 \\ 0 & -1 \end{bmatrix} x' + \begin{bmatrix} 1 & 0 \end{bmatrix} x = 0. \]

\[ \begin{bmatrix} -1/3 & 0 \\ 0 & -1 \end{bmatrix} x'' + \begin{bmatrix} 0 & 1/3 \\ -1/3 & 0 \end{bmatrix} x' + \begin{bmatrix} 1 & 0 \\ 0 & 4/3 \end{bmatrix} x = 0. \quad (4.3.4) \]

for \( t \in [0,\sqrt{2}] \) and \( x(0) = \begin{bmatrix} \sqrt{2} \\ 1 \end{bmatrix}, \quad x(\sqrt{2}) = \frac{1}{e^2} \begin{bmatrix} \sqrt{2} \\ 1 \end{bmatrix}. \) It can be shown that the solution to (4.3.4) is
\[ x(t) = \begin{bmatrix} \sqrt{2} e^{-\sqrt{2}t} \\ e^{-\sqrt{2}t} \end{bmatrix}. \]

From which we obtain the control vector
\[ u(t) = \begin{bmatrix} -3e^{-\sqrt{2}t} \\ (1 - \sqrt{2})e^{-\sqrt{2}t} \end{bmatrix}. \]

where \((x,u)\) is the theoretical solution to (4.3.3). Table V lists the resulting errors for this problem with a step sizes, \( h: \frac{\pi}{8}, \frac{\pi}{16}, \) and \( \frac{\pi}{32}. \) Like the previous problem, we have a convergence of \( O(h^4) \) for this example.
Example 4.3 Let \( m = n = 1; \ t \in [0.1]; \ A = 0, \ B = 1, \ P(t) = \begin{cases} 1 & \text{if } t \in [0,1/2) \\ 0 & \text{if } t \in (1/2,1] \end{cases} \).

\[ R(t) = \begin{cases} 1 & \text{if } t \in [0,1/2) \\ 2 & \text{if } t \in (1/2,1] \end{cases}; \] and \( x(0) = 1 + e, \ x(1) = 2e^{12} \). Substituting into (4.1.1),

we obtain

\[ x' = u. \] (4.3.5)

We also have

\[ P_1 = P, \]

\[ Q_1 = 0, \]

\[ R_1 = R. \]

Thus, substituting these values into (4.2.5), we obtain the boundary value problem

\[
\begin{cases}
-x'' & \text{if } t \in [0,1/2) \\
-2x'' & \text{if } t \in (1/2,1]
\end{cases}
+ \begin{cases}
x & \text{if } t \in [0,1/2) \\
0 & \text{if } t \in (1/2,1]
\end{cases} = 0.
\]

\[
\begin{cases}
-x'' + x & \text{if } t \in [0,1/2) \\
-2x'' + 0 & \text{if } t \in (1/2,1]
\end{cases} = 0. \] (4.3.6)

for \( t \in [0,1] \), and \( x(0) = 1 + e, \ x(1) = 2e^{12} \). It can be shown that the solution to both

(4.3.6) and (4.3.5) is

\[
x = \begin{cases}
e^t + e^{1-t} & \text{if } t \in [0,1/2) \\
2e^{12} & \text{if } t \in (1/2,1]
\end{cases}
\]
The results from this problem can be found in Table VI. The rate of convergence around the discontinuity, \( t = 1/2 \), is not quite the optimal rate, \( O(h^1) \), for this method because of the discontinuity. From Table VI, we can see that the overall rate of convergence is \( O(h^3) \).

So far, we have only seen examples for when \( x(b) \) is fixed. It can be easily verified that the previous methods can be used to solve control problems for when \( x(b) \) is free.

**Case 2: Rank of \( B \) is less than \( n \)**

Let \( m < n \), \( p = n - m \), and rank of \( B \) be \( m \). Clearly, \( B \) is of full column rank. Thus, if \( B \) is also constant then there exists a nonsingular matrix \( G \) such that

\[
GB = \begin{bmatrix} 0 \\ \overline{B_2} \end{bmatrix},
\]

where \( \overline{B_2} \) is \( m \times m \) and nonsingular [5].

By multiplying (4.1.1) by \( G \), we have

\[
Gx' = GAx + GBu
\]

which can be written as

\[
\left( Gx \right)' = GAG^{-1}Gx + GBu.
\] (4.4.1)

Now, define

\[
w = Gx \\
\overline{A} = GAG^{-1}
\]
\[
\overline{B} = GB
\]
\[
\overline{P} = G^{-1}TPG^{-1}.
\]
Substituting into (4.4.1), the equation becomes
\[
w' = \overline{A}w + \overline{B}u.
\] (4.4.2)
where \(\overline{A}\) can be written as
\[
\begin{bmatrix}
\overline{A}_1 \\
\overline{A}_2
\end{bmatrix}.
\]
Hence, (4.4.2) has the form
\[
\begin{bmatrix}
w_1 \\
w_2
\end{bmatrix}' = \begin{bmatrix}
\overline{A}_1 \\
\overline{A}_2
\end{bmatrix}w + \begin{bmatrix}
0 \\
\overline{B}_2
\end{bmatrix}u,
\]
where \(\overline{A}_1\) is \(p \times n\), \(\overline{A}_2\) is \(m \times n\), \(w\) is \(p \times 1\), and \(w\) is \(m \times 1\).

Therefore, we can assume that (4.1.1) can be written as
\[
x'_1 = A_1x
\] (4.4.3)
\[
x'_2 = A_2x + B_2u,
\] (4.4.4)
where \(A_1\) is \(p \times n\), \(A_2\) is \(m \times n\), \(x_1\) is \(p \times 1\), and \(x_2\) is \(m \times 1\).

(If it can not be written in this form, we can find a nonsingular constant matrix that will transform (4.1.1) into the form given by (4.4.3) and (4.4.4).) [5]

Define
\[
P_2 = A_2^T B_2^{-1}T RB_2^{-1} A_2 + P
\]
\[
Q_2 = B_2^{-1}T RB_2^{-1} A_2
\]
\[
R_2 = B_2^{-1}T RB_2^{-1}.
\]
Substituting into (4.1.2), we obtain the integral

\[ J_2(x) = \int_a^b \frac{1}{2} \left( x^T P_2 x - 2x'^T Q_2 x + x'^T R_1 x \right) dt. \]  

(4.4.5)

It can be verified that if \( P \) is positive semi-definite on \([a, b]\), then we have the following [5]

- \( P_2, Q_2, \) and \( R_2 \) are piecewise continuous and bounded on \([a, b]\).
- \( P_2(t) \) is symmetric and positive semi-definite for \( t \in [a, b] \).
- \( R_2(t) \) is symmetric and positive definite for \( t \in [a, b] \).

Furthermore, Dalpatadu [5] has shown that \((\hat{x}, \hat{u})\) is the unique solution to (4.4.3) and (4.4.4) iff \( \hat{x} \) is the unique solution that minimizes (4.4.5). and where \( \hat{u} = B_2^{-1}(\hat{x}' - A_2 x) \).

Let \( \lambda \) be the Lagrange multiplier of (4.4.5), then we have

\[ J_2' (x; \lambda) = J_2(x) + \int_a^b \lambda^T (x'_1 - A_1 x) dt \]  

(4.4.6)

Hestenes [8] has shown that for this problem, \( \lambda \) is piecewise continuous. Define the function \( x_3 \) to be

\[ \int_a^b \lambda(s) ds, \]

then \( x_3 \) is \( p \times 1 \) and piecewise smooth and continuous, and

\[ x'_1(t) = \lambda(t); \]

where \( t \in [a, b] \). Define

\[ P_3 = \begin{bmatrix} P_2 & 0 \\ 0 & 0 \end{bmatrix}. \]
\[
Q_3 = \begin{bmatrix}
0 & 0 \\
Q_2 & 0 \\
A_1 & 0
\end{bmatrix},
\]

\[
R_3 = \begin{bmatrix}
0 & 0 & 1 \\
0 & R_2 & 0 \\
1 & 0 & 0
\end{bmatrix},
\]

\[
X = \begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix},
\]

where \( P_3, Q_3, \) and \( R_3 \) are \((n+p) \times (n+p)\), and \( X \) is \((n+p) \times 1\). Thus,

\[
J_2' (x; \lambda) = J_3 (X) = \int_a^b \frac{1}{2} \left( X^T P_3 X - 2 X^T Q_3 X + X^T R_3 X \right) dt. \quad (4.4.7)
\]

By rearranging the Euler-Lagrange equation for (4.4.7), we obtain

\[
(-R_3 X')' + (Q_3 X)' - Q_3' X' + P_3 X = 0; \quad (4.4.8)
\]

with the boundary conditions [5]

\[
X(a) = \begin{bmatrix}
x_a \\
0
\end{bmatrix},
\]

\[
x(b) = x_b,
\]

\[
x_1'(b) - A_1(b)x(b) = 0.
\]

\[
x_2'(b) = 0.
\]

\[
R_2(b)x_2'(b) - Q_2(b)x(b) = 0.
\]

Also, the variational form for (4.4.8) is

\[
\int_a^b \left( Y^T P_3 X - Y'^T Q_3 X - Y^T Q_3' X' + Y'^T R_3 X' \right) dt = 0, \quad (4.4.9)
\]

where \( Y \) is a piecewise smooth and continuous function, such that
\[ Y(a) = 0 \text{ : and} \]
\[
Y(b) = \begin{bmatrix} 0 \\ 0 \\ y_{3b} \end{bmatrix} \text{ if } x(b) = x_b, \text{ for } y_{3b} \in \mathbb{R}^p \text{ : or } \]
\[ Y(b) \in \mathbb{R}^{n+p} \text{ if } x(b) \in \mathbb{R}^n. \]

It has been shown that a solution to (4.4.9) is a solution to (4.4.8) which is a solution to (4.4.5) which, in turn, is a solution to our original control problem. Thus, we can find the pair \((\hat{x}, \hat{u})\) which approximates the solution to (4.1.1) by simply determining the approximate to the boundary value problem given by (4.4.8). Since (4.4.8) is of the same form as (4.2.5), we can use the previous methods to approximate the solution to (4.4.8).

In conclusion, we have shown that the solution to boundary value problems can be approximated by reformulating the boundary value problem to become a minimization problem. The solution to the minimization problem is then approximated by using cubic B-splines. Once the approximate to the minimization problem is found, that approximate is also the approximation of the solution to the boundary value problem.

Ultimately, we have shown that we can extend this technique to approximating certain optimal control problems. These optimal control problems are approximated by first converting them into two-point boundary value problems. From which, we determine the variational form and apply the methods in chapter 2 and 3 to determine the approximate of the solution. For the case when the rank of B is n, the transformation process is fairly straightforward, but for the case when the rank of B is less than n, the process becomes a little more complicated. To handle a problem of this type, u is first eliminated to obtain an equivalent variational problem. Then, Lagrange multipliers are
used to obtain a problem of full rank. Once we obtain a problem of full rank, we can use the previous numerical methods on it to approximate the solution to the resulting boundary value problem.
APPENDIX I

RESULTS

Table I - Example 3.1

<table>
<thead>
<tr>
<th>$t_k$</th>
<th>$x(t_k)$</th>
<th>Error, h</th>
<th>Error, h/2</th>
<th>Error, h/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>0.00000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
<tr>
<td>0.1000</td>
<td>-0.09000</td>
<td>8.81278D-08</td>
<td>5.56849D-09</td>
<td>3.39585D-10</td>
</tr>
<tr>
<td>0.2000</td>
<td>-0.16000</td>
<td>5.43477D-08</td>
<td>3.37425D-09</td>
<td>2.11364D-10</td>
</tr>
<tr>
<td>0.3000</td>
<td>-0.21000</td>
<td>4.90307D-08</td>
<td>2.96972D-09</td>
<td>1.89145D-10</td>
</tr>
<tr>
<td>0.4000</td>
<td>-0.24000</td>
<td>5.12972D-08</td>
<td>3.08887D-09</td>
<td>1.92598D-10</td>
</tr>
<tr>
<td>0.5000</td>
<td>-0.25000</td>
<td>1.56835D-08</td>
<td>9.75053D-10</td>
<td>6.11048D-11</td>
</tr>
<tr>
<td>0.6000</td>
<td>-0.24000</td>
<td>4.35639D-08</td>
<td>2.44919D-09</td>
<td>1.68154D-10</td>
</tr>
<tr>
<td>0.7000</td>
<td>-0.21000</td>
<td>4.54073D-08</td>
<td>2.63293D-09</td>
<td>1.75537D-10</td>
</tr>
<tr>
<td>0.8000</td>
<td>-0.16000</td>
<td>3.84609D-08</td>
<td>2.33470D-09</td>
<td>1.50448D-10</td>
</tr>
<tr>
<td>0.9000</td>
<td>-0.09000</td>
<td>6.68672D-08</td>
<td>4.09553D-09</td>
<td>2.60663D-10</td>
</tr>
<tr>
<td>1.0000</td>
<td>0.00000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
</tbody>
</table>
Table II - Example 3.2

<table>
<thead>
<tr>
<th>$t_k$</th>
<th>$x(t_k)$</th>
<th>Error. $h$</th>
<th>Error. $h/2$</th>
<th>Error. $h/4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.0000</td>
<td>-1.00000000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
<tr>
<td>-0.8000</td>
<td>-1.22778525</td>
<td>2.37110D-04</td>
<td>9.67287D-06</td>
<td>5.95585D-07</td>
</tr>
<tr>
<td>-0.6000</td>
<td>-1.31105652</td>
<td>3.05432D-05</td>
<td>1.82796D-06</td>
<td>9.48364D-08</td>
</tr>
<tr>
<td>-0.4000</td>
<td>-1.11105652</td>
<td>7.16574D-04</td>
<td>3.99792D-05</td>
<td>2.79145D-06</td>
</tr>
<tr>
<td>-0.2000</td>
<td>-0.62778525</td>
<td>8.54020D-04</td>
<td>1.29567D-04</td>
<td>1.32598D-05</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.00000000</td>
<td>1.52078D-05</td>
<td>2.51462D-06</td>
<td>4.28048D-07</td>
</tr>
<tr>
<td>0.2000</td>
<td>0.62778525</td>
<td>8.84740D-04</td>
<td>1.25709D-04</td>
<td>1.35154D-05</td>
</tr>
<tr>
<td>0.4000</td>
<td>1.11105652</td>
<td>6.96045D-04</td>
<td>5.41583D-05</td>
<td>2.75537D-06</td>
</tr>
<tr>
<td>0.6000</td>
<td>1.31105652</td>
<td>3.03087D-05</td>
<td>1.65145D-06</td>
<td>9.30448D-08</td>
</tr>
<tr>
<td>0.8000</td>
<td>1.22778525</td>
<td>2.30638D-04</td>
<td>8.82019D-06</td>
<td>5.73663D-07</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.00000000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
</tbody>
</table>
Table III - Example 3.3

<table>
<thead>
<tr>
<th>$t_k$</th>
<th>$x_1(t_k)$</th>
<th>Error. h</th>
<th>Error. h/2</th>
<th>Error. h/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000</td>
<td>1.000000000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
<tr>
<td>0.12500</td>
<td>1.13148453</td>
<td>1.61073D-04</td>
<td>9.91420D-06</td>
<td>6.26178D-07</td>
</tr>
<tr>
<td>0.25000</td>
<td>1.284025417</td>
<td>1.61918D-04</td>
<td>1.00310D-06</td>
<td>6.30012D-07</td>
</tr>
<tr>
<td>0.37500</td>
<td>1.454991415</td>
<td>5.80524D-05</td>
<td>3.56026D-06</td>
<td>2.22576D-07</td>
</tr>
<tr>
<td>0.50000</td>
<td>1.647821217</td>
<td>5.41369D-07</td>
<td>9.89932D-06</td>
<td>6.85394D-10</td>
</tr>
<tr>
<td>0.62500</td>
<td>1.868245957</td>
<td>5.70435D-05</td>
<td>1.00310D-06</td>
<td>6.9361D-07</td>
</tr>
<tr>
<td>0.75000</td>
<td>2.117000017</td>
<td>1.60751D-04</td>
<td>9.89932D-06</td>
<td>6.10385D-07</td>
</tr>
<tr>
<td>0.87500</td>
<td>2.398875294</td>
<td>1.60076D-04</td>
<td>9.78295D-06</td>
<td>6.10385D-07</td>
</tr>
<tr>
<td>1.00000</td>
<td>2.718281828</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$t_k$</th>
<th>$x_2(t_k)$</th>
<th>Error. h</th>
<th>Error. h/2</th>
<th>Error. h/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000</td>
<td>0.000000000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
<tr>
<td>0.12500</td>
<td>0.125000000</td>
<td>2.31563D-07</td>
<td>1.18039D-09</td>
<td>6.08148D-10</td>
</tr>
<tr>
<td>0.25000</td>
<td>0.250000000</td>
<td>1.40108D-07</td>
<td>7.72436D-09</td>
<td>4.00216D-10</td>
</tr>
<tr>
<td>0.37500</td>
<td>0.375000000</td>
<td>2.07881D-07</td>
<td>1.22142D-08</td>
<td>7.20379D-10</td>
</tr>
<tr>
<td>0.50000</td>
<td>0.500000000</td>
<td>1.75554D-07</td>
<td>9.77047D-09</td>
<td>5.41338D-10</td>
</tr>
<tr>
<td>0.62500</td>
<td>0.625000000</td>
<td>2.19917D-07</td>
<td>1.22142D-08</td>
<td>7.59058D-10</td>
</tr>
<tr>
<td>0.75000</td>
<td>0.750000000</td>
<td>1.64475D-07</td>
<td>9.12985D-09</td>
<td>4.97386D-10</td>
</tr>
<tr>
<td>0.87500</td>
<td>0.875000000</td>
<td>2.92047D-07</td>
<td>1.53600D-08</td>
<td>8.09459D-10</td>
</tr>
<tr>
<td>1.00000</td>
<td>1.000000000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
</tbody>
</table>
Table IV - Example 4.1

<table>
<thead>
<tr>
<th>$t_k$</th>
<th>$x(t_k)$</th>
<th>Error, h</th>
<th>Error, h/2</th>
<th>Error, h/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000</td>
<td>1.000000000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
<tr>
<td>0.12500</td>
<td>0.778800783</td>
<td>3.80050D-06</td>
<td>2.30159D-07</td>
<td>2.42128D-09</td>
</tr>
<tr>
<td>0.25000</td>
<td>0.606530660</td>
<td>3.53155D-06</td>
<td>1.98657D-07</td>
<td>1.24225D-08</td>
</tr>
<tr>
<td>0.37500</td>
<td>0.472366553</td>
<td>2.40254D-06</td>
<td>1.46628D-07</td>
<td>9.07871D-09</td>
</tr>
<tr>
<td>0.50000</td>
<td>0.367879441</td>
<td>2.04009D-06</td>
<td>1.20708D-07</td>
<td>7.88572D-09</td>
</tr>
<tr>
<td>0.62500</td>
<td>0.286504797</td>
<td>1.48589D-06</td>
<td>8.83653D-08</td>
<td>5.57215D-09</td>
</tr>
<tr>
<td>0.75000</td>
<td>0.223130160</td>
<td>1.27062D-06</td>
<td>6.63835D-08</td>
<td>3.86922D-09</td>
</tr>
<tr>
<td>0.87500</td>
<td>0.173773943</td>
<td>8.27987D-07</td>
<td>5.14701D-08</td>
<td>2.12681D-09</td>
</tr>
<tr>
<td>1.00000</td>
<td>0.135335000</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
</tbody>
</table>
Table V - Example 4.2

<table>
<thead>
<tr>
<th>$t_k$</th>
<th>$x_1(t_k)$</th>
<th>Error. $h$</th>
<th>Error. $h/2$</th>
<th>Error. $h/4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>1.414213562</td>
<td>0.000000D-00</td>
<td>0.000000D-00</td>
<td>0.000000D-00</td>
</tr>
<tr>
<td>0.17678</td>
<td>1.101390620</td>
<td>7.09658D-03</td>
<td>4.21704D-04</td>
<td>2.59846D-05</td>
</tr>
<tr>
<td>0.35355</td>
<td>0.857763885</td>
<td>3.42748D-04</td>
<td>2.05416D-05</td>
<td>1.21885D-06</td>
</tr>
<tr>
<td>0.53033</td>
<td>0.668027191</td>
<td>1.16922D-04</td>
<td>7.26341D-06</td>
<td>4.43962D-07</td>
</tr>
<tr>
<td>0.70711</td>
<td>0.520260104</td>
<td>2.29454D-04</td>
<td>1.37037D-05</td>
<td>8.79214D-07</td>
</tr>
<tr>
<td>0.88388</td>
<td>0.405178980</td>
<td>2.98430D-04</td>
<td>1.83499D-05</td>
<td>1.10942D-06</td>
</tr>
<tr>
<td>1.06066</td>
<td>0.315553709</td>
<td>3.00570D-04</td>
<td>1.79840D-05</td>
<td>1.14003D-06</td>
</tr>
<tr>
<td>1.23744</td>
<td>0.245753478</td>
<td>2.08245D-04</td>
<td>1.30001D-05</td>
<td>8.02457D-07</td>
</tr>
<tr>
<td>1.41421</td>
<td>0.191392993</td>
<td>0.000000D-00</td>
<td>0.000000D-00</td>
<td>0.000000D-00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$t_k$</th>
<th>$x_2(t_k)$</th>
<th>Error. $h$</th>
<th>Error. $h/2$</th>
<th>Error. $h/4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>1.000000000</td>
<td>0.000000D-00</td>
<td>0.000000D-00</td>
<td>0.000000D-00</td>
</tr>
<tr>
<td>0.17678</td>
<td>0.778800790</td>
<td>2.55548D-04</td>
<td>1.49672D-05</td>
<td>9.58234D-07</td>
</tr>
<tr>
<td>0.35355</td>
<td>0.606530670</td>
<td>1.07445D-04</td>
<td>6.70884D-06</td>
<td>4.10970D-07</td>
</tr>
<tr>
<td>0.53033</td>
<td>0.472366565</td>
<td>1.91113D-04</td>
<td>1.17933D-05</td>
<td>7.36551D-07</td>
</tr>
<tr>
<td>0.70711</td>
<td>0.367879454</td>
<td>1.24346D-04</td>
<td>7.76869D-06</td>
<td>4.75726D-07</td>
</tr>
<tr>
<td>0.88388</td>
<td>0.286504809</td>
<td>1.56420D-04</td>
<td>9.59913D-06</td>
<td>5.90156D-07</td>
</tr>
<tr>
<td>1.06066</td>
<td>0.223130172</td>
<td>1.51683D-04</td>
<td>9.45393D-06</td>
<td>5.90251D-07</td>
</tr>
<tr>
<td>1.23744</td>
<td>0.173773954</td>
<td>1.01983D-04</td>
<td>6.34587D-06</td>
<td>3.92387D-07</td>
</tr>
<tr>
<td>1.41421</td>
<td>0.135335283</td>
<td>0.000000D-00</td>
<td>0.000000D-00</td>
<td>0.000000D-00</td>
</tr>
</tbody>
</table>
Table VI - Example 4.3

<table>
<thead>
<tr>
<th>$t_k$</th>
<th>$x(t_k)$</th>
<th>Error. h</th>
<th>Error. h/2</th>
<th>Error. h/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000</td>
<td>3.718281828</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
<tr>
<td>0.12500</td>
<td>3.532023747</td>
<td>1.65111D-05</td>
<td>1.03113D-06</td>
<td>6.42690D-08</td>
</tr>
<tr>
<td>0.25000</td>
<td>3.401025433</td>
<td>3.39509D-05</td>
<td>2.11758D-06</td>
<td>1.30617D-07</td>
</tr>
<tr>
<td>0.37500</td>
<td>3.323237372</td>
<td>5.05771D-04</td>
<td>3.89546D-05</td>
<td>2.98762D-06</td>
</tr>
<tr>
<td>0.50000</td>
<td>3.297442436</td>
<td>6.59694D-04</td>
<td>6.52927D-05</td>
<td>6.49115D-06</td>
</tr>
<tr>
<td>0.62500</td>
<td>3.297442436</td>
<td>5.06525D-04</td>
<td>4.20104D-05</td>
<td>3.46172D-06</td>
</tr>
<tr>
<td>0.75000</td>
<td>3.297442436</td>
<td>3.39105D-05</td>
<td>2.10993D-06</td>
<td>1.29562D-07</td>
</tr>
<tr>
<td>0.87500</td>
<td>3.297442436</td>
<td>1.68723D-05</td>
<td>1.04541D-06</td>
<td>6.50674D-08</td>
</tr>
<tr>
<td>1.00000</td>
<td>3.297442436</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
<td>0.00000D-00</td>
</tr>
</tbody>
</table>
APPENDIX II

FORTRAN 77 PROGRAMS

Main Program

* The following program solves the boundary value problem (BVP):
* \((-P(t)y')' + (U(t)y')' - U^T(t)y' + Q(t)y = f(t).\)
* where \(t \in [a, b]\), \(y(a) = \alpha\) and \(y(b) = \beta\); using the Rayleigh-Ritz method.
* The above BVP is changed to become a problem of solving the linear
* system of equations:
* \(Ec = b\).
* where \(e\) will be used to determine \(y\).
*
* Parameters:
* \(a_0\): endpoint \(a\) in BVP.
* \(b_0\): endpoint \(b\) in BVP.
* \(size\): size of vector \(y\).
* \(n\) denotes the number of partitioned intervals in \([a, b]\).
* \(nsize\): size of matrix \(E\) and vectors \(e\) and \(b\).
* \(m_1\): number of bands below the diagonal in \(E\).
* \(m_2\): number of bands above the diagonal in \(E\).

Program CubicB
Integer i,j,k,n,m,ncheck,qcheck,m1,m2,np,mp,mpi,size,nsize.
Double Precision PI,a0,b0,alpha,beta,u,ut
Parameter(b0=1.0,a0=0.0,size=2,m1=3*size,m2=3*size,n=16,
        nsize=(n+1)*size-1, PI=3.141592653589)
Integer indx(nsize+1),d,mcount
Double Precision t(0:n),h,Ttemp,e(0:nsize,0:nsize),
c eo(0:nsize,0:nsize),err((n-1)*size),r,p,q,b(0:nsize),bo(0:nsize),
c x(0:nsize),func,S1,S2,S3,S4,Phi,dp,fz,y
c .a(ns+1,8*size-1),al(ns+1,ml+size-1)
  External func,r,p,dp,q,f,y,z,qromb,S1,S2,S3,S4,Phi,alpha,beta,u,ut
003 format(22(E11.5,1X))
004 format(15(E8.2,1X))

* Initialize variables:

  m=1
  nn=ns+1
  h=1.0/n
  Ttemp=0.0
  check=99
  qcheck=0
  k=1
  np=ns+1
  mp=8*size-1
  mpl=ml+size-1
  mm=8*size-1

* Partitions [a,b].

  Do i=0, n
    t(i)=Ttemp
    Ttemp=Ttemp+h
  end do

* Solves for vector b and the upper triangular banded part of matrix E.

  Call VectorB(t,n,size,check,qcheck,h,b)
  Call MatrixE(t,n,size,check,qcheck,h,e)

* The lower triangular banded part of matrix E is determined

  do i=size,ns
    mcount=1+size*INT((i-size)/size)
    do j=0,mcount
      c(i,j)=e(size*INT(j/size)+MOD(i, size), i-MOD(i, size)+
        MOD(j, size))
    end do
  end do

* Stores matrix E into another matrix.

  Do i=0,ns
Do j=0,nsize 
    eo(i,j)=e(i,j) 
end do 

x(i)=0.0 
bo(i)=b(i) 
end do

* Determines whether to use gaussian elimination to solve for vector \( \mathbf{e} \). 
* or to use another method.

If (nsize.le.40) then 
    Call gaussj(e.nn,nn,b,m,m) 
else

* Subroutine Transform transforms matrix \( \mathbf{E} \) from an \((nsize + 1) \times (nsize + 1)\) into 
* matrices that make up the upper and lower bands of \( \mathbf{E} \), and stores them in \( \mathbf{a} \) and \( \mathbf{al} \), respectively. Then, subroutine \text{LUdecomp} \) decomposes \( \mathbf{E} \) into a product of 
* upper and lower triangular matrices. Finally, Bandsolver solves for \( \mathbf{c} \).

    Call Transform(a.e,m1,mm,n.size) 
    Call LUdecomp(a.nn.size,m1,m2,np,mm,al,ml,indx,d) 
    Call BandSolver(a.nn.size,m1,m2,np,mm,al,ml,indx,b)
end if

* Used to make the solution that is produced by either the subroutine 
* \text{guassj} \) or Bandsolver more accurate.

    Call Gseidel(eo,x,b,bo,nn.size)

* Determine the absolute error between the approximate and the 
* actual solution.

    Call Error(b,t,n.size,err)
end
List of Functions

* Transformation of the general BVP.

Function z(x,index)
Integer index
Double Precision z,y,x,alpha,beta,a,b
Parameter (a=0.0,b=1.0)
External y,alpha,beta
z=y( ( b-a ) * x+a, index )-beta( index ) * x-( 1.0-x ) * alpha( index )
end

* Actual solution to BVP. used to calculate the absolute error.

Function y(x,index)
Integer index
Double Precision y.x.PI.f
External f

PI=3.14159265359
If (index.eq.1) then
  y=exp(x)
else if (index.eq.2) then
  y=x
else
  print *,error in y'
end if
end

* Transformation of the RHS of the BVP.

Function r(x,index,size)
Integer index.size
Double Precision x,r,PI.q,f,alpha,beta,a,b,tempB,tempC,u.ut
Parameter(PI=3.14159265359,b=1.0,a=0.0)
External q,f,alpha.beta.u.ut

  TempC=0.0
  TempB=0.0
  do i=1,size
    TempB=TempB+q( (b-a)*x+a,index*size+i)*beta(i)*x+c(1.0-x)*alpha(i)
    TempC=TempC+(UT((b-a)*x+a,index*size+i)-c.U((b-a)*x+a,index*size+i))*(beta(i)-alpha(i))
  end do
r=((b-a)**2)*f((b-a)*x+a,index)-TempB)+(b-a)*TempC
end

* The RHS of the BVP.

Function f(x,index)
Integer index
Double Precision f x.PI.y
External y
PI=3.14159265359

if (index.eq.0) then
  f=0.0
else if (index.eq.1) then
  f=0.0
else
  print *,'error in f'
end if
end

* The value of y at a.

Function Alpha(m)
Integer m
Double Precision Alpha
if (m.eq.1) then
  Alpha=1.0
else if (m.eq.2) then
  Alpha=0.0
else
  print *,'error in Alpha'
end if
end

* The value of y at b.

Function Beta(m)
Integer m
Double Precision Beta
if (m.eq.1) then
  Beta=Exp(1.0)
else if (m.eq.2) then
  Beta=1.0
else
  print *,'error in Beta'
end if
end

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
end if
end

* Matrix P in BVP.

Function p(x,index)
Integer index
Double Precision x,p,PI
Pl=3.14159265359

If (index.eq.1) then
  p=1.0
else if (index.eq.2) then
  p=0.0
else if (index.eq.3) then
  p=-1.0
else if (index.eq.4) then
  p=2.0
else
  print*,'error in p'
end if
end

* The matrix Q in the BVP.

Function q(x,index)
Integer index
Double Precision x,q,Pl
Pl=3.14159265359
If (index.eq.1) then
  q=1.0-EXP(-x)
else if (index.eq.2) then
  q=0.0
else if (index.eq.3) then
  q=x
else if (index.eq.4) then
  q=-EXP(x)
else
  print*,'error in q'
end if
end

* The matrix U in the BVP.
Function U(x,index)
  Integer index
  Double Precision U.x

  IF (index.eq.1) then
    U=1.0
  else if (index.eq.2) then
    U=1.0
  else if (index.eq.3) then
    U=0.0
  else if (index.eq.4) then
    U=1.0
  else
    print *,'error in U'
end if
end

* The transpose of the matrix U.

Function UT(x,index)
  Integer index
  Double Precision x.UT

if (index.eq.1) then
  UT=1.0
else if (index.eq.2) then
  UT=0.0
else if (index.eq.3) then
  UT=1.0
else if (index.eq.4) then
  UT=1.0
else
  print *,'error in UT'
end if
end

* \( \phi_{t_k} \) for \( t \in [t_{k-2} \cdot t_{k-1}] \).

Function S1(x,check)
  Double Precision S1.x
  integer check

If (check.eq.99) then
  S1=(1.0/4.0)*((2.0-x)**3-4*(1.0-x)**3-6.0*x**3+4.0*
\begin{verbatim}
c    (1.0+x)**3)
else if (check.eq.999) then
    S1=(1.0/4.0)*(-3.0*(2.0-x)**2+12.0*(1.0-x)**2-18.0*
    c    x**2+12.0*(1.0+x)**2)
else
    print *. 'error in s1 check'
end if
end

* \( \phi_{i,k} \) for \( t \in [t_{k-1}, t_k] \).

Function S2(x,check)
Double Precision S2.x
integer check

If (check.eq.99) then
    S2=(1.0/4.0)*((2.0-x)**3-4.0*(1.0-x)**3-6.0*x**3)
else if (check.eq.999) then
    S2=(1.0/4.0)*(-3.0*(2.0-x)**2+12.0*(1.0-x)**2-
    c    18.0*x**2)
else
    print *. 'error in s2 check'
end if
end

* \( \phi_{i,k} \) for \( t \in [t_k, t_{k-1}] \).

Function S3(x,check)
Double Precision S3.x
Integer check

If (check.eq.99) then
    S3=(1.0/4.0)*((2.0-x)**3-4.0*(1.0-x)**3)
else if (check.eq.999) then
    S3=(1.0/4.0)*(-3.0*(2.0-x)**2+12.0*(1.0-x)**2)
else
    print *. 'error in s3 check'
end if
end

* \( \phi_{i,k} \) for \( t \in [t_{k+1}, t_{k-2}] \).
\end{verbatim}
Function $S_4(x, \text{check})$
Double Precision $S_4.x$
integer check

If (check.eq.99) then
   $S_4=(1.0/4.0)*(2.0-x)^3$
else if (check.eq.999) then
   $S_4=(1.0/4.0)*(-3.0*(2.0-x)^2)$
else
   print *. 'error in s4 check'
end if
end

* Determines: $\phi_{1,k}$

Function $\Phi(x, i, j, n, h, \text{check})$
Double Precision $\Phi.x, h, S_1, S_2, S_3, S_4$
Integer $i, j, n, \text{check}$
External $S_1, S_2, S_3, S_4$

If ((i.eq.0).AND.(i.eq.j)) then
   $\Phi=S_3(x/h, \text{check})-4.0*S_4((x+h)/h, \text{check})$
else if ((i.eq.0).AND.(j.eq.1)) then
   $\Phi=S_4(x/h, \text{check})$
else if ((i.eq.1).AND.(j.eq.0)) then
   $\Phi=S_2((x-h)/h, \text{check})-S_4((x+h)/h, \text{check})$
else if ((i.eq.1).AND.(j.eq.1)) then
   $\Phi=S_3((x-h)/h, \text{check})$
else if ((i.ge.2).AND.(i.le.n-2)).AND.((i-j).eq.2) then
   $\Phi=S_1((x-i*h)/h, \text{check})$
else if ((i.ge.2).AND.(i.le.n-2)).AND.((i-j).eq.1) then
   $\Phi=S_2((x-i*h)/h, \text{check})$
else if ((i.ge.2).AND.(i.le.n-2)).AND.(i.eq.j) then
   $\Phi=S_3((x-i*h)/h, \text{check})$
else if ((i.ge.2).AND.(i.le.n-2)).AND.((j-i).eq.1) then
   $\Phi=S_4((x-i*h)/h, \text{check})$
else if ((i.eq.n-1).AND.(j.eq.n-3)) then
   $\Phi=S_1((x-i*h)/h, \text{check})$
else if ((i.eq.n-1).AND.(j.eq.n-2)) then
   $\Phi=S_2((x-i*h)/h, \text{check})$
else if ((i.eq.n-1).AND.(i.eq.j)) then
   $\Phi=S_3((x-i*h)/h, \text{check})-S_1((x-(i+2)*h)/h, \text{check})$
else if ((i.eq.n).AND.(j.eq.n-2)) then
   $\Phi=S_1((x-i*h)/h, \text{check})$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
else if ((i.eq.n).AND.(j.eq.n-l)) then
  Phi=S2((x-i*h)/h,check)-4.0*S1((x-(i+1)*h)/h,check)
else
  Phi=0.0
end if
end

*  Solves the integral given by (2.6.5) in thesis.

Function func(x,i,j,k,n,h,check,qcheck)
Double Precision func,x,h,Phi,r,p,q,a
Parameter (b=1.0,a=0.0)
Integer i,j,n,check,k,qcheck
External Phi,r,p,q

If (check.eq.99) then
  func=Phi(x,i,j,n,h,check)
else if ((check.eq.1000).AND.(qcheck.eq.0)) then
  func=(Phi(x,i,j,n,h,999)*Phi(x,k,j,n,h,999))/h**2
else if ((check.eq.1000).AND.(qcheck.eq.1)) then
  func=Phi(x,i,j,n,h,99)*Phi(x,k,j,n,h,99)
else if ((check.eq.1000).AND.(qcheck.eq.2)) then
  func=(Phi(x,i,j,n,h,999)*Phi(x,k,j,n,h,99))
else if ((check.eq.1000).AND.(qcheck.eq.3)) then
  func=(Phi(x,i,j,n,h,99)*Phi(x,k,j,n,h,999))
else
  print *, 'error in func'
endif
end

Subroutine BandSolver

Subroutine BandSolver(a,n,size,m1,m2,np,mp,al,mpl,indx,b)
Integer m1,m2,mp,mpl,n,np,indx(n),size
Double Precision a(np,mp),al(np, mpl),b(n),dum
Integer i,k,l,mm
mm=8*size-l
If(mm.gt.mp.or.m1.gt.mpl.or.n.gt.np) pause 'bad ars in BandSolver'
l=m1
do k=1,n
  i=indx(k)
  if(i.ne.k)then
    dum=b(k)
    b(k)=b(i)
    b(i)=dum
  end if
  if(l.lt.n)l=l+1
  do i=k+1,l
    b(i)=b(i)-al(k,i-k)*b(k)
  end do
end do
l=1
do i=n,1,-1
  dum=b(i)
  do k=2,l
    dum=dum-a(i,k)*b(k+i-1)
  end do
  b(i)=dum/a(i,1)
  if(l.lt.mm)l=l+1
end do
return
end
Subroutine-Error

Subroutine Error(b,t,n,size,err)
Integer n,size,nn,Timecount,count
Double Precision aprox,actuLerr(0:(n-1)*size-1),
c b(0:(n+1)*size-1).t(0:n),z,y
External y,z

nn=n*size
count=1
Timecount=1
Do i=size.nn-1
  aprox=0.25*b(i-size)+b(i)+0.25*b(i+size)
  If (count.lt.size) then
    actuL=z(t(Timecount),count)
    count=count+1
  Else
    actuL=z(t(Timecount),count)
    count=1
    Timecount=Timecount+1
  end if
  err(i-size)=ABS(aprox-actuL)
  print *, aprox,actuL,err(i-size)
end do
return
end
SUBROUTINE gaussj(a,n,np,b,m,mp)
INTEGER m,mp,n,np,NMAX
Double Precision a(np,np),b(np,mp)
PARAMETER (NMAX=50)
INTEGER i,icol,irow,j,k,l,ll,indxc(NMAX),indxr(NMAX),
ipiv(NMAX)
REAL big,dum,pivinv
DO 11 j=1,n
   ipiv(j)=0
11 CONTINUE
DO 22 i=1,n
   big=0.
   DO 13 j=1,n
      IF(ipiv(j).NE.1)THEN
         DO 12 k=1,n
            IF(ipiv(k).EQ.0)THEN
               IF(abs(a(j,k)).GE.big)THEN
                  big=abs(a(j,k))
                  irow=j
                  icol=k
                  ENDIF
               ELSE IF(ipiv(k).GT.1)THEN
                  PAUSE 'Singular matrix in gaussj'
                  ENDIF
            ENDIF
12 CONTINUE
            IF(ipiv(icol).NE.ipiv(icol)+1)
               IF(irow.NE.icol)THEN
                  DO 14 l=1,n
                     dum=a(irow,l)
                     a(irow,l)=a(icol,l)
                     a(icol,l)=dum
14 CONTINUE
                  DO 15 l=1,m
                     dum=b(irow,l)
                     b(irow,l)=b(icol,l)
                     b(icol,l)=dum
15 CONTINUE
               ENDIF
            ENDIF
   ENDIF
22 CONTINUE
END
if (a(icol,icol).eq.0.) pause 'singular matrix in gaussj'
pivinv=1./a(icol,icol)
a(icol,icol)=1.
do 16 l=1.n
   a(icol,l)=a(icol,l)*pivinv
16 continue
do 17 l=1.m
   b(icol,l)=b(icol,l)*pivinv
17 continue
do 21 ll=1.n
   if(ll.ne.icol)then
      dum=a(ll,icol)
      a(ll,icol)=0.
do 18 l=1.n
      a(ll,l)=a(ll,l)-a(icol,l)*dum
18 continue
do 19 l=1.m
   b(ll,l)=b(ll,l)-b(icol,l)*dum
19 continue
endif
21 continue
22 continue
do 24 l=n.l.-1
   if(indxr(l).ne.indxc(l))then
      do 23 k=1,n
         dum=a(k,indxr(l))
         a(k,indxr(l))=a(k,indxc(l))
         a(k,indxc(l))=dum
23 continue
endif
24 continue
return
END
SUBROUTINE GSeidel(a,x,xo,b,n,size)  
Double Precision Tol max, diff1, diff2  
Integer NMAX,k,i,j,n,size  
Parameter(NMAX=40000,Tol=1.0e-06)  
Double Precision x(n),xo(n),a(n,n),b(n)  

k=1  
max=1.0  
do while(k.le.NMAX.AND.max.gt.Tol)  
do i=1,n  
diff1=0.0  
diff2=0.0  
do j=i-1  
   if(ABS(i-j).le.4*size-1) diff1=diff1-a(i,j)*x(j)  
   end do  
do j=i+1,n  
   if(ABS(i-j).le.4*size-1) diff2=diff2-a(i,j)*xo(j)  
   end do  
x(i)=(diff1+diff2+b(i))/a(i,i)  
end do  
max=ABS(x(1)-xo(1))  
do i=2,n  
   if(max.le.ABS(x(i)-xo(i))) max=ABS(x(i)-xo(i))  
end do  
k=k+1  
do i=1,n  
oxo(i)=x(i)  
end do  
return  
end
Subroutine-LUdecomp

Subroutine LUdecomp(a,n,size,m1,m2,mp,np,al,mpi,indx,d)
Integer m1,m2,mp,np,nindx(n),size
Double Precision d, a(np,mp), al(np,mpi), Tiny, dum
Parameter (Tiny=1.0e-20)
Integer i,j,k,l,mm
mm=8*size-1
if (mm.gt.mp.or.m1.gt.mp.or.n.gt.mp) pause 'bad args in LU'
l=m1
do i=1,m1
   do j=m1+2-i,mm
      a(i,j-1)=a(i,j)
   end do
   l=l-1
   do j=mm-l,mm
      a(i,j)=0.0
   end do
end do
d=1.
l=m1
do k=1,n
   dum=a(k,1)
i=k
   if (l.lt.n)=l+1
   do j=k+1,l
      if (ABS(a(j,1)).gt.ABS(dum))then
         dum=a(j,1)
i=j
      end if
   end do
   indx(k)=i
   if (dum.eq.0.) a(k,1)=Tiny
   if (i.ne.k)then
      d=-d
   do j=1,mm
      dum=a(k,j)
a(k,j)=a(i,j)
a(i,j)=dum
   end do
end if
   do i=k+1,l
      dum=a(i,1)/a(k,1)
al(k,i-k)=dum
   end do
end if
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
do j=2,mm
    a(i,j-1)=a(i,j)-dum*a(k,j)
end do
a(i,mm)=0.
end do
return
end
Subroutine-MatrixE

Subroutine MatrixE(t,n,size,check,qcheck,h,e)
Integer n,check,qcheck,size,count
Double Precision p,q,u,s,ba,t(0:n),h.
c e(0:(n+1)*size-1,0:(n+1)*size-1).b0,a0
External qromb,p,q,u,ut

b0=1.0
a0=0.0

do i=0:n
  do k=i,n
    qcheck=0
    do l=1,4
      if (i.eq.0) then
        s=0.0
        do j=0,1
          call qromb(t(j),t(j+l),ba,i,j,k,n,h.
          check,qcheck)
          s=s+ba
        end do
      else if (i.eq.1) then
        s=0.0
        do j=0.2
          call qromb(t(j),t(j+l),ba,i,j,k,n,h.
          check,qcheck)
          s=s+ba
        end do
      else if ((i.ge.2).AND.(i.le.n-2)) then
        s=0.0
        do j=i-2,i+l
          call qromb(t(j),t(j+l),ba,i,j,k,n,h.
          check,qcheck)
          s=s+ba
        end do
      else if (i.eq.n-1) then
        s=0.0
        do j=i-2,i
          call qromb(t(j),t(j+1),ba,i,j,k
          ,n,h,check,qcheck)
          s=s+ba
        end do
      else if (i.eq.n) then

s = 0.0

\begin{verbatim}
do j = i - 2, i - 1
    call qromb(t(j), t(j + 1), ba, i, j, k, n, h, check, qcheck)
    s += ba
end do
else
    print *, 'error in e'
end if

print *, s
count = 1
Do ii = i * size, (i + 1) * size - 1
    Do jj = k * size, (k + 1) * size - 1
        If (qcheck .eq. 0) then
            e(ii, jj) = s * p((b0 - a0) * t(i) + a0, count)
        else if (qcheck .eq. 1) then
            e(ii, jj) = e(ii, jj) + s * q((b0 - a0) * t(i) + a0, count) * (b0 - a0)**2
        else if (qcheck .eq. 2) then
            e(ii, jj) = e(ii, jj) - s * u((b0 - a0) * t(i) + a0, count) * (b0 - a0)
        else if (qcheck .eq. 3) then
            e(ii, jj) = e(ii, jj) - s * ut((b0 - a0) * t(i) + a0, count) * (b0 - a0)
        else
            print *, 'error in qcheck'
        end if
        count = count + 1
    end do
end do
qcheck = qcheck + 1
end do
qcheck = 0
end do
end
\end{verbatim}
Subroutine-Polint

Subroutine polint(xa,ya,n,x,y,dy)
Integer n,Nmax, i, m, ns
Double Precision x,y,xa(n),ya(n),dy
Parameter (nmax=10)
Double Precision den,dif,dift,ho,hp,w,c(nmax),d(nmax)
ns=1
dif=ABS(x-xa(1))
do i=1,n
dift=ABS(x-xa(i))
if (dift.lt.dif) then
   ns=i
   dif=dift
endif
c(i)=ya(i)
d(i)=ya(i)
enddo
y=ya(ns)
ns=ns-1
do m=1,n-1
do i=1,n-m
   ho=xa(i)-x
   hp=xa(i+m)-x
   w=c(i+1)-d(i)
   den=ho-hp
   if (den.eq.0) pause 'failure inpolint'
   den=w/den
   d(i)=hp*den
   c(i)=ho*den
end do
if (2*ns.lt.n-m) then
dy=c(ns+1)
else
dy=d(ns)
ns=ns-1
end if
y=y+dy
end do
return
end
Subroutine-Qromb

Subroutine qromb(a,b,ss,il,jh,kk,nn,hh,check,qcheck)
Integer jmax,jmaxp,K,KM,il,jh,nn,kk,check,qcheck
Double Precision a,b,func,ss,eps,hh
External func
Parameter (eps=1.e-6,jmax=20,jmaxp=jmax+1,K=5,Km=k-1)
Integer j
Double Precision dss,h(jmaxp),s(jmaxp)
h(1)=1.0
do j=1,Jmax
   call trapzd(func,a,b,s(j),j,il,jh,kk,nn,hh,check)
      if (j.ge.K) then
         call polint(h(j-km),s(j-km),k.0.,ss,dss)
         if (ABS(dss).le.0.5*ABS(ss)) return
      end if
   s(j+1)=s(j)
   h(j+1)=0.25*h(j)
end do
pause 'too many steps in qromb'
end
Subroutine-Transform

Subroutine Transform(a.e.ml.mm.n.size)
Integer n,m,ml,mm.size,k
Double Precision a((n+1)*size.mm).
ce(0:(n+1)*size-1:0:(n+1)*size-1)

Do i=1.(n+1)*size
   do j=1.mm
      a(i,j)=0.0
   end do
end do
Do i=0.(n+1)*size-1
   do j=0.(n+1)*size-1
      k=4*size+(j-i)
      If (k.ge.1.and.k.le.mm) a(i+1,k)=e(i,j)
   end do
end do
return
end
Subroutine-Trapzd

Subroutine trapzd(func,a,b,s,n,il,jh,kk,nn,hh,check.
c qcheck)
Integer n,il,jh,nn,kk,check,qcheck
Double Precision a,b,s,func.hh
External func
Integer it,j
Double Precision del,sum,tnm,x
if (n.eq.1) then
  s=0.5*(b-a)*(func(a,il,jh,kk,nn,nn,check,qcheck)
c +func(b,il,jh,kk,nn,nn,check,qcheck))
else
  it=2**(n-2)
  tnm=it
  del=(b-a)/tnm
  x=a+0.5*del
  sum=0.
  do j=1, it
    sum=sum+func(x,il,jh,kk,nn,nn,check,qcheck)
    x=x+del
  end do
  s=0.5*(s+(b-a)*sum/tnm)
end if
return
end [11]
Subroutine VectorB

Subroutine VectorB(t,n,size,check,qcheck,h,b)
Integer n,check,qcheck,size
Double Precision t(0:n),h,b(0:(n+1)*size-1),s,ba,r
External r

Do i=0,n
  if (i.eq.0) then
    s=0.0
    do j=0,1
      call qromb(t(j),t(j+l),ba,i,j,k,n,h,check, qcheck)
      s=s+ba
    end do
  else if (i.eq.1) then
    s=0.0
    do j=0,2
      call qromb(t(j),t(j+l),ba,i,j,k,n,h,check, qcheck)
      s=s+ba
    end do
  else if ((i.ge.2).AND.(i.le.n-2)) then
    s=0.0
    do j=i-2,i+l
      call qromb(t(j),t(j+l),ba,i,j,k,n,h,check, qcheck)
      s=s+ba
    end do
  else if (i.eq.n-1) then
    s=0.0
    do j=n-3,n-l
      call qromb(t(j),t(j+l),ba,i,j,k,n,h,check, qcheck)
      s=s+ba
    end do
  else if (i.eq.n) then
    s=0.0
    do j=n-2,n-l
      call qromb(t(j),t(j+l),ba,i,j,k,n,h,check, qcheck)
      s=s+ba
    end do
  else
    Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
print *.'error in b'
end if
do k=0.size-1
  b(i*size+k)=s*r(t(i).k.size)
end do
end do
check=1000
return
end
REFERENCES


VITA

Graduate College
University of Nevada, Las Vegas

Deborah A. Skidmore

Local Address:
8236 Carmen Blvd.
Las Vegas, NV 89128

Home Address:
8236 Carmen Blvd.
Las Vegas, NV 89128

Degrees:
Bachelor of Science, Mathematics, 1996
University of Nevada, Las Vegas

Thesis Title: Application of Cubic B-Splines to Boundary Value and Optimal Control Problems

Thesis Examination Committee:
Chairperson, Dr. Rohan J. Dalpatadu, Ph.D.
Committee Member, Dr. Ashok K. Singh, Ph.D.
Committee Member, Dr. Ching-Shyang Chen, Ph.D.
Graduate Faculty Representative, Dr. Laxmi P. Gewali, Ph.D.