SOLUTION STRATEGIES FOR SECOND ORDER, NONLINEAR, ONE DIMENSIONAL, TWO POINT BOUNDARY VALUE PROBLEMS
BY FEM ANALYSIS

by

Baird S. Ritter

December 1990

Thesis Advisor D. Salinas

Approved for public release; distribution is unlimited.
This research demonstrates the Galerkin FEM’s ability to provide approximate solutions of second order, nonlinear, one dimensional, two point boundary value problems. The research concentrates on the development of linearization, iteration, interpolation strategies for the solution of differential equations containing the nonlinear u^2 term. Additionally, various numerical considerations are explored. Over 2000 cases were analyzed using various strategies and results detailing the efficacy of strategy combinations are presented. A linearization strategy known as quasilinearization consistently yielded excellent approximate solutions of the nonlinear differential equations investigated. It converged in a minimum number of iterations was capable of solving equations which have large function order and activity over their specified domain.
Solution Strategies for Second Order, Nonlinear, One Dimensional, Two Point Boundary Value Problems by FEM Analysis

by

Baird S. Ritter
Lieutenant, United States Coast Guard
B.S., United States Coast Guard Academy, 1980

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1990

Author:

Baird S. Ritter

Approved by:

D. Salinas, Thesis Advisor

Anthony J. Healy, Chairman,
Department of Mechanical Engineering
ABSTRACT

This research demonstrates the Galerkin FEM's ability to provide approximate solutions of second order, nonlinear, one dimensional, two point boundary value problems. The research concentrates on the development of linearization, iteration, and interpolation strategies for the solution of differential equations containing the nonlinear \( u^2 \) term. Additionally, various numerical considerations are explored. Over 2000 cases were analyzed using various strategies and results detailing the efficacy of strategy combinations are presented. A linearization strategy known as quasilinearization consistently yielded excellent approximate solutions of the nonlinear differential equations investigated. It converged in a minimum number of iterations and was capable of solving equations which have large function order and activity over their specified domain.
# TABLE OF CONTENTS

I. INTRODUCTION ................................................................. 1  
   A. LINEAR DIFFERENTIAL EQUATIONS ..................................... 1   
   B. NONLINEAR DIFFERENTIAL EQUATIONS ................................ 2  

II. LINEAR APPLICATIONS OF THE GALERKIN FINITE ELEMENT  
   METHOD (FEM) ................................................................. 4  
   A. GENERAL FEM PROCEDURE ............................................... 4  
   B. RESULTS ........................................................................... 11  
   C. CONCLUSIONS ................................................................. 16  

III. ONE DIMENSIONAL SECOND ORDER NONLINEAR SYSTEMS ........... 18  
   A. LINEARIZATION STRATEGIES ............................................. 19  
      1. Constant Linearization ............................................... 19  
      2. Classical Linearization .............................................. 19  
      3. Quasilinearization .................................................... 20  
   B. INTERPOLATION STRATEGIES ............................................ 20  
      1. Linearization Vector ................................................... 21  
         a. Midpoint Lumped Approximation .................................. 21  
         b. 1/4 - 3/4 Lumped Approximation .................................. 22  
         c. Linear Approximation .............................................. 23  
      2. Linearization Matrix ................................................... 23  
         a. Midpoint Lumped Approximation .................................. 24  
         b. Linear Approximation .............................................. 24  
   C. ITERATION STRATEGIES .................................................. 24  
      1. Initial Iteration ....................................................... 25  
      2. Subsequent Iterations ................................................ 27  
         a. Previous Value Strategy .......................................... 27  
         b. Average Value Strategy .......................................... 28  
         c. Additional Strategies ............................................ 28  
   D. NUMERICAL CONSIDERATIONS ......................................... 28  
      1. Convergence/Divergence ............................................. 29
2. Critical Number of Degrees of Freedom (DOF) ................................................. 29
3. Stability ........................................................................................................ 29
4. Multiplicity of Solutions ............................................................................. 30
5. Boundary Condition Effects ...................................................................... 30
6. Computational Efficiency .......................................................................... 30

IV. APPLICATIONS .......................................................................................... 32
A. PRELIMINARIES ......................................................................................... 32
   1. Equations, Domains, and Boundary Conditions .................................... 32
   2. Related Engineering Phenomena .......................................................... 33
      a. Bar Problem ....................................................................................... 33
      b. Heat Fin Problem ............................................................................... 34
   3. Function Order and Domain Nondimensionalization .............................. 36
   4. General Solution Procedure .................................................................. 36
   5. Initial Iteration Strategy ........................................................................ 37
B. CONSTANT LINEARIZATION ..................................................................... 44
   1. Problem Formulation ............................................................................. 44
   2. Results .................................................................................................... 46
      a. General ............................................................................................... 46
      b. Boundary Conditions ........................................................................ 48
      c. Initial Iteration Strategy .................................................................... 48
      d. Subsequent Iteration Strategy ......................................................... 48
      e. Interpolation Strategy ....................................................................... 53
      f. Overall Performance .......................................................................... 53
   3. Conclusions ............................................................................................. 53
C. CLASSICAL LINEARIZATION .................................................................... 54
   1. Problem Formulation ............................................................................. 54
   2. Results .................................................................................................... 55
      a. General ............................................................................................... 55
      b. Boundary Conditions ........................................................................ 60
      c. Initial Iteration Strategy .................................................................... 61
      d. Subsequent Iteration Strategy ......................................................... 61
      e. Interpolation Strategy ....................................................................... 61
      f. Overall Performance .......................................................................... 62
   3. Conclusions ............................................................................................. 62
D. QUASILINEARIZATION ........................................ 62
  1. Problem Formulation ........................................ 62
  2. Results ................................................... 63
     a. General ........................................... 63
     b. Boundary Conditions .................................. 76
     c. Initial Iteration Strategy ............................ 76
     d. Subsequent Iteration Strategy ....................... 78
     e. Interpolation Strategy ................................ 78
     f. Overall Performance .................................. 79
  3. Conclusions ............................................. 79
E. FINAL REMARKS ........................................... 79

APPENDIX A. FORCING FUNCTION FORMULATION STRATEGIES ... 81

APPENDIX B. PROGRAM LISTINGS AND RESULTS FOR THE LINEAR
APPLICATION OF THE GALERKIN FEM ............................. 83

APPENDIX C. LINEARIZATION VECTORS FOR CONSTANT
LINEARIZATION TECHNIQUE ........................................ 103

APPENDIX D. LINEARIZATION MATRICES FOR THE CLASSICAL
LINEARIZATION TECHNIQUE ......................................... 105

APPENDIX E. PROGRAM LISTINGS FOR CONSTANT LINEARIZATION 106

APPENDIX F. PROGRAM LISTINGS FOR CLASSICAL LINEARIZATION 125

APPENDIX G. PROGRAM LISTINGS FOR QUASILINEARIZATION .... 144

LIST OF REFERENCES ............................................. 164

INITIAL DISTRIBUTION LIST .................................... 165
LIST OF TABLES

Table 1. FORCING FUNCTIONS FOR VARIOUS F(X) ....................... 12
Table 2. FUNCTION ORDER OF EQUATIONS (4.1) AND (4.2) ............... 36
Table 3. CONSTANT LINEARIZATION ELEMENT VECTORS .................... 45
Table 4. ELEMENT FORCE VECTORS FOR EQUATIONS (4.1) AND (4.2) .......... 46
Table 5. NUMBER OF ELEMENTS UPON WHICH SOLUTION PROCEDURE RESULTS ARE BASED ..................... 47
Table 6. SOLUTION PROCEDURES AND RESULTS USING CONSTANT LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN ONE .................................................. 49
Table 7. SOLUTION PROCEDURES AND RESULTS USING CONSTANT LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAINS TWO AND THREE ................................. 50
Table 8. SOLUTION PROCEDURES AND RESULTS USING CONSTANT LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN ONE .................................................. 51
Table 9. SOLUTION PROCEDURES AND RESULTS USING CONSTANT LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAINS TWO AND THREE ................................. 52
Table 10. CLASSICAL LINEARIZATION ELEMENT MATRICES ................ 55
Table 11. SOLUTION PROCEDURES AND RESULTS USING CLASSICAL LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN ONE .................................................. 56
Table 12. SOLUTION PROCEDURES AND RESULTS USING CLASSICAL LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAINS TWO AND THREE ................................. 57
Table 13. SOLUTION PROCEDURES AND RESULTS USING CLASSICAL LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN ONE .................................................. 58
Table 14. SOLUTION PROCEDURES AND RESULTS USING CLASSICAL LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAINS TWO AND THREE ................................. 59
Table 15. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN ONE ................................................................. 65
Table 16. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN ONE (CONT.) .................................................. 66
Table 17. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN TWO ............................................................... 67
Table 18. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN THREE ............................................................. 68
Table 19. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN THREE (CONT.) .................................................. 69
Table 20. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN ONE ................................................................. 70
Table 21. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN ONE (CONT.) .................................................. 71
Table 22. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN TWO ............................................................... 72
Table 23. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN TWO (CONT.) .................................................. 73
Table 24. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN THREE ............................................................. 74
Table 25. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN THREE (CONT.) .................................................. 75
LIST OF FIGURES

Figure 1. One Dimensional Linear Shape Functions ....................... 7
Figure 2. Illustration of Midpoint Lumped Approximation ............... 9
Figure 3. Illustration of 1/4 - 3/4 Lumped Approximation ............. 10
Figure 4. Illustration of Linear Approximation .......................... 10
Figure 5. Global to Local Coordinate Transformation .................... 11
Figure 6. Comparison of Solutions for Equation 2.8 Using 10 Elements. 13
Figure 7. Comparison of Solutions for Equation 2.9 Using 10 Elements. 14
Figure 8. Comparison of Solutions for Equation 2.10 Using 10 Elements. 15
Figure 9. Solution of Equation 2.10 Over a Large Domain Using Two Elements . 17
Figure 10. Axially Loaded Bar Embedded in a Nonlinearly Elastic Material 34
Figure 11. Heat Fin With Nonlinear Convection .......................... 35
Figure 12. Solution Procedure for Nonlinear Differential Equations .... 38
Figure 13. Dominance of Terms in Equation (4.1) over Domain Two .... 40
Figure 14. Dominance of Terms in Equation (4.1) over Domain Three ... 41
Figure 15. Dominance of Terms in Equation (4.2) over Domain Two .... 42
Figure 16. Dominance of Terms in Equation (4.2) over Domain Three ... 43
Figure 17. Dominance of Terms in Equation (4.2) Over Domain One .... 77
LIST OF SYMBOLS

a - FEM 2x2 Elemental Differential Operator Matrix
A - FEM NxN System Differential Operator Matrix
B - FEM Nx1 Natural Boundary Condition Vector
CPU - Actual Time Required for Iteration Process
CPU' - Measure of accuracy and computational effort
D - Length of Domain
f - Internal System Excitation
f - FEM 2x1 Element Excitation Vector
F - FEM Nx1 System Excitation Vector
g - Element 2x1 Linear Shape Function Vector
G - Global Nx1 Linear Shape Function Vector
h - Arbitrary function utilized in the linearized Galerkin integral
I - FEM 2x2 Element Linearization Matrix
L - FEM NxN System Linearization Matrix
m - Arbitrary function utilized in classical linearization
N - Number of system degrees of freedom
q - Arbitrary function utilized in quasilinearization
R - Residual
u - Dependent Variable
u - FEM Solution Nx1 Vector
v - Arbitrary function utilized in constant linearization
w - Weighting factor
x - Independent Variable
x_i - Distance from origin of x to origin of element i
η - Nondimensionalized coordinate system
ζ - Element coordinate system
\mathcal{L} - Linear Differential Operator
\mathcal{L}(\cdot) - Nonlinear Differential Operator
Subscripts
i - Iteration number
j - Global nodal point number
m - Modified

Superscripts
* - Variable which changes during the iteration process
~ - Derived from FEM approximation; approximate value
T - Vector transpose
I. INTRODUCTION

Differential equations which describe natural phenomena are developed through the application of the conservation principles of mass, momentum, and/or energy to a continuous media. After application of these principles to a small or differential element of the media, the resulting equation provides a relationship between one or more derivatives of an unknown function whose behavior is desired. The differential equation is linear when it contains only derivatives of the function with either constant or independent variable function coefficients. The solutions of linear differential equations are unique based on their boundary and/or initial conditions. Various analytical and numerical strategies have been developed to obtain these solutions. The finite element method (FEM), which is used in this research, is one of the more popular and accurate techniques that has been developed to solve differential equations.

When the coefficients of the derivatives are functions of the desired unknown, (that is, the dependent variable), or when the dependent variable and/or its derivatives do not appear linearly in the differential equation, then the differential equation becomes nonlinear and conventional analytical techniques usually do not work. Nonlinear operators occur in a number of differential equations that describe the behavior of natural phenomenon, e.g., the Navier-Stokes equations for fluid flow, a beam on an inelastic foundation, the Falkner Skan equation, etc. This research investigates various strategies for solving nonlinear, second order, one dimensional, two point boundary value problems using FEM analysis. The Galerkin method of weighted residuals (MWR) with discrete basis functions is the FEM technique used to compute approximate solutions of these equations.

A. LINEAR DIFFERENTIAL EQUATIONS

Under most circumstances, this particular FEM provides excellent approximations to linear differential equations of the form

\[ \mathcal{L} u - f = 0, \quad 0 < x < D \]  

with appropriate essential and natural boundary conditions where

- \( \mathcal{L} \) is the sum of arbitrary linear operators such as \( \frac{d}{dx} () \), \( \frac{d^2}{dx^2} () \), etc.
- \( x \) is the one-dimensional independent variable
• $u$ is the dependent variable of $x$
• $f$ is an internal excitation to the system
• $D$ is the domain of the differential equation

If the dominant operator in equation (1.1) is of odd order (non self-adjoint form), then the Galerkin method doesn't work and the Petrov-Galerkin FEM must be utilized. The Galerkin FEM transforms the differential equation into a system of linear algebraic equations of the form

$$Au - F = 0 \quad (1.2.a)$$

where

• $A$ is an $N \times N$ coefficient matrix characterizing the operator(s) $\mathcal{L}$, and $N$ is the number of system degrees of freedom (DOF) in the approximation.
• $u$ is the $N \times 1$ FEM approximate solution vector
• $F$ is an $N \times 1$ FEM vector representation of the excitation function $f$

This system of equations is readily solved by matrix algebra for the response variable,

$$u = A^{-1}F \quad (1.2.b)$$

where equation (1.2.b) implies the solution of equation (1.2.a) and does not mean that $A^{-1}$ is actually obtained.

Chapter II of the research demonstrates the Galerkin Method's ability to accurately model linear differential equations, regardless of the magnitude of the solution, i.e., $u = x^2, x^3$, etc. Additionally, various methods of modeling the excitation function, $f$, are examined.

**B. NONLINEAR DIFFERENTIAL EQUATIONS**

The remaining research examines various nonlinear, second order, differential equations of the form

$$\mathcal{L}u + \mathcal{L}(u) - f = 0 \quad 0 < x < D \quad (1.3)$$

with appropriate boundary conditions where all terms are as previously defined and $\mathcal{L}(u)$ is the sum of arbitrary nonlinear terms such as $u \frac{du}{dx}$, $\sin(u)$, $u^2$, etc. Though this research concentrates on the $u^2$ nonlinear term, the principles presented should allow for the adequate analysis of any nonlinear term that might be encountered.
The first step in the solution of equation (1.3) is linearization of the nonlinear term. Once the equation is linearized with respect to the dependent variable, the Galerkin FEM is used in an iterative fashion to solve a linear system of algebraic equations in the form

\[ Au + L' u - F = 0 \]  

(1.4)

where the coefficients of \( L' \) are functions of \( u \) evaluated at the previous iteration, denoted \( u_{i-1} \), and where subscripts \( i \) and \( i-1 \) refer to the present and past iterations. To begin the iteration process, a value is initially assumed for \( u_{i-1} \) and the system of equations is solved for \( u_i \). The new values of \( u_i \) and the input values of \( u_{i-1} \) at each node are compared and tested against a convergence criteria. If convergence is not obtained, the new value of \( u_i \) is substituted back into the system of equations for \( u_{i-1} \) and the iteration process continues until convergence is obtained. The final iteration yields the FEM approximate solution of the nonlinear differential equation.

Chapter III provides the general principles and considerations which are involved in solving nonlinear differential equations. Chapter IV utilizes these principles in the solution of two different equations containing the nonlinear term \( u^2 \). Final conclusions are made based on the solution results as to which problem solving strategy yields the most accurate approximation while using the least amount of computer time.
II. LINEAR APPLICATIONS OF THE GALERKIN FINITE ELEMENT METHOD (FEM)

This section examines the Galerkin method of weighted residuals (MWR) using discrete linear shape functions, that is the Galerkin FEM, as applied to linear second order, one dimensional, differential equations, that is, two point boundary value problems. In section A, a general FEM procedure for the differential equation

$$u'' - f(x) = 0 \quad a < x < b$$  \hspace{1cm} (2.1)

with two boundary conditions, one at each end point of the domain, is presented. In particular, several strategies for the FEM representation of the excitation function $f(x)$ are developed.

In section B, the various strategies for FEM representation of the excitation function are implemented on equation (2.1) for three different excitation functions. The three $f(x)$ functions were selected to provide the solutions of $x_1$, $x_3$, $x^*$ to equation (2.1).

A. GENERAL FEM PROCEDURE

Here, a general procedure for the FEM formulation of equation (2.1) is presented. The Galerkin FEM process transforms a differential equation into a system of linear algebraic whose solution is an approximation to the exact solution of the differential equation. The transformation requires the following three steps:

- **Step 1:** Form an $N$ degree of freedom approximation, say $\tilde{u} = G^T u$, where:
  - $\tilde{u}$ is the approximate solution
  - $G^T$ is the $1xN$ transpose vector of linear shape functions
  - $u$ is the $Nx1$ vector of the FEM solution, $u$, at each node

- **Step 2:** Form the Residual $R = \tilde{u}'' - f(x)$, where:
  - $f(x)$ is the excitation function in the differential equation

- **Step 3** Form the Galerkin integral equations $\int_a^b GR dx = 0$

The evaluation of these integral equations gives a system of $N$ linear algebraic equations whose solution is the approximation, $\tilde{u}$. The details of the FEM formulation for equation (2.1) follows. Substitution of the residual $R$ into the Galerkin integral equation and separating terms yields
\[ \int_a^b G\dddot{u} \, dx - \int_a^b Gf(x) \, dx = 0 \]  \hspace{1cm} (2.2.a)

From step 1, \( \dddot{u} = (G^T u)' = (G^T)' u \). Substitution of this into the first term of equation (2.2.a) and moving the second term to the right hand side yields

\[ \int_a^b G(G^T)' u \, dx = \int_a^b Gf(x) \, dx \]  \hspace{1cm} (2.2.b)

Note that the term on the left hand side is the FEM representation of \( \mathcal{L} u = u'' \) and the term on the right hand side is the FEM representation of \( f(x) \). The result of an integration by parts of the left hand side leaves

\[ G(G^T)' u \bigg|_a^b - \int_a^b G'(G^T) u \, dx = \int_a^b Gf(x) \, dx \]  \hspace{1cm} (2.2.c)

Each term is now evaluated individually.

**Boundary Term, \( G(G^T)' u \bigg|_a^b \)**

Differentiating the equation in step 1 of the formulation process yields \( (G^T)' u = \dddot{u} \). Substitution of \( \dddot{u} \) into the above equation and evaluating it at the upper and lower limits gives

\[ G\dddot{u} \bigg|_b - G\dddot{u} \bigg|_a = \begin{bmatrix} G_1(b)\dddot{u}(b) \\ G_2(b)\dddot{u}(b) \\ \vdots \\ G_N(b)\dddot{u}(b) \end{bmatrix} - \begin{bmatrix} G_1(a)\dddot{u}(a) \\ G_2(a)\dddot{u}(a) \\ \vdots \\ G_N(a)\dddot{u}(a) \end{bmatrix} \]  \hspace{1cm} (2.3.a)

In equation (2.3.a), \( G \) denotes that it is the linear shape function associated with the \( i \)th system degree of freedom (SDOF) at the \( i \)th system nodal point. The ' \( a \)' and ' \( b \)' arguments of \( G \) and \( u' \) are the endpoint values of the domain where these functions are evaluated. These endpoints could be denoted by their system node identities, which are 1 and \( N \), at the left and right end points respectively. With this notation, equation (2.3.a) becomes:
Due to the Kroenecker delta property of the linear shape functions, all terms are equal to zero except \( G_n(N) \ddot{u}'(N) \) and \( G_i(1) \ddot{u}'(1) \). This yields a single vector comprised of the natural boundary conditions at each end of the domain and is designated \( \mathbf{B} \) where

\[
\mathbf{G}\ddot{u}'|_b - \mathbf{G}\ddot{u}'|_a = \mathbf{B} = \begin{bmatrix}
- \ddot{u}'(a) \\
0 \\
\vdots \\
0 \\
\ddot{u}'(b)
\end{bmatrix}
\]

(2.3.c)

When natural boundary conditions, \( u' \), are present, they are used for the corresponding \( \ddot{u}' \) in equation (2.3.c).

**Differential Operator, \( \int_a^b G'(G^T)' dx \) u**

The integral \( \int_a^b G'(G^T)' dx \), associated with the \( u'' \) operator, is reduced to the element coordinate level for evaluation and becomes the 2x2 element matrix

\[
\mathbf{a} = \int_0^l g'(g^T)' d\xi
\]

(2.4.a)

where \( \xi \) is the local coordinate axis, \( l \) is the length of the element, and \( g \) is the 2x1 vector of linear shape functions, given by

\[
G_{ij} = \begin{cases} 
1 & i = j \\
0 & i \neq j
\end{cases}
\]
as shown in Figure 1. Differentiation of the $g$ functions gives the $2 \times 1$ $g'$ vector,

$$g' = \begin{bmatrix} - \frac{1}{l_e} \\ \frac{1}{l_e} \end{bmatrix} \quad (2.4.c)$$

Substituting $g'$ and $(g^T)$ into the integral of equation (2.4.a) gives

$$a = \int_0^{l_e} \left[ - \frac{1}{l_e} + \frac{1}{l_e} \right] \left[ - \frac{1}{l_e} + \frac{1}{l_e} \right] d\xi = \int_0^{l_e} \left[ - \frac{1}{l_e^2} + \frac{1}{l_e^2} \right] d\xi \quad (2.4.d)$$

$$= \frac{1}{l_e} \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}$$

Figure 1. One Dimensional Linear Shape Functions

After construction in an element DO loop, the $2 \times 2$ $a$ matrix for each element is then distributed into the $N \times N$ system $A$ matrix in accordance with a correspondence table,
which relates local DOF to system DOF, and where \( N \) is the number of system degrees of freedom. Each element has two local degrees of freedom, LDOF 1 at the left end of an element, and LDOF 2 at the right end of the element. The correspondence between LDOF 1 and LDOF 2 of the \( j \)th element and the \( j \)th system degree of freedom (SDOF \( J \)) is \( J = (i - 1) + k \) where \( k \) is 1 or 2. Upon assembly of all the element matrices, we obtain the system \( Au \) term.

**Excitation**, \( \int_a^b G f(x) \, dx \)

This term determines the forcing function (or excitation) vector \( F \), that is

\[
F = \int_a^b G f(x) \, dx
\]

(2.5)

and is obtained by assembling the 2x1 element excitation vectors \( f \). The \( f \) vectors can be either modeled as a lumped approximation term in several different ways or integrated exactly to yield a consistent forcing function. In this study, two lumped approximations and the exact integration are developed and thereafter compared to determine which yields the more accurate solution. A third approximation method is also described. Although this third method is not used in the evaluation of the excitation function in this chapter on linear systems, it is used in the next chapter on the nonlinear systems portion of the research. It should be noted that as the number of elements approaches infinity and the element length approaches zero, each approximation technique described below yields the exact value of the excitation integral.

- **Midpoint Lumped Approximation of \( f(x) \)**
  The midpoint lumped approximation method for evaluation of the excitation vector is the simplest and crudest approximation. This approximation involves evaluating the function \( f(x) \) at the midpoint of the element and multiplying this value by the element length. Half of this area \( (f(l/2)l/2) \) is then placed at the left element node and the other half at the right element node as illustrated in Figure 2 on page 9 for three different arbitrary \( f(x) \). For the monotonically increasing function in Figure 2.a, this method places too much area at the left local nodal point (LNP) and not enough at the right. Conversely, when \( f(x) \) is monotonically decreasing, too little area is placed at the left node while the right node gets too much. When \( f(x) \) is concave over the element, too little area is placed at each node (Figure 2.b), while for the convex case (Figure 2.c), each LNP receives too much area.

- **1/4 - 3/4 Lumped Approximation of \( f(x) \)**
  This technique, which is a refinement of the previous one, evaluates \( f(x) \) at the quarter point and three quarter point of an element. Each value is multiplied by half of the element length and the resulting areas are placed at the left and right
element nodal points respectively as shown in Figure 3 on page 10 for the same three arbitrary functions. This method provides a better approximation than the lumped midpoint technique, especially for the monotonically increasing function in Figure 3.a, as it uses two points to capture the behavior of the curve instead of one.

\[ f(x) \approx \tilde{f}(x) = f(x_1) \left( 1 - \frac{\xi}{l_e} \right) + f(x_2) \left( \frac{\xi}{l_e} \right) \]  \hspace{1cm} (2.6)

as shown in Figure 4 on page 10. It overestimates the area for concave curves (Figure 4.a) and underestimates for convex curves as shown in Figure 4.b. Note that \( f(x_1) \) and \( f(x_2) \) can be generalized to coefficients \( f_1 \) and \( f_2 \) to give a better linear fit of the curve. Also, the linear shape functions can be replaced by higher order shape functions which provides an \( n \)th order approximation of \( f(x) \) as opposed to a linear one. This approximation method is not utilized for the evaluation of forcing functions, but is used later in the nonlinear portion of the research to approximate other types of functions.

- Consistent
  The consistent solution requires transforming \( f(x) \) into the element coordinate system, \( f(\xi) \), by substituting \( x = \alpha + \xi \) into \( f(x) \) and performing the integration over the lengths of the elements. The coordinate transformation is illustrated in Figure 5 on page 11.
Figure 3. Illustration of 1/4 - 3/4 Lumped Approximation

Figure 4. Illustration of Linear Approximation
The integration for each of the above methods is performed on the element level, producing the \(2 \times 1\) \(\mathbf{f}\) vector. The \(\mathbf{f}\) vector for each element is then distributed into the \(N \times 1\) system force vector \(\mathbf{F}\) in accordance with the local to global nodal point correspondence.

Substitution of \(\mathbf{A}, \mathbf{F},\) and \(\mathbf{B}\) into equation (2.2.c) yields a linear system of algebraic equations in the form

\[
- \mathbf{A} \mathbf{u} = \mathbf{F} - \mathbf{B}
\]

(2.7.a)

The \((\mathbf{F} - \mathbf{B})\) term can be combined into a single vector designated \(\mathbf{F}_m\) leaving

\[
- \mathbf{A} \mathbf{u} = \mathbf{F}_m
\]

(2.7.b)

which can be solved for \(\mathbf{u}\), the FEM approximate solution at each nodal point.

**B. RESULTS**

In order to obtain specific results, the following equations are analyzed over the domain \(0 < x < 2\):

\[
\begin{align*}
u'' &= 2; \quad u(0) = 0, \quad u'(2) = 4 & u_{\text{exact}} &= x^2 \\
u'' &= 6x; \quad u(0) = 0, \quad u'(2) = 12 & u_{\text{exact}} &= x^3 \\
u'' &= 12x^2; \quad u(0) = 0, \quad u'(2) = 32 & u_{\text{exact}} &= x^4
\end{align*}
\]
The excitation function in each of the above equations is evaluated using the three methods previously described in II.A. The detailed formulation of these different f vectors is shown in Appendix A for the forcing function of equation (2.9) where \( f(x) = 6x \). The other two \( f(x) \) are evaluated in a similar manner and the results for all three \( f(x) \) functions are shown in Table 1.

Table 1. FORCING FUNCTIONS FOR VARIOUS F(X)

<table>
<thead>
<tr>
<th>( f(x) )</th>
<th>Midpoint Approximation</th>
<th>1/4 - 3/4 Approximation</th>
<th>Consistent</th>
</tr>
</thead>
</table>
| 2          | \[
\begin{bmatrix}
L_x \\
L_4
\end{bmatrix}
\] | \[
\begin{bmatrix}
L_x \\
L_4
\end{bmatrix}
\] | \[
\begin{bmatrix}
L_x \\
L_4
\end{bmatrix}
\] |
| 6x         | \[
3L_x
\begin{bmatrix}
\alpha + \frac{L}{2} \\
\alpha + \frac{L}{2}
\end{bmatrix}
\] | \[
3L_x
\begin{bmatrix}
\alpha + \frac{L}{4} \\
\alpha + \frac{3L}{4}
\end{bmatrix}
\] | \[
\begin{bmatrix}
3\alpha L_x + L^2 \\
3\alpha L_x + 2L^2
\end{bmatrix}
\] |
| 12x\(^2\)  | \[
6L_x
\begin{bmatrix}
(\alpha + \frac{L}{2})^2 \\
(\alpha + \frac{L}{2})^2
\end{bmatrix}
\] | \[
6L_x
\begin{bmatrix}
(\alpha + \frac{L}{4})^2 \\
(\alpha + \frac{3L}{4})^2
\end{bmatrix}
\] | \[
\begin{bmatrix}
6\alpha^2 L_x + 4\alpha L^2 + L^4 \\
6\alpha^2 L_x + 8\alpha L^2 + 3L^4
\end{bmatrix}
\] |

The FORTRAN programs and results for a ten element analysis of each equation with the various forcing functions are provided in Appendix B. The first problem considered is that presented by equation (2.8). Due to the nature of the forcing function in this equation, i.e., a constant, all three formulation options provide the same result. The FEM approximation, shown in Figure 6, provided the exact solution at each nodal point for a 10 element analysis.

The solutions of equations (2.9) and (2.10) were then considered. The different forcing function formulations in Table 1 were used in solving these differential equations. For clarity purposes, only a portion of the plots comparing each solution process to the exact solution in an area of rapidly changing gradient are shown in Figure 7 on page 14 (for \( u'' = 6x \)) and Figure 8 on page 15 (for \( u'' = 12x^3 \)).

The midpoint approximation method for \( F \) provides a solution which is larger than the exact at each nodal point for both equations (2.9) and (2.10). The approximation is worse for equation (2.10) in Figure 8 as the midpoint method provides an overesti-
mation of the actual value of the excitation integral at each node due to its inability to account for the quadratic nature of the forcing function.

The quarter/three-quarter point approximation of \( F \) provides solutions for equations (2.9) and (2.10) that are quite close to the exact solution at each nodal point. This technique provides a much better approximation of the area under the forcing function curve because it discretizes the area into two independent sections, where as the mid-point technique did not. Thus, this technique is quite accurate in approximating excitation functions such as \( x^2 \) which are monotonic and quadratic in nature.

The consistent formulation method provides the exact solution for both equations (2.9) and (2.10) at each nodal point, even when the solution curve has a rapidly changing gradient such as \( u = x^2 \) as shown in Figure 8. It was expected that this technique would provide the most accurate solution for all cases as it yields the exact area given by the Galerkin excitation integral.

Figure 6. Comparison of Solutions for Equation 2.8 Using 10 Elements.
Figure 7. Comparison of Solutions for Equation 2.9 Using 10 Elements.
Figure 8. Comparison of Solutions for Equation 2.10 Using 10 Elements.
C. CONCLUSIONS

Linear shape functions provide an efficient and accurate interpolation for approximating second order, linear, one dimensional, differential equations using the Galerkin FEM, regardless of the magnitude of the solution. Therefore, they are utilized throughout the remainder of the research when approximating linear and nonlinear operators of second order or less. An open question which remains is whether higher order elements will work for nonlinear two point boundary value problems when linear elements do not.

Additionally, two important observations regarding the use of a consistent forcing function analysis are made.

- The use of this technique in evaluating the excitation integral provides for a very accurate solution process. Thus, this method is employed in the remainder of the research whenever possible. In those cases where the integration cannot be performed analytically, a Simpson's Rule approximation to the integral is used so that the resulting error is kept to a minimum.

- For linear problems, this method provides very accurate approximations over large domains using a minimum number of elements. To illustrate this point, equation (2.10) was solved over the domain $0 < x < 10$ using only two elements. The FEM approximation provided the "exact" solution at $x = 5$ and $x = 10$ as shown in Figure 9 on page 17.
Figure 9. Solution of Equation 2.10 Over a Large Domain Using Two Elements
III. ONE DIMENSIONAL SECOND ORDER NONLINEAR SYSTEMS

This chapter outlines various solution strategies which are later analyzed on their ability to solve second order nonlinear differential equations of the form

\[ Lu + \mathcal{L}(u) - f(x) = 0 \quad x \in D \quad (3.1.a) \]

where \( \mathcal{L} \) denotes linear operators and \( \mathcal{L}(\cdot) \) denotes nonlinear operators. Because equation (3.1.a) is nonlinear, a closed form analytical solution is, in general, not possible. Therefore numerical solutions are obtained by a variety of approximation techniques. This chapter sets out a general procedure, consisting of three steps, for obtaining numerical solutions of equation (3.1.a). The three steps, when used with the Galerkin FEM, have the effect of transforming the original nonlinear differential equation into a system of linear algebraic equations.

The first step in the procedure consists of a ‘linearization’ of the nonlinear \( \mathcal{L}(u) \) term(s). Thus, \( \mathcal{L}(u) \) is transformed to \( \mathcal{L}'u \) where \( \mathcal{L}' \) can be obtained by a number of different strategies, three of which are described in section III.A.

Once the nonlinear differential equation, equation (3.1.a), has been transformed to a linear differential equation of the form

\[ Lu + \mathcal{L}'u = f(x) \quad (3.1.b) \]

the second step is associated with how the \( \mathcal{L}'u \) term in equation (3.1.b) is evaluated in the FEM Galerkin integral equations. A number of interpolation procedures are described for this step in Section III.B.

The third step in the solution procedure is defining the iterative process by which a solution of the linear algebraic equations developed by the Galerkin FEM is obtained. In particular, the efficacy of the iterative method involves two considerations.

- the selection of an initial estimate to begin the iteration
- the methodology for subsequent iterations

Section III.C describes a number of iteration strategies.

Thus sections III.A, III.B, and III.C cover the three basic steps in the solution procedure; that is, linearization, interpolation, and iteration. The selection of a particular strategy within each of these steps defines a specific solution procedure which can
be utilized in approximating the solution of equation (3.1.a). In addition to these sections, section III.D discusses several numerical aspects which either affect, or are used in evaluating, the efficacy of a solution procedure.

A. LINEARIZATION STRATEGIES

The first step in analyzing nonlinear equations using the Galerkin FEM is linearization of the nonlinear term(s). Three different linearization strategies are investigated in this research. There is no implication that these are the only strategies that exist.

1. Constant Linearization

The constant linearization method transforms the nonlinear term into a function of $u'$ where

$$\mathcal{L}(u) \approx \mathcal{L}^* u = v(u^*) \quad (3.2.a)$$

As discussed in Chapter 1, the solution process for nonlinear equations using the Galerkin FEM is iterative in nature. On the first iteration, $u'$ is set equal to an assumed value at each node and for subsequent iterations, each nodal value of $u'$ is based on the FEM approximation. Thus, $v(u^*)$ is a known function evaluated at each node and is taken to the right hand side, leaving a linear equation of the form

$$\mathcal{L} u = f(x) - v(u^*) \quad (3.2.b)$$

As an example, consider the nonlinear term $u'u$. The constant linearization technique linearizes this term as $v(u^*) = (u')'u'$, where $(u')'$ is evaluated using finite difference techniques.

2. Classical Linearization

The classical linearization strategy linearizes the nonlinear term as a known function coefficient multiplying the dependent variable where

$$\mathcal{L}(u) \approx \mathcal{L}^* u = (m(u^*)) u \quad (3.3.a)$$

which in a sense maintains the functional nature of the dependent variable by allowing it to be kept on the left hand side of the differential equation in a linear fashion. As in the constant linearization strategy, $m(u^*)$ is a known function coefficient where the values of $u'$ are assumed for the first iteration and are based on the FEM approximation for
subsequent iterations. Substituting the results of equation (3.3.a) into equation (3.1.a), the linearized differential equation takes the form

$$\mathcal{L}u + (m(u^*))u = f(x)$$  \hspace{1cm} (3.3.b)

Using the nonlinear term given as an example in the previous section, that is \(u'u\), the classical linearization technique provides two alternatives. The first is \((m(u')) u = (u')'u\) which keeps the full effect of the dependent variable, \(u\). The second is \((m(u')) u = u'u'\) which maintains the full effect of the derivative term, \(u'\).

3. Quasilinearization

Quasilinearization is covered extensively by Bellman and Kalaba in [Ref. 1: p. 36] where the nonlinear term is set equal to \(q(u)\) and linearized as

$$\mathcal{L}(u) \approx \mathcal{L}^* u = q(u^*) + (u - u^*) q'(u^*)$$

$$= \left( q'(u^*) \right) u + \left( q(u^*) - u^* q'(u^*) \right)$$  \hspace{1cm} (3.4.a)

Comparing equation (3.4.a) with equations (3.2.a) and (3.3.a), it can be seen that quasilinearization is a combination of constant and classical linearization with the coefficient functions determined in a different manner. Substituting the results of equation (3.4.a) into equation (3.1.a), the linearized differential equation becomes

$$\mathcal{L}u + \left( q'(u^*) \right) u = f(x) - \left( q(u^*) - u^* q'(u^*) \right)$$  \hspace{1cm} (3.4.b)

As an example, consider the \(u^2\) nonlinear term. The quasilinearization technique defines \(q(u) = u^2\), from which \(q(u') = (u')^2\) and \(q'(u') = 2u'\). Substitution of these functions into equation (3.4.a) yields a linearization of the form \(\mathcal{L}'u = 2u'u - (u')^2\). As \(u'\) and \(u\) begin to approach the same value at each node during the iteration process, \(\mathcal{L}'u\) begins to approach the original nonlinear term, namely \(u^2\).

B. INTERPOLATION STRATEGIES

The second step of the solution process is the evaluation of the Galerkin integrals. Based on the type of linearization strategy utilized, one or both of the following integrals are obtained.

Linearization Vector: \(\int_D G h(u^*) dx\)  \hspace{1cm} (3.5.a)
Linearization Matrix: \[ \int_D G G^T h(u^*) \, dx \, u \] (3.5.b)

The constant linearization strategy results in a linearization vector where \( h(u') \) in equation (3.5.a) is replaced by \( v(u') \) from equation (3.2.a). The classical linearization technique yields a linearization matrix where \( h(u') \) in equation (3.5.b) is replaced by \( m(u') \) from equation (3.3.a). The quasilinearization method utilizes both integrals where \( h(u') \) becomes \( q(u') - u'q'(u') \) in equation (3.5.a) and \( q'(u') \) in equation (3.5.b). Since \( u' \) is derived from an FEM approximation utilizing linear shape functions, it varies linearly between nodes over each element. Thus, when equations (3.5.a) and (3.5.b) are reduced to the element level for evaluation, there are numerous interpolation strategies available to approximate \( h(u') \). Here, a few strategies are discussed for each of the above integrals.

1. **Linearization Vector**

   Reducing the integral in equation (3.5.a) to the element level yields

   \[ \int_0^1 g h(u^*) \, d\xi \] (3.6)

   which is quite similar to the forcing function integral discussed in Chapter II. Three interpolation techniques similar to those used for the forcing function are examined in this research. The integral in equation (3.6) yields a 2x1 element vector denoted as \( f' \); the \(^*\) superscript meaning that the vector changes with each iteration as the values of \( u' \) are updated. The \( f' \) are then distributed into a system linearization vector denoted as \( F' \) in accordance with the local to global nodal point correspondence.

   a. **Midpoint Lumped Approximation**

   This method evaluates \( h(u') \) at the midpoint of the element and brings this value outside the integral as a constant. Since \( u' \) varies linearly over the element, its value at the midpoint of the element is simply the average of the values of \( u' \) at the two nodes of the element, \( (u'_i) \) and \( (u'_{i+1}) \). Substituting this into equation (3.6) leaves
\[ f^* = h \left( \frac{(u^*_j)_i + (u^*_{j+1})_i}{2} \right) \int_0^{l_e} \left[ 1 - \frac{\xi}{l_e} \right] d\xi \]  

(3.7)

where \( h() \) denotes that the function \( h \) is evaluated at the argument of \(( \ )\).

b. \( 1/4 - 3/4 \) Lumped Approximation

This method takes the \( h(u^*) \) term inside the shape function vector yielding

\[ f^* = \int_0^{l_e} \left[ h(u^*) \left( 1 - \frac{\xi}{l_e} \right) \right] d\xi \]  

(3.8.a)

In the first term, \( u^* \) is evaluated at \( l_e/4 \) while in the second term it is evaluated at \( 3l_e/4 \). These values are again easily determined due to the linear variation of \( u^* \) over the element and are given by equations (3.8.b) and (3.8.c).

\[ u^* \left( \frac{l_e}{4} \right) = (u^*_j)_i + \frac{1}{4} \left( (u^*_{j+1})_i - (u^*_j)_i \right) \]

(3.8.b)

\[ = \frac{3}{4} (u^*_j)_i + \frac{1}{4} (u^*_{j+1})_i \]

\[ u^* \left( \frac{3l_e}{4} \right) = (u^*_j)_i + \frac{3}{4} \left( (u^*_{j+1})_i - (u^*_j)_i \right) \]

(3.8.c)

\[ = \frac{1}{4} (u^*_j)_i + \frac{3}{4} (u^*_{j+1})_i \]

Substitution of these expressions for \( u^* \) into equation (3.8.a) gives
where again, \( h(\cdot) \) denotes that the function \( h \) is to be evaluated at \( \cdot \).

c. Linear Approximation

This method evaluates \( u' \) in a linear manner over the entire element as a function of the element coordinate \( \xi \) where

\[
\begin{aligned}
u^* &= (u_j)^* \left( 1 - \frac{\xi}{l_e} \right) + (u_{j+1})^* \left( \frac{\xi}{l_e} \right)
\end{aligned}
\]

(3.9.a)

Substitution of this expression into equation (3.6) yields

\[
\begin{aligned}
f^* &= \int_0^l \left[ \frac{1 - \frac{\xi}{l_e}}{\frac{\xi}{l_e}} \right] h \left( (u_j)^* \left( 1 - \frac{\xi}{l_e} \right) + (u_{j+1})^* \left( \frac{\xi}{l_e} \right) \right) d\xi
\end{aligned}
\]

(3.9.b)

Since \( h(u') \) is no longer a constant but a function of \( \xi \), this integral must be evaluated for each specific \( h(u') \). Examples of each of these approximations is provided in Appendix C.

2. Linearization Matrix

At the element level, the integral in equation (3.5.b) becomes

\[
\begin{aligned}
\int_0^l g^T h(u^*) d\xi
\end{aligned}
\]

(3.10)

which is similar to the Galerkin FEM differential operator integral discussed in Chapter II, with the \( g^T \) term producing a 2x2 element matrix. In order to preserve the 2x2 nature of this matrix, \( h(u') \) must remain as a single term multiplying each term in the matrix. Two techniques for evaluating this integral are examined in this research.
resulting 2x2 element matrix is denoted as \( I' \) where the ‘*’ superscript again indicates that the matrix is changing with each iteration. These are then distributed into the system linearization matrix \( L' \) in accordance with the local to global nodal point correspondence.

\[\begin{align*}
\text{a. Midpoint Lumped Approximation} \\
\text{This method evaluates } u' \text{ at the midpoint of the element using the same process as in the linearization vector analysis and brings } h(u') \text{ outside the integral as a constant. Substitution of this expression into equation (3.10) gives}
\end{align*}\]

\[\begin{align*}
I' &= h\left( \frac{(u^*_i) + (u^*_{j+1})}{2} \right) \int^\xi_0 \left[ 1 - \frac{\xi}{l_e} \right] \left[ 1 - \frac{\xi}{l_e} \right] d\xi \\
&= h\left( \frac{(u^*_i) + (u^*_{j+1})}{2} \right) \left[ \frac{l_e}{3} \frac{l_e}{6} \frac{l_e}{6} \right] \\
(3.11)
\end{align*}\]

where \( h(\cdot) \) denotes that the function \( h \) is to be evaluated at ( ).

\[\begin{align*}
\text{b. Linear Approximation} \\
\text{This method transforms } u' \text{ into a linear function of the element coordinate } \xi \text{ as given by equation (3.9.a). This expression is substituted into equation (3.10) giving}
\end{align*}\]

\[\begin{align*}
I' &= \int^\xi_0 \left[ 1 - \frac{\xi}{l_e} \right] \left[ 1 - \frac{\xi}{l_e} \right] h\left( (u^*_i) \left( 1 - \frac{\xi}{l_e} \right) + (u^*_{j+1}) \frac{\xi}{l_e} \right) d\xi \\
(3.12)
\end{align*}\]

As for linear evaluation of the linearization vector, equation (3.12) must be evaluated for each \( h(u') \). An example of each of these approximations is provided in Appendix D.

\[\begin{align*}
\text{C. ITERATION STRATEGIES} \\
\text{Having evaluated the Galerkin integrals and developed a set of linear algebraic equations, the last step in the solution process is the determination of an iteration strategy. The main goal of an iteration strategy is to obtain a convergent approximation in a minimum number of steps while at the same time, keeping the computational effort of the iteration process to a minimum. Two of the factors which control the rate of convergence within a specific linearization strategy are}
\end{align*}\]
• how close the initial assumed values of $u'$ are to the actual solution

• the nature in which the FEM solution, $\hat{u}$, obtained at the end of each iteration, is utilized to obtain a value of $u'$ to begin each of the subsequent iterations

1. Initial Iteration

In order to begin any iteration process, an initial estimate, or guess, must be made as to the value of the variable which is to be determined. The relative accuracy of this guess with respect to the true solution of the differential equation greatly affects the ability of the solution process to converge as well as its rate of convergence. If the starting point for the iteration process is too far away from the actual solution, the likelihood for divergence or convergence to a nonsolution of the differential equation is very high.

The first step in formulating an initial iteration strategy is developing an idea as to what the activity range of the solution of the differential equation might be, i.e., how much and how quickly is the dependent variable changing over the prescribed domain. Since the solution is not known, this information must be obtained from the boundary conditions, the domain length, and insight as to the physics of the system.

Two possible combinations of boundary conditions exist for two point boundary value problems.

• Essential-Essential (E-E), where the magnitude of the dependent variable is specified at each end of the domain

• Essential-Natural (E-N), where the magnitude of the dependent variable is specified at one end of the domain, and the slope or rate of change of the dependent variable is specified at the other end.

The essential-essential combination provides information as to the probable activity range of the solution over the system domain, which is referred to as function order in this research and is defined below.

Function Order provides a relative magnitude of the range of values in the solution function as indicated by two essential boundary conditions. This relative magnitude is determined by writing the values of the essential boundary conditions in power ten exponent form and then taking the quotient of the maximum value over the minimum value. When the minimum valued boundary condition is 0.0, it should be written as $1 \times 10^0$ in order that the quotient does not become undefined. The magnitude of the exponent in this quotient defines the function order of the solution while the decimal value provides a ranking of how different function orders of the same magnitude compare. For example, take a differential equation which has boundary conditions of $u(0) = 0$ and $u(2) = 25$. These are written in power ten exponent form as $u(0) = 1.00 \times 10^0$ and $u(2) = 0.25 \times 10^2$. The quotient of these
two values is $0.25 \times 10^3$. Thus, the solution function is said to have a function order of two.

When an E-N boundary condition combination is specified, determination of the dependent variable activity range is not as straightforward as in the E-E situation. Instead, the activity range must be determined from a knowledge of the physics of the system as well as the actual values of the specified boundary conditions. The importance of properly estimating the activity range of the dependent variable cannot be overemphasized as this estimation is utilized in determining an initial iteration strategy, which is the most critical step in the solution procedure.

When the function order is zero or one, or the activity range is determined to be small based on the physics of the system, the magnitude of the dependent variable does not change appreciably over the prescribed domain. Thus, a reasonable estimate of the dependent variable for the first iteration would involve utilizing the essential boundary condition value(s). To examine the validity of this line of reasoning, the following initial iteration strategies are utilized in the present research when the solution has a function order of one or less.

- $u'$ set equal to the value of the left essential boundary condition.
- $u'$ set equal to the value of the right essential boundary condition
- $u'$ set equal to the average value of the two essential boundary conditions

When the function order is greater than one, there is a chance that utilization of any of the above criteria for the initial iteration values could result in divergence or convergence to a nonsolution of the differential equation. Any number of guesses could be made for the initial iteration values, but the chances of a random guess providing a convergent solution of the differential equation are quite low. Instead, a systematic approach given by the following four steps is suggested.

- Examine the physical system to which the differential equation applies and determine which terms on the left hand side of the equation tend to dominate.
- Neglect the nondominant term(s) and solve the equation for the dependent variable by any means possible, i.e., separation of variables, undetermined coefficients, successive integration, etc.
- Determine the value of the dependent variable at each node and let these be the values used in the initial iteration.
- If divergence results, reexamine the physical system and reevaluate the dominance of each of the terms. Then repeat steps two and three.
The process of determining initial iteration values that lead to a convergent solution can be very time consuming and frustrating. But, in most instances when the function order is greater than one, the selection of initial iteration values that are 'reasonably close' to the actual solution is crucial if a convergent solution is to be obtained.

To amplify the above guidelines, consider a one dimensional constant cross-sectional area heat fin with nonlinear conduction given by

\[
k(T)T'' - c_i(T - T_\infty) = 0
\]

(3.13)

where \(c_i\) is a constant based on the fin geometry and the convection coefficient, and \(T_\infty\) is the temperature of the convective fluid. When the fin is short or the value of \(k(T)\) for the given range of operating temperatures is much greater than \(c_i\), conduction tends to dominate and the convection term, \(c_i(T - T_\infty)\), is negligible. A value can then be assumed for \(k(T)\) based on the the physical system and boundary condition temperature(s). The equation thus becomes linear and can be integrated twice to yield \(T'(x)\). \(T'\) is then determined at each node which provides the values used for the initial iteration.

When the constant \(c_i\) is much larger than \(k(T)\), the convection term tends to dominate and conduction can be neglected. Thus, \(T'\) is set equal to \(T_\infty\) at each node for the initial iteration. In those cases where conduction dominates over one half of the domain and convection over the other, \(T''\) can be determined by using a combination of both these initial iteration strategies. The process of determining an initial iteration strategy is further developed in IV.A.5.

2. Subsequent Iterations

After the initial values of \(u'\) are determined, the next step is to develop an iteration procedure for subsequent \(u'\) values which will result in convergence with the least amount of computational effort. To aid in this development and subsequent discussions, the following notation is utilized.

- \((u_j^i)\), is the value of \(u\) used in the iteration process where \(j\) denotes the node number and \(i\) is the iteration number.
- \((\tilde{u}_j)\), is the value of \(u\) returned by the FEM approximation where \(j\) and \(i\) are defined as above.

Two different strategies for determining \(u'\) are investigated in the present research.

\[ \text{a. Previous Value Strategy} \]

This method takes the value of \(\tilde{u}\) from the previous iteration and sets it equal to \(u'\) for the next iteration process where
This is the simplest iteration scheme, but does not take into account how \( \tilde{u} \) is changing during the iteration process.

**b. Average Value Strategy**

This method uses the average value of \( \tilde{u} \) from the last two iterations yielding

\[
(u^*_j)_i = \frac{(\tilde{u}_j)_{i-1} + (\tilde{u}_j)_{i-2}}{2}
\]

(3.14.b)

This method should enhance convergence when \( \tilde{u} \) is oscillating about the final convergent solution while at the same time not adversely affect those situations where \( \tilde{u} \) is converging monotonically.

**c. Additional Strategies**

Numerous other strategies can be utilized in an attempt to increase the rate of convergence. The following are not investigated in this research but are presented as topics requiring further research.

- **K-step Strategy** - This method takes into account k previous iterations where

\[
(u^*_j)_i = \frac{(\tilde{u}_j)_{i-1} + (\tilde{u}_j)_{i-2} + \cdots + (\tilde{u}_j)_{i-k}}{k}
\]

(3.14.c)

- **Weighted Average Strategy** - A method which assigns a weighting factor to each iterative value of \( \tilde{u} \) where

\[
(u^*_j)_i = \frac{w_1(\tilde{u}_j)_{i-1} + w_2(\tilde{u}_j)_{i-2}}{w_1 + w_2}
\]

(3.14.d)

The objective is to find the optimum combination of weighting factors, \( w_1 \) and \( w_2 \), that yields the minimum number of iterations.

- **Weighted K-step Strategy** - A combination of the previous two strategies where

\[
(u^*_j)_i = \frac{w_1(\tilde{u}_j)_{i-1} + w_2(\tilde{u}_j)_{i-2} + \cdots + w_k(\tilde{u}_j)_{i-k}}{\sum_{n=1}^{k} w_n}
\]

(3.14.e)

- **Rate of Change Strategy** - This method would require using some from of a Taylor series expansion to model how \( \tilde{u} \) changes from iteration to iteration.

**D. NUMERICAL CONSIDERATIONS**

The following numerical aspects are considered in evaluating the ability of each solution procedure to provide a convergent solution of the nonlinear differential
equations that are investigated. Again, a solution procedure is a specific combination of linearization, interpolation, and iteration strategies.

1. Convergence/Divergence

For a convergent solution, the absolute value of the difference between \((\tilde{u}_i)\) and \((\tilde{u}_i)_{-1}\) at each node decreases as the number of iterations, \(i\), increases. In order to achieve the best approximation for a given number of elements, it is desirable to let the maximum value of \(|(\tilde{u}_i) - (\tilde{u}_i)_{-1}|\) at all nodes reach some minimum value before exiting the iteration process. Thus, the following percent difference convergence criterion is utilized: convergence is reached when the absolute value of the maximum difference between \((\tilde{u}_i)\) and \((\tilde{u}_i)_{-1}\) at any node divided by \((\tilde{u}_i)\) at the same node it less than .0001 (.01%). This is shown mathematically in equation (3.15) and is but one of many convergence criteria that could be utilized.

\[
\text{Convergence Criterion: } \left| \frac{\text{MAX } |(\tilde{u}_i) - (\tilde{u}_i)_{-1}|}{(u_i)} \right| < .0001 \quad (3.15)
\]

The solution procedure is considered a failure if any of the following situations occur.

- convergence does not occur within 200 iterations
- divergence occurs, i.e., \(\tilde{u}\) increases without bound as the number of iterations increases
- convergence to a nonsolution of the differential equation

2. Critical Number of Degrees of Freedom (DOF)

In some instances, especially when the function order is greater than two or the activity range is large, there may be a specific or critical number of DOF below which the solution procedure will not converge. To examine this phenomenon, each solution strategy is initially evaluated using three DOF i.e., two elements. The number of DOF is then increased until either a grid independent solution is obtained or the number of DOF exceeds 100; and a critical number of DOF, if it exists, is determined.

3. Stability

Stability in numerical analysis applications is based on the approximation method's ability to converge as it relates to the time and displacement discretization that are utilized. For example, the finite difference explicit method is stable for \(r < \frac{1}{2}\) where \(r = \frac{\Delta t}{(\Delta x)^2}\), while the Crank-Nicholson implicit method is stable for any value of \(r\), i.e., unconditionally stable. For this research, three types of stability are defined.
• Unconditional Stability - There is no critical number of DOF required to guarantee a convergent solution.
• Conditional Stability - There is a critical number of DOF below which a convergent solution cannot be obtained.
• Unstable - The solution process diverges regardless of the number of DOF utilized.

4. Multiplicity of Solutions
Nonlinear differential equations do not necessarily have a unique solution. The equations utilized in this research were developed based on known solutions. When a solution different from them is obtained, that solution is checked at two different points in the domain by passing a parabola through the point in question and two adjacent points. If the equations developed from both of these parabolas satisfy the differential equation, the solution procedure is considered to have provided a valid approximation.

5. Boundary Condition Effects
Two point boundary value problems require a boundary condition at each end. Two combinations are valid; either an essential (Dirichlet) boundary condition at each end or an essential at one end and a natural (Cauchy) at the other. The effect that each of these combinations has on the performance of each solution procedure is investigated.

6. Computational Efficiency
This research defines the most computationally efficient solution procedure as the one which provides the most accurate results for the least amount of computer time. CPU time by itself though is not an effective measure of efficiency. For example, a solution process that uses two elements will take much less time to run than one with 40, but the 40 element solution is likely to be much more accurate. There should be some critical number of elements for a specific solution procedure beyond which the increase in CPU time for the additional calculations is not matched by a proportional increase in solution accuracy.

The different solution strategies in this research are compared using a factor defined as

\[ CPU^* = CPU \times \text{average } \% \text{ error} \]  

CPU is the amount of computer time (in seconds) required to complete the iteration process. A timing subroutine installed in the FORTRAN solution program starts when
the iteration process is entered and stops when it is exited. Average % error is the average percent difference between the FEM approximation and the exact solution given by

$$\text{avg \% error} = \frac{\sum_{i=1}^{N} |\% error|}{N} \quad N = \text{number of system degrees of freedom}$$

(3.16.b)

As the CPU time increases due to an increase in the number of elements, the percent error should decrease and approach zero. Therefore, the solution strategy yielding the minimum CPU$^*$ is defined as the most computationally efficient for a given equation and nonlinear operator.
IV. APPLICATIONS

A. PRELIMINARIES

1. Equations, Domains, and Boundary Conditions

This chapter evaluates the performance of various combinations of the linearization, interpolation, and iteration strategies previously developed in solving second order, nonlinear, one dimensional, differential equations containing the $u^2$ nonlinear term. In order to investigate the effect of the many numerical considerations described in the preceding chapter, two nonlinear differential equations over three domains with appropriate essential and natural boundary conditions are solved. The differential equations considered were

\[
\begin{align*}
    u'' - u^2 &= 6 - 9x^4 & u_{\text{exact}} &= 3x^2 \\
    u'' + u^2 &= 60x + 100x^6 & u_{\text{exact}} &= 10x^3
\end{align*}
\]  

(4.1)  

(4.2)

Domains and Boundary Conditions

- Domain 1: $0 < x < 1$
  Eqn. (4.1) $u(0) = 0, \ u(1) = 3$ or $u'(1) = 6$
  Eqn. (4.2) $u(0) = 0, \ u(1) = 10$ or $u'(1) = 30$

- Domain 2: $0 < x < 2$
  Eqn. (4.1) $u(0) = 0, \ u(2) = 12$ or $u'(2) = 12$
  Eqn. (4.2) $u(0) = 0, \ u(2) = 80$ or $u'(2) = 120$

- Domain 3: $0 < x < 5$
  Eqn. (4.2) $u(0) = 0, \ u(5) = 75$ or $u'(5) = 30$
  Eqn. (4.2) $u(0) = 0, \ u(5) = 1250$ or $u'(5) = 750$

In all cases the left end point of the domain has an essential boundary condition. Either an essential or natural boundary condition is provided at the right end of the domain in order to investigate the effect of the two different boundary condition combinations, namely essential-essential (E-E) and essential-natural (E-N) as discussed in III.D.5.

Equations (4.1) and (4.2) were developed by starting with a known solution, $u_{\text{exact}}$, and working backwards to form a second order nonlinear differential equation. The equations were kept simple, i.e., only one linear and one nonlinear term, due to the number of solution procedures that required evaluation, as well as the many numerical considerations involved. Still, equations (4.1) and (4.2) are viable representations of two
engineering phenomena that are described by second order differential equations, such as

- An axially loaded bar embedded in a nonlinear elastic medium as shown in Figure 10 on page 34.
- A constant cross-sectional area heat fin with internal heat generation and nonlinear convection as shown in Figure 11 on page 35.

2. Related Engineering Phenomena
   
   a. Bar Problem
      
      Consider the bar problem of Figure 10 subjected to distributed force \( p(x) \), embedded in a nonlinearly elastic media which exerts an opposing distributed force proportional to the square of the displacement, \( u \). A free body analysis of a differential element yields
      
      \[
      F(x) + dF(x) + p(x)dx - F(x) - u^2(x)dx = 0
      \]  
      (4.3.a)
      
      Cancelling the \( F(x) \) terms, taking the applied excitation \( p(x) \) to the right hand side of the equation and dividing by \( dx \) gives
      
      \[
      \frac{dF(x)}{dx} - u^2(x) = -p(x)
      \]  
      (4.3.b)
      
      From solid mechanics, the following relations are known,
      
      \[
      F = \sigma A
      \]  
      (4.4.a)
      
      \[
      \sigma = \varepsilon E
      \]  
      (4.4.b)
      
      \[
      \varepsilon = \frac{du}{dx}
      \]  
      (4.4.c)
      
      where \( F \) is the axial force, \( \sigma \) is the axial stress, \( A \) is the cross-sectional area, \( \varepsilon \) is the strain, and \( E \) is Young's Modulus. Substitution of equations (4.4.c) and (4.4.b) into equation (4.4.e) provides a relation for the force as
      
      \[
      F = EA\varepsilon
      \]  
      (4.4.d)
      
      Differentiating equation (4.4.d) with respect to \( x \) yields
      
      \[
      \frac{dF}{dx} = EA\varepsilon'
      \]  
      (4.4.e)
Substitution of equation (4.4.c) into equation (4.3.b) leaves

\[ E\Delta u'' - u^2 = -p(x) \]  

which is similar in form to equations (4.1) and (4.2), if the stiffness, \( E\Delta \), is set equal to unity.

Figure 10. Axially Loaded Bar Embedded in a Nonlinearly Elastic Material

b. Heat Fin Problem

Now consider the heat fin problem of Figure 11 on page 35 where \( q(x) \) is the volumetric heat generation/unit length and the heat transfer coefficient, \( h \), is a linear function of the temperature, \( T \), that is \( h = aT \). An energy balance on the fin differential element yields

\[ q_{\text{cond}} - \dot{q}(x)dx - (q_{\text{cond}} + dq_{\text{cond}}) - dq_{\text{conv}} = 0 \]  

(4.6.a)

Cancelling the \( q_{\text{cond}} \) terms, dividing by \( dx \), and rearranging terms yields

\[ - \frac{dq_{\text{cond}}}{dx} - \frac{dq_{\text{conv}}}{dx} = \dot{q}(x) \]  

(4.6.b)

The following relations are known from heat transfer principles,
\[ q_{\text{cond}} = -k\Lambda c \frac{dT}{dx} \]  \hspace{1cm} (4.7.a)

\[ q_{\text{conv}} = h\Lambda s (T - T_\infty) \]  \hspace{1cm} (4.7.b)

where \( \Lambda_c \) is the cross-sectional area of the fin and is constant, \( \Lambda_s \) is the surface area of the fin and is written as \( P \times \) where \( P \) is the perimeter of the heat fin, \( T_\infty \) is the ambient temperature of the convective media, and \( k \) is the thermal conductivity coefficient. Substitution of equations (4.7.a) and (4.7.b) into equation (4.6.b) and performing the appropriate differentiation yields

\[ k\Lambda_c T'' - hP(T - T_\infty) = \dot{q}(x) \]

Letting \( h = aT, T_\infty = 0 \), and dividing through by \( k\Lambda_c \) yields

\[ T'' - \frac{aP}{k\Lambda_c} T^2 = \frac{1}{k\Lambda_c} \dot{q}(x) \]  \hspace{1cm} (4.8.b)

which is similar to equations (4.1) and (4.2) if coefficient \( (aP/k\Lambda_c) \) is set equal to unity.

**Figure 11.** Heat Fin With Nonlinear Convection
3. Function Order and Domain Nondimensionalization

The exact solutions on which equations (4.1) and (4.2) are based, and the three domains of these equations previously described, were judiciously chosen to provide information as to the effect of function order on the performance of the solution procedures investigated in this analysis. The function order of each equation was determined over each domain based on the specified E-E boundary conditions and is provided in Table 2.

In order to determine whether an advantage is gained by the nondimensionalization of a domain, the following investigation was undertaken for both equations over domains two and three with equation (4.1) over domain three provided as an example. In this case, the dimensional variable \( x \) was replaced by the nondimensional independent variable \( \eta = x/5 \). Since \( d\eta = dx/5 \) and \( x = 5\eta \), equation (4.1) was transformed to

\[
u'' - 25u^2 = 150 - 140625u^4 \quad 0 < \eta < 1\]

where differentiation now is with respect to \( \eta \). Analysis of each nondimensionalized equation established that the transfer of 'domain activity' to 'differential equation activity' did not result in any computational gain.

<table>
<thead>
<tr>
<th>Equation</th>
<th>Domain</th>
<th>Function Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4.1)</td>
<td>One</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Two</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Three</td>
<td>2</td>
</tr>
<tr>
<td>(4.2)</td>
<td>One</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Two</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Three</td>
<td>4</td>
</tr>
</tbody>
</table>

4. General Solution Procedure

Subsequent analyses using the solution procedures discussed next, show that the efficacy of any particular solution procedure depends strongly on the function order of the problem being solved. The general solution procedure consists of the three steps;
linearization, interpolation, and iteration; shown in Figure 12 on page 38. First, equations (4.1) and (4.2) are linearized using the linearization strategies developed in III.A. Secondly, an interpolation strategy developed in III.B is used with the Galerkin FEM and the linearized differential equation is transformed into a set of linear algebraic equations. Finally, these algebraic equations are solved iteratively using various combinations of iteration strategies developed in III.C. The results of each solution procedure, where a solution procedure is a particular set of selected strategies, are then compared and evaluated based on their performance in approximating the solutions of equations (4.1) and (4.2).

5. Initial Iteration Strategy

The initial iteration strategy is the weak link in the overall solution procedure. For function order one problems, the initial iteration strategy is simply to utilize the essential boundary conditions as described in III.C.1. This is the strategy adopted for equation (4.1) over domain one, as well as equation (4.2) over the same domain because the latter problem becomes function order two only at the right boundary point of the domain.

Another initial iteration strategy is utilized for equations (4.1) and (4.2) over domains two and three because their function order is greater than one. The physical systems to which equations (4.1) and (4.2) apply are generic in nature and do not provide the necessary information for determining which of the terms on the left hand side of the equation dominate the behavior of the system. However, since each of these equations is based on a known solution, the terms on the left side can be written as specific functions of $x$. This information is utilized to devise an initial iteration strategy for each equation over domains two and three based on the indicated dominance of the linear and nonlinear terms. Normally, the solution of the nonlinear differential equation is unknown. In these cases, the physical system must be scrutinized and a determination made as to which processes and their corresponding operators dominate over each part of the domain. Having once assessed the dominance of each operator over various parts of the domain, the following strategies can be utilized to generate the initial iteration values.

- Where the $u''$ dominates, the initial iteration vector is obtained by solving $u'' = f(x)$.
- Where the $u'$ dominates, the initial iteration vector is obtained by solving $u' = f(x)$.
SOLUTION PROCEDURE

NONLINEAR DIFFERENTIAL EQUATION
\[ \mathcal{L} u + \mathcal{L}(u) - f = 0 \]

LINEARIZATION STRATEGY

LINEAR DIFFERENTIAL EQUATION
\[ \mathcal{L}^* u - f = 0 \]

FEM WITH INTERPOLATION STRATEGY

RECURSIVE LINEAR ALGEBRAIC EQUATIONS
\[ (A + A^*_{i-1})u_i - F^*_{i-1} - F = 0 \]

INIT. ITER. STRAT. SUBS. ITER. STRAT. (i = ITER. C'NTER)

CONVERGED SOLN AFTER n ITER'S

Figure 12. Solution Procedure for Nonlinear Differential Equations
In equation (4.1), \( u_{x=0} = 3x^2 \) which yields \( u'' = 6 \) and \( u^2 = 9x^4 \). These functions are plotted in Figure 13 on page 40 for the domain \( 0 < x < 2 \) and in Figure 14 on page 41 for the domain \( 0 < x < 5 \). From Figure 13, the \( u'' \) term is slightly more dominant than the \( u^2 \) term over the first half of the domain. Over the second half of the domain, the \( u^2 \) term is clearly dominant. Therefore, an appropriate initial iteration strategy is to utilize a procedure which neglects the \( u^2 \) term in determining the initial values of \( (u_0')_0 \) over \( 0 < x < 1 \) and neglects the \( u'' \) term over \( 1 < x < 2 \). The first part of the initial value procedure involves integrating equation (4.1) twice without the imposition of any boundary conditions. The second part requires taking the square root of the right hand side of equation (4.1). Thus, the initial iteration values used in solving equation (4.1) over domain two are determined using equation (4.9).

\[
(u_0^*)_0 = \begin{cases} 
3x^2 - \frac{3}{10}x^6 & 0 < x < 1 \\
\sqrt{9x^4 - 6} & 1 < x < 2 
\end{cases} 
\] (4.9)

From Figure 14 it is apparent that the \( u^2 \) term dominates over a majority of the domain. Thus, the \( u'' \) term is completely neglected and the initial iteration values are determined using equation (4.10).

\[
(u_0^*)_0 = \sqrt{9x^4 - 6} 
\] (4.10)

Nonlinear differential equation (4.2) was developed using \( u_{x=0} = 10x^3 \) which yields \( u'' = 60x \) and \( u^2 = 100x^4 \). These functions are plotted in Figure 15 on page 42 for a domain of \( 0 < x < 2 \) and in Figure 16 on page 43 over domain \( 0 < x < 5 \). Examination of Figure 15 shows the \( u'' \) term slightly dominating the \( u^2 \) term over the first half of the domain and the \( u^2 \) term clearly dominating over the second half of the domain. Thus, the same procedure for determining initial iteration values as was utilized for equation (4.1) over domain two is employed and the results are given by equation (4.11).

\[
(u_0^*)_0 = \begin{cases} 
10x^3 + \frac{100}{56}x^8 & 0 < x < 1 \\
\sqrt{60x + 100x^6} & 1 < x < 2 
\end{cases} 
\] (4.11)
Figure 13. Dominance of Terms in Equation (4.1) over Domain Two
Figure 14. Dominance of Terms in Equation (4.1) over Domain Three
EQN(4.2) $U'' + U^2 = 60X + 100X^2$

**Legend**
- $U'' = 60X$
- $U^2 = 100X^2$

**Figure 15.** Dominance of Terms in Equation (4.2) over Domain Two
Figure 16. Dominance of Terms in Equation (4.2) over Domain Three
Over the domain $0 < x < 5$, Figure 16 clearly shows the domination of the $u^2$ term over a majority of the domain. Therefore the $u''$ term is neglected and the initial iteration values are determined using equation (4.12).

$$ (u^*)_0 = \sqrt{60x + 100x^5} \quad (4.12) $$

The efficacy of the above initial iteration strategies was confirmed by the analyses which were undertaken.

**B. CONSTANT LINEARIZATION**

1. **Problem Formulation**

The constant linearization strategy, described in III.A.1, transforms the nonlinear term, $u^2$, into a constant linear term as shown in equation (4.13)

$$ u^2 \approx \mathcal{L}^* u = (u^*)^2 \quad (4.13) $$

where $u'$ is determined as outlined in III.C. This process results in a linear differential equation of the form

$$ u'' = \pm (u^*)^2 + f(x) \quad x \in D \quad (4.14) $$

where '+' is for equation (4.1), '−' is for equation (4.2) and $f(x)$ represents the excitation function in each equation. The Galerkin FEM formulation process outlined in Chapter II transforms equation (4.14) into

$$ G(G^T)'u|_a^b - \int_D G'(G^T)'dx u = \pm \int_D G(u^*)^2 dx + \int_D G f(x)dx \quad (4.15) $$

where $a$ and $b$ in the first term represent the value of $x$ at the left and right boundary of the domain, respectively. The left hand side of equation (4.15) is similar to that of equation (2.2.c) and upon evaluation yields $B - Au$, where the vector $B$ is only present when a natural boundary condition is specified. The integrals on the right hand side are now evaluated.

**Linearization Vector**, $\int_D G(u^*)^2 dx$

This integral is evaluated as outlined in III.B.1 where $h(u')$ in equation (3.6) is set equal to $(u')^2$. The detailed formulation of the $2x1$ $f^*$ element vectors for the three different interpolation strategies is given in Appendix C with the final results shown in
Table 3 on page 45. The 2x1 \( f' \) vectors are then distributed into the system linearization vector, \( F' \), in accordance with the local to global nodal point correspondence. The \( F' \) vector continuously changes during the iteration process as the values of \( u' \) are revised.

Table 3. **CONSTANT LINEARIZATION ELEMENT VECTORS**

<table>
<thead>
<tr>
<th>Interpolation Strategy</th>
<th>( f' )</th>
</tr>
</thead>
</table>
| Midpoint Approximation              | \[
\left( \frac{(u'_i) + (u'_{i-1})}{2} \right)^2 \left[ \begin{array}{c} \frac{l}{2} \\ \frac{l}{2} \end{array} \right]
\] |
| 1/4 - 3/4 Approximation             | \[
\frac{l}{2} \left[ \begin{array}{c} \left( \frac{3}{4} (u'_i) + \frac{1}{4} (u'_{i-1}) \right)^2 \\ \left( \frac{1}{4} (u'_i) + \frac{3}{4} (u'_{i-1}) \right)^2 \end{array} \right]
\] |
| Linear Approximation                | \[
\frac{4}{l} \left( \frac{(u'_i)^2}{6} + \frac{(u'_i)(u'_{i-1})}{6} + \frac{(u'_{i-1})^2}{4} \right)
\] |

**Excitation, \( \int G f(x) dx \)**

Transforming \( x \) to the element coordinate system, \( \xi \), the element integral for \( f \) becomes

\[
f = \int_0^l g f(\alpha + \xi) d\xi
\]  

(4.16)

where \( \alpha_i \) is the distance from the origin of \( x \) to the origin of \( \xi \) of the element for which \( f \) is being evaluated. Equation (4.16) is evaluated using the consistent technique described in II.A, which is detailed in Appendix A, where \( f() \) is replaced by the appropriate excitation function in equations (4.1) and (4.2). The resulting 2x1 element excitation vectors are presented in Table 4 on page 46. The \( f \) vectors remain steady (constant) during the iteration process and are distributed into an Nx1 system force vector, \( F \), in accordance with the local to global DOF correspondence.
Table 4. ELEMENT FORCE VECTORS FOR EQUATIONS (4.1) AND (4.2)

<table>
<thead>
<tr>
<th>$f(x)$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$6 - 9x^4$</td>
<td>$l_3 \left[ \frac{3}{3} \right] - \frac{9}{2} \alpha l_5 \left[ \frac{1}{1} \right] - 6 \alpha l_2 \left[ \frac{2}{2} \right] - \frac{9}{2} \alpha l_1 \left[ \frac{1}{3} \right] - \frac{9}{5} \alpha l_4 \left[ \frac{1}{4} \right] - \frac{1}{30} l_5 \left[ \frac{1}{5} \right]$</td>
</tr>
<tr>
<td>$60x + 100x^4$</td>
<td>$30 \alpha l_5 \left[ \frac{1}{1} \right] + 10 \alpha l_2 \left[ \frac{2}{2} \right] + 50 \alpha l_1 \left[ \frac{1}{1} \right] + 100 \alpha l_1 \left[ \frac{1}{2} \right] + 125 \alpha l_5 \left[ \frac{1}{3} \right] + 100 \alpha l_1 \left[ \frac{1}{4} \right] + 50 \alpha l_1 \left[ \frac{1}{5} \right] + \frac{100}{7} \alpha l_6 \left[ \frac{1}{6} \right] + \frac{25}{16} l_5 \left[ \frac{1}{7} \right]$</td>
</tr>
</tbody>
</table>

Final FEM Equations

Substitution of the matrix and vector equivalents for each integral into equation (4.15) yields a system of equations given by

$$B - Au = F^* + F$$  \hspace{1cm} (4.17.a)

where the ± sign in equation (4.15) is incorporated into $F^*$. The natural boundary condition vector $B$, when present for an E-N problem does not change during the iteration process and is taken to the right side and subtracted from $F$ yielding $F_m$, where the $m$ subscript indicates that the excitation vector is modified for the given natural boundary condition. The system of equations then takes the final form of

$$-Au = F_{i-1}^* + F_m$$  \hspace{1cm} (4.17.b)

where subscripts $i$ and $i - 1$ refer to the iteration counter. Equation 4.17.b is solved iteratively for $u$, with $F^*$ changing after each iteration, until convergence is obtained.

2. Results

a. General

Equations (4.1) and (4.2) were each solved over domain one using 24 different solution procedures while 12 different procedures were utilized for both domains two and three. The FORTRAN programs utilized for the constant linearization strategy are contained in Appendix E. A summary of the strategies utilized in each solution procedure is shown in Table 6 on page 49 and Table 7 on page 50 for equation (4.1), and Table 8 on page 51 and Table 9 on page 52 for equation (4.2), with the following performance information provided in the results portion of each table.
• Convergence (III.D.1)
• Stability (III.D.3)
• Number of iterations required to obtain convergence
• Average percent error (III.D.6)
• CPU* (III.D.6)

The results for each solution procedure over each of the domains were obtained using the number of elements shown in Table 5. Though these number of elements are not necessarily the number required to obtain a grid independent solution for each solution procedure, they do provide a common baseline upon which the performance of individual solution procedures for a particular problem may be compared. This is also the number of elements utilized in the table of results for each of the solution procedures developed by the other two linearization strategies.

<table>
<thead>
<tr>
<th>Equation</th>
<th>Domain</th>
<th>Number of Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>50</td>
</tr>
</tbody>
</table>

Two general observations can be made based on the results of Table 6 through Table 9. One is that the constant linearization technique begins to fail as the function order over the given domain begins to approach two. This is clearly shown in Table 8 as only half of the solution procedures provided convergent solutions of equation (4.2), whose function order is just barely two over the domain \(0 < x < 1\). As soon as the domain length was increased to 1.1, these remaining methods diverged as shown in Table 9. This is most likely due to equation (4.2) becoming fully order two as the domain is increased past \(x = 1.0\). In order to determine where the solutions begin to breakdown for equation (4.1), the domain was extended in 0.1 increments until all solution procedures diverged. The results of this investigation showed that some solution procedures for equation (4.1) were able to provide convergent solutions over a
domain of $0 < x < 1.7$, which approaches the region where equation (4.1) shifts in function order from one to two. This failure to provide convergent solutions of equations (4.1) and (4.2) is not due to a lack of elements used in the approximation, but is a characteristic of this particular linearization strategy.

The second observation is that the number of iterations required to obtain convergence is strictly a function of the iteration strategies utilized, both initial and subsequent, and is independent of the interpolation strategy. Also, it was found that the number of iterations is independent of the number of degrees of freedom utilized. That is, the same number of iterations were required to reach convergence whether two or 20 elements were utilized. The accuracy of the approximation, on the other hand, seems to be independent of the iteration strategy, and only a function of the interpolation strategy and the number of degrees of freedom. The effect of each problem parameter in the different solution procedures for both equations (4.1) and (4.2) over domain one is now examined, as no convergent solutions were obtained for domains two and three.

b. Boundary Conditions

The use of an essential boundary condition at both ends of the domain increased the rate of convergence by a factor of two to five over those strategies which utilized an essential and natural boundary condition combination. In fact, only three of the twelve strategies which used an E-N boundary condition combination provided convergent approximations of the differential equation. Suffice it to say, the essential-natural boundary condition combination does not produce very efficient results when utilized with the constant linearization strategy and therefore this strategy should not be used for E-N problems. That being the case, the remaining comments are directed at those strategies which utilized an essential-essential boundary condition combination.

c. Initial Iteration Strategy

Each of the initial iteration strategies, described in III.C.1, provided nearly the same results within a specific subsequent iteration and interpolation strategy combination when the function order was small, as in equation (4.1). But, when the function order begins to approach two, as in equation (4.2), the use of the right essential boundary condition for the initial iteration values led to divergence, while the other two strategies provided similar convergent results.

d. Subsequent Iteration Strategy

The use of the previous value strategy (III.C.2.a) generally led to convergence using less iterations than the average value strategy (III.C.2.b). The main reason for this is most likely a result of the constant linearization strategy providing for
<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Iter. Strat.</th>
<th>Interp. Strat.</th>
<th>$u_h''$</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU' (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Midpt.</td>
<td>Prey.</td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>8</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>10.48</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>1/4-3/4</td>
<td>Prey.</td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>8</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>200+</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Linear</td>
<td>Prey.</td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>8</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>200+</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Midpt.</td>
<td>Avg.</td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>9</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>38</td>
<td>0.27</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>1/4-3/4</td>
<td>Avg.</td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>9</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>38</td>
<td>0.31</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Linear</td>
<td>Avg.</td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>9</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>38</td>
<td>0.47</td>
</tr>
</tbody>
</table>

**LEGEND:**
- E = essential boundary condition
- N = natural boundary condition
- L = left essential boundary condition
- R = right essential boundary condition
- M = average of L and R
- C = convergence
- D = divergence
- S = unconditional stability
- CS = conditional stability
- U = unstable

**NOTES:**
- Oscillates between two nonsolutions of the differential equation
Table 7. **SOLUTION PROCEDURES AND RESULTS USING CONSTANT LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAINS TWO AND THREE**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Interp.</th>
<th>B. C.'s</th>
<th>((u'_0))</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU* (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iter. Strat.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interp. Strat.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Prev. Value</td>
<td>1/4-3/4</td>
<td>E</td>
<td>E (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E</td>
<td>E (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>1/4-3/4</td>
<td>E</td>
<td>E (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E</td>
<td>E (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 5)</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 5)</td>
<td>Prev. Value</td>
<td>1/4-3/4</td>
<td>E</td>
<td>E (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 5)</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E</td>
<td>E (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 5)</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 5)</td>
<td>Avg. Value</td>
<td>1/4-3/4</td>
<td>E</td>
<td>E (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 5)</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E</td>
<td>E (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N (2)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>

**LEGEND:**
- \(E\) = essential boundary condition;
- \(N\) = natural boundary condition;
- \(L\) = left essential boundary condition;
- \(R\) = right essential boundary condition;
- \(M\) = average of \(L\) and \(R\);
- \(C\) = convergence;
- \(D\) = divergence;
- \(S\) = unconditional stability;
- \(CS\) = conditional stability;
- \(U\) = unstable;

**NOTES:**
1. See equation (4.9)
2. See equation (4.10)
Table 8. SOLUTION PROCEDURES AND RESULTS USING CONSTANT LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN ONE

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iter. Strat.</td>
<td>Interp. Strat.</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prev. Value</td>
<td>Midpt.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prev. Value</td>
<td>1/4-3/4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prev. Value</td>
<td>Linear</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. Value</td>
<td>Midpt.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. Value</td>
<td>1/4-3/4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. Value</td>
<td>Linear</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LEGEND: E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence; S = unconditional stability; CS = conditional stability; U = unstable

NOTES: a - Converges to a nonsolution of the differential equation
## Table 9. Solution Procedures and Results Using Constant Linearization to Solve Equation (4.2) Over Domains Two and Three

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Interp. Strat.</th>
<th>B. C.'s</th>
<th>(a'_i)'s</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU* (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lt</td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>1/4-3/4</td>
<td>E E (1)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Avg. Value</td>
<td>1/4-3/4</td>
<td>E E (1)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E E (2)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Prev. Value</td>
<td>1/4-3/4</td>
<td>E E (2)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E (2)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E E (2)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>1/4-3/4</td>
<td>E E (2)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E E (2)</td>
<td>D U</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
</tbody>
</table>

**LEGEND:**
- E = essential boundary condition; N = natural boundary condition;
- L = left essential boundary condition; R = right essential boundary condition;
- M = average of L and R; C = convergence; D = divergence S = unconditional stability;
- CS = conditional stability; U = unstable

**NOTES:**
1. See equation (4.11)
2. See equation (4.12)
monotonic convergence when used with an E-E combination. The essential boundary conditions provide a range that the values of \( u \) should be between and thus provide strong guidance during the iteration process. The one exception to this involves those cases where a natural boundary condition is given at the right end of the domain. In these instances, only the average value iteration strategy resulted in convergent solutions. This E-N combination does not fix the value of the dependent variable at the right end, so the approximate solution most likely oscillates about the exact solution during the iteration process. The average value strategy tends to enhance the convergence of this type of approximation, which might explain why this strategy resulted in convergence of the E-N problem while the previous value strategy did not.

e. Interpolation Strategy

The 1/4-3/4 interpolation method consistently provided the most accurate approximations as indicated by the low average percent error values. This is because the solution function for both equations (4.1) and (4.2) is a polynomial function of \( x \) and the 1/4-3/4 method was shown in Chapter II to provide better approximations of the integrals of these functions than the midpoint or linear techniques.

f. Overall Performance

The overall performance, which factors both computational effort (i.e., CPU time) and solution accuracy, of a particular solution procedure is indicated by its respective value of CPU'; where the lower the value, the more efficient the solution procedure. In Table 6, the CPU times upon which the CPU' values are based were all at or below the clock subroutine accuracy of \( \pm .03 \) seconds. Thus, the CPU' values in this table should be used with caution. A comparison of the solution procedures using average percent error values (% Dif in Table 6) indicates the previous value iteration strategy combined with the 1/4-3/4 interpolation strategy provides the most accurate solution of equation (4.1) over domain one.

Due to the increase in the number of elements and iterations required for convergence of equation (4.2), more CPU time was required by the solution procedures in Table 8. Hence, the CPU' times in Table 8 are all based on CPU times much greater than the clock subroutine accuracy and therefore are all valid. Again, the combination of previous value iteration and 1/4-3/4 interpolation strategies provided for the most efficient procedure.

3. Conclusions

The constant linearization strategy can generally provide approximations of nonlinear differential equations when the function activity is less than order two over the
given domain. The relative 'crudeness' of this linearization technique requires that as
much information as possible concerning the actual value of the dependent variable over
the given domain be known. Thus, the knowledge of an essential boundary condition
at each end of the domain is almost a prerequisite to obtain a convergent solution of a
nonlinear differential equation when using this linearization method. If these conditions
are met, an iteration strategy utilizing the magnitude of the smallest valued boundary
condition for the initial iteration and the previous value strategy for subsequent iter-
ations should result in convergence with a minimum number of iterations and expend
the least amount of CPU time for a given number of degrees of freedom. The 1/4-3/4
interpolation strategy should yield the most accurate approximation as most solutions
of engineering problems are monotonically increasing or decreasing functions, or at
worst, convex or concave over the given domain.

C. CLASSICAL LINEARIZATION

1. Problem Formulation

The classical linearization strategy transforms the $u^2$ nonlinear term into a linear
term as described in III.A.2 and shown in equation (4.18)

$$u'^2 \approx L'u = u^*u$$

where $u'$ is determined as outlined in III.C. Substitution of equation (4.18) into
equations (4.1) and (4.2) yields a linear differential equation of the form

$$u'' \pm u^*u = f(x) \quad x \in D$$

where the '−' is for equation (4.1), the '+' is for equation (4.2) and $f(x)$ is again the
respective excitation function in each equation. The Galerkin FEM formulation process
transforms equation (4.19) into

$$\begin{align*}
G(G^T)u' \big|_a^b & - \int_D G'(G^T)dx u \pm \int_D G^T u^*dx u = \int_D Gf(x)dx \\
\end{align*}$$

The first two terms on the left side of equation (4.20) again provide $B = Au$ where the
$B$ vector is present only when a natural boundary condition is provided. The integral
on the right side of equation (4.20), which was evaluated in IV.B.1, gives the system
excitation vector, $F$. The only term remaining to be evaluated is the third term on the
left side of equation (4.20), the linearization matrix integral.
Linearization Matrix, $\int_{\Omega} GG' u' dx$

This integral is evaluated as outlined in III.B.2 where $h(u')$ in equation (3.10) is replaced by $u'$. The detailed formulation of the element linearization matrices, $L'$, for the two different interpolation strategies is given in Appendix D, with the final results provided in Table 10. The $L'$ are then distributed into the system linearization matrix, $L^*$, based on the local to global nodal point correspondence. The $L^*$ matrix is updated during each iteration as the values of $u'$ are revised.

Table 10. CLASSICAL LINEARIZATION ELEMENT MATRICES

<table>
<thead>
<tr>
<th>Interpolation Strategy</th>
<th>$L'$</th>
</tr>
</thead>
</table>
| Midpoint Approximation | \[
\frac{1}{2} \begin{pmatrix}
\frac{l_3}{3} & \frac{l_4}{3} \\
\frac{l_4}{3} & \frac{l_3}{3}
\end{pmatrix}
\] |
| Linear Approximation   | \[
\frac{1}{12} \begin{pmatrix}
3(u'_1) & (u'_{-1}) & (u'_1) & (u'_{-1}) \\
(u'_1) & (u'_{-1}) & (u'_1) & (u'_{-1}) & 3(u'_{-1})
\end{pmatrix}
\] |

Substitution of the matrix and vector equivalents for each integral into equation (4.20) yields a system of equations given by

$$B - Au + L^* u = F$$ (4.21.a)

where the $\pm$ sign in equation (4.20) is incorporated into $L^*$. Again, $F$ and $B$ are combined to yield $F_m$, and the system of linear algebraic equations takes the final form

$$(-A + L_{i+1}^*)u_i = F_m$$ (4.21.b)

Equation (4.21.b) is solved iteratively for $u_i$, with $L_{i+1}^*$ being calculated using $u'_i$ values, until convergence is obtained.

2. Results

a. General

Sixteen different solution procedures were utilized to solve equations (4.1) and (4.2) over domain one and eight procedures were utilized for each equation over both domains two and three. The FORTRAN programs for constant linearization are contained in Appendix F. The different strategies utilized in each procedure and the
corresponding results are provided in Table 11 on page 56, and Table 12 on page 57 for equation (4.1); and Table 13 on page 58, and Table 14 on page 59 for equation (4.2). The number of elements utilized in each solution procedure is shown in Table 5 on page 47.

Table 11. SOLUTION PROCEDURES AND RESULTS USING CLASSICAL LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN ONE

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>B. C.'s (u')_0</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C</td>
<td>S</td>
</tr>
</tbody>
</table>

LEGEND: E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence; S = unconditional stability; CS = conditional stability; U = unstable
Table 12. SOLUTION PROCEDURES AND RESULTS USING CLASSICAL LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAINS TWO AND THREE

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Interp. Strat.</th>
<th>B. C.'s</th>
<th>(u')₀</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU* (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>17</td>
<td>0.38</td>
<td>0.0678</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E N (1)</td>
<td>C</td>
<td>S</td>
<td>77</td>
<td>0.34</td>
<td>0.2262</td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>17</td>
<td>0.25</td>
<td>0.0458</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E N (1)</td>
<td>C</td>
<td>S</td>
<td>76</td>
<td>0.29</td>
<td>0.2239</td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>12</td>
<td>0.36</td>
<td>0.0423</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E N (1)</td>
<td>C</td>
<td>S</td>
<td>18</td>
<td>0.34</td>
<td>0.0572</td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>12</td>
<td>0.23</td>
<td>0.0298</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E N (1)</td>
<td>C</td>
<td>S</td>
<td>18</td>
<td>0.25</td>
<td>0.0465</td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>C'</td>
<td>S</td>
<td>200+</td>
<td>0.60</td>
<td>2.0412</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E N (1)</td>
<td>C'</td>
<td>S</td>
<td>200+</td>
<td>0.96</td>
<td>3.2696</td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>C'</td>
<td>S</td>
<td>200+</td>
<td>0.43</td>
<td>1.4574</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E N (1)</td>
<td>C'</td>
<td>S</td>
<td>200+</td>
<td>0.65</td>
<td>2.2315</td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>13</td>
<td>0.60</td>
<td>0.1365</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E N (1)</td>
<td>C</td>
<td>S</td>
<td>10</td>
<td>0.65</td>
<td>0.0986</td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>13</td>
<td>0.42</td>
<td>0.0777</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E N (1)</td>
<td>C</td>
<td>S</td>
<td>13</td>
<td>0.42</td>
<td>0.0863</td>
<td></td>
</tr>
</tbody>
</table>

LEGEND: E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence S = unconditional stability; CS = conditional stability; U = unstable

NOTES: a - Convergence was imminent and obtained within another 50 iterations (1) - See equation (4.10)
Table 13. SOLUTION PROCEDURES AND RESULTS USING CLASSICAL LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN ONE

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Interp. Strat.</th>
<th>Iter. Strat.</th>
<th>B. C.'s</th>
<th>( (u'_1) )</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU* (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td></td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>9</td>
<td>6.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>8</td>
<td>6.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>10</td>
<td>6.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C°</td>
<td>S</td>
<td>17</td>
<td>-----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Linear</td>
<td></td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>9</td>
<td>4.61</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>8</td>
<td>4.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>10</td>
<td>4.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C°</td>
<td>S</td>
<td>200+</td>
<td>-----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td></td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>13</td>
<td>6.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>11</td>
<td>7.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>15</td>
<td>6.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C°</td>
<td>S</td>
<td>17</td>
<td>-----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Linear</td>
<td></td>
<td>E</td>
<td>E</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>12</td>
<td>4.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>11</td>
<td>4.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>E</td>
<td>R</td>
<td>C</td>
<td>S</td>
<td>15</td>
<td>4.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
<td>N</td>
<td>L</td>
<td>C°</td>
<td>S</td>
<td>17</td>
<td>-----</td>
</tr>
</tbody>
</table>

LEGEND: E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence S = unconditional stability; CS = conditional stability; U = unstable

NOTES:  
a - Converges to another solution of the differential equation  
b - Convergence to another solution of the differential equation was imminent
Table 14. SOLUTION PROCEDURES AND RESULTS USING CLASSICAL LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAINS TWO AND THREE

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Interp. Strat.</th>
<th>B. C.'s</th>
<th>(u'_s)</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>C^*</td>
<td>S</td>
<td>87</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>C^*</td>
<td>S</td>
<td>95</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>C^*</td>
<td>S</td>
<td>40</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>C^*</td>
<td>S</td>
<td>35</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E E (1) (2)</td>
<td>----</td>
<td>200+</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E (1) (2)</td>
<td>----</td>
<td>200+</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E E (1)</td>
<td>C^*</td>
<td>S</td>
<td>117</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Linear</td>
<td>E E (1)</td>
<td>C^*</td>
<td>S</td>
<td>43</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

LEGEND: E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence S = unconditional stability; CS = conditional stability; U = unstable

NOTES: a - Converges to a nonsolution of the differential equation
(1) - See equation (4.12)
(2) - Nonconvergent solution

Three general observations can be made from the results shown in these tables. The first is that, for solution procedures which yielded valid converged approximations, the number of iterations to convergence was strictly a function of the initial and subsequent iteration strategies and independent of the interpolation strategy. Additionally, the number of iterations to convergence was independent of the number of elements utilized. The accuracy of the solution, on the other hand, was strictly a function
of the interpolation strategy and the number of degrees of freedom used in the approximation.

The second observation is that the classical linearization technique provided valid approximations of equation (4.1) over domain three, but was not able to solve equation (4.2) over domain two, despite the fact both of these equations over these respective domains have the same function order. The main reason for this lies in the fact that the solution of equation (4.2) over domain two is changing more rapidly than that of equation (4.1) over domain three. Thus, it appears that the classical linearization technique can provide valid approximations of nonlinear differential equations that have a function order of two, but whose rate of change over the given domain is not 'excessively' large.

Last of all, a general comment on the magnitude of the average percent difference values in Table 13 is in order. Though these values may seem large in relation to the percent difference values in Table 11 and Table 12, it must be remembered that these are average values. In this particular situation, the values of the dependent variable over $0 < x < 0.2$ are quite small, i.e., on the order of 0.08 and less. Thus, an approximate solution at one node which is in absolute error of only 0.0023 from an exact solution value of 0.0100 yields a 23 percent error. Therefore, the larger average percent difference values stem from minor errors in the approximations at those nodes where the magnitude of the dependent variable is very small. The overall approximate solutions provided by these procedures is much better than the average percent difference values indicate as the actual percent difference values over a majority of the domain was less than 0.5 percent. The effect of each problem parameter on the performance of the various solution procedures which provided convergent solutions of equations (4.1) and (4.2) is now examined.

b. Boundary Conditions

The use of an E-E boundary condition combination provided convergence with less iterations than the E-N combination in all instances except for the solution of equation (4.1) over domain three. No explanation for this behavior could be determined and it remains an open question requiring further investigation. The accuracy of the approximations within a specific combination of iteration and interpolation strategies was not greatly affected by the boundary conditions, as the average percent difference values provided by the E-E and E-N combinations were similar.
c. Initial Iteration Strategy

In solving equations (4.1) and (4.2) over domain one, the initial iteration strategy is developed utilizing the prescribed essential boundary condition(s) as described in III.C.1. An initial iteration strategy based on the average value of the two essential boundary conditions consistently provided convergence with slightly less iterations than the other two strategies.

Over domains two and three, only convergent solutions of equation (4.1) were obtained; no valid approximate solutions were obtained of equation (4.2) over these domains. Initial iteration strategies defined by equations (4.9) and (4.10) were both utilized in the analysis over domain two to determine which was most effective. Equations (4.9) and (4.10) provided nearly the same numerical values in the approximation, but the use of equation (4.10) consistently enabled the solution procedure to converge with less iterations. This behavior requires further research and remains an open question, as it was felt that equation (4.9), which accounted for the dominance of the $u''$ term over the first part of the domain, should have provided better initial iteration values. In the solution of equation (4.2) over domains two and three, both initial iteration strategies led to convergent/nonconvergent nonsolutions of the differential equation.

d. Subsequent Iteration Strategy

Over domain one for both equations (4.1) and (4.2), the use of the previous value iteration strategy consistently provided convergence with slightly less iterations than the average value strategy. The one exception to this was when an E-N boundary condition combination was specified. In that case, both strategies yielded convergence with the same number of iterations. Over domains two and three, the average value strategy provided for convergence with significantly less iterations than the previous value method. This fact is especially evident in the solution of equation (4.1) over domain three where the previous value strategy could not converge within 200 iterations while the average value method provided convergence in 13 iterations or less. These two observations tend to indicate that the classical linearization strategy provides for monotonic convergence when the function order is one or less and oscillatory convergence when the function order is greater than one.

e. Interpolation Strategy

The linear interpolation strategy consistently provided more accurate approximations than the midpoint strategy as indicated by the lower average percent difference values in Table 11 through Table 13. The main reason for this relates to the
fact that the solutions of equations (4.1) and (4.2) are polynomial functions of $x$. The number of elements utilized in each of the solution procedures was sufficient enough to make the solution curve almost linear over each element. Thus, the linear approximation induced less error in the evaluation of the Galerkin linearization matrix integral than the midpoint method. For those cases where only a few elements were utilized, the midpoint method occasionally provided a more accurate approximation than the linear strategy. But, the overall accuracy of the approximation was poor due to the decrease in the number of system DOF.

f. Overall Performance

When the function order of the differential equation solution is one or less, a solution procedure which utilizes a previous value iteration and linear interpolation strategy tends to provide the most efficient solutions based on the average percent difference values in Table 11 and CPU' values in Table 13. The CPU' values in Table 11 should be evaluated with caution as the CPU times upon which they are based are at or below the accuracy level of the clock subroutine which is $\pm 0.03$ seconds. When the function order is greater than one, an average value iteration and linear interpolation strategy combination provide the most efficient approximation as indicated by the CPU' values in Table 12.

3. Conclusions

The classical linearization strategy can generally provide approximations of second order nonlinear differential equations when the function order is two or less and the rate of change of the dependent variable is not extremely large. Provided these conditions are met, an average value iteration strategy combined with a linear interpolation strategy should provide an efficient, valid approximation of the differential equation. If the function order is later found to be one or less, the use of a previous value iteration strategy should result in similar numerical results and converge using slightly less iterations. In either case, both the E-E and E-N boundary condition combinations can be accommodated, although the use of an E-E combination is preferred.

D. QUASILINEARIZATION

1. Problem Formulation

Quasilinearization transforms the nonlinear $u^2$ term into a linear term using the relation given by equation (3.4.a), where $q(u') = (u')^2$ and $q'(u') = 2u'$. Substitution of these functions into equation (3.4.a) yields
\[ u^2 \approx \mathcal{L}u = 2u^* u + (u^*)^2 - u^*(2u^*) \]
\[ = 2u^* u - (u^*)^2 \]  \hspace{1cm} (4.22)

where \( u^* \) is determined as outlined in III.C. Substitution of equation (4.22) into equations (4.1) and (4.2) yields a linear differential equation of the form
\[ u'' \pm 2u^* u = f(x) \pm (u^*)^2 \quad x \in D \]  \hspace{1cm} (4.23)

where both \(-\) signs are for equation (4.1), both \(+\) signs are for equation (4.2) and \( f(x) \) is the respective excitation function in each equation. The Galerkin FEM formulation process transforms equation (4.23) into
\[ G(G^T)u^b - \int_D G'(G^T)dx u \pm 2\int_D GG^T u^* dx u = \int_D Gf(x)dx \pm \int_D G(u^*)^2 dx \]  \hspace{1cm} (4.24)

The first two terms on the left side of equation (4.24) yield \( B - Au \) where the \( B \) vector is present only when a natural boundary condition is prescribed. The third term on the left side of the equation is the linearization matrix integral which was evaluated in IV.C.1 and yields \( L' \). Both integrals on the right side of equation (4.24), the excitation and linearization vector, were evaluated in IV.B.1 and yield \( F \) and \( F' \), respectively.

Substitution of the matrix and vector equivalents for each term into equation (4.24) yields a system of equations given by
\[ B - Au + 2L'u = F + F' \]  \hspace{1cm} (4.25.a)

where the \( \pm \) signs in equation (4.24) are incorporated into \( L' \) and \( F' \). Combining \( F \) and \( B \) to yield \( F_m \), the system of linear algebraic equations takes the final form
\[ (-A + 2L_{i-1}^*)u_i = F_m + F_{i-1}' \]  \hspace{1cm} (4.25.b)

Equation (4.25.b) is solved iteratively for \( u \), with both \( L' \) and \( F' \) changing after each iteration, where \( i \) is the iteration counter.

2. Results

a. General

Forty eight different solution procedures were evaluated in solving equations (4.1) and (4.2) over domain one while there were twenty four procedures available for approximating each equation over both domains two and three. The
FORTRAN programs for quasilinearization are provided in Appendix G. The different strategies utilized in each solution procedure and the corresponding results in approximating the solution of equation (4.1) are provided in Table 15 on page 65 through Table 19 on page 69. For equation (4.2), this information is provided in Table 20 on page 70 through Table 25 on page 75. The number of elements utilized in each solution procedure is given in Table 5 on page 47.

Four general observations can be made about the quasilinearization strategy based on these results. The first is that this linearization technique provided valid approximations of both equations (4.1) and (4.2) over all domains, although some individual solution procedures were much more accurate than others. The difference in performance between the various solution procedures became noticeable as the function order approached three or more and the solution function gradient became large, as in equation (4.2) over domains two and three.

The second point is that this linearization technique provides for convergence with a minimum of iterations due to its quadratic rate of convergence [Ref. 1: pp. 38-40]. Some of the solution procedures utilized to solve equations (4.1) and (4.2) over domain three converged in just two iterations and thus did not yield as accurate solutions as was anticipated. In order to determine if these solution procedures could provide more accurate approximations, the convergence criterion was changed from .0001 to .0000001, to allow for slightly more iterations. The effect of changing the convergence criterion for those applicable solution procedures is noted in Table 18, Table 19, Table 24, and Table 25.

The third point, which also relates to convergence, is that the number of iterations required for convergence is not always a function of the overall iteration strategy as it was for the previous two linearization strategies. When the function order is one or slightly over two, as in equation (4.1) over domains one and two, and equation (4.2) over domain one, the number of iterations is dictated by the iteration strategies utilized. For those situations where the function order is almost three or more, the number of iterations required for convergence also appears to be affected by the specific combination of interpolation strategies used in the solution procedure.
Table 15. **SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN ONE**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Iter. Strat.</th>
<th>Interp. Strat.</th>
<th>B. C.'s</th>
<th>((u'_c))</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>Midpt.</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.00</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E M</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.00</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E R</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.00</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E N L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>5</td>
<td>0.00</td>
<td>0.0000</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>1/4-3/4</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.60</td>
<td>0.0100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E M</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.60</td>
<td>0.0080</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E R</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.60</td>
<td>0.0100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E N L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>5</td>
<td>0.16</td>
<td>0.0023</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>Linear</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.32</td>
<td>0.0053</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E M</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.32</td>
<td>0.0064</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E R</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.32</td>
<td>0.0064</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E R</td>
<td>C</td>
<td>S</td>
<td></td>
<td>5</td>
<td>0.24</td>
<td>0.0057</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>Midpt.</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.16</td>
<td>0.0021</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E M</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.16</td>
<td>0.0021</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E R</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.16</td>
<td>0.0032</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E N L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>5</td>
<td>0.61</td>
<td>0.0122</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>1/4-3/4</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.44</td>
<td>0.0059</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E M</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.44</td>
<td>0.0059</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E R</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.44</td>
<td>0.0044</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E N L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>5</td>
<td>0.54</td>
<td>0.0108</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>Linear</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.16</td>
<td>0.0027</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E M</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.16</td>
<td>0.0027</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E E R</td>
<td>C</td>
<td>S</td>
<td></td>
<td>4</td>
<td>0.16</td>
<td>0.0027</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E N L</td>
<td>C</td>
<td>S</td>
<td></td>
<td>5</td>
<td>0.02</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

**LEGEND:** E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence S = unconditional stability; CS = conditional stability; U = unstable
Table 16. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN ONE (CONT.)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iter. Strat.</td>
<td>Interp. Strat.</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Midpt. Midpt.</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Midpt. 1/4-3/4</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Midpt. Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Midpt.</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear 1/4-3/4</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td>Avg. Value</td>
<td>Linear Linear</td>
</tr>
</tbody>
</table>

LEGEND: E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence S = unconditional stability; CS = conditional stability; U = unstable
Table 17. **SOLUTION PROCEDURES AND RESULTS USING QUASILINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN TWO**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Interp. Strat.</th>
<th>B. C.'s</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU* (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Prev. Value</td>
<td>Midpt. Midpt.</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>4</td>
<td>0.00</td>
<td>0.0001</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Prev. Value</td>
<td>Midpt. 1/4-3/4</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>4</td>
<td>0.97</td>
<td>0.0547</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Prev. Value</td>
<td>Midpt. Linear</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>4</td>
<td>0.97</td>
<td>0.0547</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Linear Value</td>
<td>Midpt. Midpt.</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>4</td>
<td>0.26</td>
<td>0.0130</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Linear Value</td>
<td>Midpt. 1/4-3/4</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>4</td>
<td>0.26</td>
<td>0.0130</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Linear Value</td>
<td>Linear Linear</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>4</td>
<td>0.27</td>
<td>0.0137</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Avg. Value</td>
<td>Midpt. Midpt.</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>6</td>
<td>0.00</td>
<td>0.0001</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Avg. Value</td>
<td>Midpt. 1/4-3/4</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>5</td>
<td>0.97</td>
<td>0.0610</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Avg. Value</td>
<td>Linear Linear</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>5</td>
<td>0.97</td>
<td>0.0610</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Avg. Value</td>
<td>Midpt. Linear</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>5</td>
<td>0.51</td>
<td>0.0308</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Avg. Value</td>
<td>Linear Midpt.</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>6</td>
<td>0.26</td>
<td>0.0198</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Avg. Value</td>
<td>Linear 1/4-3/4</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>5</td>
<td>0.71</td>
<td>0.0448</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 2$</td>
<td>Avg. Value</td>
<td>Linear Linear</td>
<td>E E (1)</td>
<td>C</td>
<td>S</td>
<td>5</td>
<td>0.26</td>
<td>0.0163</td>
</tr>
</tbody>
</table>

**LEGEND:**
- E = essential boundary condition;
- N = natural boundary condition;
- L = left essential boundary condition;
- R = right essential boundary condition;
- M = average of L and R;
- C = convergence;
- D = divergence;
- S = unconditional stability;
- CS = conditional stability;
- U = unstable

**NOTES:** (1) - See equation (4.10)
<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iter. Strat.</td>
<td>Interp. Strat.</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 5$</td>
<td>Prev. Value</td>
<td>Midpt. Midpt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prev. Value</td>
<td>Midpt. 1.4-3/4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prev. Value</td>
<td>Midpt. Linear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prev. Value</td>
<td>Linear Midpt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prev. Value</td>
<td>Linear 1/4-3/4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prev. Value</td>
<td>Linear Linear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LEGEND:**
- E = essential boundary condition; N = natural boundary condition;
- L = left essential boundary condition; R = right essential boundary condition;
- M = average of L and R; C = convergence; D = divergence S = unconditional stability;
- CS = conditional stability; U = unstable

**NOTES:**
- a - Convergence criterion changed to 0.0000001 in order to allow for additional iterations and a more efficient approximation
- b - Convergence criterion kept at .0001 as use of tighter criterion in note a leads to a less efficient or nonconvergent approximation.
- (1) - See equation (4.10)
Table 19. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.1) OVER DOMAIN THREE (CONT.)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>B. C.'s</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E</td>
<td>E</td>
<td>(1)</td>
<td>C*</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LEGEND:  E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence S = unconditional stability; CS = conditional stability; U = unstable

NOTES:  

- Convergence criterion changed to 0.0000001 in order to allow for additional iterations and a more efficient approximation.
- Convergence criterion kept at .0001 as use of tighter criterion in note a leads to a less efficient or nonconvergent approximation.
- Tightening of convergence criterion had no effect.

(1) - See equation (4.10)
Table 20. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN ONE

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Interp. Strat.</th>
<th>B. C.'s</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU* (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Midpt.</td>
<td>E E L C S 5</td>
<td>1.05</td>
<td>0.063</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Midpt. 1/4-3/4</td>
<td>E E M C S 5</td>
<td>11</td>
<td>----</td>
<td>----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Midpt. Linear</td>
<td>E E L C S 5</td>
<td>16.1</td>
<td>1.070</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E L C S 5</td>
<td>9.28</td>
<td>0.648</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E L C S 5</td>
<td>3.61</td>
<td>0.216</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E L C S 5</td>
<td>11</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E L C S 5</td>
<td>12</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E L C S 5</td>
<td>5</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E L C S 5</td>
<td>6</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 &lt; x &lt; 1$</td>
<td>Prev. Value</td>
<td>Linear</td>
<td>E E L C S 5</td>
<td>4.64</td>
<td>0.308</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LEGEND: E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence S = unconditional stability; CS = conditional stability, U = unstable

NOTES: a - Converges to another solution of the differential equation
<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>B. C.'s</th>
<th>$(u'_t)_0$</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU' (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value</td>
<td>Midpt.</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td>6</td>
<td>1.07</td>
<td>0.0852</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
<td>E E M</td>
<td>C'</td>
<td>S</td>
<td>15</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
<td>E E R</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
<td>E N L</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value Midpt.</td>
<td>Linear</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td>6</td>
<td>16.1</td>
<td>1.3399</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E E M</td>
<td>C'</td>
<td>S</td>
<td>16</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E E R</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E N L</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value Midpt.</td>
<td>Linear</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>3.61</td>
<td>0.3486</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E E M</td>
<td>C'</td>
<td>S</td>
<td>15</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E E R</td>
<td>C'</td>
<td>S</td>
<td>6</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E N L</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value Linear</td>
<td>Midpt.</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>11.4</td>
<td>0.9091</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
<td>E E M</td>
<td>C'</td>
<td>S</td>
<td>16</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
<td>E E R</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
<td>E N L</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0 &lt; x &lt; 1</td>
<td>Avg. Value Linear</td>
<td>Linear</td>
<td>E E L</td>
<td>C</td>
<td>S</td>
<td>7</td>
<td>4.60</td>
<td>0.4440</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E E M</td>
<td>C'</td>
<td>S</td>
<td>15</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E E R</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
<td>E N L</td>
<td>C'</td>
<td>S</td>
<td>7</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

**LEGEND:**
- E = essential boundary condition; N = natural boundary condition;
- L = left essential boundary condition; R = right essential boundary condition;
- M = average of L and R; C = convergence; D = divergence; S = unconditional stability;
- CS = conditional stability; U = unstable

**NOTES:**
- a - Converges to another solution of the differential equation
Table 22. SOLUTION PROCEDURES AND RESULTS USING QUASILINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN TWO

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iter. Strat.</td>
<td>Interp. Strat.</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Midpt.</td>
<td>1/4-3/4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Linear</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Midpt.</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Linear</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/4-3/4</td>
</tr>
<tr>
<td>0 &lt; x &lt; 2</td>
<td>Prev. Value</td>
<td>Linear</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
</tr>
</tbody>
</table>

**LEGEND:**
- E = essential boundary condition; N = natural boundary condition;
- L = left essential boundary condition; R = right essential boundary condition;
- M = average of L and R; C = convergence; D = divergence S = unconditional stability;
- CS = conditional stability; U = unstable

**NOTES:**
- a - Majority of error occurs over 0 < x < 0.2
- b - Divergence results when certain number of elements are utilized
- c - Majority of error occurs over 0 < x < 0.6
- d - Provides a reasonable approximation over 1 < x < 2
- e - Majority of error occurs over 0 < x < 0.5
- f - Majority of error occurs over 0 < x < 0.3
- (1) - See equation (4.11)
Table 23. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN TWO (CONT.)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Avg. Midpt.</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td>(0 &lt; x &lt; 2)</td>
<td>Avg. Value</td>
<td>Linear</td>
</tr>
</tbody>
</table>

**LEGEND:** E = essential boundary condition; N = natural boundary condition; L = left essential boundary condition; R = right essential boundary condition; M = average of L and R; C = convergence; D = divergence S = unconditional stability; CS = conditional stability; U = unstable

**NOTES:**
- a - Majority of error occurs over \(0 < x < 0.2\)
- b - Majority of error occurs over \(0 < x < 0.6\)
- c - Divergence results when certain number of elements are utilized
- d - Provides a reasonable approximation over \(1 < x < 2\)
- e - Majority of error occurs over \(0 < x < 0.5\)
- f - Majority of error occurs over \(0 < x < 0.3\)
- (1) - See equation (4.11)
### Table 24. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN THREE

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interp. Strat.</td>
<td>B. C.'s</td>
</tr>
<tr>
<td></td>
<td>Iter. Strat.</td>
<td>Lt</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 5$</td>
<td>Prev. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 5$</td>
<td>Prev. Value</td>
<td>Midpt.</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 5$</td>
<td>Prev. Value</td>
<td>1/4-3/4</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 5$</td>
<td>Prev. Value</td>
<td>Linear</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 5$</td>
<td>Prev. Value</td>
<td>Linear</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 5$</td>
<td>Prev. Value</td>
<td>Linear</td>
</tr>
<tr>
<td>$0 &lt; x &lt; 5$</td>
<td>Prev. Value</td>
<td>Linear</td>
</tr>
</tbody>
</table>

**LEGEND:**
- **E** = essential boundary condition;
- **N** = natural boundary condition;
- **L** = left essential boundary condition;
- **R** = right essential boundary condition;
- **M** = average of L and R;
- **C** = convergence;
- **D** = divergence;
- **S** = unconditional stability;
- **CS** = conditional stability;
- **U** = unstable.

**NOTES:**
- **a** - Convergence criterion changed to 0.0000001 in order to allow for additional iterations and a more efficient approximation.
- **b** - Convergence criterion kept at .0001 as use of tighter criterion in note **a** leads to a less efficient or nonconvergent approximation.
- **c** - Divergence results when certain number of elements are used.
- **d** - Provided an adequate approximation over 2.5 < $x$ < 5.0.
- **e** - Majority of error occurs over 0 < $x$ < 0.5 where the values of $u < 2.0$.
- **(1)** - See equation (4.12).
### Table 25. SOLUTION PROCEDURES AND RESULTS USING QUASI-LINEARIZATION TO SOLVE EQUATION (4.2) OVER DOMAIN THREE (CONT.)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Solution Procedure</th>
<th>B. C.'s</th>
<th>Conv.</th>
<th>Stab.</th>
<th># of Iter.</th>
<th>% Dif</th>
<th>CPU* (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt. Midpt.</td>
<td>E E (1)</td>
<td>C*</td>
<td>S</td>
<td>4</td>
<td>11.1*</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt. 1/4-3/4</td>
<td>E E (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Midpt. Linear</td>
<td>E E (1)</td>
<td>D</td>
<td>U</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Linear Midpt.</td>
<td>E E (1)</td>
<td>C*</td>
<td>S</td>
<td>2</td>
<td>1143*</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Linear 1/4-3/4</td>
<td>E E (1)</td>
<td>C*</td>
<td>S</td>
<td>2</td>
<td>672*</td>
</tr>
<tr>
<td>0 &lt; x &lt; 5</td>
<td>Avg. Value</td>
<td>Linear Linear</td>
<td>E E (1)</td>
<td>C*</td>
<td>S</td>
<td>31</td>
<td>1.46</td>
</tr>
</tbody>
</table>

**LEGEND:**
- E = essential boundary condition; N = natural boundary condition;
- L = left essential boundary condition; R = right essential boundary condition;
- M = average of L and R; C = convergence; D = divergence; S = unconditional stability;
- CS = conditional stability; U = unstable

**NOTES:**
- a - Convergence criterion changed to 0.0000001 in order to allow for additional iterations and a more efficient approximation
- b - Majority of error occurs over 0 < x < 0.5 where the values of u < 2.0
- c - Convergence criterion kept at .0001 as use of tighter criterion in note a leads to a less efficient or nonconvergent approximation.
- d - Provided an adequate approximation over 2.0 < x < 5.0
- e - Provided an adequate approximation over 1.5 < x < 5.0
- (1) - See equation (4.12)

Last of all, a general comment is required with respect to the average percent difference values in Table 22 through Table 25. As previously noted in IV.C.3, the value of the dependent variable for equation (4.2) is very small over the first part of the domain. Thus, errors in the approximation which are on the same order of magnitude as the value of the dependent variable result in large percent difference values. Therefore, average percent difference values of three or more are amplified with a superscript 75.
which advises the reader as to the true accuracy of the approximation. The term 'a majority of error' in each of these accompanying notes means that the percent difference at each node is 10 percent or more over the indicated domain. The effect of each parameter on the overall performance of the various solution procedures is now evaluated.

b. Boundary Conditions

Except in a few isolated instances, the accuracy and number of iterations to convergence for a specific combination of iteration and interpolation strategies was unaffected by the boundary condition combination utilized. The one exception to this is shown in Table 20 and Table 21 where the use of a E-N combination caused the solution procedure to converge to a second solution of the differential equation. But, when the initial iteration strategy was changed from using the value of the left boundary condition to the strategy defined by equation (4.12), valid approximations of the solution \( u = 10x^3 \) were obtained by all procedures using an E-N combination. Thus, it appears that this linearization technique is quite favorable to both E-E and E-N type boundary value problems, provided that a valid initial iteration strategy is utilized.

c. Initial Iteration Strategy

Over domain one for equation (4.1), use of the three initial iteration strategies described in III.C.1 provided convergence with nearly the same number of iterations. Over the same domain for equation (4.2), only the use of the left essential boundary condition as values for the initial iteration strategy yielded a convergent approximation of the original exact solution. The other two strategies, when utilized with an E-E boundary condition combination, converged to a second solution of equation (4.2). The reason that the use of the left essential boundary condition for the initial iteration strategy outperformed the other two strategies is because it has a value of zero. With \( (u')_b = 0 \), the first iteration solves the differential equation neglecting the effect of the nonlinear term, as \( F' \) and \( L' \) are zero. Thus, the values of \( (u'_b) \), utilized in the next iteration are very close to values of \( (u'_b)_0 \) that would have been obtained by neglecting the \( u^2 \) term and integrating the differential equation twice with the imposition of boundary conditions. From Figure 17 on page 77, it can be seen that the \( u'' \) term dominates over a majority of the domain. Thus, if equation (4.2) had been analyzed as order two over this domain and the initial iteration strategy developed by neglecting the \( u^2 \) term, similar results using one less iteration would probably have been obtained.

For equation (4.1) over domain two, the initial iteration strategies defined by equations (4.9) and (4.10) were both used in the analysis to determine which was most effective. As in the classical linearization results, they both provided almost identical
Figure 17. Dominance of Terms in Equation (4.2) Over Domain One
numerical approximations, and the use of equation (4.10) again resulted in convergence with less iterations. Likewise, equations (4.11) and (4.12) were both utilized as initial iteration strategies for equation (4.2) over domain two. They both provided the same accuracy in the approximation, but the use of equation (4.11), which divides the dominance equally over the domain, converged using less iterations as was originally expected. The use of equations (4.10) and (4.12) for the initial iteration strategies of equations (4.1) and (4.2), respectively, over domain three, lead to accurate approximations for those solution procedures which did converge.

d. Subsequent Iteration Strategy

The use of either the previous value or average value iteration strategy, in general, had no effect on the number of iterations required for convergence. This is most likely due to the quadratic rate of convergence guaranteed by the quasilinearization method, as previously mentioned in IV.D.2.a.

e. Interpolation Strategy

The quasilinearization method requires the use of two interpolation strategies; one for the linearization vector and one for the linearization matrix. In the interpolation strategy column of Table 15 through Table 25, the upper strategy is for the linearization matrix and the lower one is for the linearization vector. A general trend in the accuracy provided by the various combinations of interpolation strategies is evident. The different combinations are ranked from least to most accurate in the following list, where the first strategy indicated is for the linearization matrix and the second is for the linearization vector.

- Midpoint;1/4-3/4
- Linear;1/4-3/4
- Midpoint;Linear
- Linear;Midpoint
- Linear;Linear
- Midpoint;Midpoint

Two conclusions can be drawn from the above list. First is that the most accurate interpolation strategies utilize the same interpolation technique for both linearization integrals. Thus, if a 1/4-3/4 interpolation strategy for the linearization matrix had been developed, there is a good possibility that an overall 1/4-3/4;1/4-3/4 interpolation strategy would have provided accurate approximations. Second, the least
refined interpolation strategy, namely midpoint;midpoint, provides the most accurate approximations. This result is not very surprising, as in many situations, the simplest method provides the best results.

f. Overall Performance

In almost all cases, the solution procedure consisting of a previous iteration and a midpoint;midpoint interpolation strategy provided the most efficient approximations regardless of the function order of the equation or the boundary conditions imposed. The only case where this procedure faltered slightly was in approximating equation (4.2) over domain two. But, as shown in note a of Table 22 and Table 23, it only had a problem approximating the solution over that part of the domain where \( u < 0.1 \). The solution procedure utilizing a previous value iteration and a linear;linear interpolation strategy was not quite as efficient as the above solution procedure, but performed in an acceptable manner.

3. Conclusions

Quasilinearization provides a viable method of approximating nonlinear differential equations that contain the \( u^2 \) term, regardless of the function order of the equation and the nature of the boundary conditions imposed. The use of a previous value iteration and either a midpoint;midpoint or linear;linear interpolation strategy should provide an accurate approximation with a minimum number of iterations, provided that the initial iteration strategy is adeptly chosen. The actual shape of the solution curve and the discretization invoked, i.e., the number of elements, are the two factors most likely to determine which interpolation technique provides the more accurate approximation.

E. FINAL REMARKS

An overall solution procedure involving quasilinearization combined with a previous value iteration and either a midpoint;midpoint or linear;linear interpolation strategy provides excellent approximations of second order, nonlinear, one dimensional, differential equations which contain the \( u^2 \) nonlinear term. As the \( u^2 \) term has a more nonlinear nature than some of the other nonlinear terms encountered, i.e., \( u''u, (u')^2 \), etc.; it is felt that this solution procedure should provide viable approximations of many nonlinear, second order differential equations. It cannot be overemphasized that the success or failure of the above solution procedure depends greatly on the initial iteration strategy developed. Thus, utilization of this solution procedure requires that the user have an in-depth understanding of the physics involved in the system being analyzed.
This research has provided a fundamental baseline for the future investigation of techniques for solving nonlinear differential equations. The following steps provide a logical progression for determining the actual capabilities of a Galerkin FEM solution procedure utilizing quasilinearization.

- Conduct an analysis of second order, one dimensional, nonlinear, differential equations which contain nonlinear terms other than $u^2$. These nonlinear equations should be of an engineering nature for which experimental data exists to allow for a confirmation of the results developed by the mathematical model.

- Investigate the ability of this solution method to solve one dimensional, nonlinear, fourth order differential equations. This requires some modification of the interpolation strategies as the Galerkin FEM must utilize cubic shape functions for developing a fourth order differential equation approximation.

- Extrapolate the concepts and principals developed by this and future research to the solution of two dimensional, second and fourth order, nonlinear differential equations.
APPENDIX A. FORCING FUNCTION FORMULATION STRATEGIES

On an elemental level, $\int_0^G(6x)dx$ becomes $\int_0^G(6(\alpha + \xi))d\xi$ where

- $\xi$ is the local element coordinate, $0 \leq \xi \leq \ell$,
- $\alpha$ is the sum of all element lengths prior to the element being evaluated. For equal length elements, $\alpha = (i - 1)\ell$ where $i$ is the element number.

Midpoint Lumped Approximation

This method evaluates $6(\alpha + \xi)$ at the midpoint of the element, $\xi = \frac{\ell}{2}$ and brings it outside the integral as a constant yielding

$$f = 6\left(\alpha + \frac{\ell}{2}\right) \int_0^\ell \left[1 - \frac{\xi}{\ell}\right] d\xi$$

$$= 3\ell \left[\alpha + \frac{\ell}{2}\right]$$

$$= 3\ell \left[\frac{\alpha + \ell}{2}\right]$$

(A.1)

Quarter/Three Quarter Lumped Approximation

This method takes $6(\alpha + \xi)$ inside the shape function vector yielding

$$f = \int_0^\ell \left[(6(\alpha + \xi))\left(1 - \frac{\xi}{\ell}\right)\right] d\xi$$

(A.2a)

The first $6(\alpha + \xi)$ term is evaluated at $\xi = \frac{\ell}{4}$ and the second term at $\xi = \frac{3\ell}{4}$, yielding
Consistent Evaluation

This method calculates the exact value of the Galerkin excitation integral yielding

\[ f = \int_{0}^{l_e} \left[ \frac{6(\alpha + \frac{l_e}{4})(1 - \frac{\xi}{l_e})}{6(\alpha + \frac{3l_e}{4})(\frac{\xi}{l_e})} \right] d\xi \]

\[ = 3l_e \left[ \frac{\alpha + \frac{l_e}{4}}{\alpha + \frac{3l_e}{4}} \right] \]  

\[ (A.2.b) \]

\[ f = 6 \int_{0}^{l_e} \left[ \frac{1 - \frac{\xi}{l_e}}{\frac{\xi}{l_e}} \right] (\alpha + \xi)d\xi \]

\[ = 6 \int_{0}^{l_e} \left[ \alpha - \frac{\alpha \xi}{l_e} + \xi - \frac{\xi^2}{l_e} \right] d\xi \]

\[ = 6 \int_{0}^{l_e} \left[ \frac{\alpha \xi^2}{2l_e} + \frac{\xi^3}{3l_e} \right] d\xi \]

\[ = 6 \left[ \frac{\alpha l_e}{2} + \frac{l_e^2}{3} \right] \]

\[ = \left[ 3\alpha l_e + \frac{l_e^2}{2} \right] \]

\[ = \left[ 3\alpha l_e + 2l_e^2 \right] \]  

\[ (A.3) \]
APPENDIX B. PROGRAM LISTINGS AND RESULTS FOR THE LINEAR APPLICATION OF THE GALERKIN FEM

*                                                                                             *
*   PROGRAM LIN1                                                                                 *
*                                                                                             *
*   THIS PROGRAM SOLVES THE DIFFERENTIAL EQUATION U'' = 2., 0<X<2                                   *
*   U(0)=0; U'(2)=4 WITH UEXACT=X**2.                                                            *
*                                                                                             *
*                                                                                             *
110 COMMON A(100,100),FT(100),U(100),UEXT(100),UDIF(100),COORD(100),                           :
      ELEN,ICORR(100,2),NEL,NSNP                                                             :
C                                                                                             :
C INPUT NUMBER OF ELEMENTS AND TOTAL LENGTH OF DOMAIN                                          :
120 PRINT*,"INPUT NUMBER OF ELEMENTS DESIRED AND TOTAL LENGTH. '                             :
130 READ(6,*) NEL,TLEN                                                                      :
C                                                                                             :
C CALCULATE NUMBER OF NODAL POINTS AND DEFINE LEFT BOUNDARY OF                                 :
C DOMAIN                                                                                  :
135 NSNP=NEL+1                                                                             :
150 COORD(1)=0.                                                                           :
C                                                                                             :
C DETERMINE ELEMENT LENGTH, LOCAL TO GLOBAL NODAL POINT                                      :
C CORRESPONDENCE, AND X-COORDINATE OF EACH NODAL POINT                                        :
155 ELEN=TLEN/ FLOAT(NEL)                                                                   :
160 DO 169 IEL=1,NEL                                                                        :
162 ICORR(IEL,1)=IEL                                                                      :
163 ICORR(IEL,2)=IEL+1                                                                      :
164 COORD(IEL+1)=COORD(IEL)+ELEN                                                          :
169 CONTINUE                                                                              :
C                                                                                             :
C CALL SUBROUTINE SYM1A TO DETERMINE A MATRIX AND F VECTOR AND SOLVE                           :
C THE LINEAR SYSTEM OF EQUATIONS AU = F                                                      :
170 CALL SYM1A                                                                               :
C                                                                                             :
C CALL SUBROUTINE UX2EXT TO DETERMINE EXACT SOLUTION                                         :
190 CALL UX2EXT                                                                             :
C                                                                                             :
C CALL SUBROUTINE OUTLIN TO OUTPUT RESULTS                                                   :
200 CALL OUTLIN                                                                            :
210 END                                                                                    

83
**SUBROUTINE SYMIA**

This subroutine computes the A matrix and F vector for main program LINI and solves the linear set of equations AU = F.

---

100 SUBROUTINE SYMIA

110 COMMON A(100,100), FT(100), U(100), UEXT(100), UDIF(100), COORD(100), ELEN, ICORR(100, 2), NEL, NSNP

120 DIMENSION AE(2, 2), FS1E(2), FS2E(2), WKAREA(40600)

C ZERO OUT A MATRIX AND F VECTOR

140 DO 210 IZ = 1, NSNP

150 FT(IZ) = 0.

160 DO 200 JZ = 1, NSNP

170 A(IZ, JZ) = 0.

200 CONTINUE

210 CONTINUE

213 ALPHA = 0.

C ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR

215 DO 375 IEL = 1, NEL

C DETERMINE ELEMENTAL A MATRIX AND ELEMENTAL F VECTOR

220 AE(1, 1) = 1./ELEN

230 AE(1, 2) = (-1./ELEN)

240 AE(2, 1) = AE(1, 2)

250 AE(2, 2) = AE(1, 1)

260 FS1E(1) = ELEN

265 FS1E(2) = ELEN

C DISTRIBUTE AE MATRICES AND FE VECTORS INTO SYSTEM A MATRIX AND F VECTOR

300 DO 370 II = 1, 2

310 DO 350 JJ = 1, 2

320 IN = ICORR(IEL, II)

330 JN = ICORR(IEL, JJ)

340 A(IN, JN) = A(IN, JN) - AE(II, JJ)

350 CONTINUE

360 FT(IN) = FS1E(II) + FT(IN)

370 CONTINUE

372 ALPHA = ALPHA + ELEN

375 CONTINUE

C IMPOSE KINEMATIC AND NATURAL BOUNDARY CONDITIONS

376 A(1, 1) = 1.

377 A(1, 2) = 0.

378 FT(1) = 0.
379 \( FT(\text{NSNP}) = FT(\text{NSNP}) - 4.\)
380 \( M = 1\)
390 \( \text{IDGT} = 3\)
400 \( \text{IQ} = 100\)

C \( \text{CALL SUBROUTINE LEQT2F TO SOLVE } A U = F \)

410 \( \text{CALL LEQT2F}(A, M, \text{NSNP}, \text{IQ}, FT, \text{IDGT}, \text{WKAREA}, \text{IER}) \)
420 DO 440 NEW = 1, NSNP
430 \( \text{U(NEW)} = FT(\text{NEW})\)
440 CONTINUE
450 RETURN
460 END
SUBROUTINE UX2EXT

* THIS SUBROUTINE COMPUTES THE VALUE OF U=X**2 AT THE SPECIFIED NODAL POINTS FOR MAIN PROGRAM LIN1.

* *******************************************

100 COMMON A(100,100),FT(100),U(100),UEXT(100),UDIF(100),COORD(100),
      ELEN,ICORR(100,2),NEL,NSNP
130 DO 150 NN = 1,NSNP
140 UEXT(NN) = COORD(NN)**2
150 CONTINUE
160 RETURN
170 END
SOLUTION OF $u'' = 2.$ USING CONSISTENT FORCING FUNCTION

<table>
<thead>
<tr>
<th>X-COORD</th>
<th>U EXACT</th>
<th>U FEM</th>
<th>% DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>0.200</td>
<td>0.0400</td>
<td>0.0400</td>
<td>0.0</td>
</tr>
<tr>
<td>0.400</td>
<td>0.1600</td>
<td>0.1600</td>
<td>0.0</td>
</tr>
<tr>
<td>0.600</td>
<td>0.3600</td>
<td>0.3600</td>
<td>0.0</td>
</tr>
<tr>
<td>0.800</td>
<td>0.6400</td>
<td>0.6400</td>
<td>0.0</td>
</tr>
<tr>
<td>1.000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>1.200</td>
<td>1.4400</td>
<td>1.4400</td>
<td>0.0</td>
</tr>
<tr>
<td>1.400</td>
<td>1.9600</td>
<td>1.9600</td>
<td>0.0</td>
</tr>
<tr>
<td>1.600</td>
<td>2.5600</td>
<td>2.5600</td>
<td>0.0</td>
</tr>
<tr>
<td>1.800</td>
<td>3.2400</td>
<td>3.2400</td>
<td>0.0</td>
</tr>
<tr>
<td>2.000</td>
<td>4.0000</td>
<td>4.0000</td>
<td>0.0</td>
</tr>
</tbody>
</table>
*************** PROGRAM LIN2 ***************

* THIS PROGRAM SOLVES THE DIFFERENTIAL EQUATION U'' = 6X, 0<X<2 *
* U(0)=0; U'(2)=12 WITH UEXACT=U**3. *

*********************** COMMON A(100,100),FT(100),U(100),UEXT(100),UDIF(100),COORD(100), :
*:ELEN,ICORR(100,2),NEL,NSNP

C INPUT NUMBER OF ELEMENTS AND TOTAL LENGTH OF DOMAIN

120 PRINT*, 'INPUT NUMBER OF ELEMENTS DESIRED AND TOTAL LENGTH.'
130 READ(6,*) NEL,TLEN

C CALCULATE NUMBER OF NODAL POINTS AND DEFINE LEFT BOUNDARY OF
C DOMAIN

135 NSNP=NEL+1
150 COORD(1)=0.

C DETERMINE ELEMENT LENGTH, LOCAL TO GLOBAL NODAL POINT
C CORRESPONDENCE, AND X-COORDINATE OF EACH NODAL POINT

155 ELEN=TLEN/FLOAT(NEL)
160 DO 169 IEL=1,NEL
162 ICORR(IEL,1)=IEL
163 ICORR(IEL,2)=IEL+1
164 COORD(IEL+1)=COORD(IEL)+ELEN
169 CONTINUE

C CALL SUBROUTINE SYM2A TO DETERMINE A MATRIX AND F VECTOR AND SOLVE
C THE LINEAR SYSTEM OF EQUATIONS AU = F

170 CALL SYM2A

C CALL SUBROUTINE UX3EXT TO DETERMINE EXACT SOLUTION
190 CALL UX3EXT

C CALL SUBROUTINE OUTLIN TO OUTPUT RESULTS

200 CALL OUTLIN
210 END
**SUBROUTINE SYM2A**

* THIS SUBROUTINE COMPUTES THE A MATRIX AND F VECTOR FOR MAIN PROGRAM LIN2 AND SOLVES THE LINEAR SET OF EQUATIONS AU = F.

```fortran
100 SUBROUTINE SYM2A
110 COMMON A(100,100),FT(100),U(100),UEXT(100),UDIF(100),COORD(100),
: ELEN,ICORR(100,2),NEL,NSNP
120 DIMENSION AE(2,2), FS1E(2), FS2E(2), WKAREA(40600)

C ZERO OUT A MATRIX AND F VECTOR

140 DO 210 IZ = 1,NSNP
150 FT(IZ) = 0.
160 DO 200 JZ = 1,NSNP
170 A(IZ,JZ) = 0.
200 CONTINUE
210 CONTINUE
213 ALPHA=0.
214 PRINT*,'WHAT TYPE OF FORCING FUNCTION IS TO BE USED?'
215 PRINT*,'MIDPOINT = 1; 1/4 - 3/4 APPROX = 2; CONSISTENT = 3'
216 READ(6,*) NFF

C ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR

217 DO 375 IEL=1,NEL

C DETERMINE ELEMENTAL A MATRIX AND ELEMENTAL F VECTOR

220 AE(1,1)=1./ELEN
230 AE(1,2)=(-1./ELEN)
240 AE(2,1)=AE(1,2)
250 AE(2,2)=AE(1,1)
263 IF (NFF.EQ.1) THEN
264 FS1E(1)=3.*ELEN*(ALPHA+ELEN/2.)
265 FS1E(2)=FS1E(1)
266 ELSEIF (NFF.EQ.2) THEN
267 FS1E(1)=3.*ELEN*(ALPHA+ELEN/4.)
268 FS1E(2)=3.*ELEN*(ALPHA+ELEN/4.)
269 ELSE
270 FS1E(1)=3.*ALPHA*ELEN+ELEN**2
271 FS1E(2)=3.*ALPHA*ELEN+ELEN**2
272 ENDIF

C DISTRIBUTE AE MATRICES AND FE VECTORS INTO SYSTEM A MATRIX AND F VECTOR

300 DO 370 II=1,2
310 DO 350 JJ=1,2
320 IN=ICORR(IEL,II)
330 JN=ICORR(IEL,JJ)
340 A(IN,JN)=A(IN,JN) - AE(II,JJ)
350 CONTINUE
360 FT(IN)=FS1E(II) + FT(IN)
```
CONTINUE
ALPHA=ALPHA + ELEN
CONTINUE

C IMPOSE KINEMATIC AND NATURAL BOUNDARY CONDITIONS

A(1,1)=1.
A(1,2)=0.
FT(1)=0.
FT(NSNP)=FT(NSNP)-12.
M=1
IDGT=3
IQ=100

C CALL SUBROUTINE LEQT2F TO SOLVE AU = F

CALL LEQT2F(A,M,NSNP,IQ,FT,IDGT,WKAREA,IER)
DO 440 NEW=1,NSNP
   U(NEW)=FT(NEW)
CONTINUE
RETURN
END
SUBROUTINE UX3EXT

* THIS SUBROUTINE COMPUTES THE VALUE OF U=X**3 AT THE SPECIFIED NODAL POINTS FOR MAIN PROGRAM LIN2.

C
C
C
C

SUBROUTINE UX3EXT

COMMON A(100,100),FT(100),U(100),UEXT(100),UDIF(100),COORD(100),:
ELEN,ICORR(100,2),NEL,NSNP

DO 150 NN = 1,NSNP

UEXT(NN) = COORD(NN)**3

CONTINUE

RETURN

END
SOLUTION OF \( u'' = 6x \) USING LUMPED MIDPOINT FORCING FUNCTION

<table>
<thead>
<tr>
<th>X-COORD</th>
<th>U EXACT</th>
<th>U FEM</th>
<th>% DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>0.200</td>
<td>0.0080</td>
<td>0.0120</td>
<td>50.0</td>
</tr>
<tr>
<td>0.400</td>
<td>0.0640</td>
<td>0.0720</td>
<td>12.5</td>
</tr>
<tr>
<td>0.600</td>
<td>0.2160</td>
<td>0.2280</td>
<td>5.6</td>
</tr>
<tr>
<td>0.800</td>
<td>0.5120</td>
<td>0.5280</td>
<td>3.1</td>
</tr>
<tr>
<td>1.000</td>
<td>1.0000</td>
<td>1.0200</td>
<td>2.0</td>
</tr>
<tr>
<td>1.200</td>
<td>1.7280</td>
<td>1.7520</td>
<td>1.4</td>
</tr>
<tr>
<td>1.400</td>
<td>2.7440</td>
<td>2.7720</td>
<td>1.0</td>
</tr>
<tr>
<td>1.600</td>
<td>4.0960</td>
<td>4.1280</td>
<td>0.8</td>
</tr>
<tr>
<td>1.800</td>
<td>5.8320</td>
<td>5.8680</td>
<td>0.6</td>
</tr>
<tr>
<td>2.000</td>
<td>8.0000</td>
<td>8.0400</td>
<td>0.5</td>
</tr>
</tbody>
</table>
SOLUTION OF $u'' = 6x$ USING $1/4 - 3/4$ LUMPED FORCING FUNCTION

<table>
<thead>
<tr>
<th>X-COORD</th>
<th>U EXACT</th>
<th>U FEM</th>
<th>% DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>0.200</td>
<td>0.0080</td>
<td>0.0060</td>
<td>-25.0</td>
</tr>
<tr>
<td>0.400</td>
<td>0.0640</td>
<td>0.0600</td>
<td>-6.3</td>
</tr>
<tr>
<td>0.600</td>
<td>0.2160</td>
<td>0.2100</td>
<td>-2.8</td>
</tr>
<tr>
<td>0.800</td>
<td>0.5120</td>
<td>0.5040</td>
<td>-1.6</td>
</tr>
<tr>
<td>1.000</td>
<td>1.0000</td>
<td>0.9900</td>
<td>-1.0</td>
</tr>
<tr>
<td>1.200</td>
<td>1.7280</td>
<td>1.7160</td>
<td>-0.7</td>
</tr>
<tr>
<td>1.400</td>
<td>2.7440</td>
<td>2.7300</td>
<td>-0.5</td>
</tr>
<tr>
<td>1.600</td>
<td>4.0960</td>
<td>4.0800</td>
<td>-0.4</td>
</tr>
<tr>
<td>1.800</td>
<td>5.8320</td>
<td>5.8140</td>
<td>-0.3</td>
</tr>
<tr>
<td>2.000</td>
<td>8.0000</td>
<td>7.9800</td>
<td>-0.2</td>
</tr>
</tbody>
</table>
SOLUTION OF $U'' = 6x$ USING CONSISTENT FORCING FUNCTION

<table>
<thead>
<tr>
<th>X-COORD</th>
<th>U EXACT</th>
<th>U FEM</th>
<th>% DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>0.200</td>
<td>0.0080</td>
<td>0.0080</td>
<td>0.0</td>
</tr>
<tr>
<td>0.400</td>
<td>0.0640</td>
<td>0.0640</td>
<td>0.0</td>
</tr>
<tr>
<td>0.600</td>
<td>0.2160</td>
<td>0.2160</td>
<td>0.0</td>
</tr>
<tr>
<td>0.800</td>
<td>0.5120</td>
<td>0.5120</td>
<td>0.0</td>
</tr>
<tr>
<td>1.000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>1.200</td>
<td>1.7280</td>
<td>1.7280</td>
<td>0.0</td>
</tr>
<tr>
<td>1.400</td>
<td>2.7440</td>
<td>2.7440</td>
<td>0.0</td>
</tr>
<tr>
<td>1.600</td>
<td>4.0960</td>
<td>4.0960</td>
<td>0.0</td>
</tr>
<tr>
<td>1.800</td>
<td>5.8320</td>
<td>5.8320</td>
<td>0.0</td>
</tr>
<tr>
<td>2.000</td>
<td>8.0000</td>
<td>8.0000</td>
<td>0.0</td>
</tr>
</tbody>
</table>
C******************************************************************************
C* PROGRAM LIN3
C******************************************************************************
C* THIS PROGRAM SOLVES THE DIFFERENTIAL EQUATION U'" = 12X^2. 
C* 0 < X < 2  U(0)=0; U'(2)=32 WITH UEXACT=X^4 
C******************************************************************************

110 COMMON A(100,100),FT(100),U(100),UEXT(100),UDIF(100),COORD(100),
: ELEN,ICORR(100,2),NEL,NSNP

C INPUT NUMBER OF ELEMENTS AND TOTAL LENGTH OF DOMAIN

120 PRINT*, 'INPUT NUMBER OF ELEMENTS DESIRED AND TOTAL LENGTH.'
130 READ(6,*) NEL,TLEN

C CALCULATE NUMBER OF NODAL POINTS AND DEFINE LEFT BOUNDARY OF
C DOMAIN

135 NSNP=NEL+1
150 COORD(1)=0.

C DETERMINE ELEMENT LENGTH, LOCAL TO GLOBAL NODAL POINT
C CORRESPONDENCE, AND X-COORDINATE OF EACH NODAL POINT

155 ELEN=TLEN/FLOAT(NEL)
160 DO 169 IEL=1,NEL
162 ICORR(IEL,1)=IEL
163 ICORR(IEL,2)=IEL+1
164 COORD(IEL+1)=COORD(IEL)+ELEN
169 CONTINUE

C CALL SUBROUTINE SYM3A TO DETERMINE A MATRIX AND F VECTOR AND SOLVE
C THE LINEAR SYSTEM OF EQUATIONS AU = F

170 CALL SYM3A

C CALL SUBROUTINE UX4EXT TO DETERMINE EXACT SOLUTION

190 CALL UX4EXT

C CALL SUBROUTINE OUTLIN TO OUTPUT RESULTS

200 CALL OUTLIN
210 END
**SUBROUTINE SYM3A**

* THIS SUBROUTINE COMPUTES THE A MATRIX AND F VECTOR FOR MAIN PROGRAM LIN3 AND SOLVES THE LINEAR SET OF EQUATIONS AU = F.

****************************************************

C

*** ZERO OUT A MATRIX AND F VECTOR ***

100 SUBROUTINE SYM3A
110 COMMON A(100,100),FT(100),U(100),UEXT(100),UDIF(100),COORD(100),
      :ELEN,ICORR(100,2),NEL,NSNP
120 DIMENSION AE(2,2), FS1E(2), FS2E(2), WKAREA(40600)

C

DO 210 IZ = 1,NSNP
150 FT(IZ) = 0.
160 DO 200 JZ = 1,NSNP
170 A(IZ,JZ) = 0.
200 CONTINUE
210 CONTINUE
213 ALPHA=0.
214 PRINT*,'WHAT TYPE OF FORCING FUNCTION IS TO BE USED?'
215 PRINT*,'MIDPOINT = 1; 1/4 - 3/4 APPROX = 2; CONSISTENT = 3'
216 READ(6,*) NFF

C

** ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR **

217 DO 375 IEL=1,NEL

C

** DETERMINE ELEMENTAL A MATRIX AND ELEMENTAL F VECTOR **

220 AE(1,1)=L./ELEN
230 AE(1,2)=(-L./ELEN)
240 AE(2,1)=AE(1,2)
250 AE(2,2)=AE(1,1)
263 IF (NFF.EQ.1) THEN
264   FS1E(1)=6.*ELEN*(ALPHA+ELEN/2. )**2
265   FS1E(2)=FS1E(1)
266 ELSEIF (NFF.EQ.2) THEN
267   FS1E(1)=6.*ELEN*(ALPHA+ELEN/4. )**2
268   FS1E(2)=6.*ELEN*(ALPHA+ELEN/4. )**2
269 ELSE
270   FS1E(1)=6.*(ALPHA**2)*ELEN+4.*ALPHA*(ELEN**2)+ELEN**3
271   FS1E(2)=6.*(ALPHA**2)*ELEN+8.*ALPHA*(ELEN**2)+3.*ELEN**3
272 ENDIF

C

** DISTRIBUTE AE MATRICES AND FE VECTORS INTO SYSTEM A MATRIX AND F VECTOR **

300 DO 370 II=1,2
310 DO 350 JJ=1,2
320 IN=ICORR(IEL,II)
330 JN=ICORR(IEL,JJ)
340 A(IN,JN)=A(IN,JN) - AE(II,JJ)
350 CONTINUE
360 FT(IN)=FS1E(II) + FT(IN)
370 CONTINUE
372 ALPHA=ALPHA + ELEN
375 CONTINUE

C IMPOSE KINEMATIC AND NATURAL BOUNDARY CONDITIONS

376 A(1,1)=1.
377 A(1,2)=0.
378 FT(1)=0.
379 FT(NSNP)=FT(NSNP)-32.
380 M=1
380 IDGT=3
400 IQ=100

C CALL SUBROUTINE LEQT2F TO SOLVE AU = F

410 CALL LEQT2F(A,M,NSNP,IQ,FT,IDGT,WKAREA,IER)
420 DO 440 NEW=1,NSNP
430 U(NEW)=FT(NEW)
440 CONTINUE
450 RETURN
460 END
C

SUBROUTINE UX4EXT

* THIS SUBROUTINE COMPUTES THE VALUE OF U=X**4 AT THE SPECIFIED NODAL POINTS FOR MAIN PROGRAM LIN3.

******************************************************************************

100 SUBROUTINE UX4EXT
110 COMMON A(100,100),FT(100),U(100),UEXT(100),UDIF(100),COORD(100),
:ELEN,ICORR(100,2),NEL,NSNP
130 DO 150 NN = 1,NSNP
140 UEXT(NN) = COORD(NN)**4
150 CONTINUE
160 RETURN
170 END
SOLUTION OF $u'' = 12x^2$ USING MIDPOINT LUMPED FORCING FUNCTION

<table>
<thead>
<tr>
<th>X-COORD</th>
<th>U EXACT</th>
<th>U FEM</th>
<th>% DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>0.200</td>
<td>0.0016</td>
<td>0.0184</td>
<td>1050.3</td>
</tr>
<tr>
<td>0.400</td>
<td>0.0256</td>
<td>0.0608</td>
<td>137.5</td>
</tr>
<tr>
<td>0.600</td>
<td>0.1296</td>
<td>0.1848</td>
<td>42.6</td>
</tr>
<tr>
<td>0.800</td>
<td>0.4096</td>
<td>0.4864</td>
<td>18.8</td>
</tr>
<tr>
<td>1.000</td>
<td>1.0000</td>
<td>1.1000</td>
<td>10.0</td>
</tr>
<tr>
<td>1.200</td>
<td>2.0736</td>
<td>2.1984</td>
<td>6.0</td>
</tr>
<tr>
<td>1.400</td>
<td>3.8416</td>
<td>3.9928</td>
<td>3.9</td>
</tr>
<tr>
<td>1.600</td>
<td>6.5536</td>
<td>6.7328</td>
<td>2.7</td>
</tr>
<tr>
<td>1.800</td>
<td>10.4976</td>
<td>10.7064</td>
<td>2.0</td>
</tr>
<tr>
<td>2.000</td>
<td>16.0000</td>
<td>16.2400</td>
<td>1.5</td>
</tr>
</tbody>
</table>
SOLUTION OF $U'' = 12x^2$ USING $1/4 - 3/4$ LUMPED FORCING FUNCTION

<table>
<thead>
<tr>
<th>X-COORD</th>
<th>U EXACT</th>
<th>U FEM</th>
<th>% DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>0.200</td>
<td>0.0016</td>
<td>0.0046</td>
<td>187.8</td>
</tr>
<tr>
<td>0.400</td>
<td>0.0256</td>
<td>0.0296</td>
<td>15.7</td>
</tr>
<tr>
<td>0.600</td>
<td>0.1296</td>
<td>0.1326</td>
<td>2.3</td>
</tr>
<tr>
<td>0.800</td>
<td>0.4096</td>
<td>0.4096</td>
<td>0.0</td>
</tr>
<tr>
<td>1.000</td>
<td>1.0000</td>
<td>0.9950</td>
<td>-0.5</td>
</tr>
<tr>
<td>1.200</td>
<td>2.0736</td>
<td>2.0616</td>
<td>-0.6</td>
</tr>
<tr>
<td>1.400</td>
<td>3.8416</td>
<td>3.8206</td>
<td>-0.5</td>
</tr>
<tr>
<td>1.600</td>
<td>6.5536</td>
<td>6.5216</td>
<td>-0.5</td>
</tr>
<tr>
<td>1.800</td>
<td>10.4976</td>
<td>10.4526</td>
<td>-0.4</td>
</tr>
<tr>
<td>2.000</td>
<td>16.0000</td>
<td>15.9400</td>
<td>-0.4</td>
</tr>
</tbody>
</table>
SOLUTION OF $U'' = 12x^2$ USING CONSISTENT FORCING FUNCTION

<table>
<thead>
<tr>
<th>X-COORD</th>
<th>U EXACT</th>
<th>U FEM</th>
<th>% DIFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>0.200</td>
<td>0.0016</td>
<td>0.0016</td>
<td>0.1</td>
</tr>
<tr>
<td>0.400</td>
<td>0.0256</td>
<td>0.0256</td>
<td>0.0</td>
</tr>
<tr>
<td>0.600</td>
<td>0.1296</td>
<td>0.1296</td>
<td>0.0</td>
</tr>
<tr>
<td>0.800</td>
<td>0.4096</td>
<td>0.4096</td>
<td>0.0</td>
</tr>
<tr>
<td>1.000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.0</td>
</tr>
<tr>
<td>1.200</td>
<td>2.0736</td>
<td>2.0736</td>
<td>0.0</td>
</tr>
<tr>
<td>1.400</td>
<td>3.8416</td>
<td>3.8416</td>
<td>0.0</td>
</tr>
<tr>
<td>1.600</td>
<td>6.5536</td>
<td>6.5536</td>
<td>0.0</td>
</tr>
<tr>
<td>1.800</td>
<td>10.4976</td>
<td>10.4976</td>
<td>0.0</td>
</tr>
<tr>
<td>2.000</td>
<td>16.0000</td>
<td>16.0000</td>
<td>0.0</td>
</tr>
</tbody>
</table>
SUBROUTINE OUTLIN

This subroutine computes the percent error between the exact and FEM values of U and outputs the appropriate information for main programs LIN1, LIN2, and LIN3.

**SUBROUTINE OUTLIN**

100 COMMON A(100,100), FT(100), U(100), UEXT(100), UDIF(100), COORD(100), ELEN, ICORR(100,2), NEL, NSNP

**SUBROUTINE OUTLIN**

110 DO 150 IK=2,NSNP
111   UDIF(IK)=100.* (U(IK)-UEXT(IK))/UEXT(IK)
112   CONTINUE
113   UDIF(1)=U(1)-UEXT(1)

**SUBROUTINE OUTLIN**

120 COMPUTE PERCENT ERROR AT EACH NODAL POINT

130 WRITE(6,180)
135 WRITE(30,180)
140 FORMAT(/,1X,'X-COORD',4X,'U EXACT',4X,'U FEM',6X,'% DIFF')
145 WRITE(6,200) (COORD(NP),UEXT(NP),U(NP),UDIF(NP), NP=1,NSNP)
150 WRITE(30,200) (COORD(NP),UEXT(NP),U(NP),UDIF(NP), NP=1,NSNP)
155 FORMAT(/,2X,F5.3,5X,F7.4,3X,F7.4,4X,F6.1)
160 RETURN
165 END

**SUBROUTINE OUTLIN**
APPENDIX C. LINEARIZATION VECTORS FOR CONSTANT LINEARIZATION TECHNIQUE

For this analysis, \( h(u') \) in equation (3.6) is replaced by \( (u')^2 \) and the element linearization vector becomes

\[
f^* = \int_0^l g(u'^2) d\xi \quad \text{(C.1)}
\]

The three approximations of this integral outlined in III.B.1 are now determined.

**Midpoint Lumped Approximation**

Replacing \( h(u') \) in equation (3.7) with \( (u')^2 \) yields

\[
f^* = \left( \frac{(u_i^*)_i + (u_{i+1})_i}{2} \right)^2 \left[ \frac{l_e}{2} \right] \quad \text{(C.2)}
\]

**1/4 - 3/4 Lumped Approximation**

Replacing \( h(u') \) in equation (3.8.d) with \( (u')^2 \) yields

\[
f^* = \frac{l_e}{2} \left[ \left( \frac{3}{4} (u_i^*)_i + \frac{1}{4} (u_{i+1})_i \right)^2 \right] \left[ \left( \frac{1}{4} (u_i^*)_i + \frac{3}{4} (u_{i+1})_i \right) \right] \quad \text{(C.3)}
\]
Linear Approximation
Replacing \( h(u') \) with \((u')^2\) in equation (3.9.b) gives

\[
f^* = \int_0^l \left[ 1 - \frac{\xi}{l_e} \right] \left( u_j^* \left( 1 - \frac{\xi}{l_e} \right) + (u_{j+1})^* \left( \frac{\xi}{l_e} \right) \right)^2 d\xi
\]

\[
= \int_0^l \left[ 1 - \frac{\xi}{l_e} \right] \left( u_j^* \left( 1 - \frac{\xi}{l_e} \right)^2 + (u_j^* + u_{j+1}^*) \left( \frac{\xi}{l_e} - \frac{\xi^2}{l_e^2} \right) + (u_{j+1}^*)^2 \right) d\xi \tag{C.4}
\]

\[
= l_e \left[ \frac{(u_j^*)^2}{4} + \frac{(u_j^*)(u_{j+1})}{6} + \frac{(u_{j+1})^2}{12} \right]
\]

\[
+ \frac{(u_j^*)^2}{12} + \frac{(u_j^*)(u_{j+1})}{6} + \frac{(u_{j+1})^2}{4}
\]
APPENDIX D. LINEARIZATION MATRICES FOR THE CLASSICAL LINEARIZATION TECHNIQUE

For this analysis, \( h(u') \) in equation (3.10) is replaced by \( u' \) and the element linearization matrix integral becomes

\[
\int_0^L g g^T u^* d\xi
\]  \( (D.1) \)

The two approximations of this integral outlined in III.B.2 are now determined.

**Midpoint Lumped Approximation**

Replacing \( h(u') \) in equation (3.11) with \( u' \) yields

\[
i_* = \left( \frac{(u_i^*)_i + (u_{j-1})_i}{2} \right) \frac{1}{l_e} \begin{bmatrix}
\frac{1}{3} & \frac{1}{6} \\
\frac{1}{6} & \frac{1}{3}
\end{bmatrix}
\]  \( (D.2) \)

**Linear Approximation**

Replacing \( h(u') \) with \( u' \) in equation (3.12) gives

\[
i_* = \int_0^L \left[ \frac{1 - \frac{\xi}{l_e}}{\frac{\xi}{l_e}} \right] \left[ 1 - \frac{\xi}{l_e} \frac{\xi}{l_e} \right] \left( (u_i^*)_i \left( 1 - \frac{\xi}{l_e} \right) + (u_{j-1})_i \left( \frac{\xi}{l_e} \right) \right) d\xi
\]  \( (D.3) \)

\[
= \int_0^L \left[ \frac{\xi}{l_e} \left( 1 - \frac{\xi}{l_e} \right) \left( \frac{\xi}{l_e} \right)^2 \right] \left( (u_i^*)_i \left( 1 - \frac{\xi}{l_e} \right) + (u_{j-1})_i \left( \frac{\xi}{l_e} \right) \right) d\xi
\]  \( (D.3) \)

\[
= \frac{l_e}{12} \begin{bmatrix}
3(u_i^*)_i + (u_{j-1})_i & (u_i^*)_i + (u_{j-1})_i \\
(u_i^*)_i + (u_{j-1})_i & (u_i^*)_i + 3(u_{j-1})_i
\end{bmatrix}
\]
APPENDIX E. PROGRAM LISTINGS FOR CONSTANT LINEARIZATION

C ************************************************************************************
C * PROGRAM NU2CAN                      *
C * THIS PROGRAM SOLVES THE NONLINEAR SECOND ORDER DIFFERENTIAL EQUATION:        *
C * U'' - U**2 = 6 - 9X**4; UEXACT=3X**2 WITH VARIABLE DOMAIN                    *
C * TREATING THE U**2 TERM AS AN EXCITATION AND TAKING IT TO THE RIGHT SIDE OF   *
C * THE EQUATION.                                                                  *
C * THE USER SELECTS:                                                             *
C * 1) NUMBER OF ELEMENTS                                                        *
C * 2) SIZE OF DOMAIN                                                             *
C * 3) X AND U(X) AT THE LEFT BOUNDARY                                           *
C * 4) U(X) OR U'(X) AT THE RIGHT BOUNDARY                                        *
C * 5) ITERATION STRATEGY FOR DETERMINING U*                                     *
C * 6) APPROXIMATION TECHNIQUE FOR THE EXCITATION INTEGRAL                       *
C ************************************************************************************

110 COMMON A(100,100), FS(100), FU(100), FT(100), U(100), UOLD(100),
     UEXT(100), UDIF(100), COORD(100), ELEN, CONV, ELTIME, ULBC, URBC,
     TLEN, ICORR(100,2), NEL, NSNP, ITYPE
115 CONV=.0001
C READ IN PARAMETERS FROM DATA FILE
130 READ(29,*) NEL, TLEN, COORD(1), ULBC, ITYPE, URBC
C CALCULATE NUMBER OF NODAL POINTS
135 NSNP=NEL+1
C DETERMINE ELEMENT SIZE OF EQUAL LENGTHS
137 ELEN=TLEN/FLOAT(NEL)
C ESTABLISH LOCAL TO GLOBAL CORRESPONDENCE AND X COORDINATE OF EACH NODE
160 DO 169 IEL=1, NEL
162 ICORR(IEL,1)=IEL
163 ICORR(IEL,2)=IEL+1
165 COORD(IEL+1)=COORD(IEL) + ELEN
169 CONTINUE
C CALL SUBROUTINE NU2CAM TO CREATE A MATRIX AND F VECTOR
170 CALL NU2CAM
C CALL SUBROUTINE NU2CAI TO PERFORM SOLUTION ITERATION
180 CALL NU2CAI(IET)
CALL SUBROUTINE U2EXTA TO COMPUTE EXACT SOLUTION \( U=3x^2 \)

CALL U2EXTA

CALL SUBROUTINE OUTPUT TO PRINT OUT DATA, COMPUTATIONAL EFFICIENCY

CALL OUTPUT(CPUSTAR,IET)

END
*** SUBROUTINE NU2CAM ***

* THIS SUBROUTINE COMPUTES THE A MATRIX AND F VECTOR FOR MAIN *
* PROGRAM NU2CAN. *

C SUBROUTINE NU2CAM
C
COMMON A(100,100),FS(100),FU(100),FT(100),U(100),UOLD(100),
:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
:TLEN,ICORR(100,2),NEL,NSNP,ITYPE
DIMENSION AE(2,2),FS1E(2),FS2E(2)

100 SUBROUTINE NU2CAM
110 COMMON A(100,100),FS(100),FU(100),FT(100),U(100),UOLD(100),
:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
:TLEN,ICORR(100,2),NEL,NSNP,ITYPE
120 DIMENSION AE(2,2), FS1E(2), FS2E(2)
122 IF (TLEN.LE.1.0) THEN
123 PRINT*,'CHOOSE BOUNDARY FOR INITIAL GUESS:
124 PRINT*,'1 = LEFT ESSENTIAL BOUNDARY CONDITION'
125 PRINT*,'2 = RIGHT ESSENTIAL BOUNDARY CONDITION'
126 PRINT*,'3 = AVERAGE OF THE TWO ESSENTIAL BOUNDARY CONDITIONS'
127 READ(6,*+) INITGS
128 ELSE
129 CONTINUE
130 ENDIF
140 DO 210 IZ = 1,NSNP

C ZERO OUT STEADY FORCE VECTOR
150 FS(IZ) = 0.

C DETERMINE INITIAL VALUE OF USTAR TO BEGIN THE ITERATION PROCESS
157 IF (INITGS.EQ.1) THEN
158 U(IZ)=ULBC
159 UOLD(IZ)=ULBC
160 ELSEIF (INITGS.EQ.2) THEN
161 U(IZ)=URBC
162 UOLD(IZ)=URBC
163 ELSEIF (INITGS.EQ.3) THEN
164 U(IZ)=(ULBC+URBC)/2.
165 UOLD(IZ)=U(IZ)
166 ELSE
167 U(IZ)=SQRT(ABS(9.*COORD(IZ)**4 - 6.))
168 UOLD(IZ)=U(IZ)
169 ENDIF

C ZERO OUT A MATRIX
170 DO 200 JZ = 1,NSNP
180 A(IZ,JZ) = 0.
200 CONTINUE
210 CONTINUE

C ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR
213 ALPHA=0.
215 DO 375 IEL=1,NEL
220 AE(1,1)=1./ELEN

108
AE(1,2)=(-l./ELEN)
AE(2,1)=AE(1,2)
AE(2,2)=AE(1,1)
FS1E(1)=3.*ELEN
FS1E(2)=FS1E(1)
F1=(ALPHA**4)*ELEN/2.
F2=2.*((ALPHA**3)*ELEN**2)/3.
F3=(ALPHA**2)*ELEN**3/2.
F4=ALPHA*(ELEN**4)/5.
F5=(ELEN**5)/30.
FS2E(1)=(-9.)*(F1 + F2 + F3 + F4 + F5)
FS2E(2)=(-9.)*(F1 + 2.*F2 + 3.*F3 + 4.*F4 + 5.*F5)
DO 370 II=1,2
   IN=ICORR(IEL,II)
   JN=ICORR(IEL,JJ)
   A(IN,JN)=A(IN,JN) - AE(II,JJ)
   CONTINUE
   FS(IN)=FS1E(II) + FS2E(II) + FS(IN)
CONTINUE
ALPHA=ALPHA + ELEN
CONTINUE
RETURN
END
C SUBROUTINE NU2CAI

C THIS SUBROUTINE PERFORMS THE ITERATIVE SOLUTION PROCESS FOR
C MAIN PROGRAM NU2CAN.

SUBROUTINE NU2CAI(IET)

COMMON A(100,100),FS(100),FT(100),U(100),UOLD(100),
: UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
: TLEN,ICORR,100,2),NEL,NSNP,ITYPE

DIMENSION WKAREA(40600),DIF(100),FUE(2),USTAR(100)

C SELECT METHOD OF DETERMINING USTAR

PRINT*,'SELECT METHOD OF U* DETERMINATION.'
PRINT*,'1: U* = U'
PRINT*,'2: U* = AVERAGE OF LAST TWO COMPUTED VALUES OF U'
READ(6,*),METHU

C SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR

PRINT*,'SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR.'
PRINT*,'1: MIDPOINT APPROXIMATION'
PRINT*,'2: 1/4 - 3/4 APPROXIMATION'
PRINT*, '3: LINEAR'
READ(6,*),METHFU

CALL SUBROUTINE SETIME TO BEGIN TIMING ITERATION PROCESS

CALL SETIME

BEGIN ITERATION PROCESS

DO 450 ITER=1,200

RESET VALUE OF UNSTEADY F VECTOR TO ZERO

DO 138 IU=1,NSNP
  FU(IU)=0.

DETERMINE VALUE OF U* AT EACH NODE

IF (METHU.EQ.1) THEN
  USTAR(IU)=U(IU)
ELSEIF (METHU.EQ.2) THEN
  USTAR(IU)=(U(IU)+UOLD(IU))/2.
ENDIF

DETERMINE UNSTEADY FORCE VECTOR

DO 210 IEL=1,NEL
  FUE(1)=(ELEN/2.)*(USTAR(IEL)+USTAR(IEL+1))/2.**2
  FUE(2)=FUE(1)

100 SUBROUTINE NU2CAI(IET)
102 COMMON A(100,100),FS(100),FT(100),U(100),UOLD(100),
: UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
: TLEN,ICORR,100,2),NEL,NSNP,ITYPE
104 DIMENSION WKAREA(40600),DIF(100),FUE(2),USTAR(100)
C SELECT METHOD OF DETERMINING USTAR
105 PRINT*,'SELECT METHOD OF U* DETERMINATION.'
106 PRINT*,'1: U* = U'
107 PRINT*,'2: U* = AVERAGE OF LAST TWO COMPUTED VALUES OF U'
109 READ(6,*),METHU
C SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR
116 PRINT*,'SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR.'
117 PRINT*,'1: MIDPOINT APPROXIMATION'
118 PRINT*,'2: 1/4 - 3/4 APPROXIMATION'
119 PRINT*, '3: LINEAR'
120 READ(6,*),METHFU
C CALL SUBROUTINE SETIME TO BEGIN TIMING ITERATION PROCESS
121 CALL SETIME
C BEGIN ITERATION PROCESS
122 DO 450 ITER=1,200
C RESET VALUE OF UNSTEADY F VECTOR TO ZERO
123 DO 138 IU=1,NSNP
124   FU(IU)=0.
C DETERMINE VALUE OF U* AT EACH NODE
125   IF (METHU.EQ.1) THEN
126     USTAR(IU)=U(IU)
127   ELSEIF (METHU.EQ.2) THEN
128     USTAR(IU)=(U(IU)+UOLD(IU))/2.
129   ENDIF
132 CONTINUE
C DETERMINE UNSTEADY FORCE VECTOR
140 DO 210 IEL=1,NEL
145   IF (METHFU.EQ.1) THEN
146     FUE(1)=(ELEN/2.)*(USTAR(IEL)+USTAR(IEL+1))/2.**2
147     FUE(2)=FUE(1)
110
ELSEIF (METHFU.EQ. 2) THEN
FUE(1)=ELEN/2.*3.*USTAR(IEL)/4. + USTAR(IEL+1)/4.)**2
FUE(2)=ELEN/2.*USTAR(IEL)/4. + 3.*USTAR(IEL+1)/4.)**2
ELSE
FUE(1)=ELEN*(USTAR(IEL)**2/4. + USTAR(IEL)*USTAR(IEL+1)/6. + USTAR(IEL+1)**2/12.)
FUE(2)=ELEN*(USTAR(IEL)**2/12. + USTAR(IEL)*USTAR(IEL+1)/6. + USTAR(IEL+1)**2/4.)
ENDIF

DO 200 II=1,2
IN=ICORR(IEL,II)
FU(IN)=FUE(II)+FU(IN)

CONTINUE

C DETERMINE TOTAL FORCE VECTOR

DO 240 NP=1,NSNP
FTCNP)=FS(NP)+FU(NP)
UOLD(NP)=U(NP)

CONTINUE

C IMPOSE BOUNDARY CONDITIONS

A(1,1)=1.
A(1,2)=0.
FT(1)=ULBC
IF (ITYPE.EQ. 1) THEN
A(NSNP,NSNP-1)=0.
A(NSNP,NSNP)=1.
FT(NSNP)=URBC
ELSE
FT(NSNP)=FT(NSNP)-URBC*TLEN
ENDIF

M=1
IDGT=3
IQ=100

C CALL SUBROUTINE LEQT2F TO SOLVE SET OF LINEAR ALGEBRAIC EQUATIONS

CALL LEQT2F(A,M,NSNP,IQ,FT,IDGT,WKAREA,IER)

DO 310 NEW=1,NSNP
U(NEW)=FT(NEW)

C TEST FOR CONVERGENCE

DIF(NEW)=ABS(U(NEW)-UOLD(NEW))
CONTINUE

DIFMAX=DIF(1)
NMAX=1
DO 330 IJ=1,NEL
IF (DIF(IJ+1).GE.DIF(IJ)) THEN
DIFMAX=DIF(IJ+1)
NMAX=IJ+1
ELSE

111
CONTINUE
ENDIF
CONTINUE
IF (ABS(DIFMAX/U(NMAX)).LT.CONV) THEN
GO TO 451
ELSE
CONTINUE
ENDIF
CONTINUE
CONTINUE
C CALL SUBROUTINE GETIME TO OBTAIN CPU TIME FOR ITERATION PROCESS
CALL GETIME(IET)
C OUTPUT HEADER INFORMATION
WRITE(6,464)
WRITE(30,464)
FORMAT(1X,'EQUATION: U'' - U**2 = 6 - 9X**4')
IF (ITYPE.EQ.1) THEN
WRITE(6,468) COORD(1),ULBC,COORD(NSNP),URBC
WRITE(30,468) COORD(1),ULBC,COORD(NSNP),URBC
FORMAT(1X,'B.C.: U(',F2.0,')&',F2.0,': DU/DX(',F3.0,')=',F4.0,')
ELSE
WRITE(6,472) COORD(1),ULBC,COORD(NSNP),URBC
WRITE(30,472) COORD(1),ULBC,COORD(NSNP),URBC
FORMAT(1X,'B.C.: U(',F2.0,')&',F2.0,': DU/DX(',F3.0,')=',F4.0,')
ENDIF
IF (METHU.EQ.1) THEN
WRITE(6,478)
WRITE(30,478)
FORMAT(1X,'ITERATION METHOD: U*=U',/)
ELSE
WRITE(6,482)
WRITE(30,482)
FORMAT(1X,'ITERATION METHOD: U*=(U+UOLD)/2',/)
ENDIF
IF (METHFU.EQ.1) THEN
WRITE(6,491)
WRITE(30,491)
FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: MIDPOINT',/)
ELSEIF (METHFU.EQ.2) THEN
WRITE(6,495)
WRITE(30,495)
FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: 1/4-3/4',/)
ELSE
WRITE(6,499)
WRITE(30,499)
FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: LINEAR',/)
ENDIF
IF (ITER.GE.200) THEN
WRITE(6,505)
WRITE(30,505)
FORMAT(1X,'CONVERGENCE NOT OBTAINED AFTER 200 ITERATIONS.')
ELSEIF (ABS(U(NMAX)).GT.(10.**20).OR.ABS(U(NSNP-1)).GT.112
:(10.**20)) THEN
  WRITE(6,509)
  WRITE(30,509)
  FORMAT(1X,'SOLUTION PROCESS DIVERGES. ')
ELSE
  WRITE(6,520) ITER,NEL
  WRITE(30,520) ITER,NEL
  FORMAT(1X,'CONVERGENCE OBTAINED AFTER ',I3,' ITERATIONS USING ',I3,' ELEMENTS. ',/)
ENDIF
RETURN
END
SUBROUTINE U2EXTA

* THIS SUBROUTINE COMPUTES THE EXACT SOLUTION, U=3X**2, FOR * 
* MAIN PROGRAM NU2CA AT THE SPECIFIED NODAL POINTS.  *

COMMON A(100,100),FS(100),FU(100),FT(100),U(100),UOLD(100),  
:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,  
:TLEN,ICORR(100,2),NEL,NSNP,ITYPE 
DO 150 NN = 1,NSNP  
140 UEXT(NN) = 3.*COORD(NN)**2  
CONTINUE  
RETURN 
END
PROGRAM NU2CBN

THIS PROGRAM SOLVES THE NONLINEAR SECOND ORDER DIFFERENTIAL EQUATION:

\[ \ddot{u} + u^2 = 60x + 100x^6; \quad \text{UEXACT}=10x^3 \]

WITH VARIABLE DOMAIN, TREATING THE \( u^2 \) TERM AS AN EXCITATION AND TAKING IT TO THE RIGHT SIDE OF THE EQUATION. THE USER SELECTS:

1) NUMBER OF ELEMENTS
2) SIZE OF DOMAIN
3) X AND U(X) AT THE LEFT BOUNDARY
4) U(X) OR \( \dot{U}(X) \) AT THE RIGHT BOUNDARY
5) ITERATION STRATEGY FOR DETERMINING \( U^* \)
6) APPROXIMATION TECHNIQUE FOR THE EXCITATION INTEGRAL

COMMON A(100,100),FS(100),FU(100),FT(100),U(100),UOLD(100),UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,TLEN,ICORR(100,2),NEL,NSNP,ITYPE

CONV=.0001

READ IN PARAMETERS FROM DATA FILE

READ(29,*) NEL,TLEN,COORD(1),ULBC,ITYPE,URBC

CALCULATE NUMBER OF NODAL POINTS

NSNP=NEL+1

DETERMINE ELEMENT SIZE OF EQUAL LENGTHS

ELEN=TLEN/FLOAT(NEL)

ESTABLISH LOCAL TO GLOBAL CORRESPONDENCE AND X COORDINATE OF EACH NODE

DO 169 IEL=1,NEL
   ICORR(IEL,1)=IEL
   ICORR(IEL,2)=IEL+1
   COORD(IEL+1)=COORD(IEL)+ELEN
CONTINUE

CALL SUBROUTINE NU2CBM TO CREATE A MATRIX AND F VECTOR

CALL NU2CBM

CALL SUBROUTINE NU2CBI TO PERFORM SOLUTION ITERATION

CALL NU2CBI(IET)

CALL SUBROUTINE U2EXTB TO COMPUTE EXACT SOLUTION \( U=10x^3 \)

CALL U2EXTB

CALL SUBROUTINE OUTPUT TO PRINT OUT DATA, COMPUTATIONAL EFFICIENCY
200  CALL OUTPUT(CPUSTAR, IET)
210  END
**SUBROUTINE NU2CBM**

* THIS SUBROUTINE COMPUTES THE A MATRIX AND F VECTOR FOR MAIN PROGRAM NU2CBN.

**----------------------------------------------------------------------------------------------------------------------------------**

100 COMMON A(100,100),FS(100),FU(100),FT(100),U(100),UOLD(100),
: UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
: TLEN,ICORR(100,2),NEL,NSNP,ITYPE
120 DIMENSION AE(2,2), FS1E(2), FS2E(2)
122 IF (TLEN.LT. 1.0) THEN
123 PRINT*, 'CHOOSE BOUNDARY FOR INITIAL GUESS: '
124 PRINT*, '1 = LEFT ESSENTIAL BOUNDARY CONDITION'
125 PRINT*, '2 = RIGHT ESSENTIAL BOUNDARY CONDITION'
126 PRINT*, '3 = AVERAGE OF THE TWO ESSENTIAL BOUNDARY CONDITIONS'
127 READ(6,*) INITGS
128 ELSE
129 CONTINUE
130 ENDIF
140 DO 210 IZ = 1,NSNP

C ZERO OUT STEADY FORCE VECTOR

150 FS(IZ) = 0.

C DETERMINE INITIAL VALUE OF USTAR TO BEGIN THE ITERATION PROCESS

157 IF (INITGS.EQ.1) THEN
158 U(IZ)=ULBC
159 UOLD(IZ)=ULBC
160 ELSEIF (INITGS.EQ.2) THEN
161 U(IZ)=URBC
162 UOLD(IZ)=URBC
163 ELSEIF (INITGS.EQ.3) THEN
164 U(IZ)=(ULBC+URBC)/2.
165 UOLD(IZ)=U(IZ)
166 ELSE
167 U(IZ)=SQRT(60.*COORD(IZ) + 100.*COORD(IZ)**6)
168 UOLD(IZ)=U(IZ)
169 ENDIF

C ZERO OUT A MATRIX

170 DO 200 JZ = 1,NSNP
175 A(IZ,JZ) = 0.
200 CONTINUE

C ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR

213 ALPHA=0.
215 DO 375 IEL=1,NEL
220 AE(1,1)=1./ELEN
AE(1,2) = (-1. / ELEN)
AE(2,1) = AE(1,2)
AE(2,2) = AE(1,1)

FS1E(1) = 30. * ALPHA * ELEN + 10. * ELEN**2
FS1E(2) = 30. * ALPHA * ELEN + 20. * ELEN**2

F1 = 50. *(ALPHA**6) * ELEN
F2 = 100. *(ALPHA**5) * (ELEN**2)
F3 = 125. *(ALPHA**4) * (ELEN**3)
F4 = 100. *(ALPHA**3) * (ELEN**4)
F5 = 50. *(ALPHA**2) * (ELEN**5)
F6 = 100. * ALPHA * (ELEN**6) / 7.

F7 = 25. *(ELEN**7) / 16.

FS2E(1) = F1 + F2 + F3 + F4 + F5 + F6 + F7

DO 370 II = 1, 2
DO 350 JJ = 1, 2
     IN = ICORR(I, II)
     JN = ICORR(I, JJ)
     A(IN, JN) = A(IN, JN) - AE(II, JJ)
     CONTINUE
     FS(IN) = FS1E(II) + FS2E(II) + FS(IN)
     CONTINUE
ALPHA = ALPHA + ELEN
CONTINUE
RETURN
END
**SUBROUTINE NU2CBI**

* * *

**THIS SUBROUTINE PERFORMS THE ITERATIVE SOLUTION PROCESS FOR**

* * *

**MAIN PROGRAM NU2CBN.**

* * *

********************************************************************************************************************************************

100 SUBROUTINE NU2CBI(IET)

102 COMMON A(100,100),FS(100),FU(100),FT(100),U(100),UOLD(100),
: UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
: TLEN,ICORR(100,2),NEL,NSNP,ITYPE

104 DIMENSION WKAREA(40600), DIF(100), FUE(2), USTAR(100)

C SELECT METHOD OF DETERMINING USTAR

105 PRINT*, 'SELECT METHOD OF U* DETERMINATION.'

106 PRINT*, '1: U* = U'

107 PRINT*, '2: U* = AVERAGE OF LAST TWO COMPUTED VALUES OF U'

109 READ(6,*) METHU

C SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR

116 PRINT*, 'SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR.'

117 PRINT*, '1: MIDPOINT APPROXIMATION'

118 PRINT*, '2: 1/4 - 3/4 APPROXIMATION'

119 PRINT*, '3: CONSISTENT'

120 READ(6,*) METHFU

C CALL SUBROUTINE SETIME TO BEGIN TIMING ITERATION PROCESS

121 CALL SETIME

C BEGIN ITERATION PROCESS

122 DO 450 ITER=1,200

C RESET VALUE OF UNSTEADY F VECTOR TO ZERO

123 DO 138 IU=1,NSNP

124 FU(IU)=0.

C DETERMINE VALUE OF U* AT EACH NODE

125 IF (METHU.EQ.1) THEN

126 USTAR(IU)=U(IU)

127 ELSE

128 USTAR(IU)=(U(IU)+UOLD(IU))/2.

132 ENDIF

138 CONTINUE

C DETERMINE UNSTEADY FORCE VECTOR

140 DO 210 IEL=1,NEL

145 IF (METHFU.EQ.1) THEN

146 FUE(1)=(ELEN/2.)*((USTAR(IEL)+USTAR(IEL+1))/2.)**2

147 FUE(2)=FUE(1)

119
ELSEIF (METHFU. EQ. 2) THEN
    FUE(1) = (ELEN/2.)*(3.*USTAR(IEL)/4. + USTAR(IEL+1)/4.)**2
    FUE(2) = (ELEN/2.)*(USTAR(IEL)/4. + 3.*USTAR(IEL+1)/4.)**2
ELSE
    FUE(1) = ELEN*(USTAR(IEL)**2/4. + USTAR(IEL)*USTAR(IEL+1)/6.
             + USTAR(IEL+1)**2/12.)
    FUE(2) = ELEN*(USTAR(IEL)**2/12. + USTAR(IEL)*USTAR(IEL+1)/6.
             + USTAR(IEL+1)**2/4.)
ENDIF

DO 200 II=1,2
   IN=ICORR(IEL,II)
   FU(IN)=FLIE(II) + FU(IN)
CONTINUE

CONTINUE

C DETERMINE TOTAL FORCE VECTOR

DO 240 NP=1,NSNP
   FT(NP)=FS(NP)-FU(NP)
   UOLD(NP)=U(NP)
CONTINUE

C IMPOSE BOUNDARY CONDITIONS

A(1,1)=1.
A(1,2)=0.
FT(1)=ULBC
IF (ITYPE.EQ. 1) THEN
   A(NSNP,NSNP-1)=0.
   A(NSNP,NSNP)=1.
   FT(NSNP)=URBC
ELSE
   FT(NSNP)=FT(NSNP)-URBC
ENDIF

M=1
IDGT=-3
IQ=100

CALL LEQT2F(A,M,NSNP,IQ,FT,IDGT,WKAREA,IER)
DO 310 NEW=1,NSNP
   U(NEW)=FT(NEW)
   WRITE(*,*), 'U(NEW)=', U(NEW)
C TEST FOR CONVERGENCE

DIF(NEW)=ABS(U(NEW)-UOLD(NEW))
CONTINUE
DIFMAX=DIF(1)\nNMAX=1
DO 390 IJ=1,NEL
   IF (DIF(IJ+1).GE.DIF(IJ)) THEN
      DIFMAX=DIF(IJ+1)
      NMAX=IJ+1
   ELSE
      DIFMAX=DIF(IJ+1)
      NMAX=IJ+1
   CONTINUE
   WRITE(*,*), 'UNEW=', U(NEW)
C CALL SUBROUTINE LEQT2F TO SOLVE SET OF LINEAR ALGEBRAIC EQUATIONS

CALL LEQT2F(A,M,NSNP,IQ,FT,IDGT,WKAREA,IER)
DO 310 NEW=1,NSNP
   U(NEW)=FT(NEW)
   WRITE(*,*), 'U(NEW)=', U(NEW)
C TEST FOR CONVERGENCE

DIF(NEW)=ABS(U(NEW)-UOLD(NEW))
CONTINUE
DIFMAX=DIF(1)
NMAX=1
DO 390 IJ=1,NEL
   IF (DIF(IJ+1).GE.DIF(IJ)) THEN
      DIFMAX=DIF(IJ+1)
      NMAX=IJ+1
   ELSE
      DIFMAX=DIF(IJ+1)
      NMAX=IJ+1
   CONTINUE
CONTINUE
ENDIF
CONTINUE
IF (ABS(DIFMAX/U(NMAX)).LT.CONV) THEN
   GO TO 460
ELSE
   CONTINUE
ENDIF
CONTINUE
CONTINUE
CALL SUBROUTINE GETIME TO OBTAIN CPU TIME FOR ITERATION PROCESS

CALL GETIME(IET)

C OUTPUT HEADER INFORMATION
WRITE(6,464)
WRITE(30,464)
FORMAT(1X,'EQUATION: U" + U**2 = 60X + 100X**6')
IF (ITYPE.EQ.1) THEN
   WRITE(6,468) COORD(1),ULBC,COORD(NSNP),URBC
   WRITE(30,468) COORD(1),ULBC,COORD(NSNP),URBC
   FORMAT(1X,'B.C.: U(',F2.0,')=',F3.0,'; U(',F2.0,')=','F4.0,/) ELSE
   WRITE(6,472) COORD(1),ULBC,COORD(NSNP),URBC
   WRITE(30,472) COORD(1),ULBC,COORD(NSNP),URBC
   FORMAT(1X,'B.C.: U(',F2.0,')=','F3.0,DU/DX(',F2.0,')=','F4.0,/) ENDIF
IF (METHU.EQ.1) THEN
   WRITE(6,478)
   WRITE(30,478)
   FORMAT(1X,'ITERATION METHOD: U*=U',/)
ELSE
   WRITE(6,482)
   WRITE(30,482)
   FORMAT(1X,'ITERATION METHOD: U*=(U+UOLD)/2',/)
ENDIF
IF (METHFU.EQ.1) THEN
   WRITE(6,491)
   WRITE(30,491)
   FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: MIDPOINT',/)
ELSEIF (METHFU.EQ.2) THEN
   WRITE(6,495)
   WRITE(30,495)
   FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: 1/4-3/4',/)
ELSE
   WRITE(6,499)
   WRITE(30,499)
   FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: LINEAR',/)
ENDIF
IF (ITER.GE.200) THEN
   WRITE(6,505)
   WRITE(30,505)
   FORMAT(1X,'CONVERGENCE NOT OBTAINED AFTER 200 ITERATIONS.') ELSEIF (ABS(U(NMAX)).GT.(10.**20).OR.ABSCU(NSNP-1)).GT.121
:(10.**20)) THEN
507    WRITE(6,509)
508    WRITE(30,509)
509    FORMAT(1X,'SOLUTION PROCESS DIVERGES. ')
510    ELSE
511    WRITE(6,520) ITER,NEL
515    WRITE(30,520) ITER,NEL
520    FORMAT(1X,'CONVERGENCE OBTAINED AFTER ',I3,' ITERATIONS USING '

525    ENDIF
530    RETURN
540    END
SUBROUTINE U2EXTB

* THIS SUBROUTINE COMPUTES THE EXACT SOLUTION, U = 10X**3,
* FOR MAIN PROGRAM NU2CBN AT THE SPECIFIED NODAL POINTS.

100 SUBROUTINE U2EXTB
110 COMMON A(100,100),FS(100),FU(100),FT(100),U(100),UOLD(100),
: UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
: TLEN,ICORR(100,2),NEL,NSNP,ITYPE
130 DO 150 NN = 1,NSNP
140 UEXT(NN) = 10.*COORD(NN)**3
150 CONTINUE
160 RETURN
170 END
**SUBROUTINE OUTPUT**

This subroutine computes the per cent error between the exact and FEM solutions, CPU* for the iteration process, and prints out all data in tabular form for programs NU2CAN and NU2CBN.

```fortran
100 SUBROUTINE OUTPUT(CPUSTAR,IET)
110 COMMON A(100,100),FS(100),FU(100),FT(100),U(100),UOOLD(100),
 :UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
 :TLEN,ICORR(100,2),NEL,NSNP,ITYPE
115 SUMDIF=0.

C CALCULATE PER CENT ERROR AT EACH NODE AND SUM THE ABSOLUTE VALUE
C OF ALL THE ERRORS
120 DO 150 IK=2,NSNP
130 UDIF(IK)=100.*(U(IK)-UEXT(IK))/UEXT(IK)
140 SUMDIF=SUMDIF+ABS(UDIF(IK))
150 CONTINUE
160 UDIF(1)=U(1)-UEXT(1)

C COMPUTE THE ELAPSED TIME OF THE ITERATION PROCESS
164 ELTIME=IET*.000026
165 WRITE(6,169)ELTIME
166 WRITE(30,169)ELTIME
169 FORMAT(1X,'ELAPSED TIME FOR THE ITERATION PROCESS IS ',F9.4,
 :' SECONDS.')

C OUTPUT DATA IN TABULAR FORMAT
170 WRITE(6,180)
175 WRITE(30,180)
180 FORMAT(/,1X,'X-COORD',3X,'U EXACT',3X,'U FEM',4X,'% DIFF')
190 WRITE(6,200) (COORD(NP),UEXT(NP),U(NP),UDIF(NP), NP=1,NSNP)
195 WRITE(30,200) (COORD(NP),UEXT(NP),U(NP),UDIF(NP), NP=1,NSNP)
200 FORMAT(/,2X,F5.3,5X,F7.4,3X,F7.4,4X,F5.1)

C CALCULATE CPU* FOR THE ITERATION PROCESS
205 CPUSTAR=ELTIME*SUMDIF/NSNP
210 WRITE(6,220) CPUSTAR
215 WRITE(30,220) CPUSTAR
220 FORMAT(/,1X,'CPU* FOR THE ITERATION PROCESS IS ',F9.4,' SECONDS.')
230 RETURN
240 END
```
APPENDIX F. PROGRAM LISTINGS FOR CLASSICAL LINEARIZATION

C **************************************************************
C * PROGRAM NU2KA *
C * THIS PROGRAM SOLVES THE NONLINEAR SECOND ORDER DIFFERENTIAL *
C * EQUATION: *
C * U'' - U**2 = 6 - 9X**4; UEXACT=3X**2 WITH VARIABLE DOMAIN *
C * BY LINEARIZING THE U**2 TERM AS USTAR*U AND KEEPING IT ON THE *
C * LEFT SIDE OF THE EQUATION. THE USER SELECTS: *
C * 1) NUMBER OF ELEMENTS *
C * 2) SIZE OF DOMAIN *
C * 3) X AND U(X) AT THE LEFT BOUNDARY *
C * 4) U(X) OR U'(X) AT THE RIGHT BOUNDARY *
C * 5) ITERATION STRATEGY FOR DETERMINING U* *
C * 6) APPROXIMATION TECHNIQUE FOR THE EXCITATION INTEGRAL *
C **************************************************************

110 COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),
:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC, 
:TLEN,ICORR(100,2),ITYPE,NEL,NSNP
115 CONV=.0001

C READ IN PARAMETERS FROM DATA FILE
130 READ(29,*) NEL,TLEN,COORD(1),ULBC,ITYPE,URBC
C CALCULATE NUMBER OF NODAL POINTS
135 NSNP=NEL+1
C DETERMINE ELEMENT SIZE OF EQUAL LENGTHS
137 ELEN=TLEN/FLOAT(NEL)
C ESTABLISH LOCAL TO GLOBAL CORRESPONDENCE AND X COORDINATE OF
C EACH NODE
160 DO 169 IEL=1,NEL
162 ICORR(IEL,1)=IEL
163 ICORR(IEL,2)=IEL+1
164 COORD(IEL+1)=COORD(IEL)+ELEN
169 CONTINUE
C CALL SUBROUTINE NU2CAM TO CREATE A MATRIX AND F VECTOR
170 CALL NU2KAM
C CALL SUBROUTINE NU2CAI TO PERFORM SOLUTION ITERATION
180 CALL NU2KAI(IET)

125
CALL SUBROUTINE U2EXTA TO COMPUTE EXACT SOLUTION U=3X**2

CALL CLEXTA

CALL SUBROUTINE OUTPUT TO PRINT OUT DATA, COMPUTATIONAL EFFICIENCY

CALL CLOTPT(CPUSTAR, IET)

END
**SUBROUTINE NU2KAM**

* THIS SUBROUTINE COMPUTES THE A MATRIX AND F VECTOR FOR MAIN PROGRAM NU2KA.

**COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100), : UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC, : TLEN,ICORR(100,2),ITYPE,NEL,NSNP**

**DIMENSION AE(2,2), FS1E(2), FS2E(2)**

**IF (TLEN.LE.1.0) THEN**

**PRINT*: 'CHOOSE BOUNDARY FOR INITIAL GUESS:'**

**PRINT*: '1 = LEFT ESSENTIAL BOUNDARY CONDITION'**

**PRINT*: '2 = RIGHT ESSENTIAL BOUNDARY CONDITION'**

**PRINT*: '3 = AVERAGE OF THE TWO ESSENTIAL BOUNDARY CONDITIONS'**

**READ(6,*) INITGS**

**ELSE**

**CONTINUE**

ENDIF

**DO 210 IZ = 1,NSNP**

**ZERO OUT STEADY FORCE VECTOR**

**FS(IZ) = 0.**

**Determine initial value of USTAR to begin the iteration process**

**IF (INITGS.EQ.1) THEN**

**U(IZ) = ULCB**

**UOLD(IZ) = U(IZ)**

**ELSEIF (INITGS.EQ.2) THEN**

**U(IZ) = URBC**

**UOLD(IZ) = U(IZ)**

**ELSEIF (INITGS.EQ.3) THEN**

**U(IZ) = (ULBC+URBC)/2.**

**UOLD(IZ) = U(IZ)**

**ELSE**

**U(IZ) = SQRT(ABS(9.*COORD(IZ)**4 - 6.))**

**UOLD(IZ) = U(IZ)**

**ENDIF**

**ZERO OUT ALL MATRICES**

**DO 200 JZ = 1,NSNP**

**A(IZ,JZ) = 0.**

**B(IZ,JZ) = 0.**

**C(IZ,JZ) = 0.**

**CONTINUE**

**CONTINUE**

**ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR**

**ALPHA = 0.**

**DO 375 IEL=1,NEL**

**AE(1,1) = 1./ELEN**

127
230 \ AE(1,2)=(-1./ELEN)
240 \ AE(2,1)=AE(1,2)
250 \ AE(2,2)=AE(1,1)
260 \ FS1E(1)=3.*ELEN
270 \ FS1E(2)=FS1E(1)
272 \ F1=(ALPHA**4)*ELEN/2.
274 \ F2=2.*((ALPHA**3)*(ELEN**2))/3.
276 \ F3=((ALPHA**2)*(ELEN**3))/2.
278 \ F4=ALPHA*(ELEN**4)/5.
280 \ F5=(ELEN**5)/30.
287 \ FS2E(1)=(-9.)*(F1 + F2 + F3 + F4 + F5)
290 \ FS2E(2)=(-9.)*(F1 + 2.*F2 + 3.*F3 + 4.*F4 + 5.*F5)
300 \ DO 370 II=1,2
310 \ \ \ DO 350 JJ=1,2
320 \ \ \ IN=ICORR(IEL,II)
330 \ \ \ JN=ICORR(IEL,JJ)
340 \ \ \ A(IN,JN)=A(IN,JN) - AE(II,JJ)
350 \ \ \ CONTINUE
360 \ \ \ CONTINUE
370 \ \ \ FS(IN)=FS1E(II) + FS2E(II) + FS(IN)
372 \ \ \ ALPHA=ALPHA + ELEN
375 \ \ \ CONTINUE
420 \ RETURN
430 \ END
**SUBROUTINE NU2KAI**

* THIS SUBROUTINE PERFORMS THE ITERATIVE SOLUTION PROCESS FOR *
* MAIN PROGRAM NU2KA. *

*************************************************************************

SUBROUTINE NU2KAI(IET)

COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),
UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
TLEN,ICORR(100,2),ITYPE,NEL,NSNP

DIMENSION WKAREA(40600), DIF(100), BE(2,2), USTAR(100), FT(100)

C SELECT METHOD OF DETERMINING USTAR

PRINT*, 'SELECT METHOD OF DETERMINATION.'
PRINT*, '1: U* = U'
PRINT*, '2: U* = AVERAGE OF LAST TWO COMPUTED VALUES OF U'
READ(6,*) METHU

C SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR

PRINT*, 'SELECT METHOD OF DETERMINING UNSTEADY B MATRIX.'
PRINT*, '1: MIDPOINT APPROXIMATION FOR U OVER THE ELEMENT.'
PRINT*, '2: U LINEARIZED OVER THE LENGTH OF THE ELEMENT.'
READ(6,*) METHBM

C CALL SUBROUTINE SETIME TO BEGIN TIMING ITERATION PROCESS

CALL SETIME

C BEGIN ITERATION PROCESS

DO 450 ITER=1,200

C DETERMINE VALUE OF U* AT EACH NODE

DO 138 IU=1,NSNP
IF (METHU.EQ.1) THEN
USTAR(IU)=U(IU)
ELSE
USTAR(IU)=(U(IU)+UOLD(IU))/2.
ENDIF
138 CONTINUE

C DETERMINE UNSTEADY ELEMENT B MATRIX

DO 210 IEL=1,NEL
IF (METHBM.EQ.1) THEN
BE(1,1)=(ELEN/6.)*(USTAR(IEL)+USTAR(IEL+1))
BE(1,2)=(ELEN/12.)*(USTAR(IEL)+USTAR(IEL+1))
ELSE
BE(1,1)=BE(1,2)
BE(2,1)=BE(1,2)
BE(2,2)=BE(1,1)
ENDIF
210 CONTINUE
BE(2,1)=BE(1,2)
BE(2,2)=\frac{ELEN}{12.}\ast (USTAR(IEL) + 3.\ast USTAR(IEL+1))
ENDIF
DO 200 II=1,2
DO 195 JJ=1,2
IN=ICORR(IEL,II)
JN=ICORR(IEL,JJ)
B(IN,JN)=BE(II,JJ) + B(IN,JN)
CONTINUE
DO 200 CONTINUE
DO 210 CONTINUE
C DETERMINE TOTAL SYSTEM MATRIX
DO 240 IP=1,NSNP
DO 232 JP=1,NSNP
CONTINUE
DO 240 CONTINUE
C RESET B MATRIX TO ZERO AND LET UOLD=U AND FT=FS
B(IP,JP)=0.
CONTINUE
UOLD(IP)=U(IP)
FT(IP)=FS(IP)
CONTINUE
C IMPOSE BOUNDARY CONDITIONS
C(1,1)=1.
C(1,2)=0.
FT(1)=ULBC
IF (ITYPE.EQ.1) THEN
C(NSNP,NSNP-1)=0.
C(NSNP,NSNP)=1.
FT(NSNP)=URBC
ELSE
FT(NSNP)=FT(NSNP)-URBC
ENDIF
M=1
IDGT=3
IQ=100
C CALL SUBROUTINE LEQT2F TO SOLVE SET OF LINEAR ALGEBRAIC EQUATIONS
CALL LEQT2F(C,M,NSNP,IQ,FT,IDGT,WKAREA,IER)
DO 310 NEW=1,NSNP
U(NEW)=FT(NEW)
WRITE(*,*),'U(NEW)=-',U(NEW)
C TEST FOR CONVERGENCE
DIF(NEW)=ABS(U(NEW)-UOLD(NEW))
CONTINUE
DIFMAX=DIF(1)
NMAX=1
DO 390 IJ=1,NEL
340 IF (DIF(IJ+1).GE.DIF(IJ)) THEN
350   DIFMAX=DIF(IJ+1)
355   NMAX=IJ+1
360 ELSE
370   CONTINUE
380 ENDIF
390 CONTINUE
405 IF (ABS(DIFMAX/U(NMAX)).LT.CONV) THEN
410   GO TO 460
420 ELSE
430   CONTINUE
440 ENDIF
450 CONTINUE
460 CONTINUE

C CALL SUBROUTINE GETIME TO OBTAIN CPU TIME FOR ITERATION PROCESS

461 CALL GETIME(IET)

C OUTPUT HEADER INFORMATION

462 WRITE(6,464)
463 WRITE(30,464)
464 FORMAT(1X,'EQUATION: U" - U**2 = 6 - 9X**4')
465 IF (ITYPE.EQ.1) THEN
466   WRITE(6,468) COORD(1),ULBC,COORD(NSNP),URBC
467   WRITE(30,468) COORD(1),ULBC,COORD(NSNP),URBC
468   FORMAT(1X,'B.C.: U(',F2.0,')=',F2.0,'; DU/DX(',F2.0,')=',F4.0,/)  
469 ELSE
470   WRITE(6,472) COORD(1),ULBC,COORD(NSNP),URBC
471   WRITE(30,472) COORD(1),ULBC,COORD(NSNP),URBC
472   FORMAT(1X,'B.C.: U(',F2.0,')=',F2.0,'); DU/DX(',F2.0,')=',F4.0,/)  
473 ENDIF
475 IF (METHU.EQ.1) THEN
476   WRITE(6,478)
477   WRITE(30,478)
478   FORMAT(1X,'ITERATION METHOD: U*='U',/)  
479 ELSE
480   WRITE(6,482)
481   WRITE(30,482)
482   FORMAT(1X,'ITERATION METHOD: U*=(U+UOLD)/2',/)  
487 ENDIF
488 IF (METHBM.EQ.1) THEN
489   WRITE(6,491)
490   WRITE(30,491)
491   FORMAT(1X,'METHOD OF B MATRIX INTEGRAL EVALUATION: MIDPOINT',/)  
496 ELSE
497   WRITE(6,499)
498   WRITE(30,499)
499   FORMAT(1X,'METHOD OF B MATRIX INTEGRAL EVALUATION: LINEAR',/)  
500 ENDIF
502 IF (ITER.GE.200) THEN
503   WRITE(6,505)
504   WRITE(30,505)
505   FORMAT(1X,'CONVERGENCE NOT OBTAINED AFTER 200 ITERATIONS.')  
506 ELSEIF (ABS(U(NMAX)).GT.(10.**20).OR.ABS(U(NSNP-1)).GT.1.0E-30) THEN
131
: (10. ** 20)) THEN
507   WRITE(6, 509)
508   WRITE(30, 509)
509   FORMAT(1X, 'SOLUTION PROCESS DIVERGES.')
510 ELSE
511   WRITE(6, 520) ITER, NEL
515   WRITE(30, 520) ITER, NEL
520   FORMAT(1X, 'CONVERGENCE OBTAINED AFTER ', I3, ' ITERATIONS USING ',
525   : I3, ' ELEMENTS.')
525   ENDIF
530 RETURN
540 END
SUBROUTINE CLEXTA

* THIS SUBROUTINE COMPUTES THE EXACT SOLUTION, U=3X**2, FOR
* MAIN PROGRAM NU2KA AT THE SPECIFIED NODAL POINTS.

100 SUBROUTINE CLEXTA
110 COMMON A(100,100),B(100,100),C(100,100),U(100),UOLD(100),
:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
:TLEN,ICORR(100,2),ITYPE,NEL,NSNP
130 DO 150 NN = 1,NSNP
140 UEXT(NN) = 3.*COORD(NN)**2
150 CONTINUE
160 RETURN
170 END
PROGRAM NU2KB

THIS PROGRAM SOLVES THE NONLINEAR SECOND ORDER DIFFERENTIAL EQUATION:

\[ U'' + U^2 = 60X + 100X^6; \quad \text{UEXACT}=10X^3 \]

DOMAIN BY LINEARIZING THE \( U^2 \) TERM AS \( U^* U \) AND KEEPING IT ON THE LEFT SIDE OF THE EQUATION. THE USER SELECTS:

1) NUMBER OF ELEMENTS
2) SIZE OF DOMAIN
3) \( X \) AND \( U(X) \) AT THE LEFT BOUNDARY
4) \( U(X) \) OR \( U'(X) \) AT THE RIGHT BOUNDARY
5) ITERATION STRATEGY FOR DETERMINING \( U^* \)
6) APPROXIMATION TECHNIQUE FOR THE EXCITATION INTEGRAL

COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),
:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
:ULBC,ITYPE,NEL,NSNP

CONV=.0001

READ IN PARAMETERS FROM DATA FILE

READ(29,*) NEL,TLEN,COORD(1),ULBC,ITYPE,URBC

CALCULATE NUMBER OF NODAL POINTS

NSNP=NEL+1

DETERMINE ELEMENT SIZE OF EQUAL LENGTHS

ELEN=TLEN/FLOAT(NEL)

ESTABLISH LOCAL TO GLOBAL CORRESPONDENCE AND X COORDINATE OF EACH NODE

DO 169 IEL=1,NEL
162 ICORR(IEL,1)=IEL
163 ICORR(IEL,2)=IEL+1
164 COORD(IEL+1)=COORD(IEL)+ELEN
169 CONTINUE

CALL SUBROUTINE NU2KBM TO CREATE A MATRIX AND F VECTOR

CALL NU2KBM

CALL SUBROUTINE NU2KBI TO PERFORM SOLUTION ITERATION

CALL NU2KBI(IET)

CALL SUBROUTINE CLEXTB TO COMPUTE EXACT SOLUTION \( U=3X^2 \)

CALL CLEXTB

CALL SUBROUTINE CLOTPT TO PRINT OUT DATA, COMPUTATIONAL EFFICIENCY
200  CALL CLOTPT(CPUSTAR,IET)
210  END
*** SUBROUTINE NU2KBM ***

* THIS SUBROUTINE COMPUTES THE A MATRIX AND F VECTOR FOR MAIN PROGRAM NU2KB.

*** SUBROUTINE NU2KBM ***

** COMMON A(100,100),B(100,100),C(100,100),U(100),UOLD(100),
: UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
: TLEN,ICORR(100,2),ITYPE,NEL,NSNP

DIMENSION AE(2,2), FS1E(2), FS2E(2)

ZERO OUT A MATRIX AND ALL VECTORS

DO 210 IZ = 1,NSNP
FS(IZ) = 0.
U(IZ) = SQRT(60.*COORD(IZ) + 100.*COORD(IZ)**6)
UOLD(IZ) = 0.
DO 200 JZ = 1,NSNP
ACIZ,JZ) = 0.
B(IZ,JZ) = 0.
C(IZ,JZ) = 0.
200 CONTINUE
210 CONTINUE

ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR

ALPHA=0.
DO 375 IEL=1,NEL
AE(1,1)=1./ELEN
AE(1,2)=(-1./ELEN)
AE(2,1)=AE(1,2)
AE(2,2)=AE(1,1)
FS1E(1)=30.*ALPHA*ELEN + 10.*ELEN**2
FS1E(2)=30.*ALPHA*ELEN + 20.*ELEN**2
F1=50.*ALPHA**6)*ELEN
F2=100.*ALPHA**5)*(ELEN**2)
F3=125.*ALPHA**4)*(ELEN**3)
F4=100.*ALPHA**3)*(ELEN**4)
F5=50.*ALPHA**2)*(ELEN**5)
F6=100.*ALPHA*(ELEN**6)/7.
F7=25.*(ELEN**7)/16.
FS2E(1)=F1 + F2 + F3 + F4 + F5 + F6 + F7
FS2E(2)=F1 + 2.*F2 + 3.*F3 + 4.*F4 + 5.*F5 + 6.*F6 + 7.*F7
DO 370 II=1,2
DO 350 JJ=1,2
IN=ICORR(IEL,II)
JN=ICORR(IEL,JJ)
A(IN,JN)=A(IN,JN) - AE(II,JJ)
340 CONTINUE
350 CONTINUE
FS(IN)=FS1E(II) + FS2E(II) + FS(IN)
370 CONTINUE
372 ALPHA=ALPHA + ELEN
375 CONTINUE
420      RETURN
430      END
SUBROUTINE NU2KBI

* THIS SUBROUTINE PERFORMS THE ITERATIVE SOLUTION PROCESS FOR *
* MAIN PROGRAM NU2KA. *

SUBROUTINE NU2KBI(IET)

COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
:TLEN,ICORR(100,2),ITYPE,NEL,NSNP

DIMENSION WKAREA(40600), DIF(100), BE(2,2), USTAR(100), FT(100)

SELECT METHOD OF DETERMINING USTAR

PRINT*,'SELECT METHOD OF DETERMINATION.'
PRINT*,'1: U* = U'
PRINT*,'2: U* = AVERAGE OF LAST TWO COMPUTED VALUES OF U'
PRINT*,'3: U* = WEIGHTED AVERAGE OF LAST TWO COMPUTED VALUES OF U'
READ(6,*) METHU
IF (METHU.EQ.3) THEN
PRINT*,'CHOOSE WEIGHTING VALUES A AND B'
READ(6,*) AW,BW
ELSE
CONTINUE
ENDIF

SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR

PRINT*,'SELECT METHOD OF DETERMINING UNSTEADY B MATRIX.'
PRINT*,'1: MIDPOINT APPROXIMATION FOR U OVER THE ELEMENT.'
PRINT*,'2: U LINEARIZED OVER THE LENGTH OF THE ELEMENT.'
READ(6,*) METHBM

CALL SUBROUTINE SETIME TO BEGIN TIMING ITERATION PROCESS

CALL SETIME

BEGIN ITERATION PROCESS

DO 450 ITER=1,200

DETERMINE VALUE OF U* AT EACH NODE

DO 138 IU=1,NSNP
IF (METHU.EQ.1) THEN
USTAR(IU)=U(IU)
ELSEIF (METHU.EQ.2) THEN
USTAR(IU)=(U(IU)+UOLD(IU))/2.
ELSE
USTAR(IU)=(AW*U(IU)+BW*UOLD(IU))/(AW+BW)
ENDIF
CONTINUE

DETERMINE UNSTEADY ELEMENT B MATRIX
DO 210 IEL=1,NEL
   IF (METHBM.EQ.1) THEN
      BE(1,1)=(ELEN/6.)*(USTAR(IEL)+USTAR(IEL+1))
      BE(1,2)=(ELEN/12.)*(USTAR(IEL)+USTAR(IEL+1))
      BE(2,1)=BE(1,2)
      BE(2,2)=BE(1,1)
   ELSE
      BE(1,1)=(ELEN/12.)*(3.*USTAR(IEL) + USTAR(IEL+1))
      BE(1,2)=(ELEN/12.)*(USTAR(IEL) + USTAR(IEL+1))
      BE(2,1)=BE(1,2)
      BE(2,2)=(ELEN/12.)*(USTAR(IEL) + 3.*USTAR(IEL+1))
   ENDIF
   DO 200 II=1,2
      IN=ICORR(IEL,II)
      JN=ICORR(IEL,JJ)
      B(IN,JN)=BECII,JJ)+BCIN,JN)
   CONTINUE
200 CONTINUE
210 CONTINUE

C DETERMINE TOTAL SYSTEM MATRIX

DO 240 IP=1,NSNP
   DO 232 JP=1,NSNP
   CONTINUE
232 CONTINUE
240 CONTINUE

C RESET B MATRIX TO ZERO AND LET UOLD=U AND FT=FS

B(IP,JP)=0.
   CONTINUE
235 UOLD(IP)=U(IP)
236 FT(IP)=FS(IP)
240 CONTINUE

C IMPOSE BOUNDARY CONDITIONS

C(1,1)=1.
C(1,2)=0.
FT(1)=ULBC
IF (ITYPE.EQ.1) THEN
   C(NSNP,NSNP-1)=0.
   C(NSNP,NSNP)=1.
   FT(NSNP)=URBC
ELSE
   FT(NSNP)=FT(NSNP)-URBC
ENDIF
M=1
IDGT=3
IQ=100

C CALL SUBROUTINE LEQT2F TO SOLVE SET OF LINEAR ALGEBRAIC EQUATIONS

CALL LEQT2F(C,M,NSNP,IQ,FT,IDGT,WKAREA,IER)
DO 310 NEW=1,NSNP
U(NEW) = FT(NEW)
WRITE(*,*) 'UNEW=', U(NEW)

C TEST FOR CONVERGENCE
DIF(NEW) = ABS(U(NEW) - UOLD(NEW))
CONTINUE
DIFMAX = DIF(1)
NMAX = 1
DO 390 IJ = 1, NEL
   IF (DIF(IJ+1) .GE. DIF(IJ)) THEN
      DIFMAX = DIF(IJ+1)
      NMAX = IJ+1
   ELSE
      CONTINUE
   ENDIF
390 CONTINUE
DIFMAX = -DIF(NMAX)
NMAX = NMAX + 1
IF (ABS(DIFMAX/U(NMAX)) .LT. CONV) THEN
   GO TO 460
ELSE
   CONTINUE
ENDIF
460 CONTINUE

C CALL SUBROUTINE GETIME TO OBTAIN CPU TIME FOR ITERATION PROCESS
CALL GETIME(IET)

C OUTPUT HEADER INFORMATION
WRITE(6,464)
WRITE(30,464)
FORMAT(1X, 'EQUATION: U'' + U**2 = 60X + 100X**6')
IF (ITYPE .EQ. 1) THEN
   WRITE(6,468) COORD(1), ULBC, COORD(NSNP), URBC
   WRITE(30,468) COORD(1), ULBC, COORD(NSNP), URBC
   FORMAT(1X, 'B.C.: U(',F2.0,')=',F3.0,'; U(',F2.0,')=',F5.0,'/)
ELSE
   WRITE(6,472) COORD(1), ULBC, COORD(NSNP), URBC
   WRITE(30,472) COORD(1), ULBC, COORD(NSNP), URBC
   FORMAT(1X, 'B.C.: U(',F2.0,')=',F3.0,'; DU/DX(',F2.0,')=',F5.0,'/)
ENDIF
IF (METHU .EQ. 1) THEN
   WRITE(6,478)
   WRITE(30,478)
   FORMAT(1X, 'ITERATION METHOD: U*=U',/)
ELSEIF (METHU .EQ. 2) THEN
   WRITE(6,482)
   WRITE(30,482)
   FORMAT(1X, 'ITERATION METHOD: U*=(U+UOLD)/2',/)
ELSE
   WRITE(6,486) AW, BW, AW, BW
   WRITE(30,486) AW, BW, AW, BW
   FORMAT(1X, 'ITERATION METHOD: U*=' ,F3.0,'*U +',F3.0,'*UOLD)/(',

ENDIF

IF (ITER.GE.200) THEN
WRITE(6,505)
WRITE(30,505)
FORMAT(1X,'CONVERGENCE NOT OBTAINED AFTER 200 ITERATIONS.')
ELSEIF (ABS(U(NMAX)).GT. (10. **20).OR. ABS(U(NSNP-1)).GT. (10. **20)) THEN
WRITE(6,509)
WRITE(30,509)
FORMAT(1X,'SOLUTION PROCESS DIVERGES.')
ELSE
WRITE(6,520) ITER,NEL
WRITE(30,520) ITER,NEL
FORMAT(1X,'CONVERGENCE OBTAINED AFTER ',I3,' ITERATIONS USING ', I3,' ELEMENTS. ')
ENDIF
RETURN
END
*SUBROUTINE CLEXTB*

*THIS SUBROUTINE COMPUTES THE EXACT SOLUTION, U=10X**3, FOR*

*MAIN PROGRAM NU2KB AT THE SPECIFIED NODAL POINTS.*

```fortran
100 SUBROUTINE CLEXTB
110 COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),
:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
 :TLEN,ICORR(100,2),ITYPE,NEL,NSNP
130 DO 150 NN = 1,NSNP
140 UEXT(NN) = 10.*COORD(NN)**3
150 CONTINUE
160 RETURN
170 END
```
**SUBROUTINE CLOTPT**

* This subroutine computes the per cent error between the exact and FEM solutions, CPU* for the iteration process, and prints out all data in tabular form for main programs NU2KA & NU2KB.

**----------------------------------------------------------------------------------------------------------------------------------**

**SUBROUTINE CLOTPT(CPUSTAR, IET)**

**COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLED(100),**
**:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,**
**:TLEN,ICORR(100,2),ITYPE,NEL,NSNP**

**SUMDIF=0.**

**CALCULATE PER CENT ERROR AT EACH NODE AND SUM THE ABSOLUTE VALUE OF ALL THE ERRORS**

**DO 150**

**Ik=2,NSNP**

**UDIF(IK)=100.*((U(IK)-UEXT(IK))/UEXT(IK))**

**SUMDIF=SUMDIF+ABS(UDIF(IK))**

**CONTINUE**

**UDIF(1)=U(1)-UEXT(1)**

**COMPUTE THE ELAPSED TIME OF THE ITERATION PROCESS**

**ELTIME=IET*. 000026**

**WRITE(6,169) ELTIME**

**WRITE(30,169)ELTIME**

**FORMAT(1X,'ELAPSED TIME FOR THE ITERATION PROCESS IS ',F9.4,**

**: SECONDS.')**

**OUTPUT DATA IN TABULAR FORMAT**

**WRITE(6,180)**

**WRITE(30,180)**

**FORMAT(/,1X,'COORD',3X,'U EXACT',3X,'U FEM',7X,'% DIFF')**

**WRITE(6,200) (COORD(NP),UEXT(NP),U(NP),UDIF(NP), NP=1,NSNP)**

**WRITE(30,200) (COORD(NP),UEXT(NP),U(NP),UDIF(NP), NP=1,NSNP)**

**FORMAT(/,2X,FS.3,4X,F9.4,3X,F9.4,4X,F. 1)**

**CALCULATE CPU* FOR THE ITERATION PROCESS**

**CPUSTAR=ELTIME*SUMDIF/NSNP**

**WRITE(6,220) CPUSTAR**

**WRITE(30,220? CPUSTAR**

**FORMAT(/,1X,'CPU* FOR THE ITERATION PROCESS IS ',F9.4,' SECONDS.')**

**RETURN**

**END**
APPENDIX G. PROGRAM LISTINGS FOR QUASILINEARIZATION

C ******************** ******************************
C * PROGRAM NU2QA *
C * THIS PROGRAM SOLVES THE NONLINEAR SECOND ORDER DIFFERENTIAL *
C * EQUATION: *
C * U'' - U**2 = 6 - 9X**4; UEXACT=3X**2 WITH VARIABLE DOMAIN *
C * BY THE PROCESS OF QUASILINEARIZATION. THE USER SELCTS: *
C * 1) NUMBER OF ELEMENTS *
C * 2) SIZE OF DOMAIN *
C * 3) X AND U(X) AT THE LEFT BOUNDARY *
C * 4) U(X) OR U'(X) AT THE RIGHT BOUNDARY *
C * 5) ITERATION STRATEGY FOR DETERMINING U* *
C * 6) INTERPOLATION STRATEGY FOR THE B MATRIX INTEGRAL *
C * 7) INTERPOLATION STRATEGY FOR THE EXCITATION INTEGRAL *
C ******************** ******************************

110 COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UO(100), 
:UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC, 
:ICORR(100,2),ITYPE,NEL,NSNP
115 CONV=.0001
C READ IN PARAMETERS FROM DATA FILE
130 READ(29,*) NEL,TLEN,COORD(1),ULBC,ITYPE,URBC
C CALCULATE NUMBER OF NODAL POINTS
135 NSNP=NEL+1
C DETERMINE ELEMENT SIZE OF EQUAL LENGTHS
137 ELEN=TLEN/FLOAT(NEL)
C ESTABLISH LOCAL TO GLOBAL CORRESPONDENCE AND X COORDINATE OF 
C EACH NODE
160 DO 169 IEL=1,NEL
162 ICORR(IEL,1)=IEL
163 ICORR(IEL,2)=IEL+1
164 COORD(IEL+1)=COORD(IEL)+ELEN
169 CONTINUE
C CALL SUBROUTINE NU2QAM TO CREATE A MATRIX AND F VECTOR
170 CALL NU2QAM
C CALL SUBROUTINE NU2QAI TO PERFORM SOLUTION ITERATION
180 CALL NU2QAI(IET)
C CALL SUBROUTINE QLEXTA TO COMPUTE EXACT SOLUTION U=3X**2
190 CALL QLEXTA

C CALL SUBROUTINE QLOTPT TO PRINT OUT DATA, COMPUTATIONAL EFFICIENCY

200 CALL QLOTPT(CPUSTAR, IET)
210 END
**SUBROUTINE NU2QAM**

**THIS SUBROUTINE COMPUTES THE A MATRIX AND STEADY F VECTOR FOR**
**MAIN PROGRAM NU2QA.**

ZERO OUT A MATRIX AND ALL VECTORS

DO 210 IZ = 1, NSNP
FS(IZ) = 0.
U(IZ) = SQRT(ABS(9. * COORD(IZ)**4 - 6.))
UOLD(IZ) = 0.
DO 200 JZ = 1, NSNP
A(IZ, JZ) = 0.
B(IZ, JZ) = 0.
C(IZ, JZ) = 0.
CONTINUE
210 CONTINUE

ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR

ALPHA = 0.
DO 375 IEI = 1, NEL
AE(1, 1) = 1. / ELEN
AE(1, 2) = (-1. / ELEN)
AE(2, 1) = AE(1, 2)
AE(2, 2) = AE(1, 1)
FS1E(1) = 3. * ELEN
FS1E(2) = FS1E(1)
FS2E(1) = (-9.)* (F1 + F2 + F3 + F4 + F5)
FS2E(2) = (-9.)* (F1 + 2.*F2 + 3.*F3 + 4.*F4 + 5.*F5)
DO 370 II = 1, 2
IN = ICORR(IEI, II)
JN = ICORR(IEI, JJ)
A(IN, JN) = A(IN, JN) - AE(II, JJ)
CONTINUE
FS(IN) = FS1E(II) + FS2E(II) + FS(IN)
CONTINUE
ALPHA = ALPHA + ELEN
CONTINUE
RETURN
END
SUBROUTINE NU2QAI

THIS SUBROUTINE PERFORMS THE ITERATIVE SOLUTION PROCESS FOR THE MAIN PROGRAM NU2QA.

COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,ICORR(100,2),ITYPE,NEL,NSNP
DIMENSION WKAREA(40600), DIF(100), BE(2,2), USTAR(100), FT(100), FUE(2),FU(100)

SELECT METHOD OF DETERMINING USTAR

SELECT METHOD OF DETERMINING UNSTEADY B MATRIX

SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR

CALL SUBROUTINE SETIME TO BEGIN TIMING ITERATION PROCESS

CALL SETIME

BEGIN ITERATION PROCESS

DO 450 ITER=1,200

DETERMINE VALUE OF U* AT EACH NODE AND SET VALUE OF UNSTEADY FORCE VECTOR TO ZERO

DO 138 IU=1,NSNP
FU(IU)=0.
IF (METHU.EQ.1) THEN
USTAR(IU)=U(IU)
ELSEIF (METHU.EQ.2) THEN
USTAR(IU)=(U(IU)+UOLD(IU))/2.
ELSE
USTAR(IU)=(AW*U(IU)+BW*UOLD(IU))/(AW+BW)
ENDIF
CONTINUE

C DETERMINE UNSTEADY ELEMENT B MATRIX

DO 210 IEL=1,NEL
IF (METHBM.EQ.1) THEN
    BE(1,1)=(ELEN/3.)*(USTAR(IEL)+USTAR(IEL+1))
    BE(1,2)=(ELEN/6.)*(USTAR(IEL)+USTAR(IEL+1))
    BE(2,1)=BE(1,2)
    BE(2,2)=BE(1,1)
ELSE
    BE(1,1)=(ELEN/6.)*(3.*USTAR(IEL)+USTAR(IEL+1))
    BE(1,2)=(ELEN/6.)*(USTAR(IEL)+USTAR(IEL+1))
    BE(2,1)=BE(1,2)
    BE(2,2)=(ELEN/6.)*(USTAR(IEL)+3.*USTAR(IEL+1))
ENDIF

C DETERMINE SYSTEM B MATRIX BY DISTRIBUTING ELEMENT B MATRICES
C ACCORDING TO THE LOCAL TO GLOBAL CORRESPONDENCE

DO 163 II=1,2
    DO 162 JJ=1,2
        IN=ICORR(IEL,II)
        JN=ICORR(IEL,JJ)
        B(IN,JN)=BE(II,JJ) + B(IN,JN)
    CONTINUE
163 CONTINUE

C DETERMINE UNSTEADY ELEMENT FORCE VECTOR

IF (METHFU.EQ.1) THEN
    FUE(1)=(ELEN/2.)*(USTAR(IEL)+USTAR(IEL+1))/2.)**2
    FUE(2)=FUE(1)
ELSEIF (METHFU.EQ.2) THEN
    FUE(1)=(ELEN/2.)*(3.*USTAR(IEL)+USTAR(IEL+1))/4. + USTAR(IEL+1)/4.)**2
    FUE(2)=(ELEN/2.)*(USTAR(IEL)+3.*USTAR(IEL+1))/4.)**2
ELSE
    FUE(1)=ELEN*(USTAR(IEL)**2/4. + USTAR(IEL)*USTAR(IEL+1)/6. + USTAR(IEL+1)**2/12.)
    FUE(2)=ELEN*(USTAR(IEL)**2/12. + USTAR(IEL)*USTAR(IEL+1)/6. + USTAR(IEL+1)**2/4.)
ENDIF

C DETERMINE UNSTEADY SYSTEM FORCE VECTOR BY DISTRIBUTING ELEMENTAL
C FORCE VECTORS ACCORDING TO THE LOCAL TO GLOBAL CORRESPONDENCE

DO 180 II=1,2
    IN=ICORR(IEL,II)
    FU(IN)=FUE(II) + FU(IN)
CONTINUE
CONTINUE

C DETERMINE TOTAL SYSTEM MATRIX

DO 240 IP=1,NSNP
   DO 232 JP=1,NSNP
   CONTINUE

C RESET B MATRIX TO ZERO

B(IP,JP)=0.
CONTINUE

C UPDATE VALUE OF U AT THE PREVIOUS ITERATION

UOLD(IP)=U(IP)

C DETERMINE TOTAL SYSTEM FORCE VECTOR

FT(IP)=FS(IP)-FU(IP)
CONTINUE

C IMPOSE BOUNDARY CONDITIONS

C(1,1)=1.
C(1,2)=0.
FT(1)=ULBC
IF (ITYPE.EQ.1) THEN
   C(NSNP,NSNP-1)=0.
   C(NSNP,NSNP)=1.
   FT(NSNP)=URBC
ELSE
   FT(NSNP)=FT(NSNP)-URBC
ENDIF

M=1
IDGT=3
IQ=100

C CALL SUBROUTINE LEQT2F TO SOLVE SET OF LINEAR ALGEBRAIC EQUATIONS

CALL LEQT2F(C,M,NSNP,IQ,FT,IDGT,WKAREA,IER)

DO 310 NEW=1,NSNP
   U(NEW)=FT(NEW)
   WRITE(*,*),'UNEW=',U(NEW)
CONTINUE

DIF(MAX)=ABS(U(NEW)-UOLD(NEW))
CONTINUE

DIFMAX=DIF(1)
NMAX=1
DO 390 IJ=1,NEL
   IF (DIF(IJ+1).GE.DIF(IJ)) THEN
      DIFMAX=DIF(IJ+1)
      NMAX=IJ+1
   ENDIF
   DO 390 IJ=1,NEL
      IF (DIF(IJ+1).GE.DIF(IJ)) THEN
         DIFMAX=DIF(IJ+1)
         NMAX=IJ+1
      ENDIF
   CONTINUE

149
ELSE
    CONTINUE
ENDIF
CONTINUE
IF (U(NMAX).EQ.0.) GO TO 450
IF (ABS(DIFMAX/U(NMAX)).LT.CONV) THEN
    GO TO 460
ELSE
    CONTINUE
ENDIF
CONTINUE

CALL SUBROUTINE GETIME TO OBTAIN CPU TIME FOR ITERATION PROCESS

CALL GETIME(IET)

C OUTPUT HEADER INFORMATION

WRITE(6,464)
WRITE(30,464)
FORMAT(1X,'EQUATION:  U'' - U*U = 6 - 9X**4')
IF (ITYPE.EQ.1) THEN
    WRITE(6,468) COORD(1),ULBC,COORD(NSNP),URBC
    WRITE(30,468)
    FORMAT(1X,'B.C.: U(',F2.0,')=',F2.0,','; U(',F2.0,')=',F4.0,')/
ELSE
    WRITE(6,472) COORD(1),ULBC,COORD(NSNP),URBC
    WRITE(30,472)
    FORMAT(1X,'B.C.: U(',F2.0,')=',F2.0,','; DU/DX(',F2.0,')=',F4.0,')/
ENDIF
IF (METHU.EQ.1) THEN
    WRITE(6,478)
    WRITE(30,478)
    FORMAT(1X,'ITERATION METHOD: U*=-U',/) 
ELSEIF (METHU.EQ.2) THEN
    WRITE(6,482)
    WRITE(30,482)
    FORMAT(1X,'ITERATION METHOD: U*=(U+UOLD)/2',/) 
ELSE
    WRITE(6,486)AW,BW,AW,BW
    WRITE(30,486)
    FORMAT(1X,'ITERATION METHOD: U*=-(',F4.1,'*U + ',F4.1,'*UOLD)/(',F4.1,')',/) 
ENDIF
IF (METHBM.EQ.1) THEN
    WRITE(6,491)
    WRITE(30,491)
    FORMAT(1X,'METHOD OF B MATRIX INTEGRAL EVALUATION: MIDPOINT',/) 
ELSE
    WRITE(6,495)
    WRITE(30,495)
    FORMAT(1X,'METHOD OF B MATRIX INTEGRAL EVALUATION: LINEAR',/) 
ENDIF
IF (METHFU.EQ.1) THEN
    WRITE(6,504)
WRITE(30,504)
504 FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: MIDPOINT',/)
505 ELSEIF (METHFU.EQ.2) THEN
506 WRITE(6,508)
507 WRITE(30,508)
508 FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: 1/4-3/4',/)
509 ELSE
510 WRITE(6,512)
511 WRITE(30,512)
512 FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: LINEAR',/)
513 ENDIF
514 IF (ITER.GE.200) THEN
515 WRITE(6,518)
516 WRITE(30,518)
517 FORMAT(1X,'CONVERGENCE NOT OBTAINED AFTER 200 ITERATIONS. ')
518 ELSEIF (ABS(U(NMAX)).GT.(10.**20).OR.ABS(U(NSNP-1)).GT.
519 : (10.**20)) THEN
520 WRITE(6,522)
521 WRITE(30,522)
522 FORMAT(1X,'SOLUTION PROCESS DIVERGES. ')
523 ELSE
524 WRITE(6,526) ITER,NEL
525 WRITE(30,526) ITER,NEL
526 FORMAT(1X,'CONVERGENCE OBTAINED AFTER ',I3,' ITERATIONS USING ',
527 :I3,' ELEMENTS. ',/)
530 ENDIF
540 RETURN
550 END
**SUBROUTINE QLEXTA**

* THIS SUBROUTINE COMPUTES THE EXACT SOLUTION, \( U=3x^2 \), FOR MAIN PROGRAM NU2QA AT THE SPECIFIED NODAL POINTS.

100 SUBROUTINE QLEXTA
110 COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),
: UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
: ICORR(100,2),ITYPE,NEL,NSNP
130 DO 150 NN = 1,NSNP
140 UEXT(NN) = 3.*COORD(NN)**2
150 CONTINUE
160 RETURN
170 END
PROGRAM NU2QB

THIS PROGRAM SOLVES THE NONLINEAR SECOND ORDER DIFFERENTIAL EQUATION:

\[ \frac{\text{d}^2 U}{\text{d}x^2} + U^2 = 60x + 100x^6; \quad \text{U}_{\text{EXACT}}=10x^3 \]

DOMAIN BY THE PROCESS OF QUASILINEARIZATION. THE USER SELCTS:

1) NUMBER OF ELEMENTS
2) SIZE OF DOMAIN
3) X AND U(X) AT THE LEFT BOUNDARY
4) U(X) OR U'(X) AT THE RIGHT BOUNDARY
5) ITERATION STRATEGY FOR DETERMINING U*
6) INTERPOLATION STRATEGY FOR THE B MATRIX INTEGRAL
7) INTERPOLATION STRATEGY FOR THE EXCITATION INTEGRAL

110 COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),
:UXEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
:ICORR(100,2),ITYPE,NEL,NSNP
115 CONV=.0000001

READ IN PARAMETERS FROM DATA FILE

130 READ(29,*) NEL,TLEN,COORD(1),ULBC,ITYPE,URBC

CALCULATE NUMBER OF NODAL POINTS

135 NSNP=NEL+1

DETERMINE ELEMENT SIZE OF EQUAL LENGTHS

137 ELEN=TLEN/FLOAT(NEL)

ESTABLISH LOCAL TO GLOBAL CORRESPONDENCE AND X COORDINATE OF EACH NODE

160 DO 169 IEL=1,NEL
162 ICORR(IEL,1)=IEL
163 ICORR(IEL,2)=IEL+1
164 COORD(IEL+1)=COORD(IEL)+ELEN
169 CONTINUE

CALL SUBROUTINE NU2QBM TO CREATE A MATRIX AND F VECTOR

170 CALL NU2QBM

CALL SUBROUTINE NU2QBI TO PERFORM SOLUTION ITERATION

180 CALL NU2QBI(IET)

CALL SUBROUTINE QLEXTB TO COMPUTE EXACT SOLUTION U=3X**2

190 CALL QLEXTB

CALL SUBROUTINE QLOTPT TO PRINT OUT DATA, COMPUTATIONAL EFFICIENCY
200 CALL QLOTPT(CPUSTAR, IET)
210 END
***SUBROUTINE NU2QBM***

**THIS SUBROUTINE COMPUTES THE A MATRIX AND STEADY F VECTOR FOR**

**MAIN PROGRAM NU2QB.**

**ZERO OUT A MATRIX AND ALL VECTORS**

DO 210 IZ = 1,NSNP
   FS(IZ) = 0.
   IF (COORD(IZ).LE.1.) THEN
      U(IZ)=0.
   ELSE
      G1(IZ) = 60.*COORD(IZ) + 100.*COORD(IZ)**6
      G2(IZ) = 1500.*COORD(IZ)**4/SQRT(G1(IZ))
      G3(IZ) = (60. +600.*COORD(IZ)**5)**2/(4.*GlCIZ)**1.5)
      U(IZ)=SQRT(ABS(G1(IZ)-G2(IZ)+G3(IZ)))
   ENDIF

U(IZ) = SQRT(60.*COORD(IZ) + 100.*COORD(IZ)**6)
UOLD(IZ)= U(IZ)

DO 200 JZ = 1,NSNP
   A(IZ,JZ) = 0.
   B(IZ,JZ) = 0.
   C(IZ,JZ) = 0.

CONTINUE

C ELEMENTAL DO LOOP TO DETERMINE A MATRIX AND F VECTOR

DO 375 IEL=1,NEL
   AE(1,1)=1./ELEN
   AE(1,2)=(-1./ELEN)
   AE(2,1)=AE(1,2)
   AE(2,2)=AE(l,1)
   FS1E(1)=30.*ALP{A*ELEN + 10.*ELEN**2
   FS1E(2)=30. *ALPJ{A*ELEN + 20. *ELEN**2
   F1=50. *(ALPHA**(6))*ELEN
   F2=100. *(ALPHA**(5))*(ELEN**2)
   F3=125. *(ALPHA**(4))*(ELEN**3)
   F4=100. *(ALPHA**(3))*(ELEN**4)
   F5=50. *(ALPHA**(2))*(ELEN**5)
   F6=100. *ALPHA*(ELEN**6)/7.
   F7=25. *(ELEN**7)/16.
   FS2E(1)=F1 + F2 + F3 + F4 + F5 + F6 + F7
   FS2E(2)=F1 + 2.*F2 + 3.*F3 + 4.*F4 + 5.*F5 + 6.*F6 + 7*F7

CONTINUE

ALPHA=0.

DO 375 IEL=1,NEL
   AE(1,1)=1./ELEN
   AE(1,2)=(-1./ELEN)
   AE(2,1)=AE(1,2)
   AE(2,2)=AE(l,1)
   FS1E(1)=30.*ALP{A*ELEN + 10.*ELEN**2
   FS1E(2)=30. *ALPJ{A*ELEN + 20. *ELEN**2
   F1=50. *(ALPHA**(6))*ELEN
   F2=100. *(ALPHA**(5))*(ELEN**2)
   F3=125. *(ALPHA**(4))*(ELEN**3)
   F4=100. *(ALPHA**(3))*(ELEN**4)
   F5=50. *(ALPHA**(2))*(ELEN**5)
   F6=100. *ALPHA*(ELEN**6)/7.
   F7=25. *(ELEN**7)/16.
   FS2E(1)=F1 + F2 + F3 + F4 + F5 + F6 + F7
   FS2E(2)=F1 + 2.*F2 + 3.*F3 + 4.*F4 + 5.*F5 + 6.*F6 + 7*F7

CONTINUE

IN=ICORR(IEL,II)
330           JN=ICORR(IEL,JJ)
340           A(IN,JN)=A(IN,JN) - AE(II,JJ)
350           CONTINUE
360           FS(IN)=FS1E(II) + FS2E(II) + FS(IN)
370           CONTINUE
372           ALPHA=ALPHA + ELEN
375           CONTINUE
420           RETURN
430           END
SUBROUTINE NU2QBI

* THIS SUBROUTINE PERFORMS THE ITERATIVE SOLUTION PROCESS FOR *
* MAIN PROGRAM NU2QA. *

100 SUBROUTINE NU2QBI(IET)
102 COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),
      :UXEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
      :ICORR(100,2),ITYPE,NEL,NSNP
104 DIMENSION WKAREA(40600), DIF(100), BE(2,2), USTAR(100), FT(100),
      :FUE(2),FU(100)

C SELECT METHOD OF DETERMINING USTAR

105 PRINT*, 'SELECT METHOD OF U* DETERMINATION.'
106 PRINT*, '1: U* = U'
107 PRINT*, '2: U* = AVERAGE OF LAST TWO COMPUTED VALUES OF U'
108 PRINT*, '3: U* = WEIGHTED AVERAGE OF LAST TWO COMPUTED VALUES OF U'
109 READ(6,*) METHU
110 IF (METHU.EQ.3) THEN
111   PRINT*, 'CHOOSE WEIGHTING VALUES A AND B'
112   READ(6,*) AW,BW
113 ELSE
114   CONTINUE
115 ENDIF

C SELECT METHOD OF DETERMINING UNSTEADY B MATRIX

116 PRINT*, 'SELECT METHOD OF DETERMINING UNSTEADY B MATRIX.'
117 PRINT*, '1: MIDPOINT APPROXIMATION FOR U OVER THE ELEMENT.'
118 PRINT*, '2: U LINEARIZED OVER THE LENGTH OF THE ELEMENT.'
119 READ(6,*) METHBM

C SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR

121 PRINT*, 'SELECT METHOD OF DETERMINING UNSTEADY FORCE VECTOR.'
122 PRINT*, '1: MIDPOINT APPROXIMATION'
123 PRINT*, '2: 1/4 - 3/4 APPROXIMATION'
124 PRINT*, '3: LINEAR'
125 READ(6,*) METHFU

C CALL SUBROUTINE SETIME TO BEGIN TIMING ITERATION PROCESS

127 CALL SETIME

C BEGIN ITERATION PROCESS

128 DO 450 ITER=1,200

C DETERMINE VALUE OF U* AT EACH NODE AND SET VALUE OF UNSTEADY
C FORCE VECTOR TO ZERO

129 DO 138 IU=1,NSNP
130   FU(IU)=0.
131   IF (METHU.EQ.1) THEN
USTAR(IU) = U(IU)
ELSEIF (METHU.EQ.2) THEN
    USTAR(IU) = (U(IU) + UOLD(IU))/2.
ELSE
    USTAR(IU) = (AW*U(IU) + BW*UOLD(IU))/(AW+BW)
ENDIF
CONTINUE

C DETERMINE UNSTEADY ELEMENT B MATRIX

DO 210 IEL = 1, NEL
    IF (METHBM.EQ.1) THEN
        BE(1,1) = (ELEN/3.)*(USTAR(IEL) + USTAR(IEL+1))
        BE(1,2) = (ELEN/6.)*(USTAR(IEL) + USTAR(IEL+1))
        BE(2,1) = BE(1,2)
        BE(2,2) = BE(1,1)
    ELSE
        BE(1,1) = (ELEN/6.)*(3.*USTAR(IEL) + USTAR(IEL+1))
        BE(1,2) = (ELEN/6.)*(USTAR(IEL) + USTAR(IEL+1))
        BE(2,1) = BE(1,2)
        BE(2,2) = (ELEN/6.)*(USTAR(IEL) + 3.*USTAR(IEL+1))
    ENDIF

C DETERMINE SYSTEM B MATRIX BY DISTRIBUTING ELEMENTAL B MATRICES
C ACCORDING TO THE LOCAL TO GLOBAL CORRESPONDENCE

DO 163 II = 1, 2
    DO 162 JJ = 1, 2
        IN = ICORR(IEL,II)
        JN = ICORR(IEL,JJ)
        B(IN,JN) = BE(II,JJ)
    CONTINUE
CONTINUE

C DETERMINE UNSTEADY ELEMENT FORCE VECTOR

IF (METHFU.EQ.1) THEN
    FUE(1) = (ELEN/2.)*(USTAR(IEL) + USTAR(IEL+1))/2.*2
    FUE(2) = FUE(1)
ELSEIF (METHFU.EQ.2) THEN
    FUE(1) = (ELEN/2.)*(3.*USTAR(IEL) + USTAR(IEL+1))/4. + USTAR(IEL+1)/4.**2
    FUE(2) = (ELEN/2.)*(USTAR(IEL))/4. + 3.*USTAR(IEL+1)/4.**2
ELSE
    FUE(1) = ELEN*(USTAR(IEL)**2/4. + USTAR(IEL)*USTAR(IEL+1)/6.
                      + USTAR(IEL+1)**2/12.)
    FUE(2) = ELEN*(USTAR(IEL)**2/12. + USTAR(IEL)*USTAR(IEL+1)/6.
                      + USTAR(IEL+1)**2/4.)
ENDIF

C DETERMINE UNSTEADY SYSTEM FORCE VECTOR BY DISTRIBUTING ELEMENTAL
C FORCE VECTORS ACCORDING TO THE LOCAL TO GLOBAL CORRESPONDENCE

DO 180 II = 1, 2
    IN = ICORR(IEL,II)
    FU(IN) = FUE(II) + FU(IN)
C DETERMINE TOTAL SYSTEM MATRIX

DO 240 IP=1,NSNP
  DO 232 JP=1,NSNP

C RESET B MATRIX TO ZERO

B(IP,JP)=0.

C UPDATE VALUE OF U AT THE PREVIOUS ITERATION

UOLD(IP)=U(IP)

C DETERMINE TOTAL SYSTEM FORCE VECTOR

FT(IP)=FS(IP)+FU(IP)

C IMPOSE BOUNDARY CONDITIONS

C(1,1)=1.
C(1,2)=0.
FT(1)=ULBC
IF (ITYPE.EQ.1) THEN
C(NSNP,NSNP-1)=0.
C(NSNP,NSNP)=1.
FT(NSNP)=URBC
ELSE
FT(NSNP)=FT(NSNP)-URBC
ENDIF
M=1
IQ=100

C CALL SUBROUTINE LEQT2F TO SOLVE SET OF LINEAR ALGEBRAIC EQUATIONS

CALL LEQT2F(C,M,NSNP,IQ,FT,IDGT,WKAREA,IER)

DO 310 NEW=1,NSNP
  U(NEW)=FT(NEW)
C WRITE(*,*),'UNEW=',U(NEW)

C TEST FOR CONVERGENCE

DIF(NEW)=ABS(U(NEW)-UOLD(NEW))
CONTINUE
DIFMAX=DIF(1)
NMAX=1
DO 390 IJ=1,NEL
  IF (DIF(IJ+1).GE.DIF(IJ)) THEN
    DIFMAX=DIF(IJ+1)
    NMAX=IJ+1
  ENDIF
CONTINUE
ELSE
    CONTINUE
ENDIF
CONTINUE
ENDIF
CONTINUE
ENDIF
CONTINUE
RETAIN
CONTINUE
RETURN
END

C CALL SUBROUTINE GETIME TO OBTAIN CPU TIME FOR ITERATION PROCESS

CALL GETIME(IET)

C OUTPUT HEADER INFORMATION

WRITE(6,464)
WRITE(30,464)
FORMAT(1X,'EQUATION: U'' + U**2 = 60X + 100X**6')

IF (ITYPE.EQ.1) THEN
  WRITE(6,468) COORD(1),ULBC,COORD(NSNP),URBC
  WRITE(30,468) COORD(1),ULBC,COORD(NSNP),URBC
  FORMAT(1X,'B.C.: UC',F2.0,')=&,F2.0,'; U(',F2.0,')',F5.0,/) ENDIF
ELSE
  WRITE(6,472) COORD(1),ULBC,COORD(NSNP),URBC
  WRITE(30,472) COORD(1),ULBC,COORD(NSNP),URBC
  FORMAT(1X,'B.C.: U(',F2.0,')=',F2.0,'; DU/DX(',F2.0,')=&,F4.0,/) ENDIF

IF (METHU.EQ.1) THEN
  WRITE(6,478)
  WRITE(30,478)
  FORMA'T(1X,'ITERATION METHOD: U*=U',/)
ELSEIF (METHU.EQ.2) THEN
  WRITE(6,482)
  WRITE(30,482)
  FORMAT(1X,'ITERATION METHOD: U*=(U+UOLD)/2',/)
ELSE
  WRITE(6,486)AW,BW,AW,BW
  WRITE(30,486)AW,BW,AW,BW
  FORMAT(1X,'ITERATION METHOD: U*=(' ,F4.1,'*U +I ,F4.1,'*UOLD)/(',
    F4.1,')',F4.1,')',/)
ENDIF
IF (METHB.EQ.1) THEN
  WRITE(6,491)
  WRITE(30,491)
  FORMA'T(1X,'METHOD OF B MATRIX INTEGRAL EVALUATION: MIDPOINT',/)
ELSE
  WRITE(6,499)
  WRITE(30,499)
  FORMA'T(1X,'METHOD OF B MATRIX INTEGRAL EVALUATION: LINEAR',/)
ENDIF
IF (METHU.EQ.1) THEN
  WRITE(6,504)
  WRITE(30,504)
FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: MIDPOINT',/)
ELSEIF (METHFU.EQ.2) THEN
WRITE(6,508)
WRITE(30,508)
FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: 1/4-3/4',/)
ELSE
WRITE(6,512)
WRITE(30,512)
FORMAT(1X,'METHOD OF EXCITATION INTEGRAL EVALUATION: LINEAR',/)
ENDIF
IF (ITER.GE.200) THEN
WRITE(6,518)
WRITE(30,518)
FORMAT(1X,'CONVERGENCE NOT OBTAINED AFTER 200 ITERATIONS. ')
ELSEIF (ABS(U(NMAX)).GT.(10.**20).OR.ABS(U(NSNP-1)).GT.(10.**20)) THEN
WRITE(6,522)
WRITE(30,522)
FORMAT(1X,'SOLUTION PROCESS DIVERGES. ')
ELSE
WRITE(6,526) ITER,NEL
WRITE(30,526) ITER,NEL
FORMAT(1X,'CONVERGENCE OBTAINED AFTER ',13,' ITERATIONS USING ',I3,' ELEMENTS.',/)
ENDIF
RETURN
SUBROUTINE QLEXTB

* THIS SUBROUTINE COMPUTES THE EXACT SOLUTION, U=10X**3, FOR *
* MAIN PROGRAM NU2QB AT THE SPECIFIED NODAL POINTS. *

100 SUBROUTINE QLEXTB
110 COMMON A(100,100),FS(100),B(100,100),C(100,100),U(100),UOLD(100),
: UEXT(100),UDIF(100),COORD(100),ELEN,CONV,ELTIME,ULBC,URBC,
: ICORR(100,2),ITYPE,NEL,NSNP
130 DO 150 NN = 1,NSNP
140 UEXT(NN) = 10.*COORD(NN)**3
150 CONTINUE
160 RETURN
170 END
*SUBROUTINE QLOTPT*

**This subroutine computes the per cent error between the exact and FEM solutions, CPU* for the iteration process, and prints out all data in tabular form for main program NU2QA and NU2QB.**

```fortran
100 SUBROUTINE QLOTPT(CPUSTAR, IET)
110 COMMON A(100,100), FS(100), B(100,100), C(100,100), U(100), UOLD(100),
111 UEXT(100), UDIF(100), COORD(100), ELEN, CONV, ELTIME, ULBC, URBC,
112 ICORR(100,2), ITYPE, NEL, NSNP
115 SUMDIF=0.

C CALCULATE PER CENT ERROR AT EACH NODE AND SUM THE ABSOLUTE VALUE
C OF ALL THE ERRORS

120 DO 150 IK=2,NSNP
130 UDIF(IK)=100.*(U(IK)-UEXT(IK))/UEXT(IK)
140 SUMDIF=SUMDIF + ABS(UDIF(IK))
150 CONTINUE
160 UDIF(1)=U(1)-UEXT(1)

C COMPUTE THE ELAPSED TIME OF THE ITERATION PROCESS

164 ELTIME=IET*.000026
165 WRITE(6,169) ELTIME
166 WRITE(30,169)ELTIME
169 FORMAT(X,'ELAPSED TIME FOR THE ITERATION PROCESS IS ',F9.4,
170 :' SECONDS. ')

C OUTPUT DATA IN TABULAR FORMAT

170 WRITE(6,180)
175 WRITE(30,180)
180 FORMAT(/,1X,'X-COORD',4X,'U EXACT',7X,'U FEM',4X,'% DIFF')
190 WRITE(6,200) (COORD(NP), UEXT(NP), U(NP), UDIF(NP), NP=1,NSNP)
195 WRITE(30,200) (COORD(NP), UEXT(NP), U(NP), UDIF(NP), NP=1,NSNP)
200 FORMAT(/,2X,F5.3,4X,F9.4,3X,F10.4,4X,F.1)

C CALCULATE CPU* FOR THE ITERATION PROCESS

205 CPUSTAR=ELTIME*SUMDIF/NSNP
210 WRITE(6,220) CPUSTAR
215 WRITE(30,220)CPUSTAR
220 FORMAT(/,1X,'CPU* FOR THE ITERATION PROCESS IS ',F9.4,' SECONDS. ')
230 RETURN
240 END
```

163
LIST OF REFERENCES

# INITIAL DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>No.</th>
<th>Copies</th>
<th>Recipient</th>
<th>Address</th>
</tr>
</thead>
</table>
| 1.  | 2      | Defense Technical Information Center  
      Cameron Station  
      Alexandria, VA  22304-6145 |
| 2.  | 2      | Library, Code 52  
      Naval Postgraduate School  
      Monterey, CA  93943-5002 |
| 3.  | 1      | Department Chairman, Code ME  
      Department of Mechanical Engineering  
      Naval Postgraduate School  
      Monterey, CA  93943 |
| 4.  | 1      | Naval Engineering Curricular Office, Code 34  
      Department of Mechanical Engineering  
      Naval Postgraduate School  
      Monterey, CA  93943 |
| 5.  | 4      | Professor David Salinas, Code ME/Sa  
      Department of Mechanical Engineering  
      Naval Postgraduate School  
      Monterey, CA  93943 |
| 6.  | 1      | Professor Young Kwon, Code ME/Kw  
      Department of Mechanical Engineering  
      Naval Postgraduate School  
      Monterey, CA  93943 |
| 7.  | 1      | Professor Robert E. Newton, Code ME/Ne  
      Department of Mechanical Engineering  
      Naval Postgraduate School  
      Monterey, CA  93943 |
| 8.  | 2      | Commandant (G-PTE-2)  
      U.S. Coast Guard Headquarters  
      2100 2nd Street S.W.  
      Washington, D.C.  20593 |
| 9.  | 1      | LT Baird S. Ritter, USCG  
      U.S. Coast Guard Marine Safety Center  
      400 7th Street S.W.  
      Washington, D.C.  20590 |