A Generalization on the Solution of the Second Order Ordinary Linear Boundary Value Problem by the Finite Element Method

M. R. Hossain 1,*, Touhid Hossain 2, Md. Mamun-Ur-Rashid Khan 3
1Department of Applied Mathematics, 2Department of Mathematics
University of Dhaka, Dhaka-1000, Bangladesh
* Corresponding Author: M. R. Hossain

Abstract: In this paper we introduce a generalized MATHEMATICA code for solving one dimensional second order ordinary boundary value problems by Modified Galerkin’s technique of finite element methods using the standard basis functions of different degrees and any number of elements.

Keywords: Finite element method, Modified Galerkin’s method, Standard shape function, MATHEMATICA

I. Introduction

The second order ordinary linear differential equations (ODE) arise in many applications including the spring-mass system [1], the string theory [2] and the general wave mechanics [3]. To solve such ODE’s, a number of different mathematical techniques, such as, the finite difference method (FDM), finite element methods (FEM) and the control volume method (CVM), have been used [4-7]. Depending on the physical problems, these equations, may be appeared in the general form in the self-adjoint form [1], may be required different types of spatial grid schemes, or may be accompanied by different types of boundary conditions. For example, in the FEM, the choices of the types of the number of the elements or degree of the elements are very crucial. If the grid schemes or the boundary conditions are required to modify within a simulation, one has to reformulate the whole solution procedure accordingly, which is laborious and subject to the extra computational cost. Also to get the better accuracy by increasing the number or order of the elements, the computational work becomes lengthy and time consuming. To overcome these difficulties, this paper aims to generalize the FEM for 1D boundary value problems. A set of MATHEMATICA modules have also been presented for further reference.

II. Generalization

Consider the general 2nd order one dimensional ordinary BVP

\[ a_1(x) \frac{d^2 u}{dx^2} + a_2(x) \frac{du}{dx} + a_3(x) u = a_4(x), \quad a \leq x \leq b \]  

(1)

By converting which can also be written as the self-adjoint form as

\[ - \frac{d}{dx} \left[ p(x) \frac{du}{dx} \right] + q(x) u(x) = r(x), \quad a \leq x \leq b. \]  

(2)

The boundary conditions are

\[ u(a) = \gamma_1, \quad u(b) = \gamma_2 \]  

(1a)

or, \( u(a) = \gamma_3, \quad u(b) = \gamma_4 \)  

(1b)

or, \( u'(a) = \gamma_5, \quad u(b) = \gamma_6 \)  

(1c)

or, \( u'(a) = \gamma_7, \quad u'(b) = \gamma_8 \)  

(1d)

By taking the trial solution of the form

\[ \tilde{u}(x) = \sum_{j=1}^{n} a_j N_j(x) \]

And using Modified Galerkin’s [4-6] formulation we found the general system of equations

DOI: 10.9790/5728-1402025560 www.iosrjournals.org 55 | Page
A Generalization on the Solution of the Second Order Ordinary Linear Boundary Value Problem ....

\[
\begin{bmatrix}
K_{11} & \ldots & K_{1n} \\
\vdots & \ddots & \vdots \\
K_{n1} & \ldots & K_{nn}
\end{bmatrix}
\begin{bmatrix}
a_1 \\
\vdots \\
a_n
\end{bmatrix} =
\begin{bmatrix}
F_1 \\
\vdots \\
F_2
\end{bmatrix}
\] (3)

where,
\[
K_{ij} = \int [e] \frac{dN_i}{dx} p(x) \frac{dN_j}{dx} \, dx + \int [e] N_i(x) q(x) N_j(x) \, dx , \quad \text{and}
\]

\[
F_i = \int [e] r(x) N_i(x) \, dx + \left[ \left( p(x) \frac{dN_i}{dx} \right) N_i(x) \right]_{[e]}.
\]

The matrices \( K = [K_{ij}] \) and \( F = [F_i] \) are known as Stiffness matrix and local force term [1], respectively. Here, \( N_j(x), j = 1, 2, \ldots \) are the standard shape functions of different order. For example, the linear shape functions are \( N_1(\xi) = \frac{1-\xi}{2}, N_2(\xi) = \frac{1+\xi}{2} \), the quadratic shape functions are \( N_1(\xi) = \frac{\xi}{2}(\xi - 1), N_2(\xi) = 1 - \xi^2, \) \( N_3(\xi) = \frac{\xi}{2}(\xi + 1) \) and so on.

If we use the linear shape functions and 4 elements to solve a problem of the form (2) with any boundary condition the stiffness matrix becomes a 5x5 matrix and the local force term consists of 5 elements, similarly if we use linear shape functions and 8 elements then the stiffness matrix becomes a 9x9 matrix. Also if we use quadratic shape functions and 3 quadratic elements the stiffness matrix becomes a 7x7 matrix. From the above observations we can conclude the following lemma as:

**Lemma:** If we consider elements and \((n - 1)^{th}\) degree shape functions, then we have to calculate \((m \times n - m + 1) \times (m \times n - m + 2)\) integral values to obtain matrices \( K \) and \( F \).

## III. Program Modules

The MATHEMATICA [8-10] code that we develop in this paper initially takes the values of the boundaries, degree of the shape functions and the number of elements as inputs. If the given differential equation is in the general form (1), then the code will convert it to self-adjoint form (2). Next the code will generate the standard shape functions, as require, and compute the entries of the matrices \( K \) and \( F \) by taking the boundary conditions (essential or suppressible) given. Finally, it shows the calculated numerical results with absolute error (if the exact solution is available) in the tabular form and represents the results graphically.

### IV. Numerical Examples

**Example 4.1:**
Consider the following BVP [1]
\[
\frac{d}{dx} \left( x \frac{du}{dx} \right) = \frac{2}{x^2}, \quad 1 < x < 2, \quad u(1) = 2, \quad u'(2) = -\frac{1}{4}
\]

**Input:**
\[a = 1, \quad b = 1, \quad p(t) = -t, \quad q(t) = 0, \quad r(t) = \frac{2}{t^2}, \quad u(1) = 2, \quad u'(2) = -\frac{1}{4}\]

**Output:**
The matrices \( K \) and \( F \) are:
\[
K = \begin{bmatrix}
-9/2 & 9/2 & 0 & 0 & 0 \\
9/2 & -10 & 11/2 & 0 & 0 \\
0 & 11/2 & -12 & 13/2 & 0 \\
0 & 0 & 13/2 & -14 & 15/2 \\
0 & 0 & 0 & 15/2 & -15/2
\end{bmatrix}
\]
\[
F = \begin{bmatrix}
2 + du[1] - 8 \log \left( \frac{2}{e} \right) \\
-8 \log \left( \frac{2}{e} \right) + 8 \log \left( \frac{2}{e} \right) \\
-8 \log \left( \frac{2}{e} \right) + 8 \log \left( \frac{2}{e} \right) \\
-8 \log \left( \frac{2}{e} \right) + 8 \log \left( \frac{2}{e} \right) \\
-1 - 2 du[2] + 8 \log \left( \frac{2}{e} \right)
\end{bmatrix}
\]

**Table 1:** Results using Linear Shape function and 4 elements:

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Approx. Solution</th>
<th>Exact Solution</th>
<th>Absolute Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1.25</td>
<td>1.71441</td>
<td>1.71157</td>
<td>0.00283969</td>
</tr>
<tr>
<td>1.5</td>
<td>1.54013</td>
<td>1.53607</td>
<td>0.00405968</td>
</tr>
<tr>
<td>1.75</td>
<td>1.42732</td>
<td>1.42267</td>
<td>0.00465969</td>
</tr>
<tr>
<td>2.0</td>
<td>1.35156</td>
<td>1.34657</td>
<td>0.00498432</td>
</tr>
</tbody>
</table>

DOI: 10.9790/5728-1402025560  www.iosrjournals.org  56 | Page
A Generalization on the Solution of the Second Order Ordinary Linear Boundary Value Problem

Table 2: Results using Quadratic Shape function and 4 elements:

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Approx. Solution</th>
<th>Exact Solution</th>
<th>Absolute Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1.125</td>
<td>1.83673</td>
<td>1.83667</td>
<td>0.0000566035</td>
</tr>
<tr>
<td>1.25</td>
<td>1.71159</td>
<td>1.71157</td>
<td>0.000048841</td>
</tr>
<tr>
<td>1.375</td>
<td>1.61381</td>
<td>1.61377</td>
<td>0.0000343892</td>
</tr>
<tr>
<td>1.5</td>
<td>1.53609</td>
<td>1.53607</td>
<td>0.0000192144</td>
</tr>
<tr>
<td>1.625</td>
<td>1.47355</td>
<td>1.47352</td>
<td>0.0000272429</td>
</tr>
<tr>
<td>1.75</td>
<td>1.42269</td>
<td>1.42267</td>
<td>0.0000207598</td>
</tr>
<tr>
<td>1.875</td>
<td>1.381</td>
<td>1.38097</td>
<td>0.000024504</td>
</tr>
<tr>
<td>2.0</td>
<td>1.34659</td>
<td>1.34657</td>
<td>0.0000213972</td>
</tr>
</tbody>
</table>

Table 3: Results using Cubic Shape function and 4 elements:

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Approx. Solution</th>
<th>Exact Solution</th>
<th>Absolute Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1.08333</td>
<td>1.88616</td>
<td>1.88618</td>
<td>0.0000197765</td>
</tr>
<tr>
<td>1.16667</td>
<td>1.79134</td>
<td>1.79136</td>
<td>0.0000201209</td>
</tr>
<tr>
<td>1.25</td>
<td>1.71157</td>
<td>1.71157</td>
<td>6.50232×10⁻⁸</td>
</tr>
<tr>
<td>1.33333</td>
<td>1.64383</td>
<td>1.64384</td>
<td>7.07551×10⁻⁸</td>
</tr>
<tr>
<td>1.41667</td>
<td>1.58591</td>
<td>1.58592</td>
<td>7.17107×10⁻⁸</td>
</tr>
<tr>
<td>1.5</td>
<td>1.53607</td>
<td>1.53607</td>
<td>7.77409×10⁻⁸</td>
</tr>
<tr>
<td>1.58333</td>
<td>1.49292</td>
<td>1.49292</td>
<td>2.96977×10⁻⁸</td>
</tr>
<tr>
<td>1.66667</td>
<td>1.45541</td>
<td>1.45541</td>
<td>3.00205×10⁻⁸</td>
</tr>
<tr>
<td>1.75</td>
<td>1.42267</td>
<td>1.42267</td>
<td>8.10082×10⁻⁸</td>
</tr>
<tr>
<td>1.83333</td>
<td>1.39398</td>
<td>1.39398</td>
<td>1.38426×10⁻⁸</td>
</tr>
<tr>
<td>1.91667</td>
<td>1.36877</td>
<td>1.36877</td>
<td>1.39677×10⁻⁸</td>
</tr>
<tr>
<td>2.0</td>
<td>1.34657</td>
<td>1.34657</td>
<td>8.20266×10⁻⁸</td>
</tr>
</tbody>
</table>

Graphical Results:

Figure 1a: Solution using 4-Linear elements  Figure 1b: Solution using 4-Quadratic elements
A Generalization on the Solution of the Second Order Ordinary Linear Boundary Value Problem ······

Figure 1c: Solution using 4-Cubic elements

Figure 1d: Solution using 8-Cubic elements

Example 4.2:
Consider the BVP [11]
\[ \frac{d^2 u}{dx^2} + \frac{4}{x} \frac{du}{dx} + \frac{2}{x^2} u = \frac{2}{x^2} \ln x , \quad 1 < x < 2 , \quad u(1) = \frac{1}{2} , \quad u(2) = \ln 2 \]

Input:
\[ a = 1 , \quad b = 2 , \quad a_2(t) = 1 , \quad a_1(t) = \frac{4}{t} , \quad a_0(t) = \frac{2}{t^2} , \quad c(t) = \frac{2}{t^2} \ln t , \quad u_1(t) = \frac{1}{2} , \quad u_2(t) = \ln 2 \]

Graphical Results:

Figure 2a: Solution using 4-Linear elements

Figure 2b: Solution using 4-Quadratic elements

Figure 2c: Solution using 4-Cubic elements

Figure 2d: Solution using 8-Cubic elements

V. Discussion

In example 4.1, we see that if we take 4 elements and use linear shape functions we get the approximate solutions (node wise) with percentage errors within ±0.1% to ±0.3% (Table 1). Whereas, if we consider 4 elements with quadratic shape functions (although it requires tiresome hand calculations) we get the approximate solutions with percentage errors within ±0.001% to ±0.003% (Table 2). With the same number of elements and cubic shape functions we get more accurate results (Table 3).
A Generalization on the Solution of the Second Order Ordinary Linear Boundary Value Problem......

VI. Conclusion

To get better approximate solutions (with graphical representation and error analysis, if exact solution is available) of any form of second order BVP’s by using Modified Galerkin’s technique of FEM with any number of elements and standard shape functions of any degrees by avoiding time consuming hand calculations (impossible in some cases), we can use the MATHEMATICA code that we developed in this paper.

References


Appendix

Mathematica Code:

FEM:=Module[{a=Input["Enter the value of a"],
b=Input["Enter the value of b"],
deg=Input["Enter the value of the degree of the standard shape function"],
m=Input["Enter the number of elements"],
equform=Input["Enter the value of the degree of the standard shape function"],
if exact solution is available) of any form of second order BVP’s by using Modified Galerkin’s technique of FEM with any number of elements and standard shape functions of any degrees by avoiding time consuming hand calculations (impossible in some cases), we can use the MATHEMATICA code that we developed in this paper.

References


Appendix

Mathematica Code:

FEM:=Module[{a=Input["Enter the value of a"],
b=Input["Enter the value of b"],
deg=Input["Enter the value of the degree of the standard shape function"],
m=Input["Enter the number of elements"],
equform=Input["Enter the value of the degree of the standard shape function"],
if exact solution is available) of any form of second order BVP’s by using Modified Galerkin’s technique of FEM with any number of elements and standard shape functions of any degrees by avoiding time consuming hand calculations (impossible in some cases), we can use the MATHEMATICA code that we developed in this paper.

References


Appendix

Mathematica Code:

FEM:=Module[{a=Input["Enter the value of a"],
b=Input["Enter the value of b"],
deg=Input["Enter the value of the degree of the standard shape function"],
m=Input["Enter the number of elements"],
equform=Input["Enter the value of the degree of the standard shape function"],
if exact solution is available) of any form of second order BVP’s by using Modified Galerkin’s technique of FEM with any number of elements and standard shape functions of any degrees by avoiding time consuming hand calculations (impossible in some cases), we can use the MATHEMATICA code that we developed in this paper.

References


Appendix

Mathematica Code:

FEM:=Module[{a=Input["Enter the value of a"],
b=Input["Enter the value of b"],
deg=Input["Enter the value of the degree of the standard shape function"],
m=Input["Enter the number of elements"],
equform=Input["Enter the value of the degree of the standard shape function"],
if exact solution is available) of any form of second order BVP’s by using Modified Galerkin’s technique of FEM with any number of elements and standard shape functions of any degrees by avoiding time consuming hand calculations (impossible in some cases), we can use the MATHEMATICA code that we developed in this paper.
A Generalization on the Solution of the Second Order Ordinary Linear Boundary Value Problem

If [\(l\in n\), F\([l+j]\)=F\([l+j]\)+\(\int_{\xi} \{p[z[n+j]] \ast du[z[n+j]]\} F\([l+j]\)+\int_{\xi} \{p[z[n+j]] \ast du[z[n+j]]\} F\([l+j]\)]

For \(c=1, cSn, c++\),
\(K[l+j][c+j]=K[l+j][c+j]+\int_{\xi} \{p[z[n+j]] \ast du[z[n+j]]\} F\([l+j]\)+\int_{\xi} \{p[z[n+j]] \ast du[z[n+j]]\} F\([l+j]\); \)
\(j=j+n-1\);

\(g=\text{Input"input} \\n1 \text{ if both boundary conditions are essential} \\n2 \text{ if last condition is suppresible} \\n3 \text{ if first condition is suppresible} \ \n4 \text{ if both boundary conditions are suppresible"}; \)
\(\text{anasol=Input"Enter } \\text{n 1 if analytical solution is known} \ \n0 \text{ if analytical solution is unknown"}; \)
\(\text{If } g\in 1, \text{u[a]=Input"u[a]"; u[b]=Input"u[b]"; } \)
\(\text{s4=LinearSolve[Table[K[i,j]/N,\{i,2,m*n-m\},\{j,2,m*n-m\}], Table[(F[i]-u[a]*K[i,1]-u[b]*K[i,m*n-m+1])/N,\{i,2,m*n-m\}])}; \)
\(\text{asol[1]=u[a]; asol[m*n-m+1]=u[b]; } \)
\(\text{For[i=2,i\le m*n-m, i++, asol[i]=s4[[i-1]]]}; \)
\(\text{If } g\in 2, \text{u[a]=Input"u[a]"; } \)
\(\\beta =\text{Input"u'[b]"; } \)
\(\text{s4=LinearSolve[Table[K[i,j]/N,\{i,2,m*n-m+1\},\{j,2,m*n-m+1\}], Table[(F[i]/.du[b]\to \beta -u[a]*K[i,1])/N,\{i,2,m*n-m+1\}])}; \)
\(\text{asol[1]=u[a]; For[i=2,i\le m*n-m, i++, asol[i]=s4[[i-1]]]}; \)
\(\\text{If } g\in 3, \text{u[b]=Input"u[b]"; } \)
\(\text{\beta =Input"u'[a]";} \)
\(\text{s4=LinearSolve[Table[K[i,j]/N,\{i,1,m*n-m\},\{j,1,m*n-m\}], Table[(F[i]/.du[a]\to \beta -u[b]*K[i,m*n-m+1])/N,\{i,1,m*n-m\}])}; \)
\(\text{asol[m*n-m+1]=u[b]; For[i=1,i\le m*n-m, i++, asol[i]=s4[[i]]]}; \)
\(\text{If } g\in 4, \text{\beta =Input"u'[a]"; } \)
\(\\gamma =\text{Input"u'[b]";} \)
\(\text{s4=LinearSolve[Table[K[i,j]/N,\{i,1,m*n-m+1\},\{j,1,m*n-m+1\}], Table[(F[i]/.du[a]\to \beta -du[b]\to \gamma)/N,\{i,1,m*n-m+1\}])]; } \)
\(\text{For[i=1,i\le m*n-m+1, i++, asol[i]=s4[[i]]])}; \)
\(\text{If } \text{anasol}\in 1, \text{f[t_]=Input"Analytical solution u(t) in variable t"}; } \)
\(\text{Print"The matrices K and F are: \"n\}; } \)
\(\text{Print"K", MatrixForm[Table[N[K[i,j]],\{i,1,m*n-m+1\},\{j,1,m*n-m+1\}]]}; \)
\(\text{Print"F", MatrixForm[Table[N[F[i]],\{i,1,m*n-m+1\}]]}}; \)
\(\text{If } \text{anasol}\in 1, \text{PrintTableForm[Table[N[z[i]],\{i,1,m*n-m+1\}], error[i]=Abs[f[N[z[i]]]-asol[i]], \{i,1,m*n-m+1\}], TableHeadings\to \{\text{Nodes}, "Approx. Solutions"}, TableAlignments\to \{\text{Left}]}]; \)
\(\text{Show[\{Plot[f[t],\{t,a,b\}], \text{PlotStyle}\to \text{Red}, \text{ListPlot[Table[\{N[z[i]], asol[i]\},\{i,1,m*n-m+1\}], \text{PlotMarkers}\to \"\",15\}], \text{Epilog}\to \text{Inset[\{LineLegend[\{Red\}, \"Exact soln\"], LegendMarkerSize\to 10, LabelStyle\to \text{Red}, PointLegend[\{Blue\}, \"Approx. Soln\"], LegendMarkers\to \"\", LegendMarkerSize\to 10, LabelStyle\to \text{Blue}\}], RoundingRadius\to 9, \text{Scaled[\{.8,.7\}]\}]}, TableForm[Table[\{N[z[i]], asol[i]\},\{i,1,m*n-m+1\}], TableHeadings\to \{None, \"Nodes\", \"Approx. Solutions\"\}, TableAlignments\to \{\text{Left}\}]\}]; \)

FEM