Name: Luyando Kwenda
Class: ENM 5020
Due Date: 26 April 2023
Assignment: 4

Galerkin Finite Element Method to Solve 1D Diffusion Problem

Contents

1 Introduction

Finite Element Methods (FEM) is numerical method used to solve Partial Differential Equations (PDEs). FEM involves breaking down a problem into smaller parts (or shapes) called finite elements through discretization forming a mesh and solved over their respective domains. The finite elements are connected at points called nodes and we a can form a system of equations that's can be solved mathematically. One variation of FEM is the Galerkin method involves approximating the solution of a PDE or integral equation by a linear combination of basis functions. These basis functions are chosen to satisfy certain properties such as being smooth, orthogonal, or piece-wise continuous. The approximate solution is then obtained by minimizing the residual error, which is the difference between the exact solution and the approximate solution.

We have been tasked to analyse a Boundary Value Problem (BVP) in one dimension in the domain $D = (0 \leq x \leq 1)$. The particular BVP we are solving is a form of the Poisson Equation that is not equal to zero i.e $\alpha \neq 0$ with a Boundary Condition $T = 0$ on all boundaries.

$$\frac{d^2T}{dx^2} + \alpha = 0 \tag{1}$$

The generalized version of the Galerkin we are using in this assignment of the method of Mean Weighted Residuals (MWR). A function is multiplied by by a set of weighted functions and integrated over a given domain. In this case, the residuals are orthogonal to the weight functions. Given a PDE $R(c) = 0$, we can say that the MWR is

$$R_k = \int_D w_k(x) R(T(x)) dx = 0 \tag{2}$$

Our choice of the weighted function will determine the projection method.The weighting functions are chosen to be the same as the approximate solution function used to approximate the solution of the PDE. In the Galerkin MWM we have:

$$w_k(x) = \phi^k(x) \tag{3}$$

Once all the equations are set up, we are to use 3 point gaussian quadrature to solve an integral

2 Methods

2.1 Finite Element Method

We first divide the given problem into a finite number of elements which we will label as N_{el} and try to obtain a solution over each element. The number of unknown nodes will be denoted by N which are the temperature values of the mesh. The differential equation in Eq. (1) with its boundary conditions is known as the strong form and as we convert it into a weighted residual it will be in the weak form which is obtained from the method of MWR. Combining Eq. (1) and Eq. (2) into Eq. (3), we obtain the residual function

$$R(x) = \int_0^1 \phi^i(x) \left(\frac{d^2T(x)}{dx^2} + \alpha \right) dx = 0 \tag{4}$$

$$R(x) = \int_0^1 \left(\frac{d^2T(x)}{dx^2} \phi^i(x) + \alpha \phi^i(x) \right) dx = 0 \tag{5}$$

The solution is an continuous function and this much more complicated to solve in this state, so we apply integration by parts on the diffusion term.

$$\int_0^1 \phi^i(x) \frac{d^2T(x)}{dx^2} dx = \phi^i(x) \frac{dT}{dx} \bigg|_0^1 - \int_0^1 \frac{dT}{dx} \frac{d\phi^i}{dx} dx \tag{6}$$

Because we know the that boundary conditions are zero at x=0 and x=1, the first term in the above equation goes to zero leaving us with

$$\int_0^1 \phi^i(x) \frac{d^2T(x)}{dx^2} dx = -\int_0^1 \frac{dT}{dx} \frac{d\phi^i}{dx} dx \tag{7}$$

Eq. (5) is the rewritten as Eq. (8) which is the weak form of the problem statement.

$$R(x) = \int_0^1 \left(\alpha\phi^i(x) - \frac{dT}{dx}\frac{d\phi^i}{dx} \right) dx = 0 \tag{8}$$

We can also approximate the solution using

$$T(x) \approx \sum_{j=0}^{N_{el}+1} T_j \phi^j(x) \tag{9}$$

This then further reduces the equation to

$$\sum_{j=0}^{N_{el}+1} T_j \int_0^1 \frac{d\phi^i}{dx}\frac{d\phi^j}{dx} dx = \int_0^1 \alpha\phi^i dx \tag{10}$$

This is a linear equation in the form Ax = b where x = T, which can be easily solved. The next step will be finding a way of writing the A term into a matrix. A and b will be the global matrices but we can use smaller elements in the local frame to simplify the method.

2.2 Gaussian Quadrature

Gaussian quadrature is a numerical integration method for approximating the definite integral of a function. It involves using a weighted sum of function values at carefully chosen points within the interval of integration. These points are known as quadrature points or nodes, and the weights are chosen to optimize the accuracy of the approximation. Gaussian quadrature is very useful because of it high accuracy. Given an integral, I, we can rewrite this in the form on weights

$$I = \int_{-1}^{+1} f(x)dx = \sum_{i=1}^{n} w_i f(x_i) \tag{11}$$

Using 3 point quadrature, we have the values of the weights and the Gauss points

$w_i = -\sqrt{\frac{3}{5}}$	$x_i = \frac{5}{9}$
$w_i = 0$	$x_i = \frac{8}{9}$
$w_i = \sqrt{\frac{3}{5}}$	$x_i = \frac{5}{9}$

This method only works for the domain [-1 1] but we are able to map to the real domain. We evaluate each N_{el} element in the local domain, then map to the real domain.

$$x = \sum_{i=1}^{3} x_i \phi^i(\xi) \tag{12}$$

The basis functions of this domain are

$$\phi^{(1)} = \frac{1}{2}(1-\xi) \tag{13}$$

$$\phi^{(2)} = \frac{1}{2}(1+\xi) \tag{14}$$

With their derivatives given by

$$\frac{\phi^{(1)}}{d\xi} = -\frac{1}{2} \tag{15}$$

$$\frac{\phi^{(2)}}{d\xi} = \frac{1}{2} \tag{16}$$

With the derived equations, we can now rewrite the weak form of the residual. For the matrix A, we have:

$$\int_l \frac{d\phi^i}{dx}\frac{d\phi^j}{dx} dx = \int_{-1}^{+1} f(\xi)d\xi = \sum_{i=1}^{N_{el}} w_i f(\xi) \tag{17}$$

$$Alocal_{ij} = \int_{-1}^{+1} \frac{d\phi^i}{d\xi}\frac{d\phi^j}{d\xi}(\frac{dx}{d\xi})^{-1} \tag{18}$$

Putting all these equations together we get

$$\sum_{l=1}^{l+1}\sum_{j=1}^{N} T_j \int_{-1}^{+1} \frac{d\phi^i}{d\xi}\frac{d\phi^j}{d\xi}(\frac{dx}{d\xi})^{-1}d\xi = \sum_{l=1}^{N}\int_{-1}^{+1}\phi^i\alpha\frac{dx}{d\xi}d\xi \tag{19}$$

3 Results

After keeping the value of alpha constant, we varied the number of elements in the fEM process and it is notably different from the shape of the graph.

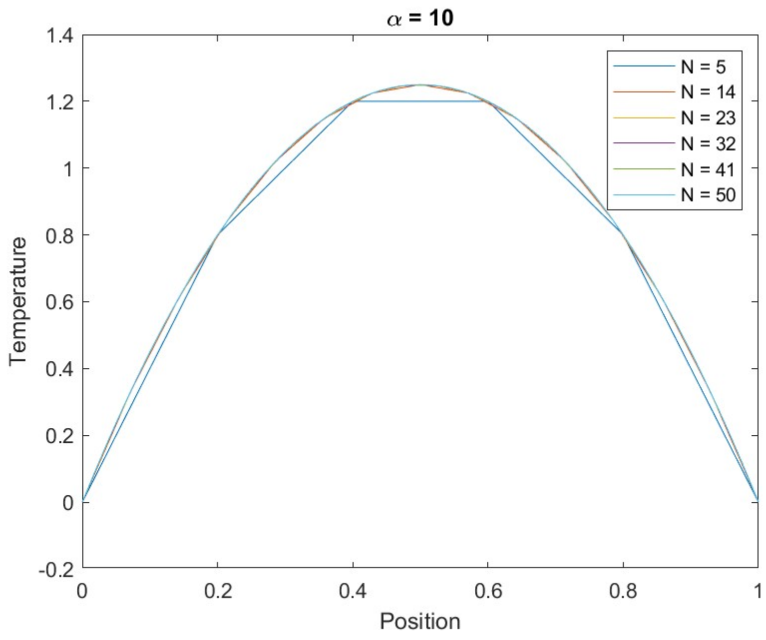
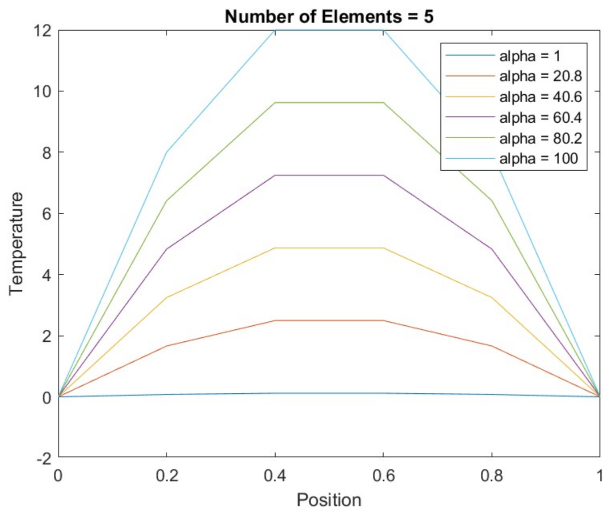


Figure 1: constant alpha

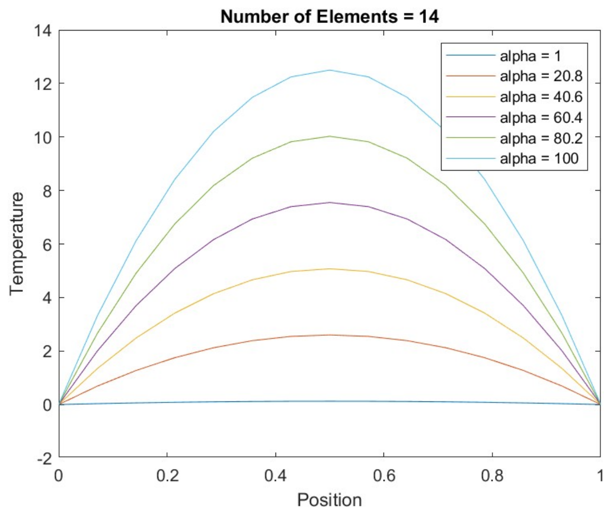
As we vary the number of elements, we can see that the curve becomes more smooth. This result is expected as we observed in assignment 1. As we increased the number of grid points, we saw the curve of the line become smooth, increasing its resolution. So we can conclude that having a greater number of elements produces smoother results. As the number of elements increases, there is a point where there isn't much of a difference like the case when the number of elements is equal to 32 and 50.

As we increase the value of alpha, we see greater temperature peaks which makes sense when we compare it to the case when alpha is equal to zero, This is just a regular second order differential equation that produces a straight line.

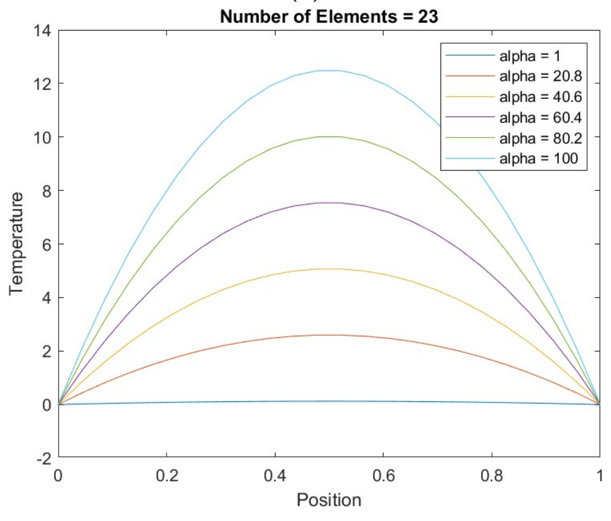
The error in FEM generally decreases as the number of elements used in the mesh is increased. This is because increasing the number of elements means that the approximation of the solution becomes more refined and accurate. In this case, we expect the error to be really close to zero.



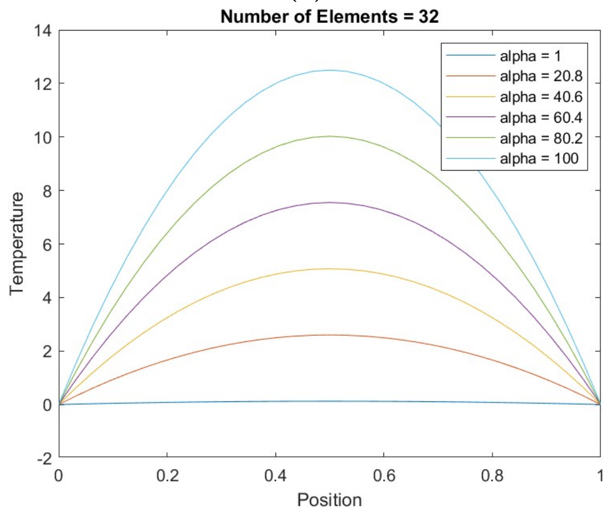
(a)



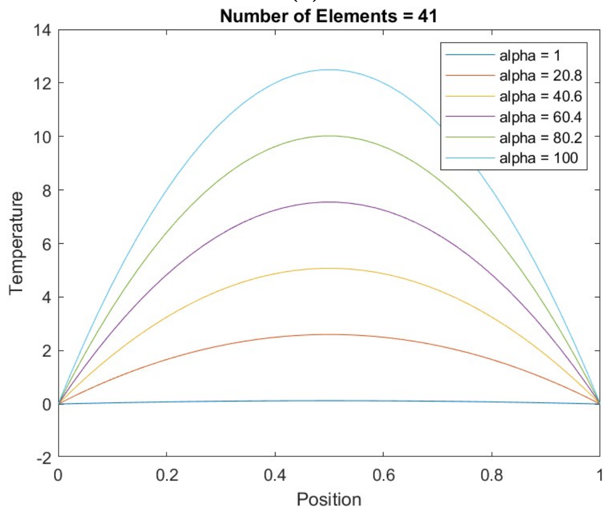
(b)



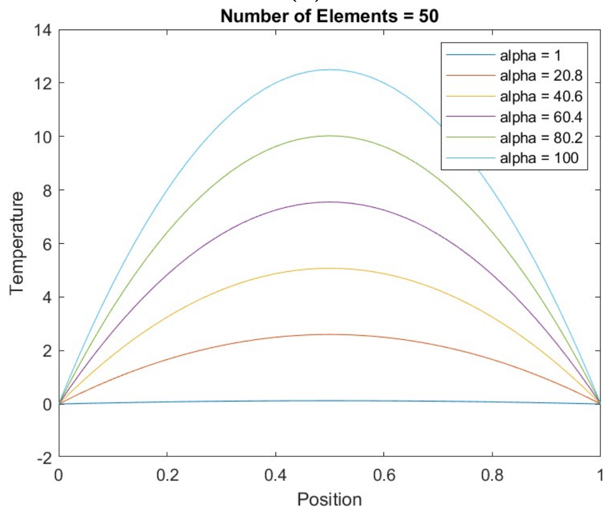
(c)



(d)



(e)



(f)

Figure 2: 2D BVP Surface Plots

4 Conclusion

In a broad overview, we have previously been able to solve a 1D heat equation using LU decomposition, Now we have solved this using Finite Element Methods. This involved breaking the problem into smaller elements by converting the problem from a string form to a weak form using the Galerkin Method of Weighted Residuals to simplify the integration process. To solve the integrals, we use 3 point Gaussian Quadrature using Gauss-Legendre Polynomial. The main advantage of Gaussian quadrature over other numerical integration methods is that it can achieve high accuracy using relatively few function evaluations. This is because the method is designed to exactly integrate polynomials of a certain degree, and since many functions can be approximated well by polynomials, the method can provide accurate approximations for a wide range of functions.Once this occurs, we obtain a linear equation in the form Ax=b which can be solved by the mldivide function in Matlab.

We observe that increasing the number of elements increases the resolution of the results but there is a limit to the extent in which we increase the number of elements because at some Nel value the graphs start to look the same.

5 Appendix

Below is the code used in the analysis of the problem

5.1 Main Script

```matlab
%this script changes the Nel at specific alpha values

%here we will choose a specific alpha value and vary the number of elements
alpha = 10;
Nel = linspace(5,50,6);
f1 = figure;
for ii = 1:length(Nel)
    nElements = Nel(ii) ;
    [T,x] = FEM1D(nElements,alpha);

    plot(x,T,'DisplayName',"N = "+num2str(nElements))
    hold on
end
legend
title("\alpha = " + num2str(alpha))
xlabel('Position')
ylabel('Temperature')
exportgraphics (gcf,'manyNel.jpg')
close(f1)

alphavec = linspace(1,100,6);

for ii = 1:length(Nel)
    nElements = Nel(ii) ;

    f = figure;
    for jj = 1:length(Nel)
        alpha = alphavec(jj);
        [T,x] = FEM1D(nElements,alpha);
        plot(x,T,'DisplayName',"alpha = "+num2str(alpha))
        hold on
    end
    legend
    title("Number of Elements = " + num2str(nElements))
    xlabel('Position')
    ylabel('Temperature')
    exportgraphics (gcf,"Nel="+ num2str(nElements) +".jpg")
    close(f)
end
```

5.2 FEM

```matlab


function [T,x] = FEM1D(Nel,alpha)

%Define the number of nodes
N = Nel+1;
```

```matlab

%create mesh
x = linspace(0,1,N);

%weights and xi for mapping
gx = [-sqrt(3/5) 0 sqrt(3/5)];
gw = [5/9 8/9 5/9];

%construct the Global and Local matrices
Aij_gobal = zeros(N,N);
b_global = zeros(N,1);

%create local to global mapping matrix
ltog = [(1:N-1)', (2:N)'];

%calculate basis functions. Store the index 1 in row 1 and index 2 in row 2
phi = zeros(2,length(gx));
for ll = 1:length(phi)
    phi(1,ll) = 0.5*(1-gx(ll));
    phi(2,ll) = 0.5*(1+gx(ll));
end

%we also know the derivatives of each
dphi_xi=[-0.5 0.5];

%loop through the elements Nel then stor in global matrix
for kk = 1:Nel
    x1 = x(kk);
    x2 = x(kk+1);
    %find dx/dxi
    dx_dxi = x1*dphi_xi(1) + x2*dphi_xi(2);
    Alocal = zeros(2,2);
    blocal = zeros(2,1);
    for ii =1:2
        blocal(ii) = sum(gw.*phi(1,:)*alpha*dx_dxi);
        for jj =1:2
            dphii_xi= basis_func(ii);
            dphj_xi = basis_func(jj);
            for pp =1:3
                Alocal(ii,jj) = Alocal(ii,jj) + gw(pp)*dphii_xi*dphj_xi/dx_dxi;
            end
        end
    end
%store values in global matrix
Aij_gobal(kk:kk+1,kk:kk+1) = Aij_gobal(kk:kk+1,kk:kk+1)+ Alocal;
b_global(kk:kk+1) = b_global(kk:kk+1) + blocal;

end

%impose boundary conditions
b_global(1) = 0;
b_global(end) = 0;

%make first and last row zero
Aij_gobal(1,:) = 0;
Aij_gobal(end,:) = 0;
Aij_gobal(1,1) = 1;
Aij_gobal(end,end) = 1;

%solve for T
T= Aij_gobal\b_global;



end
```

5.3 Basis Functions Function

```
%This function determines the basis functions and its derivative
function phi_derivative = basis_func(ii)
%we only have two basis functions at ii=1 and ii=2
if ii == 1
phi_derivative = -0.5;

elseif ii==2

phi_derivative = 0.5;
end

end
```