1D FINITE ELEMENT METHODS

ARDEN RASMUSSEN

ABSTRACT. Finite Element Methods (FEM) is a process that is utilized to numerically approximate solutions to other wise impossible partial differential equations. This paper will serve as an explanation and introduction to the concept of FEM localized in one dimension.

1. Introduction

We follow the process presented by [KH15], for the Galerkin method for generating a finite element approximation in one dimension.

For simplicity we will only examine FEM in one dimension. There ten stages for the implementation of the finite element analysis. The first four of which are theoretical, and must be done prior to numerical computation. These theoretical stages is used to setup the general mathematical formulas for computation, and is significantly more difficult to automate, and thus must be done analytically. The ten stages are listed below.

(i) Problem Statement
(ii) Variational Formulation
(iii) Discretization
(iv) Localization
(v) Mesh Generation
(vi) Assembly of the Global System
(vii) Incorporating Boundary Conditions
(viii) Numerically Solving Algebraic System
(ix) Repeating appropriate steps for time-dependent problems.
(x) Post processing.

Due to the close dependence on the specific partial differential equation considered with the theoretical stages of the process, the results from these stages is closely dependent on the problem at hand. Because of this, we will use the steady 1D heat transfer model as our problem statement for this explanation.

2. Problem Statement

We consider the steady 1D heat equation, which is a simplification of the unsteady multidimensional heat transfer model. Since the time derivative is neglected, there is no need for an initial condition, only a boundary condition for T or $\frac{\partial T}{\partial x}$ at the endpoints of our considered interval is necessary

$$\rho c_p u \frac{\partial T}{\partial x} - \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = f(x). \tag{2.1}$$

Where ρ, c_p, u, and k are constant. We take $f(x)$ to be the forcing function for the heat equation. We consider this on the domain $\Omega = (0, c)$. And we will prescribe the Dirichlet boundary conditions

$$T(0) = T_0$$
$$T(c) = T_c.$$

3. Variational Formulation

We will define the residual of equation (2.1) as

$$R(x) = \rho c_p u \frac{\partial T}{\partial x} - k\frac{\partial^2 T}{\partial x^2} - f.$$

Clearly this equals 0 if and only if T is the exact solution. We then multiply the residual by some test function w obtaining what is called the weighted residual

$$\int_0^c w(x)R(x)dx = 0.$$

For the exact solution, this will always be zero because $R(x) = 0$. Thus we can select any function for our test function $w(x)$ and this will be true.

This can be rewritten using integration by parts to obtain

$$\rho c_p u \int_0^c w\frac{\partial T}{\partial x}dx + k\int_0^c \frac{\partial w}{\partial x}\frac{\partial T}{\partial x}dx - kw\frac{\partial T}{\partial x}\Big|_0^c = \int_0^c wf dx.$$

We now revise our problem to be finding a function T that satisfies this expression. Because we apply our boundary conditions to the test function w we know that the test function is zero in the neighborhood of the boundaries by our definition of the test function $w(0) = w(c) = 0$. Applying this fact provides the equation

Date: November 22, 2018.

$$\rho c_p u \int_0^c w \frac{\partial T}{\partial x} dx + k \int_0^c \frac{\partial w}{\partial x}\frac{\partial T}{\partial x} dx = \int_0^c w f dx. \tag{3.1}$$

4. Discretization

Since the Sobolev spaces that the solution is contained in is infinitely dimensional, we will take the finite dimensional subspace $V_h \subset H^1((0,c))$ to create our approximation in. The Sobolev space $H^1((0,c))$ is the space of continuous functions that are infinitely differentiable. We will take a set of basis function $\{\varphi_j\}$. We leave these basis functions as arbitrary functions for now, and will show the process for selecting our basis functions. We can then write our approximation T_h of T as a linear combination of this finite subspace

$$T_h = \sum_{j=1}^{N} T_j \varphi_j(x). \tag{4.1}$$

Where T_j are constants that we will solve for in a later stage. Substituting this into (3.1) and taking our test function in this subspace $w_h = \varphi_i$, we find

$$\sum_{j=1}^{N}\left[\rho c_p u \int_0^c \varphi_i \frac{\partial \varphi_j}{\partial x} dx\right] T_j + \sum_{j=1}^{N}\left[k\int_0^c \frac{\partial \varphi_i}{\partial x}\frac{\partial \varphi_j}{\partial x} dx\right] T_j = \int_0^c \varphi_i f dx.$$

We can write this system of equations for $i = 1,\ldots,N$ as

$$(C+K)T = F. \tag{4.2}$$

Where T is vector in $\mathbb{R}^N$ of T_j constants, and the elements of the matrices C, K, F are given defined as

$$\begin{aligned} c_{ij} &= \rho c_p u \int_0^c \varphi_i \frac{\partial \varphi_j}{\partial x} dx && i,j = 1,\ldots,N \\ k_{ij} &= k\int_0^c \frac{\partial \varphi_i}{\partial x}\frac{\partial \varphi_j}{\partial x} dx && i,j = 1,\ldots,N \\ F_i &= \int_0^c \varphi_i f dx && i = 1,\ldots,N. \end{aligned}$$

Note that it is simpler to consider the matrix $A = C + K$ and use A instead.

5. Localization

It is possible to find these matrices on the global domain, but that proves to be extremely difficult in higher dimensions, so we will use a generalizable method for the generation of these matrices. Since we are able to select our basis functions φ_j we can carefully construct a set of basis functions such that it is possible to split the domain into elements. Where our element is a subset of the domain. We then determine the matrices for every element. Then using an algorithm we combine the element matrices to construct the global matrices.

We discretize the domain using N grid points $x_1 < x_2 < \cdots < x_N$ such that $x_1 = 0$ and $x_N = c$. Note that N is the dimension of our basis functions $\{\varphi_j\}$ that were selected in section 4. This will allow us to construct a mesh of $N-1$ elements defined as

$$E_e = [x_e, x_{e+1}] \qquad e = 1,\ldots,N-1.$$

For simplicity we will assume that all elements have the same size h, although different sizes can be easily handled.

In order to construct our approximate solution, we define the shape function. We want all of our element basis functions to be of this shape function, in order to simplify the computations for the system

$$T_h^{(e)} = T_h\Big|_{E_e} \qquad e = 1,\ldots,N-1.$$

We approximate this shape function as some first-order polynomial. Note that for higher accuracy it is possible to approximate the shape function as a higher order polynomial, but for simplicity we utilize first-order polynomials

$$T_h^{(e)}(x) = \alpha_1^{(e)} + \alpha_2^{(e)} x \qquad x \in [x_e, x_{e+1}]. \tag{5.1}$$

Then because we want T_h to be continuous, we must apply the boundaries of the element to the shape function

$$\begin{aligned} T_h^{(e)}(x_e) &= T_1^{(e)} \equiv T_h(x_e) \\ T_h^{(e)}(x_{e+1}) &= T_2^{(e)} \equiv T_h(x_{e+1}). \end{aligned}$$

Evaluating this expression and solving for $\alpha_1^{(e)}$ and $\alpha_2^{(e)}$ results in

$$\begin{aligned} \alpha_1^{(e)} &= eT_1^{(e)} - (e-1)T_2^{(e)} \\ \alpha_2^{(e)} &= \frac{1}{h}\left(T_2^{(e)} - T_1^{(e)}\right). \end{aligned}$$

Using these results we substitute them into (5.1), and find

$$\begin{aligned}
T_h^{(e)} &= eT_1^{(e)} - (e-1)T_2^{(e)} + \frac{1}{h}\left(T_2^{(e)} - T_1^{(e)}\right)x \\
&= T_1^{(e)}\left(\frac{-x}{h} + e\right) + T_2^{(e)}\left(\frac{x}{h} - e + 1\right) \\
&= T_1^{(e)}\varphi_1^{(e)}(x) + T_2^{(e)}\varphi_2^{(e)}(x).
\end{aligned}$$

Thus we define our local basis functions $\varphi_1^{(e)}$ and $\varphi_2^{(e)}$ as

$$\varphi_1^{(e)} = \frac{-x}{h} + e \tag{5.2}$$

$$\varphi_2^{(e)} = \frac{x}{h} - e + 1. \tag{5.3}$$

Thus for every element there is a pair of local basis functions, and a local shape function. A plot of these local basis functions is shown in figure 1.

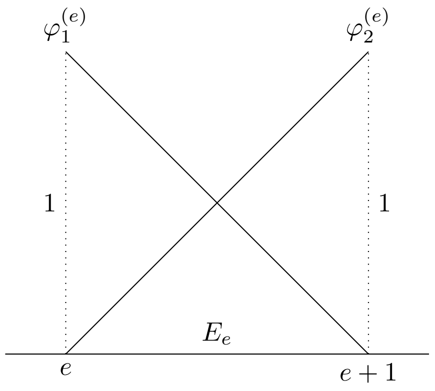

FIGURE 1. Local basis functions $\varphi_1^{(e)}$ and $\varphi_2^{(e)}$ on element e.

The global approximation is then found as a linear combination of global basis functions, which we now define as a piecewise combination of local basis functions. We construct our global basis functions $\varphi_j(x)$ as the combination of local basis functions such that at grid point j it is equal to 1 and at all other grid points is zero, as seen in figure 2.

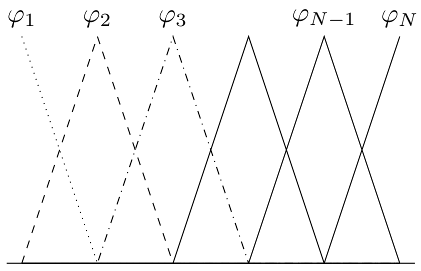

FIGURE 2. Global basis functions $\varphi_j(x)$.

Because of this construction of the global basis functions, we are able to represent them in terms of our local basis functions

$$\varphi_1(x) = \begin{cases} \varphi_1^{(1)}(x) & x \in E_1 = (0,h) \\ 0 & elsewhere \end{cases}$$

$$\varphi_2(x) = \begin{cases} \varphi_2^{(1)}(x) & x \in E_1 = (0,h) \\ \varphi_1^{(2)}(x) & x \in E_2 = (h,2h) \\ 0 & elsewhere \end{cases}$$

$$\vdots$$

$$\varphi_N(x) = \begin{cases} \varphi_2^{(N-1)}(x) & x \in E_{N-1} \\ 0 & elsewhere \end{cases}.$$

Thus from this construction we find that our general expression for the global basis function which is not on a boundary such as $j = 0$ and $j = N$ is as follows

$$\varphi_j(x) = \begin{cases} \frac{x}{h} - j + 2 & x \in ((j-2)h, (j-1)h) \\ \frac{-x}{h} + j & x \in ((j-1)h, jh) \\ 0 & elsewhere \end{cases}. \tag{5.4}$$

We are now able to use this definition of our basis function to apply in the construction of an element specific system of equations from (4.2).

Thus we construct our local system of equations in matrix form as

$$\left(C^{(e)} + K^{(e)}\right)T^{(e)} = F^{(e)}. \tag{5.5}$$

Here the elements of each matrix is determined by our local basis functions, and the size of the matrix is equivalent to the number of local degrees of freedom, which in this case is two (One for each end point of an element). The elements of these matrices are found through the following

$$\begin{aligned}
c_{ij}^{(e)} &= \rho c_p u \int_{E_e} \varphi_i^{(e)} \frac{\partial \varphi_j^{(e)}}{\partial x} dx && i,j = 1,2 \\
k_{ij}^{(e)} &= k \int_{E_e} \frac{\partial \varphi_i^{(e)}}{\partial x} \frac{\partial \varphi_j^{(e)}}{\partial x} dx && i,j = 1,2 \\
F_i^{(e)} &= \int_{E_e} \varphi_i^{(e)} f dx && i = 1,2.
\end{aligned}$$

6. Mesh Generation

To generate a mesh in 1D is to simply split the domain of our problem into a set of finite elements. We select the number of grid points that are desired N.

Then the number of elements that will be produced is $N-1$. For the 1D problem we will simply divide the domain $\Omega = (0, c)$ into $N-1$ evenly sized elements. Thus each interval with have length $h = \frac{c}{N-1}$. Note that the greater the number of elements will results in a higher accuracy approximation, but will be at the cost of computational complexity.

7. Assembly of the Global System

It is possible to directly construct the global matrix based, however this method is ineffective for practical computation and so in this introduction will be ignored.

The method that will be used is an incremental construction of the global matrix, by the addition of elements from the local matrices. We begin by initializing the matrix $A = C + K$ and F of zeros, in one dimension both of these matrices are $N \times N$. Then the corresponding contributions of the element matrices $A^{(e)} = C^{(e)} + K^{(e)}$ and $F^{(e)}$ are added in a loop over all elements.

Algorithm 1 element-by-element assembly

```
Let $A$ be a $N \times N$ matrix of zeros.
Let $F$ be a $N \times 1$ matrix of zeros.
for all $E_e$ do
    Let $A^{(e)}$ be a $2 \times 2$ matrix, such that
    $A^{(e)}_{IJ} = c^{(e)}_{IJ} + k^{(e)}_{IJ}, \quad I, J = 1, 2$
    Determine relation between global and local node numbers.
    $i = node(e, I) \quad j = node(e, J) \quad I, J = 1, 2$
    Update the global $A$ matrix.
    $A_{ij} = A_{ij} + A^{(e)}_{IJ} \quad I, J = 1, 2$
    Update the global $F$ matrix.
    $F_i = F_i + F^{(e)}_I \quad I = 1, 2$
    In our construction of the element mesh the relation between global and local is given like so
    $node(e, 1) = e \quad node(e, 2) = e + 1$
end for
return $A$, and $F$.
```

We demonstrate the results of this algorithm with an example.

Example 7.1. We will construct a global matrix for the element mesh with three elements. Thus $N = 4$.

We take $A^{(e)}$ to be a 2×2 matrix with elements defined as

$$A^{(e)}_{ij} = \rho c_p u \int_{E_e} \varphi^{(e)}_i \frac{\partial \varphi^{(e)}_j}{\partial x} dx + k \int_{E_e} \frac{\partial \varphi^{(e)}_i}{\partial x} \frac{\partial \varphi^{(e)}_j}{\partial x} dx.$$

And $F^{(e)}$ is a 2×1 matrix with elements defined as

$$F^{(e)}_i = \int_{E_e} \varphi^{(e)}_i f dx.$$

Using these two matrices, we construct the global matrix for $N = 4$ to be

$$A = \begin{pmatrix} A^{(1)}_{11} & A^{(1)}_{12} & 0 & 0 \\ A^{(1)}_{21} & A^{(1)}_{22} + A^{(2)}_{11} & A^{(2)}_{12} & 0 \\ 0 & A^{(2)}_{21} & A^{(2)}_{22} + A^{(3)}_{11} & A^{(3)}_{12} \\ 0 & 0 & A^{(3)}_{21} & A^{(3)}_{22} \end{pmatrix}$$
$$F = \begin{pmatrix} F^{(1)}_1 \\ F^{(1)}_2 + F^{(2)}_1 \\ F^{(2)}_2 + F^{(3)}_1 \\ F^{(3)}_2 \end{pmatrix}.$$

Using this algorithm, it is simple to construct the global A, and F matrices. It also becomes clear that the matrix of A is a tridiagonal matrix (At least in this one dimensional case).

8. Dirichlet Boundary Conditions

With our global system of equations constructed in the general form of

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_N \end{pmatrix}. \tag{8.1}$$

This solution is not well behaved, as we have neglected the prescribed boundary conditions until now. This means that our matrix is over determined and cannot be solved, to fix this we must apply our boundary conditions. As we know from our Dirichlet boundary conditions that

$$T_1 = T_0$$
$$T_N = T_c.$$

There are several ways of applying these boundary conditions to the system of equations, but we will focus on a single method. This method is to leave the boundary values of T_i as if they were unknown, and simply modify the system of equations such that they are forced to appear. This is done by modifying two of the equations of the system. This modification can be done to any two equations of the system, but in order to maintain the tridiagonality of this matrix, we will modify the first and last equations in the system. This can be done like so

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(N-1)1} & a_{(N-1)2} & \cdots & a_{(N-1)N} \\ 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N-1} \\ T_N \end{pmatrix} = \begin{pmatrix} T_0 \\ F_2 \\ \vdots \\ F_{N-1} \\ T_c \end{pmatrix}.$$

Using this system of equations is easy to implement, and allows for easy solving as it retains the tridiagonal nature of the matrix A.

9. Solving the Algebraic System

In order to solve the system of equations we must find the values for all a_{ij} and the values for F_i. The process for finding each of these is given, and since $f(x)$ is given as an input as the forcing function, we are able to numerically solve for all of these values.

Due to the tridiagonal nature of the matrix, it allows the implementation of extremely efficient algorithms to find the solution, that can obtain a solution in $O(n)$ operations instead of the normal $O\left(n^3\right)$ required by Gaussian elimination.

This algorithm is outlined below for a system of equations of the general form

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \cdots & 0 \\ 0 & a_3 & b_3 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & c_{n-1} \\ 0 & 0 & \cdots & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{pmatrix}.$$

Algorithm 2 Thomas algorithm

```
for $i \in 1, 2, \ldots, n-1$ do
    if $i = 1$ then
        $c_i = \frac{c_i}{b_i}$
        $d_i = \frac{d_i}{b_i}$
    else if $i \neq n$ then
        $c_i = \frac{c_i}{b_i - a_i c_{i-1}}$
        $d_i = \frac{d_i - a_i d_{i-1}}{b_i - a_i c_{i-1}}$
    else if $i = n$ then
        $d_i = \frac{d_i - a_i d_{i-1}}{b_i - a_i c_{i-1}}$
    end if
end for
for $i \in n, \ldots, 1$ do
    if $i = n$ then
        $x_i = d_i$
    else
        $x_i = d_i - c_i x_{i+1}$
    end if
end for
return $X$
```

This algorithm will efficiently determine the solution to our system of equations. A note about tridiagonal matrices is they are extremely efficient computationally. As only the three diagonals are needed to be stored as all other elements are zero. Because of this the memory for an $N \times N$ matrix can be reduced to the memory needed for a $N \times 3$ matrix. Which for very large number of elements will be extremely substantial.

This along with the efficiency of computing solutions to tridiagonal systems of equations indicates that maintaining the tridiagonality of the matrix is very useful.

We can then construct our final approximation of T like so

$$T_h(x) = T_0\varphi_1(x) + T_c\varphi_N(x) + \sum_{j=2}^{N-1} T_j\varphi_j(x). \tag{9.1}$$

10. Further Iterations

Because of our choice of differential equation, and our decision to neglect the time derivative, indicates that $T_h(x)$ is our final solution.

11. Postprocessing

This stage is to display the approximation of the solution, and to generate graphs or calculate errors. This step is entirely dependent on the example at hand and the desired use of FEA. Due to this we will skip this step for now.

12. Example Problem

For additional clarity we will implement this process on an example problem to demonstrate the stages, and the problem specific steps.

12.1. **Problem Statement.**

We will continue to use the steady 1D heat equation

$$\rho c_p y \frac{\partial T}{\partial x} - \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) = f.$$

We will use the domain $\Omega = (0, c) = (0, 5)$.

We will define $f(x)$ to be

$$f(x) = \frac{\xi\pi}{c}\left[\rho c_p u \cos\left(\frac{\xi\pi}{c}x\right) + k\frac{\xi\pi}{c}\sin\left(\frac{\xi\pi}{c}x\right)\right].$$

Where $\xi \in \mathbb{N}$, in our example we will take $\xi = 2$. And we will take $N = 4$. And finally we apply our Dirichlet boundary conditions to be

$$T_1 = T(0) = 0$$
$$T_2 = T(5) = 0.$$

12.2. **Variational Formulation.** As the variational formulation only depended on the differential equation, which we have kept the same, this step is identical to as it was in the explanation.

12.3. **Discretization.** Again, the discretization of the problem was only dependent on the differential equation. Thus we can use the same expression for the global system $(C + K)T = F$ that were found in the explanation.

12.4. **Localization.** The localization process is only dependent on the dimension of the problem. So for any FEA contained in one dimension will utilize the same $\varphi_i^{(e)}(x)$ and $\varphi_i(x)$. The only difference could come from the construction of the local matrix $C^{(e)}$, $K^{(e)}$, and $F^{(e)}$. But this process stems from the discretization process, and thus only depends on the differential equation. Which in our case has remained the same.

12.5. **Mesh Generation.** This is the first stage where some difference occurs, as it is the first computational stage. We are able to generate the mesh of elements, which in the one dimensional case is trivial. We take the set of node points

$$x = \left\{0, \frac{5}{3}, \frac{10}{3}, 5\right\}.$$

12.6. **Assembly of the Global System.** The process for assembling the global system is identical as in the explanation. However, with our example problem we are able to numerically solve for our global A and F matrices. These matrices become

$$A = \begin{pmatrix} 0.1 & -0.1 & 0.0 & 0.0 \\ -1.1 & 1.2 & -0.1 & 0.0 \\ 0.0 & -1.1 & 1.2 & -0.1 \\ 0.0 & 0.0 & -1.1 & 1.1 \end{pmatrix}$$
$$F = \begin{pmatrix} 1.453 \\ 0.842 \\ -2.275 \\ -0.021 \end{pmatrix}.$$

Thus our global system can be represented as

$$\begin{pmatrix} 0.1 & -0.1 & 0.0 & 0.0 \\ -1.1 & 1.2 & -0.1 & 0.0 \\ 0.0 & -1.1 & 1.2 & -0.1 \\ 0.0 & 0.0 & -1.1 & 1.1 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \begin{pmatrix} 1.453 \\ 0.842 \\ -2.275 \\ -0.021 \end{pmatrix}.$$

12.7. **Dirichlet Boundary Conditions.** Once again we use the process to apply our Dirichlet boundary conditions to the system of equations. Thus obtaining

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ -1.1 & 1.2 & -0.1 & 0.0 \\ 0.0 & -1.1 & 1.2 & -0.1 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \begin{pmatrix} 0.000 \\ 0.842 \\ -2.275 \\ 0.000 \end{pmatrix}.$$

12.8. **Solving the Algebraic System.** Using our algorithm for solving tridiagonal matrices, we are able to find the values for T_i to be

$$T = \begin{pmatrix} 0.000000 \\ 0.589228 \\ -1.355743 \\ 0.000000 \end{pmatrix}.$$

Thus resulting in our final approximation to be

$$T_h(x) = 0.589228\varphi_2(x) - 1.355743\varphi_3(x).$$

12.9. **Further Iterations.** Again as we neglect the time derivative $T_h(x)$ is our final solution.

12.10. **Postprocessing.** Now that we have an actual solution to our expression we are able to do some post processing.

We claim that a solution to this differential equation can be found analytically to be

$$T(x) = \sin\left(\frac{\xi\pi}{c}x\right).$$

Using the analytical solution, and our FEM approximation, we are able to plot each solution and compare the error function between the actual and our approximate solutions.

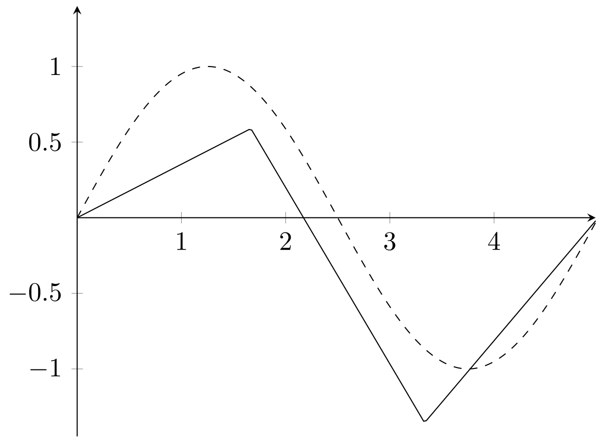

FIGURE 3. $T_h(x)$ compared to $\sin\left(\frac{2\pi}{5}x\right)$, when $N = 4$.

We can clearly see that the error at this number of grid points is quite substantial. We will compute the error by using L^2. Where we define our error function as

$$err = \left(\int_0^c \left| T_h(x) - \sin\left(\frac{2\pi}{5}x\right)\right|^2 dx \right)^{\frac{1}{2}}.$$

With our approximation of $T_h(x)$ we find that $err = 0.802665$. This amount of error is unacceptable in our problem. That is why FEA is much more effective with larger numbers of elements. We can directly view this, by automating the construction of T_h for many different N.

Using computational power to generate the plot of errors for each approximation.

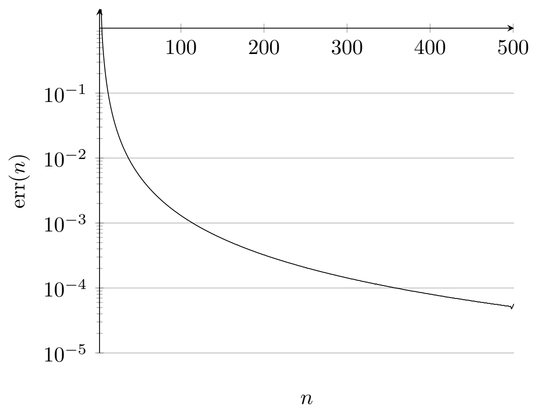

FIGURE 4. Plot of $err(n)$ for $n \in (2, 502)$.

The plot of the error function in figure 4 clearly shows that even with a few number of elements, the approximation quickly converges to the analytical solution. However, as it is only an approximation the error never reaches zero. Even with over 200 elements the error is approximately 1.49×10^{-4}, which for most situations is more than sufficient. However, with more complicated analytical solutions, the error function will take significantly longer to converge to zero.

In our one dimensional problem, the computational complexity for more N is linear. This is due to the fact that there the overall complexity of this process in one dimension is $\approx O(3N)$ Thus the increase in N results in an approximately linear increase in computation time. This can be seen in the graph in figure 5.

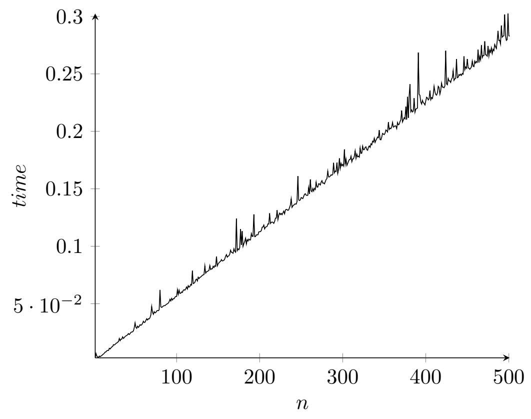

FIGURE 5. Plot of the time in seconds required to compute the approximation for $n \in (2, 502)$.

This can be somewhat decreased through parallel processing, as the computation of the local system of equations is independent. Thus this process can be split into multiple processes. However, due to the efficiency of this system on one dimension, this will only reduce the complexity to $\approx O(2N)$.

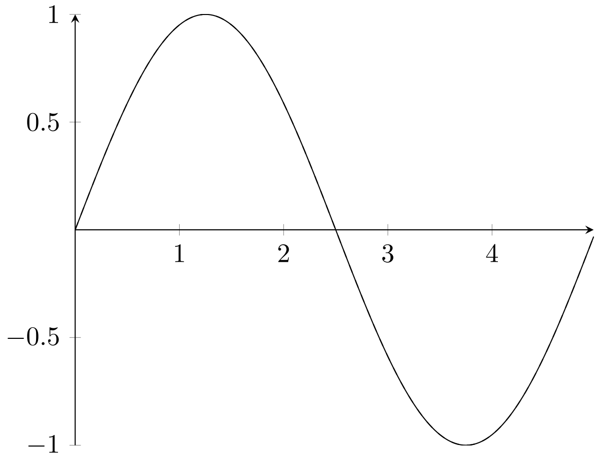

FIGURE 6. $T_h(x)$ compared to $\sin\left(\frac{2\pi}{5}x\right)$, when $N = 500$.

Figure 6 demonstrates the accuracy of the approximation for very large N values. In this case with $N = 500$ the error can be neglected, as it is on the order of 10^{-5}. This simple demonstration shows the strength of FEA, and how it can be applied to a simple one dimensional differential equation.

13. CONCLUSION

The method of finite element analysis constrained in one dimension is not extensively useful, as the analytic solutions for most differential equations is possible to compute without much difficulty. However, this demonstration and explanation of FEA shows the ability of of the process in comparison to an actual solution. And provides a generalizable process that can be expanded to higher dimensional problems.

As it was noted in the example problem, much of this process can be automated. The only parts that require manual work are the theoretical stages, and after than all other stages of the approximation process can be automated by a computer. This allows for much simpler process, as human work is only necessary for a change of dimension or a change of differential equation. All other altercations to the initial conditions, or the forcing function can be handled by computers.

References

[KH15] Dmitri Kuzmin and Jari Hämäläinen. *Finite Element Methods for Computational Fluid Dynamics.* Society for Industrial and Applied Mathematics, Philadelphia, 2015.