Abstract

- The Finite Element Method (FEM) is a numerical approach to approximate the solutions of boundary value problems involving second-order differential equations.
- Our boundary value problem was as follows:
  \[ \int_{0}^{1} f(x) = -u''(x) \quad 0 < x < 1 \]
  \[ u(0) = u(1) = 0 \]
- Due to the nature of the FEM, the problem is completely discrete, and is therefore able to be solved with normal computing methods.
- This method can be used to estimate the solutions to problems unsolvable by more traditional means, such as exact analytical methods.
- In our code, we used MATLAB’s FEM implementation, which is based on the FEM approach.

Details of Approach

- We expand \( u(x) \) in the basis of functions \( \psi_i, i = 1 \ldots n \), resulting in:
  \[ u_{FEM}(x) = \sum_{i=1}^{n} c_i \psi_i(x) = C \]
- Matrix \( A \) is the stiffness matrix of the equation, size \( n \times n \), and requires only knowledge of the basis to generate:
  \[ A = \int_0^1 \psi_i'(x) \psi_j'(x) \, dx \]
- The FEM approximate solution is obtained by solving a system of linear equations for the unknown \( C \):
  \[ AC = B \]
- Any change made to \( f(x) \) will not affect the values of \( A \).

Method Formulation

- This implementation of the FEM is based on the weak formulation of Equation (1), equal to:
  \[ \int_0^1 u'(x) v'(x) = \int_0^1 f(x) v(x) \]
- \( v(x) \) is a test function, which we then choose to be an element of a basis function. For our implementation, we chose the following hat function for each basis element, given \( n = 2^N \) number of nodes:
  \[ \psi_i(x) = \begin{cases} nx - i + 1 & \text{if } x \in \left[ \frac{i-1}{n}, \frac{i}{n} \right] \\ -nx + i + 1 & \text{if } x \in \left[ \frac{i}{n}, \frac{i+1}{n} \right] \\ 0 & \text{otherwise} \end{cases} \]
- The nodes are points of the form \( x_i = \frac{i}{n} \), \( i = 0 \ldots n \).
- This basis of functions is countable, which allows the solution \( u(x) \) to be solved via a system of linear equations, represented by the matrix equation for the unknown \( C \): \[ AC = B \]
- Where \( A, B \) are both found using the chosen basis of functions and Equation (2).

Stiffness Matrix A

- Matrix \( A \) is the stiffness matrix of the equation, size \( n \times n \), and requires only knowledge of the basis to generate:
  \[ A_{ij} = \int_0^1 \psi_i'(x) \psi_j'(x) \, dx = [A_{ij}] \]
- Due to our choice of basis, our stiffness matrix could be calculated explicitly, and took a form similar to the following:
  \[ A = \begin{bmatrix} 2n & -n & \ldots & 0 \\ -n & 2n & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & -n & 2n \end{bmatrix} \text{ for } n \text{ nodes} \]

Results and Error

- The most useful measure of error for this program was the infinity norm error, which is equal to:
  \[ E_\infty = \max_{x \in [0,1]} |u(x) - u_{FEM}(x)| \]
- By nature of our approach, the infinity error decreases by \( \frac{1}{2} \) every time you increase \( N \), or every time you double the number of used nodes.
- Below is a graphical demonstration of our initial problem for increasing values of \( N \) along with a table of increasing values of \( N \) and corresponding errors.

Source Vector B

- \( B \) is a vector, \( n \) values long, generated with a combination of the chosen basis functions, and the input function \( f(x) \):
  \[ B_i = \int_0^1 f(x) \psi_i(x) \, dx \]
- While generalizing the code for solving generic \( f(x) \), a numerical integration is required to calculate \( B \). The Simpson’s Method was used for this step.

Calculating & Displaying Results

- Once \( A \) and \( B \) are calculated, \( C = A^{-1} B \). It is important to note that the stiffness matrix is always invertible for our choice of basis.
- In our code, we used MATLAB’s Gaussian Elimination to calculate the \( C \) vector.
- Once the values of \( C \) are calculated, Equation (3) is used to find \( u(x) \).
- The values of \( u_{FEM}(x) \) are then graphed at the node points, and in a case where the answer to the initial problem can be found exactly, the value of the node points can be superimposed on top of the known solution for error calculation.

Sources and Thanks


Special thanks to Dr. Chun Liu and Ms. Carol Gaertner for their contributions to this research project.