Matlab 1D Finite Element Method (FEM) Tutorial

Cheuk Lau

Febrary 5, 2014

Overview

The goal of this tutorial is ***not*** to provide a mathematically rigorous background on the finite element method (FEM). For those interested in advanced FEM topics and a deeper understanding of its mathematical background I would direct you to a couple references:

1. Grossmann, Roos and Stynes, *Numerical Treatment of Partial Differential Equations*, Springer.
2. Ern, Guermond, *Theory and Practice of Finite Elements*, Applied Mathematical Sciences, Vol. 159, Springer (2004).

The goals of this tutorial is to:

1. Explain the ***basic*** concepts of the finite element method (FEM).
2. Show how FEM is applied to solve a ***simple*** 1D partial differential equation (PDE).
3. Discuss the provided Matlab files.

The provided Matlab files may serve as a launching point for writing one's own code to solve 1D FEM problems. Extending the code to multi-dimensions follows the same priciples. A 2D FEM code will be published in the future.

Basic Idea

FEM is a numerical method for solving PDEs over a spatial domain. The following steps make up the FEM:

1. Choose a ***test function*** to derive the ***weak form*** of your PDE.
2. ***Discretize*** your problem domain into elements.
3. Choose a ***shape function*** to interpolate your solution over each element.
4. Solve the resulting linear system using a chosen ***iterative method***.

The best way to learn is by example so lets go through how to apply the FEM to solve a simple 1D problem.

Simple 1D Problem

Problem Description

Consider the one-dimensional problem:

$$-\left(a\left(x\right)u'\left(x\right)\right)' + u = f,\ x \in (0,2) \quad (1)$$

where:

$$a(x) = 1 + x$$

with boundary conditions:

$$u(0) = 1$$

$$u'(2) = 3$$

Lets choose$f$ such that the solution is:

$$u(x) = 1 + 3x + x(2 - x)^2 \quad (2)$$

Inserting eqn 2 into eqn 1 gives:

$$f = 2 + 17x - 13x^2 + x^3 \quad (3)$$

Knowing the analytical solution for $u(x)$ allows us to calculate error convergence as a function of mesh size for code validation. As a side note, this is known as the method of manufactured solutions (MMS).

Step 1: Weak Formulation

The first step is to derive the weak form of eqn 1. We start by selecting a ***test function*** $v \in V$ as follows:

$$V = \left\{ v : \begin{array}{c} v(x) \text{ is continuous on } (0,2) \\ v'(x) \text{ is piecewise continuous on } (0,2) \\ v(0) = 0 \end{array} \right\}$$

The weak form of eqn 1 can now be derived as follows:

1. Multiply eqn 1 by *any* $v \in V$ and integrate over the problem domain:

$$-\int_0^2 (a(x)u'(x))' v(x)\,dx + \int_0^2 u(x)v(x)\,dx = \int_0^2 f(x)v(x)\,dx \quad (4)$$

2. Integrate the first term in eqn 4 by parts:

$$\Rightarrow \int_0^2 a(x)u'(x)v'(x)\,dx - a(2)u'(2)v(2) + a(0)u'(0)v(0) + \int_0^2 u(x)v(x)\,dx = \int_0^2 f(x)v(x)\,dx$$

Noting $v(0) = 0$ and $u'(2) = 3$:

$$\therefore \int_0^2 a(x)u'(x)v'(x) + u(x)v(x)\,dx = \int_0^2 f(x)v(x)\,dx + 3a(2)v(2) \quad (5)$$

3. Eqn 5 is the ***weak form*** of eqn 1. It can be further reduced by defining the operators:

$$a(u,v) = \int_0^2 a(x)u'(x)v'(x) + u(x)v(x)\,dx$$

$$L(v) = \int_0^2 f(x)\, v(x)\, dx + 3a(2)\, v(2)$$

resulting in:

$$\therefore a(u, v) = L(v) \quad (6)$$

The integrals in eqn 6 are performed using any standard numerical quadrature e.g., Gauss.

Steps 2 and 3: Discretization and Shape Functions

The next step is to divide the spatial domain into elements. For simplicity, we assume elements have the same length. We also choose ***shape functions*** to interpolate the solution over each element. For example, linear shape functions assume the solution is linear through each element. When working with finite elements, it is much easier to define a reference element defined over the reference domain:

$$\hat{\tau} \in [0, 1]$$

The reference element is mapped to each physical element in the problem. This is called ***affine transformation***. For our 1D problem this is simply:

$$T_i : [0, 1] \to [x_i, x_{i+1}]$$

$$T_i(\hat{x}) = x_i + \hat{x}(x_{i+1} - x_i), \hat{\tau} \in [0, 1] \quad (7)$$

where $\hat{x}$ is the position in the reference element and i denotes the physical element edges. For simplicity, we will use linear shape functions:

$$p(\hat{x}_i) = a + b\hat{x}_i,\ i = 0, 1 \quad (8)$$

where $a, b \in \mathbb{R}$, $\hat{x}_0$ is the left reference element edge and $\hat{x}_1$ is the right reference element edge. In matrix notation:

$$\begin{bmatrix} p(\hat{x}_0) \\ p(\hat{x}_1) \end{bmatrix} = \begin{bmatrix} 1 & \hat{x}_0 \\ 1 & \hat{x}_1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} p(\hat{x}_0) \\ p(\hat{x}_1) \end{bmatrix}$$

Lets define p as:

$$p_i(\hat{x}_j) = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases} \quad i, j = 0, 1$$

Then for p_0:

$$\begin{bmatrix} a_0 \\ b_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\therefore p_0(\hat{x}) = 1 - \hat{x} \quad (9)$$

Then for p_1:

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\therefore p_1(\hat{x}) = \hat{x} \quad (10)$$

Eqns 9 and 10 are the left and right linear shape functions over the reference element.

Aside: Numerical Quadrature

The integrals appearing in eqn 6 are computed using a numerical quadrature:

$$\int_{\hat{\tau}} f d\hat{x} \approx \sum_{i=1}^{M} w_i f(\hat{x}_i) \quad (11)$$

where $\{\hat{x}_l\} \subset [0,1]$ are the quadrature nodes and $\{w_i\}$ are the quadrature weights. For simplicity, we will use a simple two-point Gaussian quadrature:

$$w_1 = w_2 = 0.5 \quad (12)$$

$$\zeta_1 = 0.5 - \frac{1}{2\sqrt{3}}, \zeta_2 = 0.5 + \frac{1}{2\sqrt{3}} \quad (13)$$

Step 4: Iterative Method

Steps 1 through 4 have effectively created a linear system that can be solved using an iterative method (e.g., Jacobi, Gauss-Seidel, SOR, CG, GMRES). The choice of iterative solver is beyond the scope of this tutorial. The provided code will use the built-in Matlab GMRES solver.

Description of the Matlab Files

In the previous section we went over the necessary tools for using the FEM to solve a PDE. In this section we will go over each of the provided Matlab files to get a better understanding of how everything is actually put together.

ref_quad.m

- Input: none
- Output:
  - ref_quad_pos: quadrature positions on the ***reference*** element.

  - quad_weights: quadrature weights.

- Function:

  - Generates the two-point Gauss quadrature position and weights over the reference element.

dof.m

- Input:

  - num_edges: number of finite element edges over the physical domain.

- Output:

  - dof_pos: the location of the degrees of freedom for each finite element.

- Function:

  - Computes the location of the degrees of freedom for each finite element. For linear basis functions these are at the finite element edges.

eval_shape.m

- Input:

  - ref_quad_pos: Gauss quadrature positions on the reference element.

- Output:

  - ev: basis function (eqns 9 and 10) values at the Gauss quadrature positions on the reference element.

- Function:

  - Solves eqns 9 and 10 for the basis function values at the Gauss quadrature positions on the reference element:

$$f_0(\zeta_1) = 1 - \left(0.5 - \frac{1}{2\sqrt{3}}\right) = 0.789$$

$$f_0(\zeta_2) = 1 - \left(0.5 + \frac{1}{2\sqrt{3}}\right) = 0.211$$

$$f_1(\zeta_1) = 0.5 - \frac{1}{2\sqrt{3}} = 0.211$$

$$f_1(\zeta_2) = 0.5 + \frac{1}{2\sqrt{3}} = 0.789$$

eval_dir_shape.m

- Input: none
- Output:
  - evder: the basis function derivative values at the Gauss quadrature positions on the reference element.
- Function:
  - Solves the derivatives of eqns 9 and 10 at the Gauss quadrature positions on the reference element. The derivative of f_0 (eqn 9) is:

$$f_0'(x) = -1$$

The derivative of f_1 (eqn 10) is:

$$f_1'(x) = 1$$

The derivative of f_o and f_1 at the two-point Gaussian quadrature reference nodes are

$$f_0'(\zeta_1) = f_0'(\zeta_2) = -1$$

$$f_1'(\zeta_1) = f_1'(\zeta_2) = 1$$

act_quad.m

- Input:
  - dof_pos: degrees of freedom positions over physical domain.
  - ref_quad_pos: Gauss quadrature positions on reference element.
  - num_edges: number of degrees of freedom over physical domain.
- Output:
  - act_quad_pos: quadrature positions on each physical element.
- Function:
  - Maps the quadrature positions from the reference element to each physical element.

Table 1: loc_glob using six degrees of freedom

Element number i	x_i	x_{i+1}
0	-5	1
1	1	2
2	2	3
3	3	4
4	4	6

glob_to_loc.m

- Input:
  - num_edges: number of finite element edges.
- Output:
  - loc_glob: this is a table of size number of elements $\times 2$. Each row contains the left and right degrees of freedom index for the corresponding element. The first element in the table corresponds to the leftmost degree of freedom. This element is negative to flag an essential boundary condition exists i.e., $u(0) = 1$. It is worth noting that the last element corresponding to the rightmost degree of freedom does not need to be flagged because it is a natural boundary condition. To better illustrate the loc_glob matrix consider the degrees of freedom locations $\{x_0, x_1, x_2, ..., x_{N-1}, x_N\}$. The global degrees of freedom are numbered from left to right as $\{-N, 1, 2, ..., N-1, -(N+1)\}$. The table 1 shows loc_glob with six degrees of freedom. loc_glob will be used later to assemble the right-hand vector of eqn 6 i.e., $L(v)$.

compute_RHS_local.m

- Input:
  - quad_weights: Gauss quadrature weights.
  - dof_pos: degrees of freedom positions.
  - act_quad_pos: quadrature positions mapped onto the physical domain.
  - ev: shape function values at quadrature positions on reference element.
  - num_edges: number of finite element edges.
- Output:

- RHS_local: this is a matrix of size number of elements $\times 2$. Each row represents the numerical integration of each shape function over the element using the Gauss quadrature. Recall the right-hand side of the system i.e., eqn 6 is:

$$L(v) = \int_0^2 f(x)\, v(x)\, dx + 3a(2)\, v(2) \quad (14)$$

If we consider a general function $f(x)$ the integrated contribution to element j for basis function i is:

$$F_{ij} = (x_{j+1} - x_j) \int_0^1 f(x)\, \hat{\phi}(\hat{x})\, d\hat{x} \approx (x_{j+1} - x_j) \sum_{l=1}^{M} w_l f(x_{j,l})\, \hat{\phi}_i(\zeta_l) \quad (15)$$

Eqn 15 is applied to eqn 14 to form RHS_local.

compute_RHS_global.m

The input, output and function of *RHS_global.m* are:

- Input:
  - RHS_local: local integrated contibution of the RHS integral in eqn 6 for each basis function over each element.
  - loc_glob: matrix of edge indices for each element.
  - num_edges: number of finite element edges.
- Output:
  - RHS_global: this is a matrix of size number of elements $\times 1$. The contributions from RHS_loc are added up for each degree of freedom using loc_glob.

compute_stiff_local.m

- Input:
  - dof_pos: location of degrees of freedom in physical domain.
  - quad_weights: two-point Gauss quadrature weights.
  - act_quad_pos: Gauss quadrature nodes on physical domain.
  - ev: shape function values at quadrature nodes on reference element.
  - evder: shape function derivative values at quadrature nodes on reference element.
  - num_edges: number of finite element edges.
- Output:

- stiff_local: this is a number of elements $\times 9$ matrix. Each row represents the 3×3 local stiffness matrix for each element. Recall the left-hand side of the system we are solving for i.e., eqn 6:

$$a(u,v) = \int_0^2 a(x)\, u'(x)\, v'(x) + u(x)\, v(x)\, dx \quad (16)$$

The local stiffness matrix for element j is:

$$M_{j,il} = a_j(\phi_l, \phi_i)$$

$$\Rightarrow M_{j,il} = \int_{x_j}^{x_{j+1}} a(x)\, \phi_l^{'} \phi_i^{'} dx$$

$$\Rightarrow M_{j,il} = (x_{j+1} - x_j)^{-1} \int_0^1 a(x)\, \hat{\phi}_l^{'} \hat{\phi}_i^{'} d\hat{x}$$

$$\Rightarrow M_{j,il} \approx (x_{j+1} - x_j)^{-1} \sum_{t=1}^{M} w_t a(x_{j,t})\, \hat{\phi}_l^{'}(\zeta_t)\, \hat{\phi}_i^{'}(\zeta_t) \quad (17)$$

Eqn 17 is applied to eqn 16 to form stiff_local.

compute_stiff_global

- Input:
  - stiff_local: element-wise local stiffness matrices.
  - num_edges: number of finite element edges.
- Output:
  - stiff_global: this is a number of elements $\times$number of elements matrix. It is the combination of the local stiffness matrices.

a_def.m

Defines $a(x)$ given in eqn 1.

F_def.m

Defines $f(x)$ given in eqn 3.

A few words on the application of the left BC

In the global stiffness matrix I did not include the left-hand boundary condition. This becomes obvious when I remove the first element from the global_RHS vector before calling GMRES to solve the system. To incorporate the left-hand boundary condition back into the final solution we must add its value to every computed element in the solution vector. This is done in the *master.m* file on line 70.

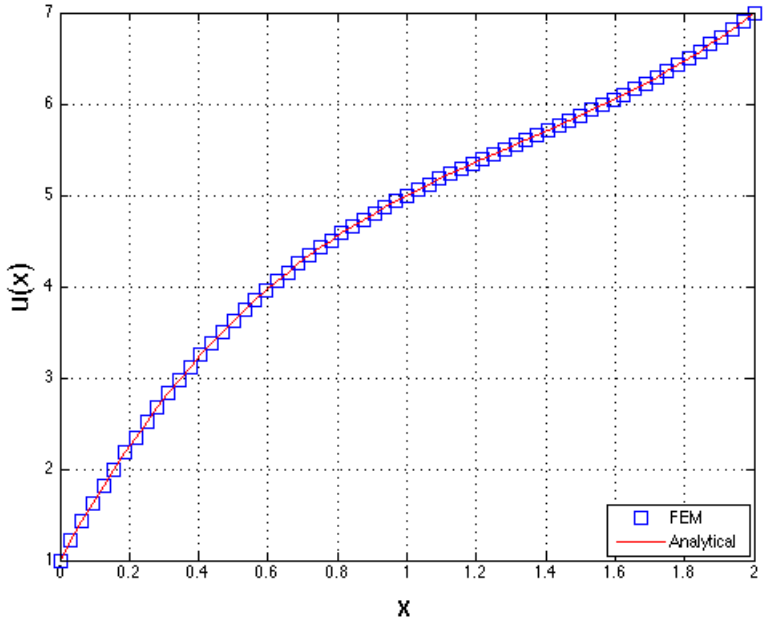

Figure 1: FEM computed and analytical solutions using $h = \frac{1}{32}$

A few words on error calculation

The error in the FEM solution is ***NOT, I REPEAT NOT,*** the error at the degrees of freedom! A common mistake is to assume the error in the solution is simply the norm of the errors at each degree of freedom. However, FEM is a numerical method that attempts to preserve the *integral* of the PDE over the problem domain. Therefore the error in the FEM solution is the *error* in the integral over each finite element. To compute the error over each finite element, project the solution at the degrees of freedom over each element using the left and right linear shape functions. Then integrate the projection over the element using Gauss quadrature. You can then take the norm (e.g., Euclidean) over all the element errors.

Running the code

Go to *master.m* and adjust the element width size if desired. Hit run. A plot of the computed and analytical (eqn 2) results will appear. Fig 1 provides the results you should get using a mesh size of $h = \frac{1}{32}$.