Introduction to Numerical Analysis
Final exam project:
Finite element method for 1D second order elliptic equation

Dr. Xiaoming He*

Complete this project INDEPENDENTLY. Show all relevant work in detail to justify your conclusions. Partial credit depends upon the work you show. For each numerical experiment, all the .m files of your Matlab code should be electronically submitted to hex@mst.edu together with a .txt file which copies all the information in the Matlab command window when you run the code to obtain the numerical results.

Consider the following 1D second order elliptic equation with Dirichlet boundary conditions:

$$\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \le x \le b), u(a) = g_a, u(b) = g_b$$

where $u(x)$ is the unknown function, g_a and g_b are the Dirichlet boundary values, $c(x)$ is a given coefficient function and $f(x)$ is a given source function. See the theorem 10.1 in the textbook for the existence and uniqueness of the solution.

In this project, we will apply the methods we learn in this course in a quick introduction of the finite element methods for the above equation. For this purpose, we will simplify the general implementation framework of finite elements and reduce the theoretical part, especially the use of functional spaces. More details and key ideas in both implementation and theory will be discussed in Math 5602 (Mathematical Foundation for Finite Element Methods). You can also find a short introduction in section 10.1.3 of the textbook.

1 Introduction to the finite element methods

The main components of the finite element methods include weak formulation of the equation, mesh, finite element space, finite element discretization which leads to a linear system, assembly of the matrix and vector of the linear system, treatment of the boundary conditions, and solver of the linear system. The piecewise polynomial approximation we learn in Chapter 4 will be used to construct the finite element space; the numerical integration we learn in Chapter 6 will be used to compute the integrals in the finite element discretization; the direct/iterative methods we learn in Chapter 3 will be used to solve the linear system arising from the finite element methods. In the following, we will introduce these components one by one.

1.1 Weak Formulation

Let's first use the target problem as an example to introduce the weak formulation. First, multiply a test function $v(x)$ and then take the integral from a to b on both sides of the equation $\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)$, we obtain

$$\int_a^b \frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) v(x)\ dx = \int_a^b f(x)v(x)\ dx.$$

Here $u(x)$ is called a trial function. Second, using integration by parts, we obtain

$$\int_a^b \frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) v\ dx \quad = \quad \int_a^b v\ d(cu') = cu'v|_a^b - \int_a^b cu'\ dv = c(b)u'(b)v(b) - c(a)u'(a)v(a) - \int_a^b cu'v'\ dx.$$

*Department of Mathematics and Statistics, Missouri University of Science and Technology, Rolla, MO 65409, hex@mst.edu

Hence

$$c(b)u'(b)v(b) - c(a)u'(a)v(a) - \int_a^b cu'v' \, dx = \int_a^b fv \, dx.$$

Third, since the Dirichlet boundary conditions $u(a) = g_a, u(b) = g_b$ directly provide the exact solution at the two ends, we don't need to do any test at a and b. Then we may choose the test function $v(x)$ such that $v(a) = v(b) = 0$. Therefore, the weak formulation is to find $u \in H^1[a, b]$ such that

$$-\int_a^b cu'v' \, dx = \int_a^b fv \, dx$$

for any $v \in H_0^1[a, b]$. Here $H^1[a, b]$ is the functional space whose functions are first weakly differentiable in $[a, b]$. And $H_0^1[a, b]$ is a functional space whose functions are first weakly differentiable in $[a, b]$ and also vanish at the two ends.

Remark: Since the definition of weakly differentiable functions is not introduced in our course, we may use $C^1[a, b]$ to replace $H^1[a, b]$ here for simplification. More details about functional spaces will be discussed in Math 5602 (Mathematical Foundation for Finite Element Methods). Here we also recall the following two definitions from the lecture slides of Chapter 1: Let $C[a, b]$ denote the function space of continuous functions on $[a, b]$; let $C^k[a, b]$ denote the function space of k^{th} differentiable functions on $[a, b]$; that is, if $f(x) \in C^k[a, b]$, then $f^{(i)}(x)$ $(i = 0, 1, \cdots, k)$ are continuous on $[a, b]$.

The basic idea of the Galerkin method is to use a finite dimensional space to approximate the infinite dimensional space in the weak formulation in order to form a finite system for the computers. For example, $H^1[a, b]$ needs to be approximated by a finite dimensional space V for the above weak formulation. When the space V is chosen to be a finite element space, the Galerkin method is called finite element method.

Therefore, the finite element methods highly reply on the weak formulation and the corresponding functional spaces, which are varying for different equations with different boundary conditions. In the following let's study and practice on the weak formulations for several fundamental cases.

Usually we can assume the 0 values of the test function $v(x)$ only at the Dirichlet boundary $(u(a) = g_a, u(b) = g_b)$, but not at the Neumann boundary $(u'(a) = p_a, u'(b) = p_b)$ or the Robin boundary $(u'(a)+q_a u(a) = p_a, u'(b)+q_b u(b) = p_b)$. Instead, the Neumann boundary conditions and the Robin boundary conditions should be imposed in the weak formulation directly and naturally.

Problem #1: Consider

$$\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x) \ (a \le x \le b).$$

(a) Derive the weak formulation with the boundary condition

$$u'(a) = p_a, u(b) = g_b.$$

(b) Derive the weak formulation with the boundary condition

$$u'(a) = p_a, u'(b) + q_b u(b) = p_b.$$

(c) Derive the weak formulation with the boundary condition

$$u'(a) = p_a, u'(b) = p_b.$$

Is the solution unique? Why?

For higher dimensional equations, the weak formulation and the corresponding functional spaces can be derived similarly. For example, for the 2D or 3D Poisson's equation, the following Green's formula (integration by parts in multi-dimension) can be used to derive the weak formulation:

$$\int_D \nabla \cdot (c\nabla u)v \, dxdy = \int_{\partial D} (c\nabla u \cdot \vec{n}) \, v \, dS - \int_D c\nabla u \cdot \nabla v \, dxdy.$$

Here $\vec{n}$ is the outward unit normal vector of the boundary of D, $\nabla u \cdot \nabla v = u_x v_x + u_y v_y$ for 2D case and $\nabla u \cdot \nabla v = u_x v_x + u_y v_y + u_z v_z$ for 3D case.

Problem #2: Consider the Poisson equation

$$\nabla \cdot (c \nabla u) = f \ \text{ in } D$$

where D is a 2D or 3D domain, f and c are given functions on D, and u is the unknown function.

(a) Derive the weak formulation with the boundary condition

$$u = g \ \text{ on } \partial D$$

where g is a given function on ∂D, the boundary of the domain D.

(b) Derive the weak formulation with the boundary condition

$$\nabla u \cdot \vec{n} + qu = p \ \text{ on } \partial D$$

where p and q are given functions on ∂D.

Remark: The above derivation and problems are just a quick introduction to the weak formulations of partial differential equations. More details for different equations with different types of boundary conditions will be discussed in Math 5602 (Mathematical Foundation for Finite Element Methods).

1.2 Mesh

For finite element methods, the problem domain needs to be partitioned into finite number of elements to construct a mesh. Then at least two matrices for the mesh information need to be formed, which are well known as the P and T matrices in finite element community. The matrix P consists of the coordinates of all mesh nodes. Here we use the j^{th} column of the matrix P to store the coordinates of the j^{th} mesh node. The matrix T consists of the indices of the mesh nodes of all the elements. Here we use the n^{th} column of the matrix T to store the indices of the mesh nodes of the n^{th} mesh element.

For the one-dimensional problem discussed in this project, assume that we have a uniform partition of $[a, b]$ into N elements with mesh size $h = \frac{b-a}{N}$. Let $x_j = a + (j-1)h$ $(j = 1, \cdots, N+1)$ denote the mesh nodes and $E_n = [x_n, x_{n+1}]$ $(n = 1, \cdots, N)$ denote the mesh elements. Then it is straightforward to form these two matrices as follows.

$$P = \begin{pmatrix} x_1 & x_2 & \cdots & x_N & x_{N+1} \end{pmatrix}, \ T = \begin{pmatrix} 1 & 2 & \cdots & N-1 & N \\ 2 & 3 & \cdots & N & N+1 \end{pmatrix}.$$

Remark: The mesh information stored in matrices P and T plays a key role in the "local assembly" idea in the general implementation framework of finite elements, which will be introduced in section 1.4 later. For a 1D problem, the mesh information stored in P and T is straightforward. Hence it is not necessary to form these two matrices in order to use the mesh information for a 1D problem. Therefore, in order to simplify the introduction for finite elements in this project, we will not bother to use them for the "local assembly" idea. However, for higher dimensional problems with more complicated problem domains, the mesh generation is much more complicated and the corresponding mesh information stored matrices P and T provide a very convenient way for the "local assembly" idea. Furthermore, for a problem whose discretization formulation involves with integrals on the mesh edges, then a matrix E for the information of mesh edges is also needed. More details about these issues will be discussed in Math 5602 (Mathematical Foundation for Finite Element Methods).

1.3 Finite element space

In this section, we will use the piecewise polynomial approximations we learn in Chapter 4 to construct a finite element space U_h, which is critical for the discretization of the weak formulation.

Define the finite element space

$$U_h = \{\phi \in C[a,b] : \ \phi(x) \text{ is linear on each } [x_n, x_{n+1}] \ (n = 1, 2, \cdots, N)\}.$$

Recall the following theorem from the lecture slides of Chapter 4.

Theorem: U_h is an $(N+1)-$dimensional subspace of $C[a,b]$.

From the proof of the above theorem in the lecture slides of Chapter 4, we can see that

$$U_h = span\{\phi_j\}_{j=1}^{N+1}$$

where

$$\begin{aligned}
\phi_1(x) &= \begin{cases} \frac{x_2 - x}{h}, & if \ x_1 \le x \le x_2, \\ 0, & otherwise, \end{cases} \\
\phi_j(x) &= \begin{cases} \frac{x - x_{j-1}}{h}, & if \ x_{j-1} \le x \le x_j, \\ \frac{x_{j+1} - x}{h}, & if \ x_j \le x \le x_{j+1}, \\ 0, & otherwise. \end{cases} \\
& (i = 2, \cdots, N) \\
\phi_{N+1}(x) &= \begin{cases} \frac{x - x_N}{h}, & if \ x_N \le x \le x_{N+1}, \\ 0, & otherwise, \end{cases}
\end{aligned}$$

are the finite element basis functions.

We will use this space to discretize the weak formulation in order to form the finite element discretization in the next section. However, before we do this, we need to analyze the approximation capability of this space by using the finite element interpolation error estimates. Define $I_h u(x) = \sum_{i=1}^{N+1} u(x_i)\phi_i(x)$ to be the 1D linear finite element interpolation of u. Then the error estimates in the following problem, which can be proved by using the approximation theory we learn in Chapter 4, show that the space U_h is a good approximation for the discretization.

Problem #3: If $u \in C^2[a,b]$, then prove the following conclusions

$$\begin{aligned}
\|u - I_h u\|_\infty &\le \frac{1}{8} h^2 \|u''\|_\infty, \\
\|(u - I_h u)'\|_\infty &\le \frac{1}{2} h \|u''\|_\infty.
\end{aligned}$$

Remark: The above finite element space is just the simplest one for 1D problems. More finite element spaces for higher dimensional problems and different other purposes will be discussed in detail in Math 5602 (Mathematical Foundation for Finite Element Methods).

1.4 Finite element discretization

Recall that the weak formulation is to find $u \in H^1[a,b]$ such that

$$-\int_a^b cu'v' \ dx = \int_a^b fv \ dx$$

for any $v \in H_0^1[a,b]$. Then by using the finite element space U_h to approximate the space $H^1[a,b]$ in the weak formulation, we can define the finite element formulation as follows: find $u_h \in U_h$ such that

$$-\int_a^b cu_h'v_h' \ dx = \int_a^b fv_h \ dx$$

for any $v_h \in U_h$.

Since $u_h \in U_h = span\{\phi_j\}_{j=1}^{N+1}$, then

$$u_h = \sum_{j=1}^{N+1} u_j \phi_j$$

for some coefficients u_j $(j = 1, \cdots, N+1)$. If we can set up a linear algebraic system for u_j $(j = 1, \cdots, N)$ and solve it, then we can obtain the finite element solution u_h. Therefore, we choose the test function $v_h = \phi_i$ $(i = 1, \cdots, N+1)$. Then the finite element formulation gives

$$\begin{aligned} & -\int_a^b c \left(\sum_{j=1}^{N+1} u_j \phi_j \right)' \phi_i' \, dx = \int_a^b f \phi_i \, dx, \ i = 1, \cdots, N+1 \\ \Rightarrow \quad & \sum_{j=1}^{N+1} u_j \left[-\int_a^b c \phi_j' \phi_i' \, dx \right] = \int_a^b f \phi_i \, dx, \ i = 1, \cdots, N+1. \end{aligned}$$

Define the stiffness matrix $A = [a_{ij}]_{i,j=1}^{N+1} = [-\int_a^b c\phi_j'\phi_i' \, dx]_{i,j=1}^{N+1}$, load vector $\vec{b} = [b_i]_{i=1}^{N+1} = [\int_a^b f\phi_i \, dx]_{i=1}^{N+1}$, and unknown vector $\vec{X} = [u_j]_{j=1}^{N}$. Then we obtain the linear algebraic system $A\vec{X} = \vec{b}$. Once this system is solved to obtain u_j $(j = 1, \cdots, N+1)$, the finite element solution $u_h = \sum_{j=1}^{N+1} u_j \phi_j$ is obtained. In fact, since

$$\phi_j(x_k) = \delta_{jk} = \begin{cases} 0, & \text{if } j \neq k, \\ 1, & \text{if } j = k. \end{cases}$$

then

$$u_h(x_k) = \sum_{j=1}^{N+1} u_j \phi_j(x_k) = u_k.$$

Hence the coefficient u_j is actually the numerical solution at the node x_j $(j = 1, \cdots, N+1)$.

For other types of partial differential equations and higher dimensional problems, the finite element discretization can be similarly derived. More details will be discussed in Math 5602 (Mathematical Foundation for Finite Element Methods). Here we just practice on a simple 2D problem.

Problem #4: Consider the weak formulation obtained in problem 2(a). Assume that we have already generated a mesh for D, constructed the corresponding finite element basis functions ϕ_j $(i = 1, \cdots, N+1)$, and formed the finite element space $U_h = span\{\phi_j\}_{j=1}^{N+1}$. Define the corresponding finite element discretization and derive the linear system $A\vec{X} = \vec{b}$ in the same way as above. (Write down the matrix A and vector $\vec{b}$ in the above explicit way)

1.5 Assembly of the stiffness matrix and load vector

We will discuss the solver for the linear system $A\vec{X} = \vec{b}$ in the next section. In this section, we need to investigate how to compute the integrals and assemble the stiffness matrix and load vector, which involves with numerical integration and the "local assembly" idea in the general implementation framework of finite elements.

From the definition of ϕ_j $(j = 1, \cdots, N+1)$, we can see that ϕ_j are non-zero only on the elements adjacent to the node x_j, but 0 on all the other elements. This observation motivates us to think about

$$a_{ij} = -\int_a^b c\phi_j'\phi_i' \, dx \quad = \quad -\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx, \ i, j = 1, \cdots, N+1.$$

This idea changes the integral on the whole domain $[a, b]$ to the summation of integrals on "local" elements, which will lead to the "local assembly" idea for the stiffness matrix. In the following, we will discuss different cases to compute $\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx$ $(i, j = 1, \cdots, N+1, n = 1, \cdots, N)$.

When $|i-j|>1$, x_i and x_j are not neighboring mesh nodes. Then on any element $[x_n, x_{n+1}]$ $(n=1,\cdots,N)$, at least one of ϕ_j and ϕ_i is 0. Hence

$$\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = 0\ (n=1,\cdots,N) \quad\Rightarrow\quad a_{ij} = -\int_a^b c\phi_j'\phi_i'\,dx = -\sum_{n=1}^{N}\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = 0 \ \text{ when } |i-j|>1.$$

When $i=j+1$ $(j=1,\cdots,N)$, the only element, on which both ϕ_j and ϕ_i are not zero, is $[x_j, x_{j+1}]$. Hence

$$\begin{aligned}
&\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = 0\ (n=1,\cdots,j-1,j+1,\cdots,N)\\
\Rightarrow\quad & a_{ij} = -\int_a^b c\phi_j'\phi_i'\,dx = -\sum_{n=1}^{N}\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = -\int_{x_j}^{x_{j+1}} c\phi_j'\phi_i'\,dx, \quad \text{when } i=j+1\\
\Rightarrow\quad & a_{j+1,j} = -\int_{x_j}^{x_{j+1}} c(x)\left(\frac{x_{j+1}-x}{h}\right)'\left(\frac{x-x_j}{h}\right)'\,dx = \frac{1}{h^2}\int_{x_j}^{x_{j+1}} c(x)\,dx\ (j=1,\cdots,N).
\end{aligned}$$

When $i=j-1$ $(j=2,\cdots,N+1)$, the only element, on which both ϕ_j and ϕ_i are not zero, is $[x_{j-1}, x_j]$. Hence

$$\begin{aligned}
&\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = 0\ (n=1,\cdots,i-1,i+1,\cdots,N)\\
\Rightarrow\quad & a_{ij} = -\int_a^b c\phi_j'\phi_i'\,dx = -\sum_{n=1}^{N}\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = -\int_{x_{j-1}}^{x_j} c\phi_j'\phi_i'\,dx, \quad \text{when } i=j-1,\\
\Rightarrow\quad & a_{j-1,j} = -\int_{x_{j-1}}^{x_j} c(x)\left(\frac{x-x_{j-1}}{h}\right)'\left(\frac{x_j-x}{h}\right)'\,dx = \frac{1}{h^2}\int_{x_{j-1}}^{x_j} c(x)\,dx\ (j=2,\cdots,N+1).
\end{aligned}$$

When $i=j$ $(j=2,\cdots,N)$, the only two elements, on which both ϕ_j and ϕ_i are not zero, are $[x_{j-1}, x_j]$ and $[x_j, x_{j+1}]$. Hence

$$\begin{aligned}
&\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = 0\ (n=1,\cdots,j-2,j+1,\cdots,N)\\
\Rightarrow\quad & a_{ij} = -\int_a^b c\phi_j'\phi_i'\,dx = -\sum_{n=1}^{N}\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = -\int_{x_{j-1}}^{x_j} c\phi_j'\phi_i'\,dx - \int_{x_j}^{x_{j+1}} c\phi_j'\phi_i'\,dx, \quad \text{when } i=j\\
\Rightarrow\quad & a_{jj} = -\int_{x_{j-1}}^{x_j} c(x)\left(\frac{x-x_{j-1}}{h}\right)'\left(\frac{x-x_{j-1}}{h}\right)'\,dx - \int_{x_j}^{x_{j+1}} c(x)\left(\frac{x_{j+1}-x}{h}\right)'\left(\frac{x_{j+1}-x}{h}\right)'\,dx\\
&\qquad = -\frac{1}{h^2}\int_{x_{j-1}}^{x_j} c(x)\,dx - \frac{1}{h^2}\int_{x_j}^{x_{j+1}} c(x)\,dx\ (j=2,\cdots,N).
\end{aligned}$$

When $i=j=1$, the only element, on which both ϕ_j and ϕ_i are not zero, is $[x_1, x_2]$. Hence

$$\begin{aligned}
&\int_{x_n}^{x_{n+1}} c\phi_1'\phi_1'\,dx = 0\ (n=2,\cdots,N)\\
\Rightarrow\quad & a_{11} = -\int_a^b c\phi_1'\phi_1'\,dx = -\sum_{n=1}^{N}\int_{x_n}^{x_{n+1}} c\phi_1'\phi_1'\,dx = -\int_{x_1}^{x_2} c\phi_1'\phi_1'\,dx,\\
\Rightarrow\quad & a_{11} = -\int_{x_1}^{x_2} c(x)\left(\frac{x_2-x}{h}\right)'\left(\frac{x_2-x}{h}\right)'\,dx = -\frac{1}{h^2}\int_{x_1}^{x_2} c(x)\,dx.
\end{aligned}$$

When $i=j=N+1$, the only element, on which both ϕ_j and ϕ_i are not zero, is $[x_N, x_{N+1}]$. Hence

$$\begin{aligned}
&\int_{x_n}^{x_{n+1}} c\phi_{N+1}'\phi_{N+1}'\,dx = 0\ (n=1,\cdots,N-1)\\
\Rightarrow\quad & a_{N+1,N+1} = -\int_a^b c\phi_{N+1}'\phi_{N+1}'\,dx = -\sum_{n=1}^{N}\int_{x_n}^{x_{n+1}} c\phi_{N+1}'\phi_{N+1}'\,dx = -\int_{x_N}^{x_{N+1}} c\phi_{N+1}'\phi_{N+1}'\,dx,\\
\Rightarrow\quad & a_{N+1,N+1} = -\int_{x_N}^{x_{N+1}} c(x)\left(\frac{x-x_N}{h}\right)'\left(\frac{x-x_N}{h}\right)'\,dx = -\frac{1}{h^2}\int_{x_N}^{x_{N+1}} c(x)\,dx.
\end{aligned}$$

From the above discussion, we can see that most of the integrals $\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx$ $(i,j=1,\cdots,N+1, n=1,\cdots,N)$ are 0 and most of the elements a_{ij} $(i,j=1,\cdots,N+1)$ are 0. Hence, we only need to compute the non-zero local integrals by using numerical integration and then assemble them into the stiffness matrix A. Here is the corresponding algorithm:

- Initialize the matrix: $A = sparse(N+1, N+1)$;
- Compute the integrals and assemble them into A:

  $FOR\ j = 1,\cdots,N+1$:

  $IF\ j \le N, THEN$

  Compute $A(j+1,j) = \frac{1}{h^2}\int_{x_j}^{x_{j+1}} c(x)\,dx$ by numerical integration;

  END

  $IF\ j \ge 2, THEN$

  Compute $A(j-1,j) = \frac{1}{h^2}\int_{x_{j-1}}^{x_j} c(x)\,dx$ by numerical integration;

  END

  $IF\ 2 \le j \le N, THEN$

  Compute $A(j,j) = -\frac{1}{h^2}\int_{x_{j-1}}^{x_j} c(x)\,dx - \frac{1}{h^2}\int_{x_j}^{x_{j+1}} c(x)\,dx$ by numerical integration;

  END

  Compute $A(1,1) = -\frac{1}{h^2}\int_{x_1}^{x_2} c(x)\,dx$ by numerical integration;

  Compute $A(N+1,N+1) = -\frac{1}{h^2}\int_{x_N}^{x_{N+1}} c(x)\,dx$ by numerical integration;

  END

Remark: The above algorithm uses the "local" idea, but it does NOT completely follow the "local assembly" idea in the general implementation framework of finite elements. This is because we are able to use the above special conclusions for the 1D problem to dramatically simplify it. Hence the above algorithm is actually simpler than the traditional one arising from the "local assembly" idea. Since the main purpose of this project is to apply the methods we learn in this course in a quick introduction of the finite element methods, we prefer this simpler algorithm here. However, for higher dimensional problems, we do NOT have those simple special conclusions any more even though we have some similar but more complicated conclusions. Hence it will be much more efficient to completely follow the "local assembly" idea there. Basically, instead of using with the global basis functions ϕ_j $(j=1,\cdots,N+1)$ on the whole domain, the key of the "local assembly" idea is to use the corresponding local basis functions on all the elements. In order to assemble the stiffness matrix, we need to go through a loop for all of the elements one by one. On each element, we need to compute the integrals corresponding to all the possible pairs of the local trial and test basis functions, and then assemble them into the matrix. In this way, we have actually already dealt with all the possible non-zero integrals. And this process will need the mesh information matrices P and T. Since we don't use the "local assembly" algorithm in this project, more details will be discussed in Math 5602 (Mathematical Foundation for Finite Element Methods).

The idea for the assembly of the load vector is similar. First, we have

$$\vec{b}_i = \int_a^b f\phi_i\,dx \quad = \quad \sum_{n=1}^{N}\int_{x_n}^{x_{n+1}} f\phi_i\,dx,\ i=1,\cdots,N+1,$$

When $2 \le i \le N$, the only two elements, on which ϕ_i is not zero, are $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$. Then

$$\int_{x_n}^{x_{n+1}} f\phi_i\,dx = 0\ (n=1,\cdots,i-2,i+1,\cdots,N)$$

$$\Rightarrow\quad \vec{b}_i = \sum_{n=1}^{N}\int_{x_n}^{x_{n+1}} f\phi_i\,dx = \int_{x_{i-1}}^{x_i} f\phi_i\,dx + \int_{x_i}^{x_{i+1}} f\phi_i\,dx = \int_{x_{i-1}}^{x_i} f(x)\frac{x-x_{i-1}}{h}\,dx + \int_{x_i}^{x_{i+1}} f(x)\frac{x_{i+1}-x}{h}\,dx.$$

When $i = 1$, the only element, on which ϕ_1 is not zero, is $[x_1, x_2]$. Then

$$\int_{x_n}^{x_{n+1}} f\phi_1 \, dx = 0 \ (n = 2, \cdots, N)$$

$$\Rightarrow \quad \vec{b}_1 = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} f\phi_1 \, dx = \int_{x_1}^{x_2} f\phi_1 \, dx = \int_{x_1}^{x_2} f(x) \frac{x_2 - x}{h} \, dx.$$

When $i = N + 1$, the only element, on which ϕ_{N+1} is not zero, is $[x_N, x_{N+1}]$. Then

$$\int_{x_n}^{x_{n+1}} f\phi_{N+1} \, dx = 0 \ (n = 1, \cdots, N-1)$$

$$\Rightarrow \quad \vec{b}_{N+1} = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} f\phi_{N+1} \, dx = \int_{x_N}^{x_{N+1}} f\phi_{N+1} \, dx = \int_{x_N}^{x_{N+1}} f(x) \frac{x - x_N}{h} \, dx.$$

Here is the corresponding algorithm:

- Initialize the matrix: $\vec{b} = zeros(N + 1, 1)$;
- Compute the integrals and assemble them into $\vec{b}$:

  $FOR \ i = 2, \cdots, N$:

  Compute $\vec{b}(i) = \int_{x_{i-1}}^{x_i} f(x) \frac{x - x_{i-1}}{h} \, dx + \int_{x_i}^{x_{i+1}} f(x) \frac{x_{i+1} - x}{h} \, dx$ by numerical integration;

  END

  Compute $\vec{b}(1) = \int_{x_1}^{x_2} f(x) \frac{x_2 - x}{h} \, dx$ by numerical integration;

  Compute $\vec{b}(N + 1) = \int_{x_N}^{x_{N+1}} f(x) \frac{x - x_N}{h} \, dx$ by numerical integration;

From the above algorithms, we can see that when we implement the finite element methods in section 2, we will need to use numerical integration to compute the integrals in the stiffness matrix and load vectors. Hence let's practice on numerical quadratures here.

1.6 Treatment of the boundary conditions

We need different techniques to deal with different types of boundary conditions. For example, as discussed in section 1.1, the Neumann boundary conditions and the Robin boundary conditions should be imposed in the weak formulation directly and naturally. After we apply the finite element discretization to the weak formulation, we can see that the boundary conditions actually add more non-zero integrals to the stiffness matrix and the load vector. More details about these two boundary conditions will be discussed in Math 401 (Finite Elements). In the computation part of this project, we only need to deal with the Dirichlet boundary conditions.

Basically, the Dirichlet boundary conditions tell us the solutions at $x_1 = a$ and $x_{N+1} = b$. Since the coefficient u_j in the finite element solution $u_h = \sum_{j=1}^{N+1} u_j \phi_j$ is actually the numerical solution at the node x_j $(j = 1, \cdots, N + 1)$, we actually know that $u_1 = u(a) = g_a$ and $u_{N+1} = u(b) = g_b$. Therefore, we don't really need the first and last equations in the linear system since they are set up for u_1 and u_{N+1} by using ϕ_1 and ϕ_{N+1}.

One way to impose the Dirichlet boundary condition is to replace the first and last equations in the linear system by the following two equations

$$u_1 = g_a \Rightarrow 1 \cdot u_1 + 0 \cdot u_2 + \cdots + 0 \cdot u_{N+1} = g_a,$$
$$u_{N+1} = g_b \Rightarrow 0 \cdot u_1 + \cdots + 0 \cdot u_{N-1} + 1 \cdot u_{N+1} = g_b.$$

That is, the first and last rows of the matrix A should become

$$(1, 0, \cdots, 0)$$

and

$$(0, \cdots, 0, 1)$$

respectively. And the first and last elements of the vector $\vec{b}$ should become g_a and g_b respectively. In this way, u_1 and u_{N+1} are still treated as two unknowns in the linear system. But their values are actually given in the above two equations of the system. Here is the corresponding algorithm.

- Deal with the Drichlet boundary conditions:

  $A(1,:) = 0;$
  $A(1,1) = 1;$
  $A(N+1,:) = 0;$
  $A(N+1, N+1) = 1;$
  $\vec{b}(1) = g_a;$
  $\vec{b}(N+1) = g_b;$

Remark: Another way to deal with the Drichlet boundary conditions is to remove the equations corresponding to the Dirichlet boundary nodes from the system and then replace the unknowns corresponding to the Dirichlet boundary nodes by their Drichlet boundary values. More details about different ways to handle different boundary conditions will be discussed in Math 401 (Finite Elements).

1.7 Solver for a linear system

In this section, we will discuss the direct and iterative methods we learn in Chapter 3 to solve the linear system $A\vec{X} = \vec{b}$ arising from the above finite element discretiation.

For both the direct methods and iterative methods for solving the linear system, we need to make use of many good properties of different types of matrices to construct more efficient algorithms. In the following, let's practice on some fundamental properties of matrices.

Problem #5: Show that the inverse of a nonsingular lower triangular matrix is lower triangular.

When the matrix is huge, the iterative methods are usually more suitable and efficient. The following problems discuss the iterative methods we learn in class.

Problem #6: Prove that if matrix

$$A = \begin{pmatrix} \kappa & \lambda \\ \lambda & \mu \end{pmatrix}.$$

is positive definite ($\vec{x}A\vec{x} > 0$ for any $\vec{x} \neq 0$), then the Jacobi method converges for the linear system $A\vec{x} = \vec{b}$.

One advantage of the finite element methods is that the linear systems are symmetric positive definite if the original equations are symmetric positive definite. This property is very good and important for many fast matrix solvers. One famous efficient solver for symmetric positive definite matrix is the preconditioned conjugate gradient (PCG) method, which is not introduced in our class.

As we can see in section 1.5, the matrix A obtained from the finite element method for the 1D elliptic equation is actually a tri-diagonal matrix, which is discussed in section 3.3.3 in the textbook.

Problem #7: Complete at least one of the following two problems. (The highest grade of these problems will be used for your final grade)

(a) Read about the literature to learn the PCG method. Write down its algorithm in the pseudo code style we use for the algorithms in our lecture slides.

(b) Read about the textbook and the literature to learn the LU decomposition method for tri-diagonal matrices. Write down its algorithm in the pseudo code style we use for the algorithms in our lecture slides.

2 Computation for the finite element methods

In this section, we will assemble the the components introduced above together to form a package for the finite element method to solve the 1D elliptic equation. Basically the algorithm can be described as follows.

1. Input a, b, and N. Compute $h = \frac{b-a}{N}$ and $x_j = a + (j-1)h$ $(j = 1, \cdots, N+1)$.
2. Initialize the matrix $A = sparse(N+1, N+1)$ and the vector $\vec{b} = zeros(N+1, 1)$;
3. Assemble the stiffness matrix A by using the given function $c(x)$:
   $FOR\ j = 1, \cdots, N+1$:
   $IF\ j \le N, THEN$
   Compute $A(j+1, j) = \frac{1}{h^2}\int_{x_j}^{x_{j+1}} c(x)\ dx$ by numerical integration;
   END
   $IF\ j \ge 2, THEN$
   Compute $A(j-1, j) = \frac{1}{h^2}\int_{x_{j-1}}^{x_j} c(x)\ dx$ by numerical integration;
   END
   $IF\ 2 \le j \le N, THEN$
   Compute $A(j, j) = -\frac{1}{h^2}\int_{x_{j-1}}^{x_j} c(x)\ dx - \frac{1}{h^2}\int_{x_j}^{x_{j+1}} c(x)\ dx$ by numerical integration;
   END
   Compute $A(1, 1) = -\frac{1}{h^2}\int_{x_1}^{x_2} c(x)\ dx$ by numerical integration;
   Compute $A(N+1, N+1) = -\frac{1}{h^2}\int_{x_N}^{x_{N+1}} c(x)\ dx$ by numerical integration;
   END
4. Assemble the load vector $\vec{b}$ by using the given function $f(x)$:
   $FOR\ i = 2, \cdots, N$:
   Compute $\vec{b}(i) = \int_{x_{i-1}}^{x_i} f(x)\frac{x - x_{i-1}}{h}\ dx + \int_{x_i}^{x_{i+1}} f(x)\frac{x_{i+1} - x}{h}\ dx$ by numerical integration;
   END
   Compute $\vec{b}(1) = \int_{x_1}^{x_2} f(x)\frac{x_2 - x}{h}\ dx$ by numerical integration;
   Compute $\vec{b}(N+1) = \int_{x_N}^{x_{N+1}} f(x)\frac{x - x_N}{h}\ dx$ by numerical integration;
5. Deal with the Drichlet boundary conditions:
   $A(1, :) = 0$;
   $A(1, 1) = 1$;
   $A(N+1, :) = 0$;
   $A(N+1, N+1) = 1$;
   $\vec{b}(1) = g_a$;
   $\vec{b}(N+1) = g_b$;
6. Solve $A\vec{X} = \vec{b}$ for $\vec{X}$ by using a direct or iterative method.

Problem #8: Use the finite element method to solve the following equation:

$$\frac{d}{dx}\left(e^{-sin(x)}\frac{du(x)}{dx}\right) = e^{-sin(x)}\left[-cos(x) + sin(x)cos(x) - \frac{1}{4}x^{-\frac{3}{2}} - \frac{1}{2}x^{-\frac{1}{2}}cos(x)\right] \ (1 \le x \le 4),$$
$$u(1) = cos(1) + 1, u(4) = cos(4) + 2.$$

(a) Program for the finite element method. Use "\" as the matrix solver in Matlab and use 4-point Gauss quadrature to compute the integrals.

(b) Use your code to solve the equation with $h = 1/4, 1/8, 1/16, 1/32, 1/64, 1/128$. Plot the numerical solutions by using the numerical solutions at all mesh nodes. List the numerical solution at $x = 2$ and $x = 3$ in the following table.

h	numerical solutions at $x = 2$	numerical solutions at $x = 3$
1/4		
1/8		
1/16		
1/32		
1/64		
1/128		

Table 1: The numerical solutions at $x = 2$ and $x = 3$.

Problem #9: Consider the same problem as above. Replace the "\" by Gauss elimination, Jocobi, and GS methods as the matrix solver in your Matlab code. Plot the numerical solutions at all mesh nodes. For the iterative methods, set all the tolerances to be 10^{-5}, maximum number of iteration steps to be 200 and the initial guess of the solution to be a vector whose element are all 1.

The theoretical analysis of the finite element method tells us that the linear finite element solution has second order convergence in the infinity norm. That is,

$$\|u_h - u\|_\infty \le Ch^2.$$

In the following, we will numerically verify this by using the above numerical results. The infinity norm can be approximated by the maximum absolute value of the errors at all nodes.

Problem #10: The analytic solution of problem #8 and #9 is $u = cos(x) + \sqrt{x}$. In the following table with $h = 1/4, 1/8, 1/16, 1/32, 1/64, 1/128$, list the maximum absolute value of the errors at all nodes for the numerical solutions you obtain in problem #9 and #10. What do you observe? Explain your observation in term of the above conclusion of the second order convergence.

h	errors with "\"	errors with Gauss elimination	errors with Jacobi	errors with GS
1/4				
1/8				
1/16				
1/32				
1/64				
1/128				

Table 2: The maximum numerical errors at all mesh nodes.

3 Conclusions

This project introduces the finite element method for solving the 1D elliptic equation and discusses how piecewise polynomial approximation, numerical integration and solver for linear system are needed in the finite element

method. The 2D and 3D finite element methods can be similarly derived. Actually, problems 2 and 4 are for the extension to 2D and 3D cases.

Finite element methods have been proved to be efficient numerical methods for partial differential equations in many different engineering areas. The advantages of finite element methods include, but not limited to

- The framework of finite element methods are universal for all partial differential equations.
- If the original equations are symmetric positive definite, the linear systems arising from finite element methods are also symmetric positive definite, which is important for many fast matrix solvers.
- It is natural for finite element methods to deal with problem domains with curved boundary, interface or singularities once the mesh is properly constructed.
- It is natural for many finite element methods to keep the conservation law.
- The finite element methods provide piecewise functions defined on the whole problem domain as numerical solutions, not just the numerical solutions at mesh nodes.
- The finite element methods have mature frameworks for mathematical analysis.

The framework of finite element methods are universal for all partial differential equations due to the following reasons:

- The weak formulations of all partial differential equations consist of integrals in similar formats. This unifies different equations into a universal formation.
- Due to the "local assembly" idea of the general implementation framework of finite elements, all the processes in finite element methods are completely based on all the elements (handled in the element loop one by one) and the mesh information matrices P, T and E, including the construction of finite element spaces, the finite element discretization, the assembly of the matrices and vectors, and the treatment of the boundary conditions. This makes the finite element methods universal for different problem domains. As long as the matrices P, T and E are formed, the rest of the implementation is the same.
- The treatment of the boundary conditions is universal since it is also based on the weak formulation.
- Each analysis framework for finite element methods can be applied to a wide range of problems. For example, the energy method is suitable for many time-dependent problems and the Strong's second lemma is suitable for all of non-conforming finite elements.

The universal framework of finite element methods leads to similar implementations of finite element methods to different equations. Therefore, when an existing package needs to be modified for a new partial differential equation, only a small portion of the package needs to be changed.

On the other hand, this universal framework is not straightforward to understand, which is the main "disadvantage" of finite element methods. It takes a long time to understand all the components of finite element methods and assemble them together properly. However, once you completely understand and successfully implement finite element methods, the universal framework will be a convenient and powerful tool for your future use. We have learned several components from this course, including piecewise polynomial approximation, numerical integration, and matrix computation. We have also practiced on finite element methods for the 1D elliptic equation in this project. In Math 5602 (Mathematical Foundation for Finite Element Methods), we will discuss more details on all the components of finite element methods as a whole dynamic system for different types of partial differential equations.