MA587: Numerical Solutions of PDEs – Finite Element Methods

1 Introduction

1.1 The Problems

We want to solve ordinary/partial differential equations (ODE/PDE), especially linear second or fourth order ODE/PDEs. The linear differential equations can be classified as *elliptic*, *parabolic*, and *hyperbolic* equations. Below are some examples:

1. Initial value problems:

$$\begin{aligned} &u''(t) + a(t)u'(t) + b(t)u(t) = f(t), \\ &u(0) = u_0, \quad \boxed{u'(0) = v_0}. \end{aligned} \tag{1}$$

2. Boundary value problems: One dimensional example,

$$\begin{aligned} &u''(x) + a(x)u'(x) + b(x)u(x) = f(x), \\ &u(0) = u_0, \quad \boxed{u(1) = u_1}. \end{aligned} \tag{2}$$

   Two dimensional example:

$$\begin{aligned} &-(u_{xx} + u_{yy}) = f(x,y), \quad (x,y) \in \Omega \\ &u(x,y) = u_0(x,y), \quad (x,y) \in \partial\Omega. \end{aligned} \tag{3}$$

3. Boundary and initial value problems

$$\begin{aligned} &u_t = a u_{xx} + f(x,t) \\ &u(0,t) = g_1(t), \quad u(1,t) = g_2(t), \qquad \text{BC} \\ &u(x,0) = u_0(x), \qquad \text{IC} \end{aligned} \tag{4}$$

4. Eigenvalue problem:

$$\begin{aligned} &-u''(x) = \lambda u(x) \\ &u(0) = 0, \quad u(1) = 0. \end{aligned} \tag{5}$$

   Both $u(x)$ and scalar λ are unknowns.

The finite element method was created to solve the complicated equations of elasticity and structural mechanics, usually modeled by elliptic type equations, with complicated geometries. It has been developed for other applications as well.

In this course, we will focus on *elliptic type ODE/PDE boundary value problems* and often use the model equations

$$\begin{aligned} &\text{1D:} \quad -u''(x) = f(x), \quad u(0) = u_0, \quad u(1) = u_1, \\ &\text{2D:} \quad -(u_{xx} + u_{yy}) = f(x,y), \quad (x,y) \in \Omega, \quad u(x,y) = u_0(x,y), \quad (x,y) \in \partial\Omega. \end{aligned}$$

Usually the solutions of differential equations can not be expressed in terms of elementary functions (polynomials, $\sin(x)$, $\cos(x)$, $\log(x)$, a^x etc.) and hard to find. We need to look for *approximate* solutions.

- Semi-analytic methods. Sometime we can approximate the solution using series, integral equations, interpolation, perturbation techniques, asymptotic approximations etc.
- Numerical approximate solutions: We use computers to obtain approximate numbers for the solutions.

In this course, we will mainly use the second approach.

1.2 Comparison of the finite difference and the finite element methods

Consider the model problem

$$-u''(x) = f(x), \quad 0 \le x \le 1, \quad u(0) = 0, \quad u(1) = 0.$$

We will solve this problem using the finite difference method and the finite element method to see their differences.

The finite difference method

1. Generate a grid. For example, we can use a uniform Cartesian grid
$$x_i = i\,h, \quad i = 0, 1, \cdots n, \quad h = \frac{1}{n}.$$
2. Substitute the derivatives with some *finite difference* formulas to get one equation at each grid point. Notice that for a twice differential function $\phi(x)$
$$\phi''(x) = \lim_{\Delta x \to 0} \frac{\phi(x - \Delta x) - 2\phi(x) + \phi(x + \Delta x)}{\Delta x^2}$$

At each grid point x_i we approximate the differential equation by

$$-\frac{u(x_i-h)-2u(x_i)+u(x_i+h)}{h^2}=f(x_i)+error.$$

We define the approximate solution at x_i as U_i, they are the solution of the following system of equations

$$\begin{aligned}
-\frac{0-2U_1+U_2}{h^2} &= f(x_1)\\
-\frac{U_1-2U_2+U_3}{h^2} &= f(x_2)\\
-\frac{U_2-2U_3+U_4}{h^2} &= f(x_3)\\
\cdots\cdots &= \cdots\\
-\frac{U_{i-1}-2U_i+U_{i+1}}{h^2} &= f(x_i)\\
\cdots\cdots &= \cdots\\
-\frac{U_{n-3}-2U_{n-2}+U_{n-1}}{h^2} &= f(x_{n-2})\\
-\frac{U_{n-2}-2U_{n-1}+0}{h^2} &= f(x_{n-1}).
\end{aligned}$$

This system of equations can be written as the matrix and vector form:

$$\begin{bmatrix}
\frac{2}{h^2} & -\frac{1}{h^2} & & & & \\
-\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2} & & & \\
 & -\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2} & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & -\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2}\\
 & & & & -\frac{1}{h^2} & \frac{2}{h^2}
\end{bmatrix}
\begin{bmatrix} U_1\\ U_2\\ U_3\\ \vdots\\ U_{n-2}\\ U_{n-1}\end{bmatrix}
=
\begin{bmatrix} f(x_1)\\ f(x_2)\\ f(x_3)\\ \vdots\\ f(x_{n-2})\\ f(x_{n-1})\end{bmatrix}$$

3. Solve the system of equations to get the approximate solution at each grid point.

4. Error analysis (consistency + stability $\Longrightarrow$ convergence). It is pointwise convergence, that is $\lim_{h\to 0}\|u(x_i)-U_i\|_\infty=0$. The finite difference method needs the solution to have *second order derivative.*

1.3 The finite element method:

1. Derive a weak or variational formulation in *integral form.* This can be done by multiplying a testing function $v(x)$, $v(0)=0$, $v(1)=0$ to both sides of the differential equation:

$$-u''v=fv.$$

Integrate from 0 to 1, and use integration by parts to get

$$\begin{aligned}\int_0^1 (-u''v)dx &= \int_0^1 fvdx \\ -u'v\Big|_0^1 + \int_0^1 u'v'dx &= \int_0^1 fvdx \\ \int_0^1 u'v'dx &= \int_0^1 fvdx. \quad \text{This is the weak form!.}\end{aligned}$$

2. Generate a triangulation (interval). For example, we can use a uniform Cartesian grid $x_i = i\,h$, $i = 0, 1, \cdots, n$, $h = 1/n$, to get the intervals $[x_{i-1}, x_i]$, $i = 1, 2, \cdots, n$.

3. Construct a set of basis functions based on the triangulation. We can use piecewise linear functions. Define:

$$\phi_1(x) = \begin{cases} \dfrac{x}{h} & \text{if } 0 \le x \le x_1 \\ \dfrac{x_2 - x}{h} & \text{if } x_1 \le x \le x_2 \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h} & \text{if } x_{i-1} \le x \le x_i \\ \dfrac{x_{i+1} - x}{h} & \text{if } x_i \le x \le x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

They are called hat functions.

4. The approximate solution then is the linear combination of the basis functions:

$$u_h(x) = \sum_{i=1}^{n-1} c_j \phi_j(x),$$

where the coefficients c_j are *unknowns*. Notice that $u_h(x)$ is a *piecewise linear* function, and usually is not the exact solution. We need to derive a linear system of equations from the weak form $\int_0^1 u'v'dx = \int_0^1 fvdx$ with the exact solution being substituted by the approximate solution $u_h(x)$:

$$\begin{aligned}\int_0^1 u_h'v'dx &= \int_0^1 fvdx, \quad \text{The error comes in!} \\ \int_0^1 \sum_{j=1}^{n-1} c_j\phi_j'v'dx &= \int_0^1 fvdx \\ \sum_{j=1}^{n-1} c_j \int_0^1 \phi_j'v'dx &= \int_0^1 fvdx\end{aligned}$$

Now we choose $v(x)$ as ϕ_1, ϕ_2, $\cdots$, ϕ_{n-1} respectively to get:

$$\begin{array}{rcl}
\left(\int_0^1 \phi_1'\phi_1' dx\right) c_1 + \left(\int_0^1 \phi_1'\phi_2' dx\right) c_2 + \cdots + \left(\int_0^1 \phi_1'\phi_{n-1}' dx\right) c_{n-1} & = & \int_0^1 f\phi_1 dx \\
\left(\int_0^1 \phi_2'\phi_1' dx\right) c_1 + \left(\int_0^1 \phi_2'\phi_2' dx\right) c_2 + \cdots + \left(\int_0^1 \phi_2'\phi_{n-1}' dx\right) c_{n-1} & = & \int_0^1 f\phi_2 dx \\
\cdots \quad \cdots \quad \cdots\cdots \quad \cdots \quad \cdots \quad \cdots\cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots & & \cdots \\
\left(\int_0^1 \phi_i'\phi_1' dx\right) c_1 + \left(\int_0^1 \phi_i'\phi_2' dx\right) c_2 + \cdots + \left(\int_0^1 \phi_i'\phi_{n-1}' dx\right) c_{n-1} & = & \int_0^1 f\phi_i dx \\
\cdots \quad \cdots \quad \cdots\cdots \quad \cdots \quad \cdots \quad \cdots\cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots & & \cdots \\
\left(\int_0^1 \phi_{n-1}'\phi_1' dx\right) c_1 + \left(\int_0^1 \phi_{n-1}'\phi_2' dx\right) c_2 + \cdots + \left(\int_0^1 \phi_{n-1}'\phi_{n-1}' dx\right) c_{n-1} & = & \int_0^1 f\phi_{n-1} dx.
\end{array}$$

In the matrix-vector form, $AU = F$, it is:

$$\begin{bmatrix} a(\phi_1,\phi_1) & a(\phi_1,\phi_2) & \cdots & a(\phi_1,\phi_{n-1}) \\ a(\phi_2,\phi_1) & a(\phi_2,\phi_2) & \cdots & a(\phi_2,\phi_{n-1}) \\ \vdots & \vdots & \vdots & \vdots \\ a(\phi_{n-1},\phi_1) & a(\phi_{n-1},\phi_2) & \cdots & a(\phi_{n-1},\phi_{n-1}) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} (f,\phi_1) \\ (f,\phi_2) \\ \vdots \\ (f,\phi_{n-1}) \end{bmatrix},$$

where

$$a(\phi_i,\phi_j) = \int_0^1 \phi_i'\phi_j' dx, \qquad (f,\phi_i) = \int_0^1 f\phi_i dx$$

If ϕ_i are the hat functions, then we have

$$\begin{bmatrix} \frac{2}{h} & -\frac{1}{h} & & & & \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & \\ & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & \\ & & \ddots & \ddots & \ddots & \\ & & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} \\ & & & & -\frac{1}{h} & \frac{2}{h} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} \int_0^1 f\phi_1 dx \\ \int_0^1 f\phi_2 dx \\ \int_0^1 f\phi_3 dx \\ \vdots \\ \int_0^1 f\phi_{n-2} dx \\ \int_0^1 f\phi_{n-1} dx \end{bmatrix}$$

5. Solve the linear system of equations $AU = F$.

6. Error analysis.

Questions we would like to ask:

- Why/how can we change PDE/ODE to a weak form?
- How do we choose the basis functions ϕ and their boundary conditions?
- How to implement the finite element method?
- How to solve the linear system of equations?
- How do we carry out the error analysis?

1.4 Advantages and disadvantages of finite difference and finite element methods

Finite difference methods:

- Simple to use and are easy to understand.
- Easy to implement for rectangular regions in two and three dimensions.
- It is pointwise.
- Many fast solvers and packages, e.g., FFT, for regular domains (rectangular and circular regions).
- Difficult for complicated geometries.
- Have strong regularity requirements (derivatives) for the solutions.

Finite element methods:

- Very successful for structural (elliptic type) problems.
- Natural approach for problems with complicated boundaries.
- Solid theoretical foundations at least for elliptic type problems.
- *Weaker requirement* for the solutions. For example, we do not need u'' for the 1D model problem since only the integral forms are used.
- Many commercial packages.
- Usually coupled with multigrid solvers.

- Need expert knowledge to generate the triangulation which is used to be the most difficult part. Now we can use many packages, for example, Matlab, Triangle, Pltmg, Fidap, Ansys etc.

Finite Difference methods, finite element methods, finite volume methods, spectral methods are all useful for solving PDEs/ODEs.

2 Finite element methods for one dimensional elliptic equations

The model is

$$\begin{aligned} &-u''(x) = f(x), \quad 0 < x < 1, \\ &u(0) = 0, \quad u(1) = 1. \end{aligned} \tag{6}$$

The problem can be reformulated into different forms.

1. (D)-form, the original differential equation.
2. (V)-form, the variational form or weak form:

   $$\int_0^1 u'v'dx = \int_0^1 fvdx \tag{7}$$

   for any testing function $v \in H_0^1(0,1)$. The corresponding finite element method is often called the Galerkin method. *Question: What happened to the boundary condition?*
3. (M)-form, the minimization form:

   $$\min_{v(x)\in H_0^1(0,1)} \left\{ \frac{1}{2}\int_0^1 \left((v')^2 - fv \right) dx \right\}. \tag{8}$$

   The corresponding finite element method is often called the Ritz method. Under some assumptions, the three different forms are equivalent, that is, they have the same solution.

2.1 Physical reasoning

From the point of view mathematical modeling, the variational/weak form and the minimization form precedes the differential form. To see this, consider an elastic string with two end fixed and with an external force $f(x)$:

Let $u(x)$ be the position of the string at x which is unknown and depends on $f(x)$. The physical law states that the equilibrium is the state that minimizes the total energy. The potential energy due to the deformation is

$$\begin{aligned}
& \tau \cdot \text{increase in the length} \\
& \tau \left(\sqrt{(u(x+\Delta x) - u(x))^2 + \Delta x^2} - \Delta x \right) \\
= \; & \tau \left(\sqrt{ \left(u + u_x \Delta x + \frac{1}{2} \Delta x^2 u_{xx} + \cdots - u(x) \right)^2 + \Delta x^2 } - \Delta x \right) \\
\approx \; & \tau \left(\sqrt{\Delta x^2 \left(1 + u_x^2\right)} - \Delta x \right) \\
\approx \; & \tau \frac{\Delta x}{2} u_x^2,
\end{aligned}$$

where τ is the coefficient of the surface tension that we assume it is a constant. The work done due to the external force is $-f(x)u(x)$. Therefore the total energy is

$$F(u) = \int_0^1 \frac{1}{2} \tau u_x^2 dx - \int_0^1 f(x)u(x)dx.$$

The equilibrium state $u^*(x)$ must minimizes the total energy:

$$F(u^*) \le F(u)$$

for any $u \in H_0^1$. Note that u^* is the minimizer of the functional $F(u)$ (function of functions).

If we consider the balance of the force, we will get the differential equation. The forces are the external force $f(x)$ which is balanced by the tension of the elastic string. From the Hooke's law we know the tension is

$$T = \tau u_x.$$

Therefore, we have

$$\begin{aligned}
\tau \left(u_x(x+\Delta x) - u(x) \right) &= -f(x) \Delta x \\
\text{or} \quad \tau \frac{u_x(x+\Delta x) - u(x)}{\Delta x} &= -f(x) \\
\text{Let } \Delta x \to 0 \text{ to get} \quad - \tau u_{xx} &= f(x).
\end{aligned}$$

Therefore from the physical reasoning, we know that that the differential equation is equivalent to the minimization problem. Using the principal of virtual work, we also can conclude that:

$$\int_0^1 u'v'dx = \int_0^1 fvdx$$

for any function $v(x) \in H_0^1$.

2.2 Mathematical equivalence

We have proved (D) $\Longrightarrow$ (V). We are going to prove (V) $\Longrightarrow$ (V) and $\Longleftrightarrow$ (M).

Theorem 2.1 *If u_{xx} exists and continuous, then from*

$$\int_0^1 u'v'dx = \int_0^1 fvdx, \quad \forall\, v(0)=v(1)=0, \quad v \in H^1(0,1)$$

we can conclude $\; -u_{xx} = f(x)$.

Proof: Using integration by parts

$$\begin{aligned} \int_0^1 u'v'dx &= u'v\big|_0^1 - \int_0^1 u''vdx \\ \Longrightarrow \quad -\int_0^1 u''vdx &= \int_0^1 fvdx \\ \text{or} \quad \int_0^1 \big(u''+f\big)\,vdx &= 0. \end{aligned}$$

Since $v(x)$ is arbitrary and continuous, and u'' and f are continuous, we must have

$$u''+f=0, \qquad i.e. \quad -u''=f.$$

Now we prove $V \Longrightarrow$ M. Suppose u^* satisfies

$$\int_0^1 {u^*}'v'dx = \int_0^1 vfdx$$

for any $v(0)=v(1)=0$, for any $v(x) \in H_0^1$, we need to prove that

$$\begin{aligned} F(u^*) &\le F(u) \quad \text{or} \\ \frac{1}{2}\int_0^1 (u^*)_x^2dx - \int_0^1 fu^*dx &\le \frac{1}{2}\int_0^1 u_x^2dx - \int_0^1 fudx \end{aligned}$$

Proof:

$$\begin{aligned}
F(u) &= F(u^*+u-u^*) = F(u^*+w), \quad w = u-u^* \quad w(0)=w(1)=0 \\
&= \int_0^1 \left[\frac{1}{2}(u^*+w)_x^2 - (u^*+w)f \right] dx \\
&= \int_0^1 \left[\frac{(u^*)_x^2 + w_x^2 + 2(u^*)_x w_x}{2} - u^* f - wf \right] dx \\
&= \int_0^1 \left(\frac{1}{2}(u^*)_x^2 - u^* f \right) dx + \int_0^1 \frac{1}{2} w_x^2 dx + \int_0^1 \left((u^*)_x w_x - fw \right) dx \\
&= \int_0^1 \left(\frac{1}{2}(u^*)_x^2 - u^* f \right) dx + \int_0^1 \frac{1}{2} w_x^2 dx + 0 \\
&= F(u^*) + \int_0^1 \frac{1}{2} w_x^2 dx \\
&> F(u^*).
\end{aligned}$$

The proof is completed.

Finally we prove that (M) $\Longrightarrow$ (V). Assume u^* is the minimizer of the $F(u^*)$, we want to prove that

$$\int_0^1 (u^*)_x v_x dx = \int_0^1 f v dx$$

for any $v(0) = v(1) = 0$ and $v \in H^1(0,1)$.

Proof: Consider the auxiliary function:

$$g(\epsilon) = F(u^* + \epsilon v).$$

Since $F(u^*) \le F(u^* + \epsilon v)$ for any ϵ, $g(0)$ is a global/local minimum and therefore $g'(0) = 0$.

$$\begin{aligned}
g(\epsilon) &= \int_0^1 \left\{ \frac{1}{2}(u^*+\epsilon v)_x^2 - (u^*+\epsilon v)f \right\} dx \\
&= \int_0^1 \left\{ \frac{1}{2}\left((u^*)_x^2 + 2(u^*)_x v_x \epsilon + v_x^2 \epsilon^2 \right) - u^* f - \epsilon v f \right\} dx \\
&= \int_0^1 \left(\frac{1}{2}(u^*)_x^2 - u^* f \right) dx + \epsilon \int_0^1 \left((u^*)_x v_x - fv \right) dx + \frac{\epsilon^2}{2} \int_0^1 v_x^2 dx.
\end{aligned}$$

Thus

$$g'(\epsilon) = \int_0^1 \left((u^*)_x v_x - fv \right) dx + \epsilon \int_0^1 v_x^2 dx$$

and

$$g'(0) = \int_0^1 \left((u^*)_x v_x - fv \right) dx = 0$$

That is, the weak form is satisfied.

The three different forms may not be equivalent for some problems depending on the regularity of the solutions. The relations are

$$(\mathrm{D}) \Longrightarrow (\mathrm{M}) \Longrightarrow (\mathrm{V})$$

From (V) to conclude (M), we usually need the differential equations to be self-adjoint. From (M) or (V) to conclude (D), we need the solution of the differential equations to have continuous second order derivatives.

3 Finite Element Method for the 1D model problem: Method and Programming

For the model problem

$$-u''(x) = f(x), \quad 0 < x < 1, \tag{9}$$

$$u(0) = u(1) = 0. \tag{10}$$

We will discuss

- Ritz method for the minimization form.
- Galerkin method for the weak/variational form.
- Programming: assembling element by element.

3.1 Procedure

Different forms:

- Weak/variational form: $\displaystyle\int_0^1 u'v'dx = \int_0^1 fvdx$ for any $v(x)$ that has continuous first derivative and $v(0) = v(1) = 0$.
- Minimization form, find u such that

$$F(u) = \min_{v \in H_0^1(0,1)} \left\{ \frac{1}{2} \int_0^1 v_x^2 dx - \int_0^1 fvdx \right\}.$$

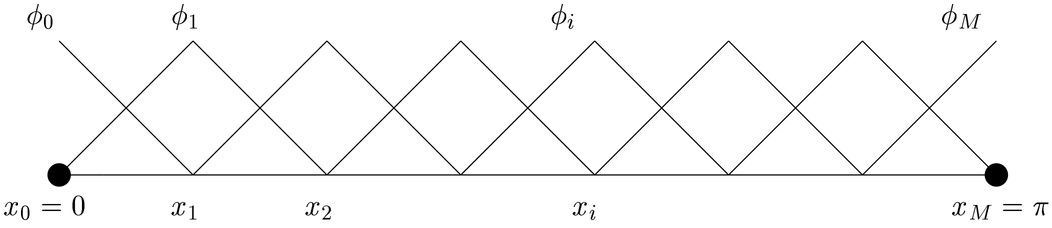

Figure 1: Diagram of a triangulation in one dimension

Triangulation:

Given a triangulation, $x_0 = 0$, $x_1, x_2, \cdots x_M = 1$. Let $h_i = x_{i+1} - x_i$, $i = 0, 1, \cdots, M-1$.

- x_i is called a *node*.
- $[x_i, x_{i+1}]$ is called an element.
- $h = \max\limits_{0 \le i \le M-1} \{ h_i \}$ measures how fine the partition is.

Define a finite dimensional space *over* the triangulation:

$$\begin{aligned} &V_h \text{ (a finite dimensional space)} \quad \subset V \text{ (the solution space,)} \\ &\text{The discrete problem} \quad \subset \text{(the continuous problem).} \end{aligned}$$

Different *finite* dimensional spaces will generate different finite element methods. Since V_h has finite dimension, we can find one set of basis functions

$$\phi_1, \quad \phi_2, \quad \cdots, \quad \phi_{M-1} \subset V_h,$$

ϕ_1, ϕ_2, $\cdots$, ϕ_{M-1} have to be *linear independent*, that is, if

$$\sum_{j=1}^{M-1} \alpha_j \phi_j = 0,$$

then $\alpha_1 = \alpha_0 = \cdots = \alpha_{M-1} = 0$; V_h is the space *spanned* by the basis functions:

$$V_h = \left\{ v_h(x), \quad v_h(x) = \sum_{j=1}^{M-1} \alpha_j \phi_j \right\}.$$

Example: The simplest finite dimensional space is the *piecewise continuous linear* function space defined over the *triangulation.*

$$V_h = \{ \, v(x), \quad v_h(x) \text{ is piecewise continuous linear over the triangulation} \quad v_h(0) = v_h(1) = 0 \} \, .$$

We want to know whether V_h has a finite or infinite dimensions. Note that there are infinite number of elements in V_h.

Find the dimension of V_h.

A linear function $l(x)$ in an interval $[x_i, x_{i+1}]$ is determined by its values at x_i and x_{i+1}

$$l(x) = l(x_i)\frac{x - x_{i+1}}{x_i - x_{i+1}} + l(x_{i+1})\frac{x - x_i}{x_{i+1} - x_i}$$

There are $M-1$ such nodal vales, $l(x_i)$s, $l(x_1)$, $l(x_2)$, $\cdots$, $l(x_{M-1})$ for a piecewise continuous linear function over the triangulation plus $l(x_0) = l(x_M) = 0$. Given a vector $[l(x_1), l(x_2), \cdots, l(x_{M-1})]^T \in R^{M-1}$, we can construct a $v_h(x) \in V_h$ by taking $v_h(x_i) = l(x_i)$. On the other hand, given a $v_h(x) \in V_h$, we get a vector $[v(x_1), v(x_2), \cdots, v(x_{M-1})]^T \in R^{M-1}$. Therefore there is *one to one* relation between V_h and R^{M-1}, so V_h has a finite dimension $M-1$.

Find a set of basis functions.

The finite dimensional space can be spanned by a set of basis functions. There are infinite number of sets of basis functions. We should choose a set of basis functions that

- are simple;
- have minimum support (zero almost everywhere);
- meet the regularity requirement (i.e. how smooth they should be).

The simplest ones are the hat functions:

$$\begin{aligned}
&\phi_1(x_1) = 1, \quad \phi_1(x_j) = 0, \quad j = 0, 2, 3, \cdots, M, \\
&\phi_2(x_2) = 1, \quad \phi_2(x_j) = 0, \quad j = 0, 1, 3, \cdots, M, \\
&\cdots\cdots\cdots\cdots \\
&\phi_i(x_i) = 1, \quad \phi_i(x_j) = 0, \quad j = 0, 1, \cdots i-1, i+1, \cdots, M, \\
&\cdots\cdots\cdots\cdots \\
&\phi_{M-1}(x_{M-1}) = 1, \quad \phi_{M-1}(x_j) = 0, \quad j = 0, 1, \cdots, M.
\end{aligned}$$

They can be written in the simple form

$$\phi_i(x_j) = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

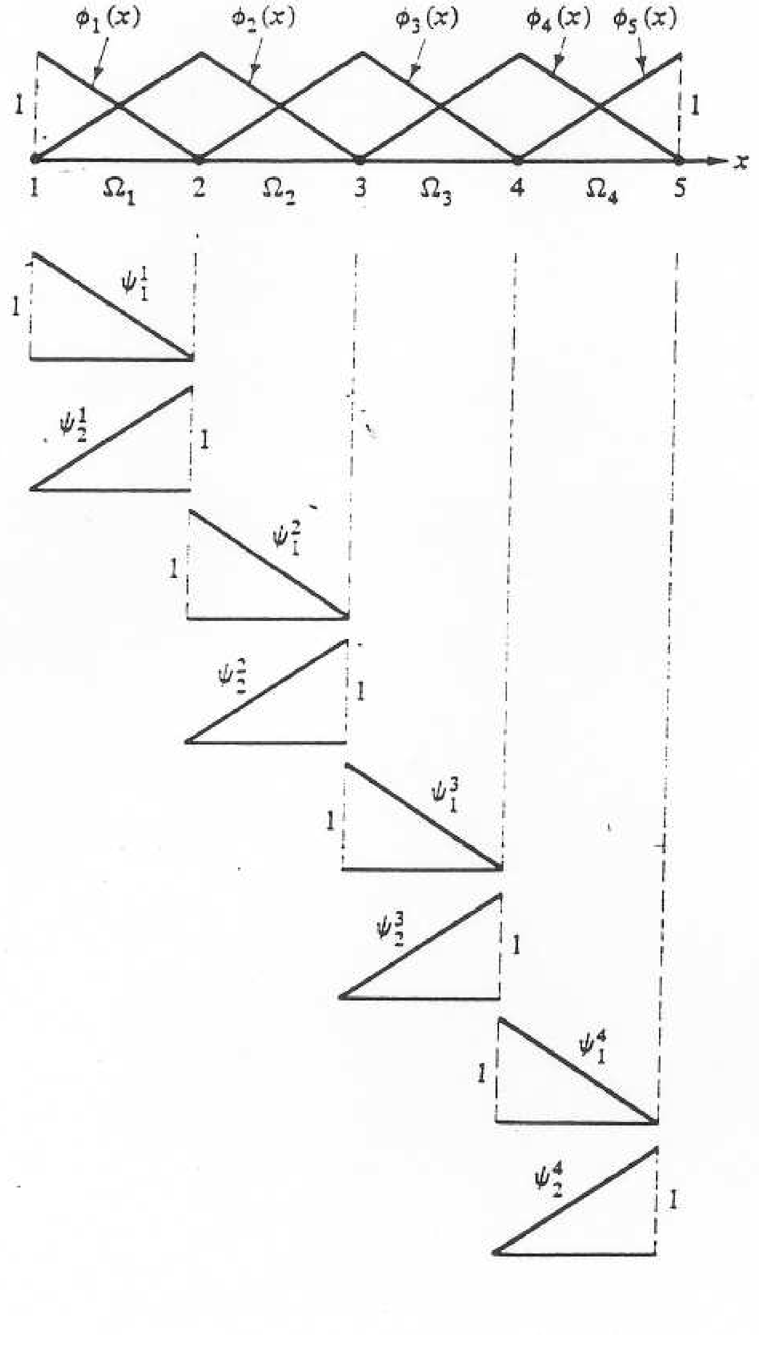

Figure 2: Piecewise linear basis functions ϕ_i for a 4-element mesh generated by linear shape functions ψ_1^e, ψ_2^e defined over each element.

The analytic form is

$$\phi_i(x) = \begin{cases} 0, & \text{if } x < x_{i-1}, \\ \dfrac{x - x_{i-1}}{h_i}, & \text{if } x_{i-1} \le x < x_i, \\ \dfrac{x_{i+1} - x}{h_{i+1}}, & \text{if } x_i \le x < x_{i+1}, \\ 0, & \text{if } x_{i+1} \le x. \end{cases} \tag{12}$$

The finite element solution that we are looking for is

$$u_h(x) = \sum_{j=1}^{M-1} \alpha_j \phi_j(x). \tag{13}$$

We can use either the minimization form (M), or weak/variational form (V), to derive a linear system of equations for the coefficients α_j. Notice that using the hat functions, we have

$$u_h(x_i) = \sum_{j=1}^{M-1} \alpha_j \phi_j(x_i) = \alpha_i \phi_i(x_i) = \alpha_i. \tag{14}$$

So α_i is an approximate solution to the exact solution at $x = x_i$.

3.2 The Ritz method (for problems that have a minimization form).

The Ritz method is the earliest form of the finite element method and is the most successful. However, not every problem has a minimization form.

The minimization form for the model problem is

$$F(v) = \frac{1}{2}\int_0^1 (v_x)^2 dx - \int_0^1 f v dx. \tag{15}$$

Therefore

$$F(v_h) = \frac{1}{2}\int_0^1 \left(\sum_{j=1}^{M-1} \alpha_j \phi_j'(x) \right)^2 - \int_0^1 f \sum_{j=1}^{M-1} \alpha_j \phi_j(x) dx. \tag{16}$$

Now $F(v_h)$ is a multi-variable function $\alpha_1, \alpha_2, \cdots, \alpha_{M-1}$

$$F(v_h) = F\left(\alpha_1, \alpha_2, \cdots, \alpha_{M-1}\right).$$

The necessary condition for a global minimum (local minimum as well) is

$$\frac{\partial F}{\partial \alpha_1} = 0, \quad \frac{\partial F}{\partial \alpha_2} = 0, \quad \cdots \quad \frac{\partial F}{\partial \alpha_i} = 0, \quad \cdots \quad \frac{\partial F}{\partial \alpha_{M-1}} = 0.$$

By taking the partial derivatives directly we have

$$\begin{aligned}\frac{\partial F}{\partial \alpha_1} &= \int_0^1 \left(\sum_{j=1}^{M-1} \alpha_j \phi_j' \right) \phi_1' dx - \int_0^1 f\phi_1 dx = 0 \\ \frac{\partial F}{\partial \alpha_i} &= \int_0^1 \left(\sum_{j=1}^{M-1} \alpha_j \phi_j' \right) \phi_i' dx - \int_0^1 f\phi_i dx = 0, \quad i = 1, 2, \cdots, M-1.\end{aligned}$$

Exchange the order of integration and the summation to get

$$\sum_{j=1}^{M-1} \left(\int_0^1 \phi_j' \phi_i' dx \right) \alpha_j = \int_0^1 f\phi_i dx, \quad i = 1, 2, \cdots M-1.$$

This is exact the same as we would get using the Galerkin method with the weak form,

$$\begin{aligned}\int_0^1 u'v'dx &= \int_0^1 fvdx \\ \int_0^1 \left(\sum_{j=1}^{M-1} \alpha_j \phi_j' \right) \phi_i' dx &= \int_0^1 f\phi_i dx, \quad i = 1, 2, \cdots M-1.\end{aligned}$$

Comparison of Ritz and Galerkin methods:

For many problems, the two methods are equivalent theoretically.

- Optimization techniques can be used in Ritz method.
- Weak form usually has weaker requirements for the differential equation and the solution. Not every problem has a minimization form, but almost all problems have some kind of weak forms. Whether the weak form is a good choice and whether the finite element method converges or not is an important issue.

4 FEM programming for 1D problem

Given a problem, say the model problem, after we have derived the minimization/weak form; constructed a triangulation and a set of basis functions, we need to form the coefficient matrix A, often called *stiffness matrix*, and the right hand side F, often called the *load vector*. How to form A and F is a very crucial part in the finite element method. For the model problem, we form A and F by *assembling element by element.* The elements are

$$\begin{array}{ccccccc} [x_0, x_1], & [x_1, x_2], & \cdots & [x_{i-1}, x_i] & \cdots & [x_{M-1}, x_M], \\ \Omega_1, & \Omega_2, & \cdots & \Omega_i, & \cdots & \Omega_M. \end{array}$$

The idea is to break up the integration element by element. For any function $g(x)$, the integration

$$\int_0^1 g(x)dx = \sum_{k=1}^{M} \int_{x_{k-1}}^{x_k} g(x)dx = \sum_{k=1}^{M} \int_{\Omega_k} g(x)dx.$$

The stiffness matrix then can be written as

$$\begin{aligned}
A &= \begin{bmatrix} \int_0^1 (\phi_1')^2 dx & \int_0^1 \phi_1'\phi_2' dx & \cdots & \int_0^1 \phi_1'\phi_{M-1}' dx \\ \int_0^1 \phi_2'\phi_1' dx & \int_0^1 (\phi_2')^2 dx & \cdots & \int_0^1 \phi_2'\phi_{M-1}' dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_0^1 \phi_{M-1}'\phi_1' dx & \int_0^1 \phi_{M-1}'\phi_2' dx & \cdots & \int_0^1 (\phi_{M-1}')^2 dx \end{bmatrix} \\
&= \begin{bmatrix} \int_{x_0}^{x_1} (\phi_1')^2 dx & \int_{x_0}^{x_1} \phi_1'\phi_2' dx & \cdots & \int_{x_0}^{x_1} \phi_1'\phi_{M-1}' dx \\ \int_{x_0}^{x_1} \phi_2'\phi_1' dx & \int_{x_0}^{x_1} (\phi_2')^2 dx & \cdots & \int_{x_0}^{x_1} \phi_2'\phi_{M-1}' dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_{x_0}^{x_1} \phi_{M-1}'\phi_1' dx & \int_{x_0}^{x_1} \phi_{M-1}'\phi_2' dx & \cdots & \int_{x_0}^{x_1} (\phi_{M-1}')^2 dx \end{bmatrix} \\
&\quad + \begin{bmatrix} \int_{x_1}^{x_2} (\phi_1')^2 dx & \int_{x_1}^{x_2} \phi_1'\phi_2' dx & \cdots & \int_{x_1}^{x_2} \phi_1'\phi_{M-1}' dx \\ \int_{x_1}^{x_2} \phi_2'\phi_1' dx & \int_{x_1}^{x_2} (\phi_2')^2 dx & \cdots & \int_{x_1}^{x_2} \phi_2'\phi_{M-1}' dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_{x_1}^{x_2} \phi_{M-1}'\phi_1' dx & \int_{x_1}^{x_2} \phi_{M-1}'\phi_2' dx & \cdots & \int_{x_1}^{x_2} (\phi_{M-1}')^2 dx \end{bmatrix} \\
&\quad + \cdots \\
&\quad + \begin{bmatrix} \int_{x_{M-1}}^{x_M} (\phi_1')^2 dx & \int_{x_{M-1}}^{x_M} \phi_1'\phi_2' dx & \cdots & \int_{x_{M-1}}^{x_M} \phi_1'\phi_{M-1}' dx \\ \int_{x_{M-1}}^{x_M} \phi_2'\phi_1' dx & \int_{x_{M-1}}^{x_M} (\phi_2')^2 dx & \cdots & \int_{x_{M-1}}^{x_M} \phi_2'\phi_{M-1}' dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_{x_{M-1}}^{x_M} \phi_{M-1}'\phi_1' dx & \int_{x_{M-1}}^{x_M} \phi_{M-1}'\phi_2' dx & \cdots & \int_{x_{M-1}}^{x_M} (\phi_{M-1}')^2 dx \end{bmatrix}.
\end{aligned}$$

Note for the hat basis functions, each interval have only *two* no-zero basis functions, therefore,

$$
\begin{aligned}
A \;=\; & \begin{bmatrix} \int_{x_0}^{x_1} (\phi_1')^2 dx & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} + \begin{bmatrix} \int_{x_1}^{x_2} (\phi_1')^2 dx & \int_{x_1}^{x_2} \phi_1' \phi_2' dx & \cdots & 0 \\ \int_{x_1}^{x_2} \phi_2' \phi_1' dx & \int_{x_1}^{x_2} (\phi_2')^2 dx & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \\
& + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & \int_{x_2}^{x_3} (\phi_2')^2 dx & \int_{x_2}^{x_3} \phi_2' \phi_3' dx & \cdots & 0 \\ 0 & \int_{x_2}^{x_3} \phi_3' \phi_2' dx & \int_{x_2}^{x_3} (\phi_3')^2 dx & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \int_{x_{M-1}}^{M} (\phi_{M-1}')^2 dx \end{bmatrix}
\end{aligned}
$$

The no-zero contribution from a particular element is

$$
K_{ij}^e = \begin{bmatrix} \int_{x_i}^{x_{i+1}} \phi_i' 2 & \int_{x_i}^{x_{i+1}} \phi_i' \phi_{i+1}' dx \\ \int_{x_i}^{x_{i+1}} \phi_{i+1}' \phi_i' dx & \int_{x_i}^{x_{i+1}} (\phi_{i+1}')^2 dx \end{bmatrix}
$$

This 2 by 2 matrix is called the *local stiffness matrix.* Similarly the *local load vector* is defined as

$$
F_i^e = \begin{bmatrix} \int_{x_i}^{x_{i+1}} f \phi_i dx \\ \int_{x_i}^{x_{i+1}} f \phi_{i+1} dx \end{bmatrix}.
$$

The global load vector can be assembled element by element

$$
F = \begin{bmatrix} \int_{x_0}^{x_1} f \phi_1 dx \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \int_{x_1}^{x_2} f \phi_1 dx \\ \int_{x_1}^{x_2} f \phi_2 dx \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \int_{x_2}^{x_3} f \phi_2 dx \\ \int_{x_2}^{x_3} f \phi_3 dx \\ \vdots \\ 0 \\ 0 \end{bmatrix} + \cdots + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \int_{x_{M-1}}^{x_M} f \phi_{M-1} dx \end{bmatrix}.
$$

Computing local stiffness matrix K_{ij}^e and local load vector F_i^e.

In the element $[x_i, x_{i+1}]$, there are only two non-zero hat functions centered at x_i and x_{i+1} respectively:

$$\psi_i^e(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i}, \qquad \psi_{i+1}^e(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

$$(\psi_i^e)' = -\frac{1}{h_i}, \qquad (\psi_{i+1}^e)' = \frac{1}{h_i}.$$

ψ_i^e and ψ_{i+1} are called the *shape functions* and they are defined only on one particular element. It is easy to verify the following:

$$\begin{aligned}
\int_{x_i}^{x_{i+1}} (\psi_i')^2 dx &= \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2} dx = \frac{1}{h_i} \\
\int_{x_i}^{x_{i+1}} \psi_i' \phi_{i+1}' dx & \int_{x_i}^{x_{i+1}} -\frac{1}{h_i^2} dx = -\frac{1}{h_i} \\
\int_{x_i}^{x_{i+1}} (\psi_{i+1}')^2 dx &= \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2} dx = \frac{1}{h_i}
\end{aligned}$$

The local stiffness matrix K_{ij}^e therefore is:

$$K_{ij}^e = \begin{bmatrix} \frac{1}{h_i} & -\frac{1}{h_i} \\ -\frac{1}{h_i} & \frac{1}{h_i} \end{bmatrix}$$

We assemble the stiffness matrix A as

$$A = 0^{M-1 \times M-1}, \quad A = \begin{bmatrix} \frac{1}{h_0} & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \quad A = \begin{bmatrix} \frac{1}{h_0} + \frac{1}{h_1} & -\frac{1}{h_1} & 0 & \cdots \\ -\frac{1}{h_1} & \frac{1}{h_1} & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix},$$

$$A = \begin{bmatrix} \frac{1}{h_0} + \frac{1}{h_1} & -\frac{1}{h_1} & 0 & 0 & \cdots \\ -\frac{1}{h_1} & \frac{1}{h_1} + \frac{1}{h_2} & -\frac{1}{h_2} & 0 & \cdots \\ 0 & -\frac{1}{h_2} & \frac{1}{h_2} & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}.$$

The final assembled matrix is a tridiagonal matrix:

$$A \;=\; \begin{bmatrix} \frac{1}{h_0}+\frac{1}{h_1} & -\frac{1}{h_1} & & & & \\ -\frac{1}{h_1} & \frac{1}{h_1}+\frac{1}{h_2} & -\frac{1}{h_2} & & & \\ & -\frac{1}{h_2} & \frac{1}{h_2}+\frac{1}{h_3} & -\frac{1}{h_3} & & \\ & & \ddots & \ddots & \ddots & \\ & & & -\frac{1}{h_{M-3}} & \frac{1}{h_{M-3}}+\frac{1}{h_{M-2}} & -\frac{1}{h_{M-2}} \\ & & & & -\frac{1}{h_{M-2}} & \frac{1}{h_{M-2}}+\frac{1}{h_{M-1}} \end{bmatrix}.$$

Remark 4.1 *If we use a uniform triangulation* $x_i = i * h$, $h = 1/M$, $i = 0, 1, \cdots M$ *and approximate*

$$\int_0^1 f(x)\phi(x)dx = \int_{x_{i-1}}^{x_{i+1}} f(x)\phi(x) \approx \int_{x_{i-1}}^{x_{i+1}} f(x_i)\phi(x)dx = f(x_i)\int_{x_{i-1}}^{x_{i+1}} \phi(x)dx = f(x_i),$$

then the system of equation we get for the model problem using the finite element method is exact the same as that derived from the finite difference method.

5 Matlab codes of FEM for 1D model problem

Matlab codes to solve the model problem:

$$-u''(x) = f(x), \quad a \le x \le b; \qquad u(a) = u(b) = 0. \tag{17}$$

using the hat basis functions can be found in

http://www4.ncsu.edu/˜zhilin/TEACHING/MA587/MATLAB/1DSIMPLE/

The global stiffness matrix and load vector are assembled element by element. The matlab codes including the following:

- $[U] = fem1d(x)$ is the main subroutine of the finite method using the hat basis functions. The input x is the vector containing the nodal points. The output U, $U(0) = U(M) = 0$ is the finite element solution at the nodal points, where $M + 1$ is the total nodal points.
- $y = hat1(x, x1, x2)$ is the local hat function in the interval $[x1, x2]$ which takes one at $x = x2$ and zero at $x = x1$.
- $y = hat2(x, x1, x2)$ is the local hat function in the interval $[x1, x2]$ which takes one at $x = x1$ and zero at $x = x2$.

- $y = int\_hata1\_f(x1, x2)$ computes the integral $\int_{x1}^{x2} fhat1dx$ using the Simpson rule.
- $y = int\_hata2\_f(x1, x2)$ computes the integral $\int_{x1}^{x2} fhat2dx$ using the Simpson rule.
- The main function is *drive.m* which solves the problem, plot the solution and the error.
- $y = f(x)$ is the right hand side of the differential equation.
- $y = soln(x)$ is the exact solution of differential equation.
- $y = fem\_soln(x, U, xp)$ evaluates the finite element solution as an arbitrary point xp in the solution domain.

5.1 Define the basis functions

In an element $[x_1, x_2]$, there are two non-zero basis functions (shape functions)

$$\psi_1^e(x) = \frac{x - x_1}{x_2 - x_1} \tag{18}$$

The Matlab code is the file hat1.m

```
function y = hat1(x,x1,x2)
% This function evaluate the hat function of the form
y = (x-x1)/(x2-x1);
return
```

The other one is

$$\psi_2^e(x) = \frac{x_2 - x}{x_2 - x_1} \tag{19}$$

and the Matlab code is the file hat2.m

```
function y = hat2(x,x1,x2)
% This function evaluate the hat function of the form
y = (x2-x)/(x2-x1);
return
```

5.2 Define $f(x)$

```
function y = f(x)
y = 1;
return
```

5.3 Main routine

```
function U = fem1d(x)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                    %
%                   Simple Matlab code for 1D FEM for                %
%                                                                    %
%                    -u’’ = f(x),   a <= x <= b, u(a)=u(b)=0         %
%    Input: x, Nodal points                                          %
%    Output: U, FEM solution at nodal points                         %
%                                                                    %
%    Function needed: f(x).                                          %
%                                                                    %
%    Matlab functions used:                                          %
%                                                                    %
%     hat1(x,x1,x2), hat function in [x1,x2] that is 1 at x2; and    %
%     0 at x1.                                                       %
%                                                                    %
%     hat2(x,x1,x2), hat function in [x1,x2] that is 0 at x1; and    %
%     1 at x1.                                                       %
%                                                                    %
%     int_hat1_f(x1,x2): Contribution to the load vector from hat1   %
%     int_hat2_f(x1,x2): Contribution to the load vector from hat2   %
%                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M = length(x);
for i=1:M-1,
  h(i) = x(i+1)-x(i);
end

A = sparse(M,M); F=zeros(M,1);           % Initialize
A(1,1) = 1; F(1)=0;
A(M,M) = 1; F(M)=0;
A(2,2) = 1/h(1); F(2) = int_hat1_f(x(1),x(2));

for i=2:M-2,                        % Assembling element by element
  A(i,i) = A(i,i) + 1/h(i);
  A(i,i+1)   = A(i,i+1) - 1/h(i);
  A(i+1,i)   = A(i+1,i) - 1/h(i);
  A(i+1,i+1) = A(i+1,i+1) + 1/h(i);
  F(i)       = F(i) + int_hat2_f(x(i),x(i+1));
  F(i+1)     = F(i+1) + int_hat1_f(x(i),x(i+1));
end

  A(M-1,M-1) = A(M-1,M-1) + 1/h(M-1);
```

```
  F(M-1)     = F(M-1) + int_hat2_f(x(M-1),x(M));

  U = A\F;                  % Solve the linear system of equation.

  return
```

5.4 Test example

First we test the results with

$$f(x) = 1, a = 0, b = 1$$

The exact solution is

$$u(x) = \frac{1}{2}x(1-x) \tag{20}$$

The drive code is

```
clear all; close all;     % Clear every thing so it won't mess up with other
                          % existing variables.

%%%%%% Generate a triangulation

x(1)=0; x(2)=0.1; x(3)=0.3; x(4)=0.333; x(5)=0.5; x(6)=0.75;x(7)=1;

U = fem1d(x);

%%%%%% Compare errors:

x2 = 0:0.05:1; k2 = length(x2);
for i=1:k2,
  u_exact(i) = soln(x2(i));
  u_fem(i) = fem_soln(x,U,x2(i));  % Compute FEM solution at x2(i)
end

error = norm(u_fem-u_exact,inf)  % Compute the infinity error

plot(x2,u_fem,':', x2,u_exact)   % Solid: the exact, %dotted: FEM solution
hold; plot(x,U,'o')              % Mark the solution at nodal points.
xlabel('x'); ylabel('u(x) & u_{fem}(x)');
title('Solid line: Exact solution, Dotted line: FEM solution')

figure(2); plot(x2,u_fem-u_exact); title('Error plot')
xlabel('x'); ylabel('u-u_{fem}');  title('Error Plot')
```

The plots produced are the following

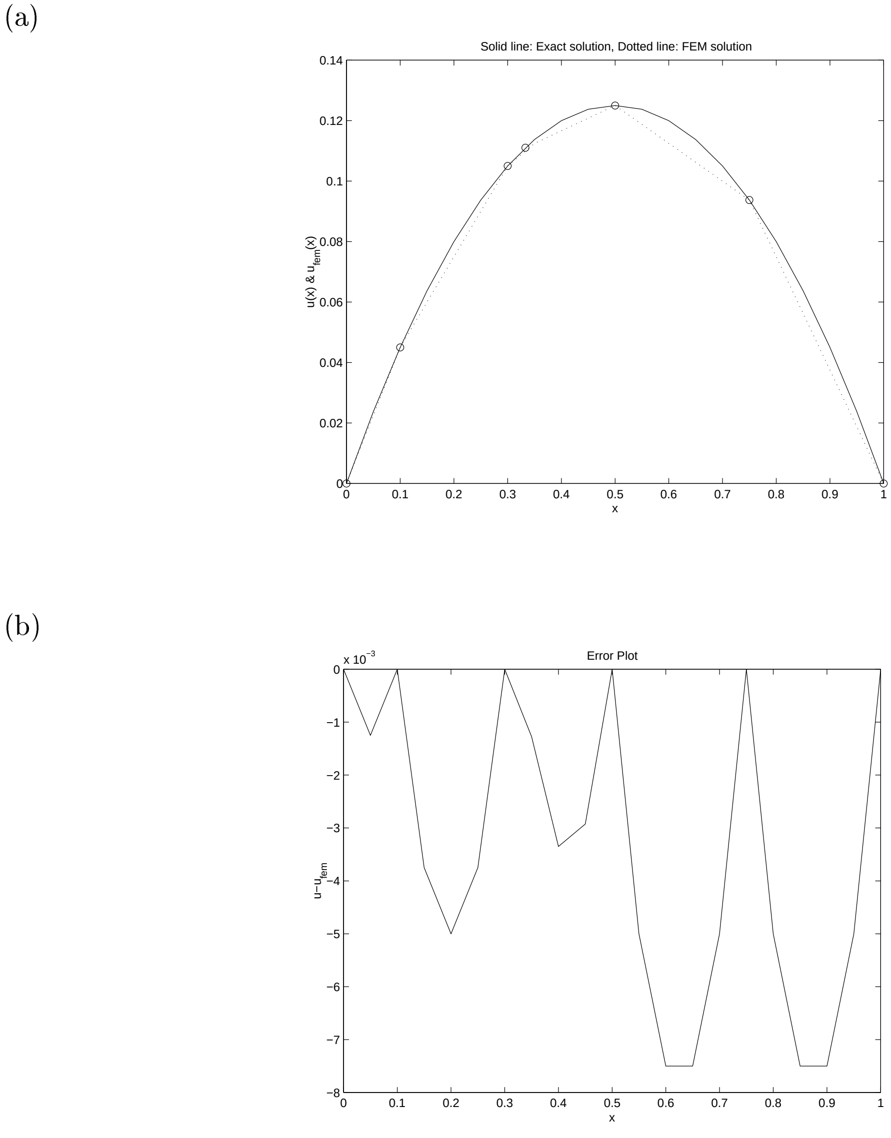

Figure 3: (a) Solution plot. (b): Error plot.