Numerical Solutions of Partial Differential Equations– An Introduction to Finite Difference and Finite Element Methods

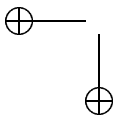
Zhilin Li[1] Zhonghua Qiao[2] Tao Tang[3]

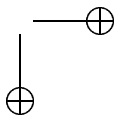
February 17, 2011

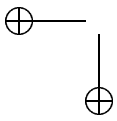
[1]Center for Research in Scientific Computation & Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205, USA

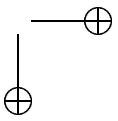
[2]Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong

[3]Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong

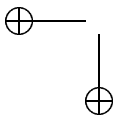


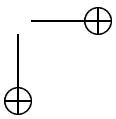






Contents





Preface

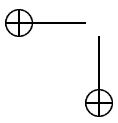
The purpose of this book is to provide an introduction to numerical methods for partial differential equations using the finite difference and finite element methods. This book aims at beginning graduate students, upper level undergraduate students, students from interdisciplinary areas, engineers, and anyone who uses numerical methods to solve computational problems modeled by differential equations.

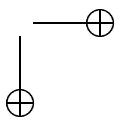
The prerequisites of this book are calculus, linear algebra, and elementary differential equations. Some knowledge of numerical analysis and partial differential equations would be helpful. The style of this book is to emphasize both the mathematical theory of the finite difference and finite element methods as well as their implementation details. The part I of this book is about finite difference methods while part II is about finite element methods. For each part, we starts with comprehensive one-dimensional discussions before moving on to two or high dimensional problems. We also list some references for those readers who wish to know more in the related areas.

This text book is based on the authors' course materials that the authors have used in teaching graduate numerical differential equation courses.

Most sample computer programming are written in Matlab. Advantages of using Matlab include its simplicity, wide range of building libraries, double precision accuracy, and many existing and emerging tool-boxes.

A web-site, http://www4.ncsu.edu/~zhilin/`FD_FEM_Book`, has been set up to post or link the computer codes in companying of this text book.





Chapter 1

Introduction.

1.1 The Problems to be solved: Differential equations of boundary value problems

We want to solve ordinary/partial differential equations (ODE/PDE), especially linear second order ODE/PDEs, and first order systems, *numerically*.

What is a differential equation? It is an equation whose unknown is a function, say $u(x)$, or $u(x, y)$, or $u(t, x)$, or $u(t, x, y, z)$ etc. The equation involves the derivatives or partial derivatives of the unknown function.

Differential equations[1] have been used intensively to model many physical problems including fluid/solid mechanics, biology, material sciences, economics, ecology, sports and computer sciences. Here are some examples, the Laplace equation for potentials, Navier Stokes equations in fluid dynamics, biharmonic equations for stresses in solid mechanics, the Maxwell equations in electro-magnetics, and many others. For more examples and mathematical theory of partial differential equations, we refer the reader to [9] and the references therein.

Unfortunately, while differential equations can describe many physical problems, only very small portion of them can be solved exactly in terms of elementary functions such as polynomials, trogonometric functions ($\sin x$, $\cos x$, ...), $\log x$, e^x, a^x etc. and their combinations. Quite often, even if a differential equation can be solved analytically, great efforts and sound mathematical theories are needed. The closed form of the solution may be too complicated to be actually useful.

If the analytic solution is not available or too difficult to get for a differential equation, we may want to find an approximate solution to the differential equation. Typically, there are two approaches:

- Semi-analytic methods. Sometime we can approximate the solution using series, integral equations, perturbation techniques, asymptotic approximations etc. The solution is often expressed as some simpler function.

- Numerical solutions which are some numbers. Those numbers are approximate solution to the original differential equation with certain accuracy. Nowadays,

[1] There are other models as well, for example, statistical models.

those numbers are obtained from computers. The development of modern computers make it possible to solve many problems that were impossible just a few decades, or years ago.

In this book, we will mainly use the second approach. In Fig.1.1, we show a flow chart of a problem solving process. In this book, we will focus on numerical solutions using computers, especially the finite difference or finite element methods for differential equations.

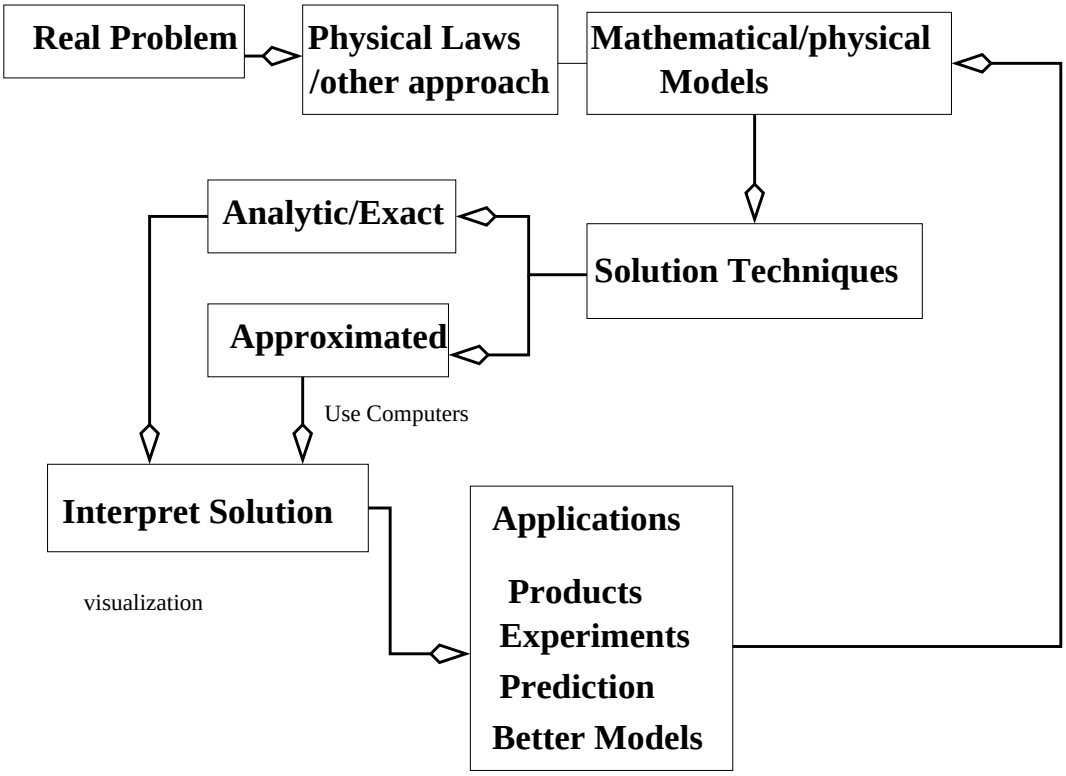


Figure 1.1. *A flow chart of a problem solving process.*

Linear differential equations can be classified as *elliptic*, *parabolic*, and *hyperbolic* equations. Below are some examples:

1. Initial value problems (IVP) . The most discussed one is the canonical first order system

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0 \tag{1.1}$$

A second orer ordinary differential equation of the following

$$\begin{aligned} &u''(t) + a(t)u'(t) + b(t)u(t) = f(t), \\ &u(0) = u_0, \quad \boxed{u'(0) = v_0}. \end{aligned} \tag{1.2}$$

can be converted a first order system if we set $y_1(t) = u$ and $y_2(t) = u'(t)$.

The system of initial value problems can often be solved using the Runge-Kutta methods with adaptive time steps. In Matlab, it is the ODE-SUITS which include ode45, ode23, ode23s, ode15s, $\cdots$ built in Matlab. For a stiff ODE system, it is recommended to use ode23s, ode15s.

2. Boundary value problems (BVP). indexBVP Here is a one-dimensional example,

$$\begin{aligned} &u''(x) + a(x)u'(x) + b(x)u(x) = f(x), \\ &u(0) = u_0, \quad \boxed{u(1) = u_1}. \end{aligned} \tag{1.3}$$

Below is a two-dimensional example:

$$\begin{aligned} &-(u_{xx} + u_{yy}) = f(x,y), \quad (x,y) \in \Omega \\ &u(x,y) = u_0(x,y), \quad (x,y) \in \partial\Omega. \end{aligned} \tag{1.4}$$

3. Boundary and initial value problems

$$\begin{aligned} &u_t = au_{xx} + f(x,t) \\ &u(0,t) = g_1(t), \quad u(1,t) = g_2(t), \qquad \text{BC} \\ &u(x,0) = u_0(x), \qquad \text{IC} \end{aligned} \tag{1.5}$$

4. Eigenvalue problems.

$$\begin{aligned} &-u''(x) = \lambda u(x) \\ &u(0) = 0, \quad u(1) = 0. \end{aligned} \tag{1.6}$$

In this example, both $u(x)$ and scalar λ are unknowns.

5. Diffusion and reaction equations:

$$\frac{\partial u}{\partial t} = \nabla\left(\cdot\beta\nabla u\right) + \mathbf{a}\cdot\nabla u + f(u) \tag{1.7}$$

where $\mathbf{a}$ is a constant vector, $\nabla\cdot\beta\nabla u$ is called a diffusion term, $\mathbf{a}\cdot\nabla u$ is called an advection term, and $f(u)$ is called a reaction term.

6. System of PDEs and non-linear PDEs. A very important one is the incompressible Navier Stokes equations

$$\begin{aligned} &\rho\left(\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = \nabla p + \mu\Delta\mathbf{u} + \mathbf{F} \\ &\nabla\cdot\mathbf{u} = 0 \end{aligned} \tag{1.8}$$

Generally, we will consider the second order *linear* partial differential equations of the form

$$a(x,y)u_{xx} + 2b(x,y)u_{xy} + c(x,y)u_{yy} + d(x,y)u_x + e(x,y)u_y + g(x,y)u(x,y) = f(x,y),$$

in a domain Ω, where the coefficients are independent of $u(x,y)$. The equation is linear with respect to u and its partial derivatives. We can classify the equation above as

- Elliptic if $b^2 - ac < 0$ for all $(x,y) \in \Omega$.
- Parabolic if $b^2 - ac = 0$ for all $(x,y) \in \Omega$.
- Hyperbolic if $b^2 - ac > 0$ for all $(x,y) \in \Omega$.

Different type equation requires different method. Another type of problems is the first order system

$$\frac{\partial \mathbf{u}}{\partial t} = A(\mathbf{x})\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \tag{1.9}$$

The classification then is determined from the eigenvalues of the matrix $A(\mathbf{x})$.

Finite difference and finite element methods are techniques to solve differential equations numerically. Other techniques include finite volume methods, collocation methods, spectral methods and many others.

1.1.1 Main features of finite difference and finite element methods

In this book, we will discuss both finite difference and finite element methods. They all extensively used. For many problems, they can be solved numerically with methods for each category. For some other problems, methods from one category may perform better. We strongly believe that a good numerical analyst should be familiar with methods from both categories. We give a brief list of features of finite difference and finite element methods.

Finite difference (FD) methods:

- They are simple to use and are easy to understand.
- They are easy to implement for regular domains such as rectangular domains in Cartesian coordinates, circles or annulus domains in polar coordinates.
- The discretization and approximate solutions are pointwise. The fundamental mathematical tool is the Taylor expansion.
- There are many fast solvers and packages for regular domains, for example, the past Poisson solver 'Fishpack [1], the claw pack [14].
- They are difficult to implement for complicated geometries.

- They have strong regularity requirements (derivatives) for the solutions.

Finite element (FE) methods:

- They are very successful for structural (elliptic type) problems.
- They are natural approach for problems with complicated boundaries.
- There are solid theoretical foundations at least for elliptic type problems using the Soblev theory.
- They require *weaker requirement* for the solutions.
- There are many commercial packages, for example, Ansys.
- They are usually coupled with multigrid solvers.
- One would 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.

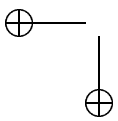
Finite Difference methods, finite element methods, finite volume methods, spectral methods are all useful for solving PDEs/ODEs.

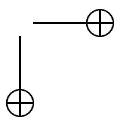
1.2 Further Reading

This text book provides an introduction to finite difference and finite methods for solving ordinary/partial differential equation with prescribed certain linear boundary conditions, or so called boundary value problems. If the readers wish to become experts in finite difference and finite methods, we recommend the following books for finite difference methods [11, 19, 15, 23, 22]; and [7] for finite methods. The text books [23, 22] are classical ones while [11, 19, 15] are relatively new. In [15], the readers can find the accompanying Matlab codes from the author's website.

There are many books on finite elements methods. The most classic one probably is the one by P. G. Ciarlet [7]. The books [13, 21] have been widely used as graduate text books. The series by Orden et. al. [5] not only presented mathematical background of finite element methods, but also gave fair amount of details about FEM programming in Fortran. New new text books have also emerged, say for example, [2, 3].

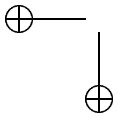
This book is written as a text book and is not focused on research progress in the related areas. As a result, the references citation is not complete.

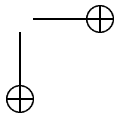




Part I

Finite Difference Methods





Chapter 2

Finite difference methods for one-dimensional problems

2.1 What is a finite difference method? A simple example

Consider a model problem

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

We will solve this problem using a finite difference method to illustrate the general procedure described below.

1. Generate a grid. For example, we can use a uniform Cartesian grid

   $$x_i = i\,h, \quad i = 0, 1, \cdots n, \quad h = \frac{1}{n}.$$

   A grid is a finite set of points where we want to find an approximate solution to the differential equation.

2. Substitute the derivatives with some *finite difference* formulas at every grid points where the solution is unknown to get an algebraic system of equations. Notice that for a twice differentiable function $\phi(x)$, we have

   $$\phi''(x) = \lim_{\Delta x \to 0} \frac{\phi(x - \Delta x) - 2\phi(x) + \phi(x + \Delta x)}{(\Delta x)^2}$$

   Therefore we can approximate $u''(x)$ using nearby function values

   $$\phi''(x) = \frac{\phi(x - \Delta x) - 2\phi(x) + \phi(x + \Delta x)}{(\Delta x)^2} + error$$

   At each grid point x_i we approximate the differential equation by

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

The *error* is called the local truncation error and will be explained later. We define the finite difference solution, or approximation, of $u(x)$ at x_i as U_i as the solution (if exists) of the following linear system of equations

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

Note that at each grid, the finite difference approximation involves the solution at three grid points x_{i-1}, x_i, and x_{i+1}. The set of these three grid points is called the *finite difference stencil.*

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

$$\begin{bmatrix}
-\frac{2}{h^2} & \frac{1}{h^2} & & & & \\
\frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & & \\
 & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \\
 & & & & \frac{1}{h^2} & -\frac{2}{h^2}
\end{bmatrix}
\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_{n-2} \\ U_{n-1} \end{bmatrix}
=
\begin{bmatrix} f(x_1) - u_a/h^2 \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) - u_b/h^2 \end{bmatrix} \tag{2.1}$$

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

4. Implement and debug the computer code. Run the program to get the output. Analyze and visualize the results (tables, plots etc.).

5. Error analysis. As we will show that the consistency along with the stability implies the convergence of the finite difference method. It is point-wise convergence, that is $\lim_{h\to 0} \|u(x_i) - U_i\|_\infty = 0$. The finite difference method needs the solution $u(x)$ to have *second order derivative.*

2.1.1 The Matlab code for the model problem

Below we show a Matlab function for the model problem called two_point.m. We use this Matlab function to illustrate how to convert an algorithm to a computer code.

```
function [x,U] = two_point(a,b,ua,ub,f,n)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    This matlab function two_point solves the following two-point     %
%    boundary value problem: u''(x) = f(x) using the center finite     %
%    difference scheme.                                                %
%    Input:                                                            %
%     a, b: Two end points.                                            %
%     ua, ub: Dirichlet boundary conditions at a and b                 %
%     f: external function f(x).                                       %
%     n: number of grid points.                                        %
%    Output:                                                           %
%     x: x(1),x(2),...x(n-1) are grid points                           %
%     U: U(1),U(2),...U(n-1) are approximate solution at grid points   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

h = (b-a)/n; h1=h*h;

A = sparse(n-1,n-1);
F = zeros(n-1,1);

for i=1:n-2,
  A(i,i) = -2/h1; A(i+1,i) = 1/h1; A(i,i+1)= 1/h1;
end
  A(n-1,n-1) = -2/h1;

for i=1:n-1,
  x(i) = a+i*h;
  F(i) = feval(f,x(i));
end
  F(1) = F(1) - ua/h1;
  F(n-1) = F(n-1) - ub/h1;

U = A\F;

return
%%%%%---------  End of the program ------------------------------------
```

We can call the Matlab function two_point directly in a Matlab command window. A better way is to put all the Matlab command in a Matlab file called an M-file, say, the file called *main.m*. The advantage of this approach is to keep a record and we can re-visit or modify the file whenever we want.

Let the interval be [0, 1], $f(x) = -\pi^2 \cos(\pi x)$, $u(0) = 0$ and $u(1) = -1$. A sample Matlab M-file is listed below.

```
%%%%%%%% Clear all unwanted variable and graphs.

clear;  close all

%%%%%%% Input

a =0; b=1; n=40;
ua = 1; ub = -1;

%%%%%% Call the solver: U is the finite difference solution at grid points.

[x,U] = two_point(a,b,ua,ub,'f',n);

%%%%%%%%%%%%%%%%%%%% Plot and error analysis: %%%%%%%%%%%%%%%%%%%

plot(x,U,'o'); hold

u=zeros(n-1,1);
for i=1:n-1,
  u(i) = cos(pi*x(i));
end
plot(x,u)

%%%%%%% Plot error

figure(2); plot(x,U-u)

norm(U-u,inf)           %%% Print out the maximum error.
```

It is easy to check that the exact solution is $\cos(\pi x)$. If we plot the computed solution that is defined only at the grid points (use plot(x,u,'o'), and the exact solution, solid line in Fig. 2.1 (a), we see no difference with naked eyes. However, if we plot the difference of the computed solution and the exact solution, which we call the error, we see there is a difference which is about $O(10^{-3})$, see Fig. 2.1 (b). So we should be happy about our results.

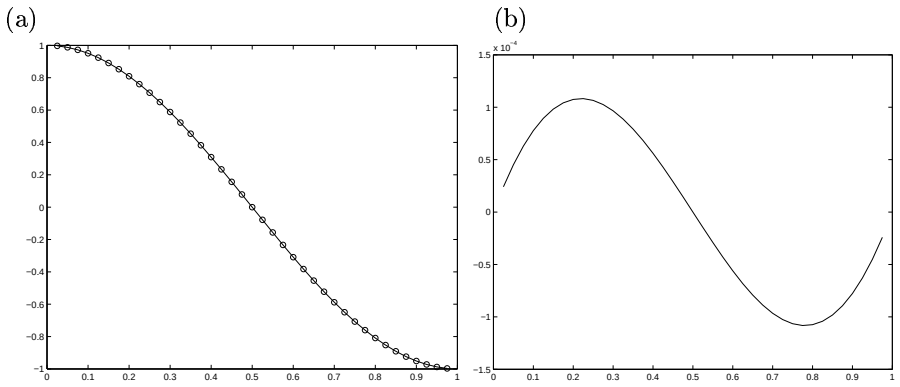


Figure 2.1. *(a) The plot of the computed solution (little 'o's), and the exact solution (solid line). (b) The plot of the errors.*

2.1.2 Questions that we should ask from this example

- Are there other finite difference formulas to approximate derivatives? If so, how do we derive them?
- How do we know a finite difference method works or not? If it works, how accurate is it? In other words, how small is the error? The answer depends on the consistency and the stability of the finite difference method.
- Do the computer errors, which are called round-off errors, affect the computed solution? If so, how much?
- How do we deal with different boundary conditions such as derivative (Neumann) or mixed (Robin) boundary conditions?
- Do we need different finite difference methods for different problems? If so, are there similar procedure?
- How do we know that we are using the most efficient method? What are the criteria? How do we implement finite difference methods efficiently?

We will address these questions in the next few chapters.

2.2 Fundamentals of Finite Difference Methods

The Taylor expansion is the most important tool in the analysis of finite difference methods. The Taylor expansion can be written as two slightly different forms

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \cdots + \frac{h^k}{k!}u^{(k)}(x) + \cdots \tag{2.2}$$

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \cdots + \frac{h^k}{k!}u^{(k)}(\xi), \tag{2.3}$$

if $|h|$ is small enough and $u(x)$ has necessary continuous high derivatives. The second one sometimes is called the extended mean value theorem.

In a finite difference method, we need to substitute the derivatives with finite difference formulas to get an linear/non-linear algebraic system. There are several different ways to substitute the derivatives with finite difference formulas.

2.2.1 Forward, backward, and central finite difference formulas for $u'(x)$

Let us first exam the first derivative $u'(x)$ of $u(x)$ at a point $\bar{x}$ using the nearby function values $u(\bar{x} \pm h)$, where h is called the step size. There are three commonly used formulas:

$$\text{Forward finite difference:} \quad \Delta_+ u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x})}{h} \sim u'(\bar{x}), \tag{2.4}$$

$$\text{Backward finite difference:} \quad \Delta_- u(\bar{x}) = \frac{u(\bar{x}) - u(\bar{x}-h)}{h} \sim u'(\bar{x}), \tag{2.5}$$

$$\text{Central finite difference:} \quad \delta u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x}-h)}{2h} \sim u'(\bar{x}). \tag{2.6}$$

Below we derive these finite difference formulas from geometric intuition and the calculus.

From calculus, we know that

$$u'(\bar{x}) = \lim_{h \to 0} \frac{u(\bar{x}+h) - u(\bar{x})}{h}$$

Assume h is small and $u'(x)$ is continuous, then we expect that $\frac{u(\bar{x}+h)-u(\bar{x})}{h}$ is close to $u'(\bar{x})$. We can use it to approximate the first derivative at $\bar{x}$.

$$\frac{u(\bar{x}+h) - u(\bar{x})}{h} \sim u'(\bar{x}). \tag{2.7}$$

Note that usually the two sides of above are not the same, and this step brings an error! In practice, h is the step size which is the distance between two grid points, so $h > 0$, we call the following formula as the ***forward finite difference***

$$\Delta_+ u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x})}{h}. \tag{2.8}$$

Geometrically, it is the slope of the secant line that connects the two points $(\bar{x}, u(\bar{x}))$ and $(\bar{x}+h, u(\bar{x}+h))$. How close is this formula to the derivative? We can use the extended mean value theorem (truncated Taylor expansion) to find it out. Assume that $u(\bar{x})$ has second order continuous derivatives. If h is small, then we have

$$u(\bar{x}+h) = u(\bar{x}) + u'(\bar{x})h + \frac{1}{2}u''(\xi)h^2. \tag{2.9}$$

Therefore we can get an error estimate

$$E_f(h) = \frac{u(\bar{x}+h) - u(\bar{x})}{h} - u'(\bar{x}) = \frac{1}{2}u''(\xi)h = O(h). \tag{2.10}$$

So the error, defined as the difference of the approximate value and the exact one, is proportional to h, such discretization is called *first order accurate.* Generally, if the error has the form

$$E(h) = Ch^p, \qquad p > 0, \tag{2.11}$$

then the method is called p-th order accurate.

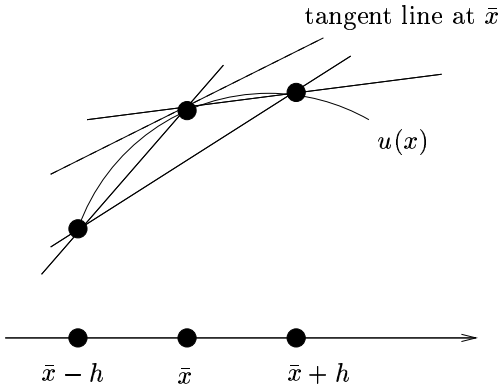


Figure 2.2. *A geometric illustration of the forward, backward, and central finite difference formulas for approximating* $u'(\bar{x})$.

Similarly, we can analyze the ***backward finite difference*** formula

$$\Delta_- u(\bar{x}) = \frac{u(\bar{x}) - u(\bar{x} - h)}{h}, \quad h > 0. \tag{2.12}$$

for approximating $u'(\bar{x})$. It is easy to get the following

$$E_b(h) = \frac{u(\bar{x}) - u(\bar{x} - h)}{h} - u'(\bar{x}) = -\frac{1}{2}u''(\xi)\, h = O(h). \tag{2.13}$$

Therefore, the backward finite difference is also first order accurate.

Geometrically, we can see that the slope of the secant line that passes through $(\bar{x} + h, u(\bar{x} + h))$ and $(\bar{x} - h, u(\bar{x} - h))$ is a better approximation to the slope of the tangent line of $u(\bar{x})$ at $(\bar{x}, u(\bar{x}))$. The formula

$$\delta u(\bar{x}) = \frac{u(\bar{x} + h) - u(\bar{x} - h)}{2h}, \quad h > 0, \tag{2.14}$$

is called the ***central finite difference*** formula for approximating first order derivatives.

In order to get a correct error estimate, we need to use more terms in the Taylor expansion

$$u(x+h) = u(x) + hu'(x) + \frac{1}{2}u''(x)h^2 + \frac{1}{6}u'''(x)h^3 + \frac{1}{24}u^{(4)}(x)h^4 + \cdots$$

$$u(x+h) = u(x) - hu'(x) + \frac{1}{2}u''(x)h^2 - \frac{1}{6}u'''(x)h^3 + \frac{1}{24}u^{(4)}(x)h^4 + \cdots$$

Therefore, we have

$$E_c(h) = \frac{u(\bar{x}+h) - u(\bar{x}-h)}{2h} - u'(\bar{x}) = \frac{1}{6}u'''(\bar{x})h^2 + h.o.t = O(h^2), \quad (2.15)$$

where $h.o.t$ stands for higher order terms. Note we have used a slightly different approach from what we used for forward and backward formulas. The central finite difference formula is second order accurate. It is easy to show that the formula above can be written as

$$\delta u() = \frac{u(\bar{x}+h) - u(\bar{x}-h)}{2h} = \frac{1}{2}\left(\Delta_+ + \Delta_-\right) u(\bar{x}).$$

A third order accurate finite difference formula for $u'(\bar{x}x)$ is

$$\delta_3 u(\bar{x}) = \frac{2u(\bar{x}+h) + 3u(\bar{x}) - 6u(\bar{x}-h) + u(\bar{x}-2h)}{6h}. \quad (2.16)$$

2.2.2 Verification and grid refinement analysis

Assume that we have learned or developed a numerical method and have had some analysis. We can write a compute code to implement the method. How do we know that our code is bug-free and our analysis is correct? One method is called the grid refinement analysis.

Typically, in a grid refinement analysis, we solve a problem in which we know the exact solution[2]. We start with a fixed h, say $h = 0.1$, then decrease h by half to see how the error changes. For a first order method, the error should be decrease by a factor of two, and for a second order method, the error should be decrease by a factor of four. We can also plot the errors versus h in the log-log scale. In the log-log scale, the slope is the order of convergence if we have equal scales in both axis.

Below we show a comparison of the forward, backward, and central finite difference formula in a Matlab script file compare.m. We choose the test function as $u(x) = \sin x$ at $x = 1$, with exact derivative being $\cos 1$. In Fig. 2.3, we plot the errors versus h in log-log scale in which the slopes indeed give correct indication of the convergence order.

```
% Compare truncation errors of the forward, backward, and central
```

[2]This is not always available. We will mention other techniques about how to validate a computed solution later.

```
% scheme for approximating u'(x). Plot the error and estimate the
% convergence order.
%    u(x) = sin(x) at x=1.   Exact derivative: u'(1) = cos(1)

clear; close all
  h = 0.1;
  for i=1:5,
    a(i,1) = h;
    a(i,2) = (sin(1+h)-sin(1))/h - cos(1);
    a(i,3) = (sin(1) - sin(1-h))/h - cos(1);
    a(i,4) = (sin(1+h)-sin(1-h))/(2*h)- cos(1);
    h = h/2;
  end

  format short e  % Use this option to see the first few significant digits.
  a % Display the result

  a = abs(a);    % Take absolute value of the matrix.
  h1 = a(:,1);    % Extract the first column which is h.
  e1 = a(:,2); e2 = a(:,3); e3 = a(:,4);

  loglog(h1,e1,h1,e2,h1,e3)
  axis('equal'); axis('square')
  axis([1e-6 1e1 1e-6 1e1])
  gtext('Slope of FW and BW = 1')
  gtext('Slope of CD =2')

%%%%%%%%%%%%%%%%% End Of Matlab Program %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Computed Results:

%      h          forward      backward     central

%   1.0000e-01  -4.2939e-02   4.1138e-02  -9.0005e-04
%   5.0000e-02  -2.1257e-02   2.0807e-02  -2.2510e-04
%   2.5000e-02  -1.0574e-02   1.0462e-02  -5.6280e-05
%   1.2500e-02  -5.2732e-03   5.2451e-03  -1.4070e-05
%   6.2500e-03  -2.6331e-03   2.6261e-03  -3.5176e-06
```

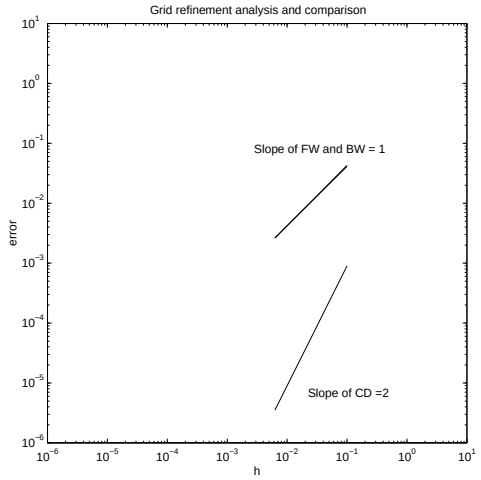


Figure 2.3. *A plot of a grid refinement analysis of the forward, backward, and central finite difference formulas for* $u'(x)$ *using* log-log *plot. The curves for forward and backward finite difference are almost identical and have slope one. The central formula is second order accurate and the slope of the plot is two.*

2.3 Deriving finite difference formulas using the method of un-determined coefficients

Suppose we want to use a one-sided finite difference to approximate $u'(x)$ at $\bar{x} = b$ which is a boundary using $u(\bar{x})$, $u(\bar{x} - h)$, $u(\bar{x} - 2h)$ with second order accuracy. We can use the method of un-determined coefficients. Let

$$u'(\bar{x}) \sim \gamma_1 u(\bar{x}) + \gamma_2 u(\bar{x} - h) + \gamma_3 u(\bar{x} - 2h).$$

We can use the Taylor expansion at $\bar{x}$ to get a system of equations for the coefficients

$$\begin{aligned}
&\gamma_1 u(\bar{x}) + \gamma_2 u(\bar{x} - h) + \gamma_3 u(\bar{x} - 2h) = \\
&\gamma_1 u(\bar{x}) + \gamma_2 \left((u(\bar{x}) - h u'(\bar{x}) + \frac{h^2}{2} u''(\bar{x}) - \frac{h^3}{6} u'''(\bar{x}) \right) + \\
&\gamma_3 \left((u(\bar{x}) - 2h u'(\bar{x}) + \frac{4h^2}{2} u''(\bar{x}) - \frac{8h^3}{6} u'''(\bar{x}) \right) + O(\max |\gamma_k| h^4)
\end{aligned}$$

The linear combination should approximate $u'(\bar{x})$. So we should set

$$\begin{aligned}
\gamma_1 + \gamma_2 + \gamma_3 &= 0 \\
-h\gamma_2 - 2h\gamma_3 &= 1 \\
h^2\gamma_2 + 2h^2\gamma_3 &= 0.
\end{aligned}$$

It is easy to check that the solution to the linear system above is

$$\gamma_1 = \frac{3}{2h}, \qquad \gamma_2 = -\frac{2}{h}, \qquad \gamma_3 = \frac{1}{2h}.$$

Therefore we get a one-sided finite difference scheme

$$u'(\bar{x}) = \frac{3}{2h}\,u(\bar{x}) - \frac{2}{h}\,u(\bar{x}-h) + \frac{1}{2h}\,u(\bar{x}-2h) + O(h^2). \tag{2.17}$$

Similarly we can get another one-sided finite difference formula by setting $h = -h$ in the formula above

$$u'(\bar{x}) = -\frac{3}{2h}\,u(\bar{x}) + \frac{2}{h}\,u(\bar{x}+h) - \frac{1}{2h}\,u(\bar{x}+2h) + O(h^2). \tag{2.18}$$

One can also differentiate a polynomials interpolation to get finite difference scheme. For example, given a sequence points $(x_i, u(x_i))$, $i = 0, 1, 2, \cdots, n$. Let the Lagrange interpolation polynomial be

$$p_n(x) = \sum_{j=0}^{n} l_j(x)u(x_j), \quad \text{where} \quad l_j(x) = \prod_{i\neq j}^{n} \frac{(x-x_j)}{(x_i-x_j)}.$$

Then $u'(\bar{x})$ can be approximated by

$$u'(\bar{x}) \sim p_n'(\bar{x}) = \sum_{j=0}^{n} l_j'(\bar{x})u(x_j).$$

2.3.1 Finite difference formula for second and high order derivatives

We can apply the finite difference operators twice to get finite difference formulas for approximating the second order derivative $u''(\bar{x})$. For example, the central finite difference formula for approximating $u''(\bar{x})$ can be obtained from

$$\begin{aligned}
\Delta_+\Delta_- u(\bar{x}) &= \Delta_+ \frac{u(x)-u(x-h)}{h} \\
&= \frac{1}{h}\left(\frac{u(x+h)-u(x)}{h} - \frac{u(x)-u(x-h)}{h}\right) \\
&= \frac{u(x-h)-2u(x)+u(x+h)}{h^2} = u''(\bar{x}) + O(h^2) \\
&= \Delta_-\Delta_+ u(\bar{x}) = \delta^2 u(\bar{x}) = \delta_u^2(\bar{x}).
\end{aligned} \tag{2.19}$$

If we use the same finite difference operator Δ_+ twice, then we get a one-sided finite difference formula for approximating $u''(\bar{x})$

$$\begin{aligned}
\Delta_+\Delta_+ u(\bar{x}) &= (\Delta_+)^2 u(\bar{x}) = \Delta_+ \frac{u(x+h)-u(x)}{h} \\
&= \frac{1}{h}\left(\frac{u(x+2h)-u(x+h)}{h} - \frac{u(x+h)-u(x)}{h}\right) \\
&= \frac{u(x)-2u(x+h)+u(x+2h)}{h^2} = u''(\bar{x}) + O(h),
\end{aligned} \tag{2.20}$$

which is only first order accurate.

In a similar way, we can use the finite difference operators to derive a finite difference formula for a cross derivative,

$$\delta_x \delta_y u(\bar{x}, \bar{y}) =$$

$$\frac{u(\bar{x}+h, \bar{y}+h) + u(\bar{x}-h, \bar{y}-h) - u(\bar{x}+h, \bar{y}-h) - u(\bar{x}-h, \bar{y}+h)}{4h^2}$$

$$\approx \frac{\partial^2 u}{\partial x \partial y}(\bar{x}, \bar{y})$$

with a uniform step size in both x and y directions. Here we use δ_x to represent the central finite difference operator in the x direction.

2.3.2 Finite difference formula for third order derivatives

We can either apply the lower order finite difference formulas or use the method of un-determined coefficients to obtain finite difference formulas for approximating third order derivatives. For example, we have

$$\begin{aligned}
\Delta_+ \delta^2 u(\bar{x}) &= \Delta_+ \frac{u(\bar{x}-h) - 2u(\bar{x}) + u(\bar{x}+h)}{h^2} \\
&= \cdots \\
&= \frac{u(\bar{x}-h) + 3u(\bar{x}) - 3u(\bar{x}+h) + u(\bar{x}+2h)}{h^3} \\
&= u'''(\bar{x}) + \frac{h}{2} u^{(4)}(\bar{x}) + \cdots
\end{aligned}$$

So the finite difference formula is first order accurate. If we use the central formula below

$$\frac{-u(\bar{x}-2h) + 2u(\bar{x}-h) - 2u(\bar{x}+h) + u(\bar{x}+2h)}{2h^3} = u'''(\bar{x}) + \frac{h^2}{4} u^{(5)}(\bar{x}) + \cdots,$$

then we can have a second order accurate scheme.

In practice, we seldom need more than fourth order derivatives. For high order differential equations, we can convert them to first order systems.

2.4 Consistency, stability, convergence, and error estimates of finite difference methods

When we use a finite difference method to solve a differential equation, we need to know how accurate the approximate solution is compared to the true solution.

2.4.1 Definition of the global error

Let $\mathbf{U} = [U_1, U_2, \cdots U_n]^T$ be the solution of the finite difference scheme (assume no round-off errors), and $\mathbf{u} = [u(x_1), u(x_2), \cdots, u(x_n)]$ be the exact solution at grid

points, $x_1, x_2, \cdots, x_n$. The global error vector is defined as $\mathbf{E} = \mathbf{U} - \mathbf{u}$. Naturally, we wish to give a smallest upper bound for the error vector. Usually we can use different norms.

- The maximum norm or the infinity norm $\|\mathbf{E}\|_\infty = \max_i\{|e_i|\}$. Usually this is regarded as the strongest measurement. If error is large at one grid point, then the maximum norm is also large.
- The 1-norm is one of average norms. It is defined as $\|\mathbf{E}\|_1 = \sum_i h_i|e_i|$ which is analogue to the continuous space $\int |e(x)|\,dx$.
- The 2-norm is another average norm. It is defined as $\|\mathbf{E}\|_2 = (\sum_i h_i|e_i|^2)^{1/2}$ which is analogue to the continuous space $(\int |e(x)|^2\,dx)^{1/2}$.

If $\|E\| \le Ch^p$, $p > 0$, we call the finite difference method is ***p-th order accurate***. Naturally, we wish to have reasonably high order method while keep the computational cost low.

Definition 2.1. *A finite difference method is called* ***convergent*** *if* $\lim_{h\to 0} \|\mathbf{E}\| = 0$.

2.4.2 Definition of the local truncation error

The intuitive definition of the local truncation error is the differential equation and the finite difference equation at a grid points. For example, for the two-point boundary value problem

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

the local truncation error of the finite difference scheme

$$\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} = f(x_i)$$

is

$$T_i = \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2} - f(x_i), \qquad i = 1, 2, \cdots, n-1.$$

Technically, we can get the local truncation error by moving the right hand side to the left, arranging/rewriting terms of the finite difference equation in a way that resembles the original differential equation; and substituting U_i with the true solution $u(x_i)$. Below we give a more rigorous definition.

Let $P(\frac{d}{dx})$ be a differential operator (take off u from the linear differential equation). Below are some examples:

- Given $u''(x) = f(x)$, then $P(\frac{d}{dx}) = \frac{d^2}{dx^2}$, and $Pu = f$.
- If $P(\frac{d}{dx}) = \frac{d^3}{dx^3} + a(x)\frac{d^2}{dx^2} + b(x)\frac{d}{dx} + c(x)$, then $Pu = f$ stands for the differential equation: $u''' + au'' + bu' + cu = f(x)$,

Let P_h be a finite difference operator. For example, for the second order differential equation $u''(x) = f(x)$, one of the finite difference operator is

$$P_h u(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}.$$

More examples will be given later.

The local truncation error is defined as

$$T(x) = P_h u - Pu. \tag{2.21}$$

Note that we use the exact solution in the definition above. For the differential equation $u''(x) = f(x)$ and the three-point central difference scheme, the local truncation error is

$$T(x) = P_h u - Pu = u''(x) - \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} \tag{2.22}$$

$$= \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} - f(x). \tag{2.23}$$

Note that the local truncation error only depends on the solution in the finite difference stencil (it is three-point in the example) but do not depends on the solution globally (far away). That is one of reasons that is called the *local* truncation error. The local truncation errors measure how well the finite difference discretization approximates the differential equation.

Definition 2.2. *A finite difference scheme is called* ***consistent*** *if*

$$\lim_{h \to 0} T(x) = \lim_{h \to 0} (P_h u - Pu) = 0 \tag{2.24}$$

Usually, we should use a consistent finite difference scheme at least for the finite difference scheme at most of grid points.

If $|T(x)| \leq Ch^p$, $p > 0$, then we say the discretization is p-th order accurate, where C $O(1)$ is an order one quantity that depends on the solution $u(x)$. To check whether a finite difference scheme is consistent or not, we usually use the Taylor expansion to expand all the terms at the master grid point x_i. For example, for the three point central finite difference scheme for $u''(x) = f(x)$, we have

$$T(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} - u''(x) = \frac{h^2}{12} u^{(4)}(x) + h.o.t. = O(h^2).$$

From this we get that $|T(x)| \leq Ch^2$, where $C = \max_{0 \leq x \leq 1} |\frac{1}{12} u^{(4)}(x)|$. Therefore the finite difference scheme is consistent and the discretization is second order accurate.

Now let us exam another finite difference scheme for $u''(x) = f(x)$. The finite difference scheme is

$$\frac{u(x_i) - 2u(x_{i+1}) + u(x_{i+2})}{h^2} = f(x_i), \quad i = 1, 2, \cdots, n-2,$$

$$\frac{u(x_{n-2}) - 2u(x_{n-1}) + u(b)}{h^2} = f(x_{n-1}).$$

The discretization at x_{n-1} is second order accurate since $T(x_{n-1}) = O(h^2)$. The local truncation error at other grid points

$$T(x_i) = \frac{u(x_i) - 2u(x_{i+1}) + u(x_{i+2})}{h^2} - f(x_i) = O(h).$$

At all grid points where the solution is unknown, we have $\lim_{h\to 0} T(x_i) = 0$. Thus the finite difference scheme is consistent. But if we implement the finite difference scheme, you may get weird results. This is because the finite difference scheme above does not use the boundary condition at $x = a$, which is apparently wrong.

So the consistency can not guarantee the convergence of a finite difference. We need another condition to determine whether a finite difference method converge or not. Such a condition is called the *stability* of a finite difference method.

For the model problem, we have

$$A\mathbf{u} = \mathbf{F} + \mathbf{T}, \qquad A\mathbf{U} = \mathbf{F}, \qquad A(\mathbf{u} - \mathbf{U}) = \mathbf{T} = -A\mathbf{E}, \tag{2.25}$$

where A is the coefficient matrix of the finite difference equations, $\mathbf{F}$ is the modified source term by taking into account the boundary condition, and $\mathbf{T}$ is the vector of the local truncation error at the grid points where the solution is unknown.

If A is non-singular, then $\|\mathbf{E}\| = \|A^{-1}\mathbf{T}\| \le \|A^{-1}\|\|\mathbf{T}\|$. If A is singular, then $\|E\|$ may be arbitrary large like the example above, which means that the finite difference method does not converge. For the central finite difference scheme, we have $\|\mathbf{E}\| \le \|A^{-1}\|h^2$. So the global error depends on both the local truncation error and $\|A^{-1}\|$.

Definition 2.3. *A finite difference method for the elliptic differential equation is stable if A is invertible and*

$$\|A^{-1}\| \le C, \quad \textit{for all } h < h_0, \tag{2.26}$$

where C and h_0 are two constants.

From the definition of the consistency and the stability, and the derivations above, we can easily get the following theorem.

Theorem 2.4. *A consistent and stable finite difference method is convergent.*

Usually it is relatively easier to prove the consistency, but it is more difficult, sometime impossible, to prove the stability.

Now we are ready to prove the convergence of the central finite difference scheme for $u''(x) = f(x)$. We need the following lemma.

Lemma 2.5. *Given a symmetric tridiagonal matrix $A \in R^{n\times n}$ whose main diagonals and off-diagonals are two constant, d and α. Then the eigenvalues of A are*

$$\lambda_j = d + 2\alpha\cos\left(\frac{\pi j}{n+1}\right), \quad j = 1, 2, \cdots n, \tag{2.27}$$

and the corresponding eigenvector

$$x_k^j = \sin\left(\frac{\pi k j}{n+1}\right), \quad k = 1, 2, \cdots n, \tag{2.28}$$

The Lemma can be proved by direct verification $A\mathbf{x}^j = \lambda_j \mathbf{x}^j$. Also noted that the eigenvectors $\mathbf{x}^j$ are orthogonal to each other in the 2-norm.

Theorem 2.6. *The central finite difference method for $u''(x) = f(x)$ with Dirichlet boundary condition is convergent and $\|E\|_\infty \leq \|E\|_2 \leq Ch^{3/2}$.*

Proof: Assume the matrix is $A \in R^{(n-1)\times(n-1)}$, then we have $d = -2/h^2$, $\alpha = 1/h^2$, the eigenvalues of A is

$$\lambda_j = -\frac{2}{h^2} + \frac{1}{h^2}\cos\left(\frac{\pi j}{n}\right) = \frac{2}{h^2}\left(\cos(\pi j h) - 1\right).$$

Note that the eigenvalues of A^{-1} are $1/\lambda_j$ and A^{-1} is also symmetric, we have

$$\|A^{-1}\|_2 = \frac{1}{\min|\lambda_j|} = \frac{h^2}{2(1-\cos(\pi h))} = \frac{h^2}{2(1-(1-(\pi h)^2/2 + (\pi h)^4/4! + \cdots)} < \frac{1}{\pi^2}.$$

Therefore, we have

$$\|\mathbf{E}\|_\infty \leq \|\mathbf{E}\|_2 \leq \|A^{-1}\|_2 \|\mathbf{T}\|_2 \leq \frac{1}{\pi^2}\sqrt{n-1}Ch^2 \leq \bar{C}h^{3/2}.$$

We can also prove that the infinity norm is also proportional to h^2 using the maximum principal, or the Green function approach, see [15].

Remark 2.1. *The eigenvectors and eigenvalues of the coefficient matrix in (2.1) can also be obtained by considering the eigenvalue problem of the Sturm-Luiville problem*

$$u''(x) + \lambda u = 0, \quad u(0) = u(1) = 0. \tag{2.29}$$

It is easy to check that the eigenvalues are

$$\lambda_k = (k\pi)^2, \quad k = 1, 2, \cdots. \tag{2.30}$$

The corresponding eigenvectors are

$$u_k(x) = \sin(k\pi x), \tag{2.31}$$

Its discrete form at grid point is

$$u_k(x_i) = \sin(k\pi i h), \quad i = 1, 2, \cdots, n-1. \tag{2.32}$$

This is one of the eigenvectors of the coefficient matrix in (2.1). The corresponding eigenvalue can be found using the definition $A\mathbf{x} = \lambda\mathbf{x}$.

2.4.3 The effect of round-off errors and the choice of machine precision.

From the knowledge of numerical linear algebra, we know that

- $\|A\|_2 = \max|\lambda_j| = \frac{2}{h^2}(1-\cos(\pi(n-1)h)) \sim \frac{4}{h^2} = 4n^2$. Therefore $\kappa(A) = \|A\|_2\|A^{-1}\|_2 \sim n^2$.
- The relative error of the computed solution $\mathbf{U}$ for a stable scheme satisfies

$$\begin{aligned}\frac{\|\mathbf{U}-\mathbf{u}\|}{\|\mathbf{u}\|} &\le \text{local truncation error} + \text{round-off error}\\ &\le \|A^{-1}\|\|\mathbf{T}\| + \bar{C}g(n)\|A\|\|A^{-1}\|\,\epsilon\\ &\le Ch^2 + \bar{C}g(n)\frac{1}{h^2}\,\epsilon\end{aligned}$$

where $g(n)$ is the growth factor of the algorithm for solving the linear system of equations, ϵ is the machine precision. For most computers, $\epsilon \sim 10^{-8}$ when we use the single precision, and $\epsilon \sim 10^{-16}$ for the double precision.

Usually the global error decreases as h decreases. With the presence of round-off errors, the error may actually increases as h decreases if h is too small. We can roughly estimate such a critical h. To make the discussion simple, assume that $C \sim O(1)$ and $g(n) \sim O(1)$, the critical h that we can expect is when the local truncation error is roughly the same as the round-off error, that is,

$$h^2 \sim \frac{1}{h^2}\epsilon, \quad \Longrightarrow \quad n \sim \frac{1}{h} = \frac{1}{\epsilon^{1/4}}$$

which is about 100 for the single precision, and $10,000$ for the double precision. Therefore, if we use the single precision, there is no point to take more than 100 grid points, and we can get roughly four significant digits at the best. That is why we usually use double precision to solve boundary value problems. Note that Matlab is using double precision by default.

2.5 Finite difference methods for 1-D self-adjoint elliptic equations.

Consider the one dimensional self-adjoint elliptic equations of the form

$$(p(x)u'(x))' - q(x)u(x) = f(x), \quad a < x < b, \tag{2.33}$$

$$u(a) = u_a, \quad u(b) = u_b, \quad \text{or other BC.} \tag{2.34}$$

The existence and uniqueness of the solution is given in the following theorem.

Theorem 2.7. *If $p(x) \in C^1(a,b)$, $q(x) \in C^0(a,b)$, $f(x) \in C^0(a,b)$, $q(x) \ge 0$, and there is a positive constant such that $p(x) \ge p_0 > 0$, then there is unique solution $u(x) \in C^2(a,b)$.*

In the theorem, $C^0(a,b)$ is the space of all continuous functions in $[a, b]$; $C^1(a,b)$ is the space of all functions that have continuous first order derivative in $[a, b]$; and so on. Note that, we have weaker conditions for finite element methods where the integral forms are used. The proof of the theorem is usually given in advanced differential equations courses. Now let us focus on our discussion on the finite difference method for such a BVP assume that the solution exists. Again, the finite difference method contains the following steps.

Step 1: Generate a grid. For simplicity, we use a uniform Cartesian grid here

$$x_i = a + ih, \quad h = \frac{b-a}{n}, \quad i = 0, 1, \cdots n.$$

Note that $x_0 = a$, $x_n = b$. Sometimes an adaptive grid may be preferred, however, we may not use the central finite difference scheme for adaptive grids directly.

Step 2: Substitute derivatives with finite difference at each grid point that the solution is unknown. This step is also called the discretization. Define $x_{i+\frac{1}{2}} = x_i + h/2$. We have $x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}} = h$. Use the central finite difference formula at a typical grid point x_i with half grid size, we can get

$$\frac{p_{i+\frac{1}{2}}u'(x_{i+\frac{1}{2}}) - p_{i-\frac{1}{2}}u'(x_{i-\frac{1}{2}})}{h} - q_i u(x_i) = f(x_i) + E_i^1$$

where $p_{i+\frac{1}{2}} = p(x_{i+\frac{1}{2}})$, $q_i = q(x_i)$, $f_i = f(x_i)$, and $E_i^1 = Ch^2$. Apply the central finite difference scheme for the first order derivative further, we get

$$\frac{p_{i+\frac{1}{2}}\frac{u(x_{i+1})-u(x_i)}{h} - p_{i-\frac{1}{2}}\frac{u(x_i)-u(x_{i-1})}{h}}{h} - q_i u(x_i) = f(x_i) + E_i^1 + E_i^2,$$

for $i = 1, 2, \cdots n-1$.

The finite difference solution $U_i \approx u(x_i)$, is defined as the solution of the linear system of equations

$$\frac{p_{i+\frac{1}{2}}U_{i+1} - (p_{i+\frac{1}{2}} + p_{i-\frac{1}{2}})U_i + p_{i-\frac{1}{2}}U_{i-1}}{h^2} - q_i U_i = f_i, \tag{2.35}$$

for $i = 1, 2, \cdots n-1$. In the matrix-vector form, the linear system above can be

written as $A\mathbf{U} = \mathbf{F}$, where

$$A = \begin{bmatrix} -\frac{p_{1/2}+p_{3/2}}{h^2} - q_1 & \frac{p_{3/2}}{h^2} & & & \\ \frac{p_{3/2}}{h^2} & -\frac{p_{3/2}+p_{5/2}}{h^2} - q_2 & \frac{p_{5/2}}{h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & & \frac{p_{n-3/2}}{h^2} & -\frac{p_{n-3/2}+p_{n-1/2}}{h^2} - q_{n-1} \end{bmatrix},$$

$$\mathbf{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_{n-2} \\ U_{n-1} \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} f(x_1) - \frac{p_{1/2}u_a}{h^2} \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) - \frac{p_{n-1/2}u_b}{h^2} \end{bmatrix}$$

It is important to note that A is symmetric, and negative definite. A is also weakly diagonally dominant and an M-matrix. Those properties guarantee that A is non-singular.

Note that the differential equation can also be written as

$$p(x)u'' + p'(x)u' - qu = f(x).$$

This is called a non-conservative form. We can directly apply second order finite difference formulas to the equation above. What are advantages and dis-advantages of this approach?

- We need to find the derivative, or its approximation, of $p(x)$.
- The coefficient matrix of the finite difference equations is not symmetric, or negative positive definite, or diagonally dominant, anymore.

Therefore we should avoid to use the non-conservative form if possible.

The local truncation error of the conservative finite difference scheme is,

$$T_i = \frac{p_{i+\frac{1}{2}}u(x_{i+1}) - (p_{i+\frac{1}{2}} + p_{i-\frac{1}{2}})u(x_i) + p_{i-\frac{1}{2}}u(x_{i-1})}{h^2} - q_i u(x_i) - f_i. \quad (2.36)$$

Note that $P\left(\frac{\partial}{dx}\right) = \frac{d}{dx}\left(p\frac{d}{dx}\right) - qu$ is the differential operator. It is easy to show that $|T_i| \leq Ch^2$. It is more difficult to show that $\|A^{-1}\| \leq C$. However, we can use the maximum principal to prove second order convergence of the finite difference scheme.

2.6 Finite difference methods for general 1D elliptic equations

Consider the differential equation

$$p(x)u''(x) + r(x)u'(x) - q(x)u(x) = f(x), \quad a < x < b, \tag{2.37}$$

$$u(a) = u_a, \quad u(b) = u_b, \quad \text{or other BC.} \tag{2.38}$$

If $p(x) = 1$, $r(x) = 0$, and $q(x) \leq 0$, such an equation is called Helmholtz equation which is difficult to solve if $q(x) \leq 0$ and $|q(x)|$ is large, say $q(x) \sim 1/h^2$.

There are two different discretization techniques that we can use.

- Central finite difference discretization for all the derivatives. The finite difference scheme is

$$p_i \frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} + r_i \frac{U_{i+1} - U_{i-1}}{2h} - q_i U_i = f_i, \tag{2.39}$$

  for $i = 1, 2, \cdots n-1$, where $p_i = p(x_i)$ and so on. The advantage of this discretization is that the method is second order accurate. The disadvantage is that the coefficient matrix may not be diagonally dominant even if $q(x) \geq 0$ and $p(x) > 0$. The term $r(x)u'(x)$ sometimes is called the advection term if u is the velocity. When the advection is strong, that is $|r(x)|$ is large, the equation behaves like a wave equation, and the numerical solutions may have non-physical oscillations, say for example, when $r_i \sim 1/h$.

- A upwinding discretization for the first order derivative term and the central finite difference scheme for diffusion term.

$$p_i \frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} + r_i \frac{U_{i+1} - U_i}{h} - q_i U_i = f_i, \text{ if } r_i \geq 0,$$

$$p_i \frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} + r_i \frac{U_i - U_{i-1}}{h} - q_i U_i = f_i, \text{ if } r_i < 0.$$

  The purpose of this approach is to increase the diagonally dominance. The finite difference scheme is only first order accurate though. It is recommended if $|r(x)|$ is very large, say $|r(x)| \sim 1/h$. It is easier and more accurate to solve the linear system of equations with either direct method or iterative methods for diagonally dominant matrices.

2.7 The ghost point method for boundary condition involve derivatives

In this section, we discuss how to treat Neumann and mixed (Robin) boundary conditions. First we consider the differential equation

$$u''(x) = f(x), \quad a < x < b,$$

$$u'(a) = \alpha, \qquad u(b) = u_b$$

Note that the solution at $x = a$ is unknown. If we use a uniform Cartesian grid $x_i = a + ih$, U_0 is one of component of the solution. We still can use

$$\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} = f_i, \quad i = 1, 2, \cdots n-1.$$

But we need an additional equation at $x_0 = a$. Certainly we should use the Neumann boundary condition at $x = a$. One intuitive approach is

$$\frac{U_1 - U_0}{h} = \alpha, \quad \text{or} \quad \frac{-U_0 + U_1}{h^2} = \frac{\alpha}{h}. \tag{2.40}$$

It works and the linear system of equations is still tri-diagonal and symmetric negative definite:

$$\begin{bmatrix} -\frac{1}{h^2} & \frac{1}{h^2} & & & & \\ \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & & \\ & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \\ & & & & \frac{1}{h^2} & -\frac{2}{h^2} \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ \vdots \\ U_{n-2} \\ U_{n-1} \end{bmatrix} = \begin{bmatrix} \frac{\alpha}{h} \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) - \frac{u_b}{h^2} \end{bmatrix} \tag{2.41}$$

But the global error is only first order accurate if $\alpha \neq 0$.

To maintain second order accuracy, we use the ghost point method by adding a ghost grid point $x_{-1} = x_0 - h = a - h$ and extending the solution to the interval $[a - h, a]$. Now we can use the central finite difference scheme for the differential equation at all grid points where the solution is unknown, that is, $i = 0, 1, \cdots n$. We have n equations and $n + 1$ unknowns. The extra one is U_{-1}. We need one more equation to close the system. The additional equation is the central finite difference equation for the Neumann boundary condition:

$$\frac{U_1 - U_{-1}}{2h} = \alpha. \tag{2.42}$$

From the equation above, we can get $U_{-1} = U_1 - 2h\alpha$. Plug this into the finite difference equation for the differential equation at $x = a$, or x_0, we have

$$\begin{aligned} \frac{U_{-1} - 2U_0 + U_1}{h^2} &= f_0 \\ \frac{U_1 - 2h\alpha - 2U_0 + U_1}{h^2} &= f_0 \\ \frac{-U_0 + U_1}{h^2} &= \frac{f_0}{2} + \frac{\alpha}{h} \end{aligned}$$

The coefficient matrix of the finite difference equation is the exact the same as (2.41). The only difference is the first component in the right hand side vector.

Now the component is $f_0/2 + \alpha/2$ compared with $\alpha/2$ of the first order method. But the results obtained from the ghost point method is much better in the sense that we have a second order method versus a first order one.

In order to discuss the stability, we need to find the eigenvalues of the coefficient matrix. Again, we can associate the eigenvalues to the continuous problem:

$$u'' + \lambda u = 0, \quad u'(0) = 0, \quad u(1) = 0. \tag{2.43}$$

It is easy to show that the eigenvectors are

$$u_k(x) = \cos\left(\frac{\pi x}{2} + k\pi x\right) \tag{2.44}$$

corresponding to the eigenvalue $\lambda_k = (\pi/2 + k\pi)^2$.

We compare the two methods in Fig. 2.4. The differential equation is $u''(x) = f(x)$ with a Dirichlet boundary condition at $x = 0$, and a Neumann boundary condition at $x = 0.5$. The exact solution is $u(x) = \cos(\pi x)$. Fig. 2.4 (a) shows the grid refinement analysis. The error in the second order method is much smaller that that of the first order method. In Fig. 2.4 (b), we show the error plot of the ghost point method. Note that the error at $x = b$ is not zero anymore.

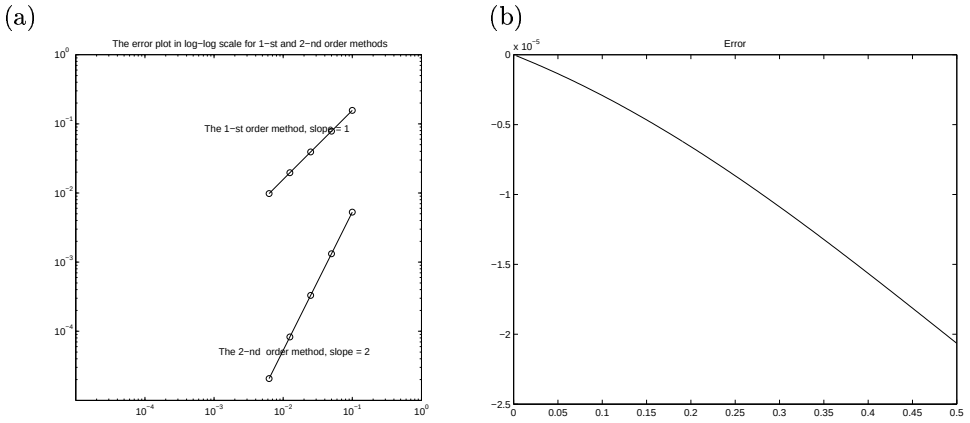


Figure 2.4. *(a) A grid refinement analysis of the ghost point method and the first order method. The slopes of the curves are the order of convergence. (b) The error plot of the computed solution from the ghost point method.*

2.7.1 A Matlab code of the ghost point method

```
function [x,U] = two_pointa(a,b,ua,uxb,f,n)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    This matlab function two_point solves the following two-point     %
%    boundary value problem: u''(x) = f(x) using the center difference %
```

```
%    scheme.                                                          %
%    Input:                                                           %
%     a, b: Two end points.                                           %
%     ua, uxb: Dirichlet and Neumann boundary conditions at a and b   %
%     f: external function f(x).                                      %
%     n: number of grid points.                                       %
%    Output:                                                          %
%     x: x(1),x(2),...x(n-1) are grid points                          %
%     U: U(1),U(2),...U(n) are approximate solution at grid points.   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

h = (b-a)/n; h1=h*h;

A = sparse(n,n);
F = zeros(n,1);

for i=1:n-1,
  A(i,i) = -2/h1; A(i+1,i) = 1/h1; A(i,i+1)= 1/h1;
end
  A(n,n) = -2/h1;
  A(n,n-1) = 2/h1;

for i=1:n,
  x(i) = a+i*h;
  F(i) = feval(f,x(i));
end
  F(1) = F(1) - ua/h1;
  F(n) = F(n) - 2*uxb/h;

U = A\F;

return
```

2.7.2 The Matlab driver program

Below is a Matlab driver code to solve the two-point boundary value problem

$$u''(x) = f(x), \quad a < x < b,$$
$$u'(a) = ua, \qquad u(b) = uxb.$$

```
%%%%%%%% Clear all unwanted variable and graphs.
```

```
clear;  close all

%%%%%%% Input

a =0; b=1/2;
ua = 1; uxb = -pi;

%%%%%% Call solver: U is

n=10;
k=1;

for k=1:5
    [x,U] = two_pointa(a,b,ua,uxb,'f',n);  %ghost-point method.
%    [x,U] = two_pointb(a,b,ua,uxb,'f',n);  %Backward Difference
  u=zeros(n,1);
  for i=1:n,
    u(i) = cos(pi*x(i));
  end

  h(k) = 1/n;
  e(k) = norm(U-u,inf); %%% Print out the maximum error.
  k = k+1; n=2*n;
end

loglog(h,e,h,e,'o'); axis('equal'); axis('square'),
%title('The error plot in log-log scale, the slope = 2'); % Ghost point
title('The error plot in log-log scale, the slope = 1'); % BW
figure(2); plot(x,U-u); title('Error')
```

2.7.3 Dealing with mixed (Robin) boundary conditions

The ghost point method can be used to discretize the mixed boundary condition, also called Robin boundary condition as well. Assume that at $x = a$, we have $\alpha u'(a) + \beta u(b) = \gamma$, where $\alpha \neq 0$. Then we can use the ghost point method to discretize the boundary condition

$$\alpha \frac{U_1 - U_{-1}}{2h} + \beta U_0 = \gamma,$$

$$\text{or} \quad U_{-1} = U_1 + \frac{2\beta h}{\alpha} U_0 - \frac{2h\gamma}{\alpha}.$$

Plugging this into the central finite difference equation at $x = x_0$ we get

$$\left(-\frac{2}{h^2} + \frac{2\beta}{\alpha h}\right) U_0 + \frac{2}{h^2} U_1 = f_0 + \frac{2\gamma}{\alpha h}, \tag{2.45}$$

$$\text{or} \quad \left(-\frac{1}{h^2} + \frac{\beta}{\alpha h}\right) U_0 + \frac{1}{h^2} U_1 = \frac{f_0}{2} + \frac{\gamma}{\alpha h}, \tag{2.46}$$

to get a symmetric coefficient matrix.

2.8 An example of a non-linear boundary value problem

By substituting the derivatives in a non-linear differential equation, we will have a non-linear algebraic system of equations. If we can solve the non-linear system, then we can get an approximate solution to the non-linear differential equation. We present an example in this section to illustrate the procedure.

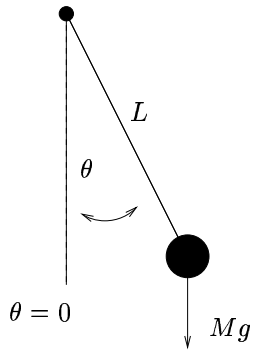


Figure 2.5. *A diagram of a pendulum.*

Consider the motion of a pendulum with mass M at the end of a rigid (but massless) bar of length L. Let $\theta(t)$ be the angle of the pendulum from vertical at time t, as illustrated in Fig. 2.5. Ignoring the mass of the bar, the friction forces, and the air resistance, the differential equation for the pendulum motion can be approximated by

$$M\mathbf{a} = -Mg \sin\theta,$$

where g is the gravitation constant. Since $S = L\theta$, where S is the arc, we have $\mathbf{v} = L\dot{\theta}$, and $\mathbf{a} = L\ddot{\theta}$, where $\mathbf{v}$ is the velocity, $\mathbf{a}$ is the acceleration. Therefore we have the following non-linear boundary value:

$$\ddot{\theta} = -\frac{g}{L} \sin\theta = K \sin\theta, \quad K = \frac{g}{L}, \tag{2.47}$$

with two reasonable boundary conditions $\theta(0) = \alpha$, say $\alpha = \pi/4$, and the observed angle at some time later $\theta(T) = \beta$.

If we apply the central finite difference scheme, then we obtain a system of non-linear equations of the following,

$$\frac{\theta_{i-1} - 2\theta_i + \theta_{i+1}}{h^2} + K \sin\theta_i = 0, \quad i = 1, 2, \cdots, n-1, \tag{2.48}$$

where $h = T/n$ is the step size. The non-linearity comes from $\sin\theta_i$.

There are several ways to solve the non-linear problems. We explain the ideas briefly for the pendulum problem.

- Use a linear model to approximate the non-linear problem. For example, we can approximate $\theta'' + K\sin\theta = 0$ using the linear equation $\theta'' + K\theta = 0$. This is valid if θ is small since $\sin\theta = \theta + O(\theta^3)$.

- Use a substitution method in which we approximate the non-linear term with the previous approximation. Given an initial guess $\theta^{(0)}(x)$, we form the iteration

$$(\theta^{k+1})'' + K\sin\theta^k = 0, \quad k = 0, 1, \cdots. \tag{2.49}$$

  At each iteration, we need to solve a linear two-point boundary value problem. The main concern of this approach is when the iterative method converge or not, and the rate of the convergence if the iterative method does converge.

- Solve the non-linear problem directly which is explained below.

In general, we will get a non-linear system of equations $\mathbf{F}(\mathbf{U}) = \mathbf{0}$ if we use finite difference method to discretize a non-linear ODE/PDE

$$\begin{cases} F_1(U_1, U_2, \cdots, U_n) = 0, \\ F_2(U_1, U_2, \cdots, U_n) = 0, \\ \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ F_n(U_1, U_2, \cdots, U_n) = 0. \end{cases} \tag{2.50}$$

A commonly used method is the Newton's method. Given an initial guess $\mathbf{U}^{(0)}$, we form the Newton's iteration

$$\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} - (J(\mathbf{U}^{(k)}))^{-1}\mathbf{F}(\mathbf{U}^{(k)}) \tag{2.51}$$

or

$$\begin{cases} J(\mathbf{U}^{(k)})\Delta\mathbf{U}^{(k)} = -\mathbf{F}(\mathbf{U}^{(k)}) \\ \mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + \Delta\mathbf{U}^{(k)}, \end{cases} \quad k = 0, 1, \cdots$$

where $J(\mathbf{U}^{(k)})$ is the Jacobi matrix defined as

$$\begin{bmatrix} \frac{\partial F_1}{\partial U_1} & \frac{\partial F_1}{\partial U_2} & \cdots & \frac{\partial F_1}{\partial U_n} \\ \frac{\partial F_2}{\partial U_1} & \frac{\partial F_2}{\partial U_2} & \cdots & \frac{\partial F_2}{\partial U_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial F_n}{\partial U_1} & \frac{\partial F_n}{\partial U_2} & \cdots & \frac{\partial F_n}{\partial U_n} \end{bmatrix}$$

For the pendulum problem, we have

$$J(\theta) = \frac{1}{h^2} \begin{bmatrix} -2 + h^2 K \cos\theta_1 & 1 & & \\ 1 & -2 + h^2 K \cos\theta_2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 + h^2 K \cos\theta_{n-1} \end{bmatrix}$$

We have implemented the Newton's method, say the Matlab code *pendulum.m*. In Fig. 2.6, we solve the pendulum problem with $\theta(0) = 0.7$, $\theta(2\pi) = 0.5$. In the left plot, we show the several approximate solutions of the Newton method. In the right plot, we show the approximate solutions using the linear and non-linear differential equations.

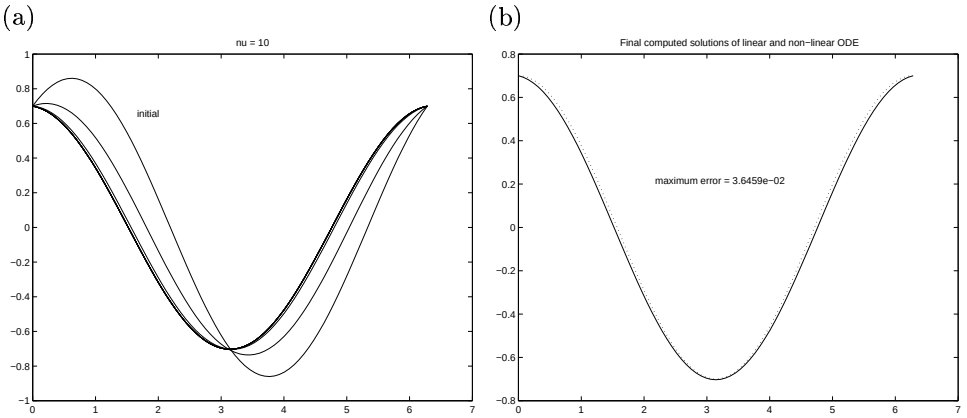


Figure 2.6. *(a) Plot of some intermediate solution to the non-linear system of equations. (b) Comparison of the solution of the linear and non-linear solution to the pendulum motion.*

It is not always easy to find $J(\mathbf{U})$ and it can be computationally expensive. Furthermore, Newton's method is called local convergent method because

it requires good initial guess to guarantee the convergence. Many methods, for example, quasi-Newton type methods, such as the Broyden and BFGS rank-one and rank-two update methods, conjugate gradient methods, can avoid evaluating the Jacobian matrix, The main issues include global convergence, the convergence order (Newton's method is quadratically convergent locally), storage etc. A well known software package called MINPACK is available through the netlib, see [12] for more complete discussions.

2.9 The grid refinement analysis technique

After we have learned or developed a numerical method, which includes convergence analysis such as consistence, stability, and order of convergence, computational complexity (operation counts, storage and other issues) etc, we need to validate and confirm the analysis numerically. We can also find out the algorithm behavior through numerically results. There are same ways to serve these purposes.

- Analyze of the output. Examine the boundary conditions, maximum/minimum values of the numerical solutions to see whether they agree with the ODE/PDE theory and intuition.
- Compare the numerical solutions with experiential data with sufficient variation of parameters.
- Do the grid refinement analysis with or without exact solutions.

Now we explain the grid refinement analysis with exact solutions. Assume a method is p-th order accurate, meaning that $\|E_h\| \sim Ch^p$ is h is small enough. We can divide h by half to get $\|E_{h/2}\|$. Then we have the following relations

$$\text{ratio} = \frac{\|E_h\|}{\|E_{h/2}\|} = \frac{Ch^p}{C(h/2)^p} = 2^p, \tag{2.52}$$

$$\log \|E_h\| = \log C + p \log h, \quad \text{and} \quad p = \frac{\log \left(\|E_h\| / \|E_{h/2}\| \right)}{\log 2}. \tag{2.53}$$

For a first order method ($p = 1$), the ratio approaches number two, and p approach number one. For a second order method ($p = 2$), the ratio approaches number four, and p approach number two, and so on. If p is some number between one and two, the method is called super-linear convergent.

For simple problems, we may come up with exact solution easily. For example, for most of linear single ODE/PDEs, we can simply set an exact solution $u_e(\mathbf{x})$, and from the exact solution, we can determine other functions and parameters such as the source term $f(\mathbf{x})$, boundary and initial conditions etc. For more complicated system of ODE/PDEs, the exact solution sometimes is difficult if not possible to construct. We can search the literature to see whether there are similar examples. Some examples in the literature have become benchmark problems. New methods may have to compare the results of the benchmark problems before they can be published.

If we do not have the exact solution, we still can estimate the order of convergence by compare the numerical solution with the one obtained from a fine mesh. Assume that the numerical solution satisfies

$$u_h = u_e + Ch^p + h.o.t. \tag{2.54}$$

where u_h is the numerical solution and u_e is the true solution. Let u_{h_*} be the solution obtained from the fine mesh,

$$u_{h_*} = u_e + C{h_*}^p + h.o.t. \tag{2.55}$$

Thus we get

$$u_h - u_{h_*} \approx C\left(h^p - {h_*}^p\right), \tag{2.56}$$

$$u_{h/2} - u_{h_*} \approx C\left((h/2)^p - {h_*}^p\right). \tag{2.57}$$

From the estimates above, we get the ratio

$$\frac{u_h - u_{h_*}}{u_{h/2} - u_{h_*}} \approx \frac{h^p - {h_*}^p}{(h/2)^p - {h_*}^p} = \frac{2^p\left(1 - (h_*/h)^p\right)}{1 - (2h_*/h)^p} \tag{2.58}$$

From this ratio, we can estimate the order of accuracy.

For example, if we double the number of grid points successively, *i.e.*,

$$\frac{h_*}{h} = 2^{-k}, \qquad k = 2, 3, \cdots, \tag{2.59}$$

then the ratio in (2.58) is

$$\frac{\tilde{u}(h) - \tilde{u}(h^*)}{\tilde{u}(\frac{h}{2}) - \tilde{u}(h^*)} = \frac{2^p\left(1 - 2^{-kp}\right)}{1 - 2^{p(1-k)}}. \tag{2.60}$$

In particular, for a first order method ($p = 1$), this becomes

$$\frac{\tilde{u}(h) - \tilde{u}(h^*)}{\tilde{u}(\frac{h}{2}) - \tilde{u}(h^*)} = \frac{2\left(1 - 2^{-k}\right)}{1 - 2^{1-k}} = \frac{2^k - 1}{2^{k-1} - 1}.$$

For $k = 2, 3, \cdots$, these ratios are

$$3, \quad \frac{7}{3} \simeq 2.333, \quad \frac{15}{7} \simeq 2.1429, \quad \frac{31}{15} \simeq 2.067, \cdots.$$

Similarly for a second order method ($p = 2$), (2.60) becomes

$$\frac{\tilde{u}(h) - \tilde{u}(h^*)}{\tilde{u}(\frac{h}{2}) - \tilde{u}(h^*)} = \frac{4\left(1 - 4^{-k}\right)}{1 - 4^{1-k}} = \frac{4^k - 1}{4^{k-1} - 1}.$$

For $k = 2, 3, \cdots$, the ratios are

$$5, \quad \frac{63}{15} = 4.2, \quad \frac{255}{63} \simeq 4.0476, \quad \frac{1023}{255} \simeq 4.0118, \cdots.$$

For one dimensional problems, we can take $n = 10, 20, 40, \cdots, 640$, depending on the size of the problem and the speed of the computers, to do the grid refinement analysis. For two dimensional problems, we can take $(10, 10)$, $(20, 20)$, $\cdots$, $(640, 640)$, or $(16, 16)$, $(32, 32)$, $\cdots$, $(512, 512)$ to do the grid refinement analysis, for three dimensional problems, we can take $(8, 8, 8)$, $(16, 16, 16)$, $\cdots$, $(128, 128, 128)$ to do the grid refinement analysis if we have enough memory.

To present the grid refinement analysis, we can tabulate the grid size n, the ratio and/or order so that we can see the order of convergence at the first glance. The other way is to plot the error versus the step size h in log-log scale, with the same scale on both x- and y- axis, the slope of the approximate line is the order of convergence.

2.10 Exercises

1. When we deal with irregular boundaries or use adaptive grids, we need to use non-uniform grids. Derive the finite difference coefficients for the following:

$$u'(\bar{x}) \approx \alpha_1 u(\bar{x} - h_1) + \alpha_2 u(\bar{x}) + \alpha_3 u(\bar{x} + h_2), \tag{2.61}$$
$$u''(\bar{x}) \approx \alpha_1 u(\bar{x} - h_1) + \alpha_2 u(\bar{x}) + \alpha_3 u(\bar{x} + h_2), \tag{2.62}$$
$$u'''(\bar{x}) \approx \alpha_1 u(\bar{x} - h_1) + \alpha_2 u(\bar{x}) + \alpha_3 u(\bar{x} + h_2). \tag{2.63}$$

   Are they consistent? In other words, as $h = \max\{h_1,\ h_2\}$ approaches zero, does the error also approach zero? If so, what are the order of accuracy? Do you see any potential problems with the schemes you have derived?

2. Consider the following finite difference scheme for solving the two point boundary value problem $u''(x) = f(x)$, $a < x < b$, $u(a) = u_a$ and $u(b) = u_b$.

$$\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} = f(x_i), \quad i = 2, 3, \cdots, n-1. \tag{2.64}$$

   where $x_i = a + ih$, $i = 0, 1, \cdots, n$, $h = (b-a)/n$. At $i = 1$, the finite difference scheme is

$$\frac{U_1 - 2U_2 + U_3}{h^2} = f(x_1). \tag{2.65}$$

   (a) Find the local truncation errors of the finite difference scheme. Is this scheme consistent?

   (b) Does this scheme converge? Justify your answer.

3. Program the central finite difference method for the self-adjoint BVP

$$(\beta(x)u')' - \gamma(x)u(x) = f(x), \quad 0 < x < 1,$$
$$u(0) = u_a, \quad au(1) + bu'(1) = c.$$

using a uniform grid and the central finite difference scheme

$$\frac{\beta_{i+\frac{1}{2}}(U_{i+1}-U_i)/h-\beta_{i-\frac{1}{2}}(U_i-U_{i-1})/h}{h}-\gamma(x_i)U_i=f(x_i). \quad (2.66)$$

Test your code for the case where

$$\beta(x)=1+x^2, \quad \gamma(x)=x, \quad a=2, \quad b=-3, \quad (2.67)$$

and the rest of functions or parameters are determined from the exact solution

$$u(x)=e^{-x}(x-1)^2. \quad (2.68)$$

Plot the computed solution and the exact solution, and the errors for a particular grid, say $n=80$. Do the grid refinement analysis to determine the order of accuracy of the global solution. Also try to answer the following questions:

- Can your code handle the case when $a=0$ or $b=0$?
- If we use the central difference scheme for the equivalent differential equation

$$\beta u''+\beta' u'-\gamma u=f, \quad (2.69)$$

what are the advantages or disadvantages?

4. Consider the finite difference scheme for the 1D steady state *convection-diffusion* equation

$$\epsilon u''-u'=-1, \quad 0<x<1 \quad (2.70)$$

$$u(0)=1, \quad u(1)=3. \quad (2.71)$$

(a) Verify the exact solution is

$$u(x)=1+x+\left(\frac{e^{x/\epsilon}-1}{e^{1/\epsilon}-1}\right). \quad (2.72)$$

(b) Compare the following two finite difference methods for $\epsilon=0.3$, 0.1, 0.05, and 0.0005, (1): Central difference scheme:

$$\epsilon\frac{U_{i-1}-2U_i+U_{i+1}}{h^2}-\frac{U_{i+1}-U_{i-1}}{2h}=-1. \quad (2.73)$$

(2): Central-upwind difference scheme:

$$\epsilon\frac{U_{i-1}-2U_i+U_{i+1}}{h^2}-\frac{U_i-U_{i-1}}{h}=-1. \quad (2.74)$$

Do grid refinement analysis for each case to determine the order of accuracy. Plot the computed solution and the exact solution for $h=0.1$, $h=1/25$, and $h=0.01$. You can use Matlab command *subplot* to put several graphs together.

(c) From you observation, give your opinion to see which method is better.

5. **Extra Credit:** Show that the finite difference method using the central formula for the BVP

$$u'' = f, \quad 0 < x < 1, \tag{2.75}$$

$$u(0) = 0, \quad u'(1) = \sigma, \tag{2.76}$$

and the *ghost point* method at $x = 1$ is stable and consistent. Find the convergence order and prove it.

6. Given a set of points (x_i, u_i), $i = 0, 1, \cdots N$, we can find the Lagrange interpolation polynomial from the formula

$$p(x) = \sum_{i=0}^{N} l_i(x)\, u_i, \qquad l_i(x) = \prod_{j=0, j\neq i}^{N} \frac{x - x_j}{x_i - x_j}. \tag{2.77}$$

By differentiating $p(x)$ with respect to x, we can get different finite difference formulas for approximating different derivatives. Assume that we have a uniform mesh, i.e. $x_{i+1} - x_i = x_i - x_{i-1} = \cdots = x_1 - x_0 = h$. Derive a central finite difference formula for $u^{(4)}$ and a one-sided finite difference formula for $u^{(3)}$.

7. Derive the finite difference method for

$$u''(x) - q(x)u(x) = f(x), \quad a < x < b, \tag{2.78}$$

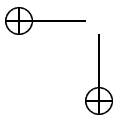
$$u(a) = u(b), \quad \text{periodic BC}, \tag{2.79}$$

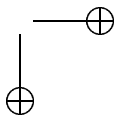
using the central difference scheme and a uniform grid. Write down the system of equations $A_h U = F$. How many unknowns are there without redundant? Is the coefficient matrix A_h tri-diagonal? **Hint:** Note that $U_0 = U_n$. Set unknowns as U_1, U_2, $\cdots$, U_n. If $q(x) = 0$, does the solution exist? Derive a compatibility condition for which the solution exists. If the solution exists, is it unique? How do we modify the finite difference method to make the solution unique?

8. Show that the central finite difference scheme for the BVP

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

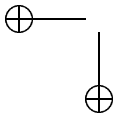
is second order accurate in the infinity norm via the Green function theory, see for example, [15].

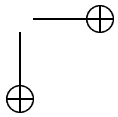




Part II

Finite Element Methods





Chapter 6

Finite element methods for one-dimensional elliptic problems

The finite element method was created to solve 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 book, we will focus on *elliptic type ODE/PDE boundary value problems.* We will start with the model equations

$$\text{1D:} \quad -u''(x) = f(x), \quad 0 < x < 1, \quad u(0) = u_0, \quad u(1) = u_1,$$

$$\text{2D:} \quad -(u_{xx} + u_{yy}) = f(x,y), \quad (x,y) \in \Omega, \quad u(x,y) = u_0(x,y), \quad (x,y) \in \partial\Omega,$$

where Ω is a domain in (x, y) plane.

6.1 An example of the finite element method for a model problem

We use a simple example to illustrate the procedures of the finite element method for solving the two-point boundary value problem,

$$-u''(x) = f(x), \quad 0 < x < 1, \quad u(0) = u_0, \quad u(1) = u_1.$$

To use a Galerkin finite element method to solve the problem, we carry out the following process.

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:

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

Integrating from 0 to 1, and using integration by parts we 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 mesh (interval). For example, we use a uniform Cartesian mesh $x_i = i\,h$, $i = 0, 1, \cdots, n$, $h = 1/n$, 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 start with a piecewise linear functions as defined by,

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

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

They are also called the hat functions. In later chapters, we will see other basis 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 $u_h(x)$:

$$\int_0^1 u_h' v' dx = \int_0^1 f v dx, \quad \text{The error comes in!}$$
$$\int_0^1 \sum_{j=1}^{n-1} c_j \phi_j' v' dx = \int_0^1 f v dx,$$
$$\sum_{j=1}^{n-1} c_j \int_0^1 \phi_j' v' dx = \int_0^1 f v dx.$$

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

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

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 $a(u,v)$ is called a bi-linear form since it is linear with each variable and (f,v) is called a lienar form,

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

If ϕ_i are the hat functions, then we can further get

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

5. Solve the linear system of equations $AU = F$ to get the coefficients and the the finite solution $u_h(x) = \sum_i c_i\phi_i(x)$.

6. Carry out the error analysis, sometimes, posterior 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?

6.2 Different mathematical formulations and equivalences for the 1D model

Let us consider the model,

$$\begin{aligned} -u''(x) &= f(x), \quad 0 < x < 1, \\ u(0) &= 0, \quad u(1) = 1. \end{aligned} \tag{6.1}$$

The problem can be reformulated into different forms.

1. (D)-form, the original differential equation.

2. (V)-form, the variational form or weak form:

$$\int_0^1 u'v'dx = \int_0^1 fvdx \tag{6.2}$$

for any testing function $v \in H_0^1(0,1)$, it is called the Sobolev space that will be explained later. It is like the C^1 space but in the integral form. The corresponding finite element method is often called the Galerkin method. *Question: What happened to the boundary condition?*

3. (M)-form, the minimization form:

$$\min_{v(x)\in H_0^1(0,1)} \left\{ \frac{1}{2}\int_0^1 \left((v')^2 - fv\right) dx \right\}. \tag{6.3}$$

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 as will be explained in the following subsections.

6.2.1 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 ends fixed and with an external force $f(x)$:

Let $u(x)$ be the position (displacement) 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

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

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

$$\tau\left(u_x(x+\Delta x)-u(x)\right)=-f(x)\Delta x$$

$$\text{or}\quad \tau\frac{u_x(x+\Delta x)-u(x)}{\Delta x}=-f(x)$$

$$\text{Let } \Delta x\to 0 \text{ to get}\quad -\tau u_{xx}=f(x).$$

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 $v(x)\in H_0^1$.

6.2.2 Mathematical equivalences

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

Theorem 6.1. *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, we have

$$\int_0^1 u'v'dx=u'v|_0^1-\int_0^1 u''vdx,$$

$$\Longrightarrow\quad -\int_0^1 u''vdx=\int_0^1 fvdx,$$

$$\text{or}\quad \int_0^1 (u''+f)\,vdx=0.$$

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

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

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

$$F(u^*)\le F(u)\quad \text{or}$$

$$\frac{1}{2}\int_0^1 (u^*)_x^2dx-\int_0^1 fu^*dx\le\frac{1}{2}\int_0^1 u_x^2dx-\int_0^1 fudx.$$

Proof:

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

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

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

Thus we have

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

and

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

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, that is, how many order derivatives of the solution can have. 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.

6.3 Procedure of the finite element method for the 1D model problem: Method and Programming

In this section, we will discuss the model problem (6.1) using the following methods,

- Galerkin method for the weak/variational form;
- Ritz method for the minimization form.

We will also discuss another important aspect about finite elements methods, that is, how to assemble the stiffness matrix using the element by element approach.

6.3.1 Procedure

The procedure is as follows.

Choose an integral form, say the weak form:

$\int_0^1 u'v'dx = \int_0^1 fvdx$ for any $v(x)$ in the Sobolev space $H_0^1(0,1)$ with $v(0) = v(1) = 0$. The Sobolev spaces and theory will be discussed in the next chapter.

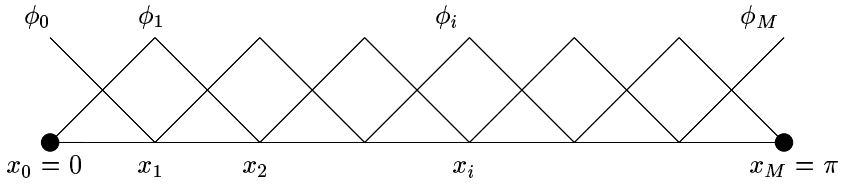


Figure 6.1. *Diagram of a triangulation in one dimension*

Generate a triangulation or mesh:

For a 1D problem, a mesh is a set of points in the interval, say, $x_0 = 0$, $x_1, x_2, \cdots x_M = 1$. 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.
- $h = \max\limits_{0 \le i \le M-1} \{ h_i \}$ is called a mesh size that measures how fine the partition is.

Define a finite dimensional space over the triangulation:

Let the solution be in the space V, for the model problem $V = H_0^1(0,1)$. We wish to construct a subspace $V_h \subset V$ based on the mesh,

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

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

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

where ϕ_1, ϕ_2, $\cdots$, ϕ_{M-1} are *linearly 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:

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

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

$$V_h = \Big\{ \; v_h(x), \quad v_h(x) \text{ is piecewise continuous linear over the triangulation}$$

$$v_h(0) = v_h(1) = 0, v_h(x) \in H_0^1(0,1) \Big\}.$$

It is easy to show that V_h has a finite dimension even although 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 uniquely determined by its values at x_i and x_{i+1} as below

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

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 $v_h(x_i) = l(x_i)$, $i = 1, \cdots, M-1$. 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 $M-1$. Because of such a relation, V_h is considered to be equivalent to R^{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, that is, zero almost everywhere except for a small region;
- meet the regularity requirement, that is, they have to be continuous, and differentiable except at nodal points.

The simplest ones are the hat functions:

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

They can be written in the simple form, $\phi_i(x_j) = \delta_i^j$, that is,

$$\phi_i(x_j) = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

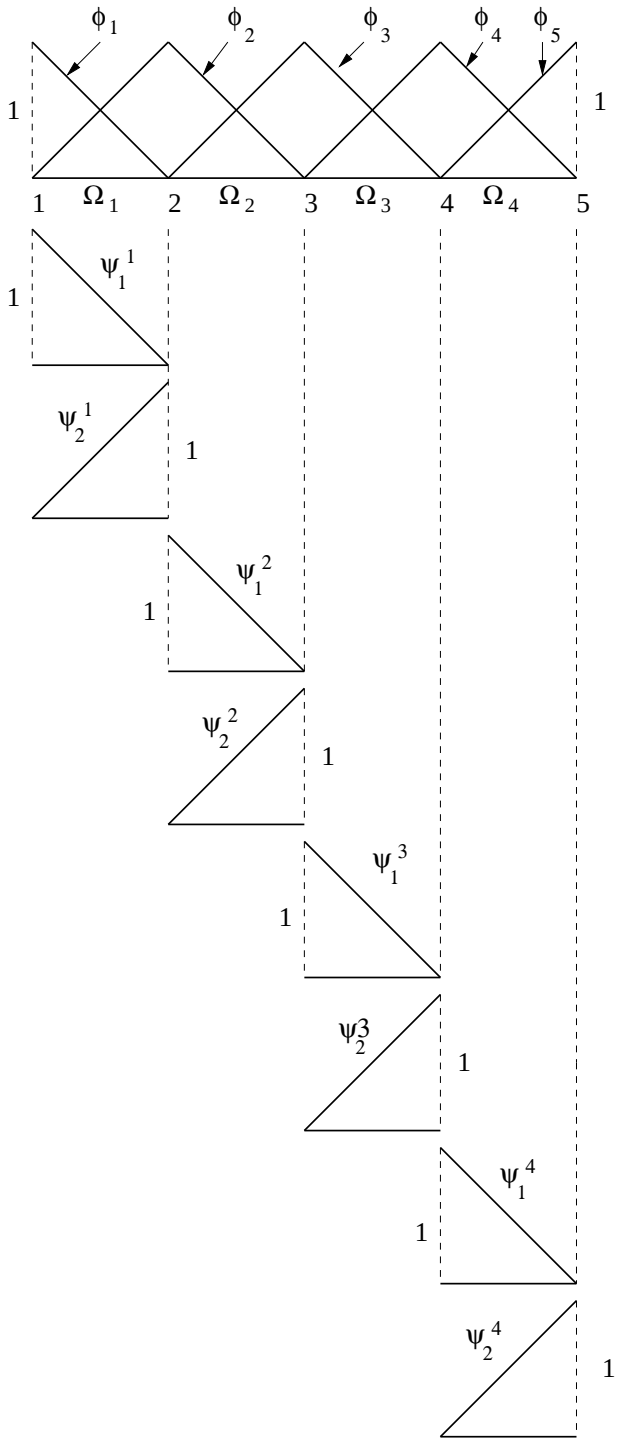


Figure 6.2. *Continuous piecewise linear basis functions ϕ_i for a 4-element mesh generated by linear shape functions ψ_1^e, ψ_2^e defined over each element. On each interior element, there are only two non-zero basis functions. The figure is adapted from [5].*

The analytic form is

$$\phi_i(x) = \begin{cases} 0, & \text{if } x < x_{i-1}, \\ \dfrac{x - x_{i-1}}{h_i}, & \text{if } x_{i-1} \le x < x_i, \\ \dfrac{x_{i+1} - x}{h_{i+1}}, & \text{if } x_i \le x < x_{i+1}, \\ 0, & \text{if } x_{i+1} \le x. \end{cases} \tag{6.5}$$

The finite element solution that we are looking for is

$$u_h(x) = \sum_{j=1}^{M-1} \alpha_j \phi_j(x). \tag{6.6}$$

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

$$u_h(x_i) = \sum_{j=1}^{M-1} \alpha_j \phi_j(x_i) = \alpha_i \phi_i(x_i) = \alpha_i. \tag{6.7}$$

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

6.3.2 The Ritz method

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

The minimization form for the model problem (6.1) is

$$\min_{v \in H_0^1(0,1)} F(v): \quad F(v) = \frac{1}{2} \int_0^1 (v_x)^2 dx - \int_0^1 f v dx. \tag{6.8}$$

As before, we look for an approximate solution of the form $u_h(x) = \sum_{j=1}^{M-1} \alpha_j \phi_j(x)$. If we plug this into the functional form above, we get

$$F(v_h) = \frac{1}{2} \int_0^1 \left(\sum_{j=1}^{M-1} \alpha_j \phi_j'(x) \right)^2 - \int_0^1 f \sum_{j=1}^{M-1} \alpha_j \phi_j(x) dx, \tag{6.9}$$

which is a multi-variable function $\alpha_1, \alpha_2, \cdots, \alpha_{M-1}$,

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

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

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

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,

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

Comparison of the Ritz and the Galerkin methods

For many problems, the two methods are equivalent theoretically.

- The Ritz method is based on the minimization form. It can be solved using optimization techniques.
- The Galerkin method 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.

6.3.3 Assembling the stiffness matrix element by element

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 the *stiffness matrix for the first order derivatives* $\{a_{ij}\} = \{\int \phi_i' \phi_j' \} dx$, and the right hand side F, often called the *load vector*. How to form A and F is a crucial part in the finite element method. For the model problem, one way to form the A and F is called *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 integrable 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

$$A = \begin{bmatrix} \int_0^1 (\phi_1')^2 dx & \int_0^1 \phi_1'\phi_2' dx & \cdots & \int_0^1 \phi_1'\phi_{M-1}' dx \\ \int_0^1 \phi_2'\phi_1' dx & \int_0^1 (\phi_2')^2 dx & \cdots & \int_0^1 \phi_2'\phi_{M-1}' dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_0^1 \phi_{M-1}'\phi_1' dx & \int_0^1 \phi_{M-1}'\phi_2' dx & \cdots & \int_0^1 (\phi_{M-1}')^2 dx \end{bmatrix}$$

$$= \begin{bmatrix} \int_{x_0}^{x_1} (\phi_1')^2 dx & \int_{x_0}^{x_1} \phi_1'\phi_2' dx & \cdots & \int_{x_0}^{x_1} \phi_1'\phi_{M-1}' dx \\ \int_{x_0}^{x_1} \phi_2'\phi_1' dx & \int_{x_0}^{x_1} (\phi_2')^2 dx & \cdots & \int_{x_0}^{x_1} \phi_2'\phi_{M-1}' dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_{x_0}^{x_1} \phi_{M-1}'\phi_1' dx & \int_{x_0}^{x_1} \phi_{M-1}'\phi_2' dx & \cdots & \int_{x_0}^{x_1} (\phi_{M-1}')^2 dx \end{bmatrix}$$

$$+ \begin{bmatrix} \int_{x_1}^{x_2} (\phi_1')^2 dx & \int_{x_1}^{x_2} \phi_1'\phi_2' dx & \cdots & \int_{x_1}^{x_2} \phi_1'\phi_{M-1}' dx \\ \int_{x_1}^{x_2} \phi_2'\phi_1' dx & \int_{x_1}^{x_2} (\phi_2')^2 dx & \cdots & \int_{x_1}^{x_2} \phi_2'\phi_{M-1}' dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_{x_1}^{x_2} \phi_{M-1}'\phi_1' dx & \int_{x_1}^{x_2} \phi_{M-1}'\phi_2' dx & \cdots & \int_{x_1}^{x_2} (\phi_{M-1}')^2 dx \end{bmatrix}$$

$$+ \cdots$$

$$+ \begin{bmatrix} \int_{x_{M-1}}^{x_M} (\phi_1')^2 dx & \int_{x_{M-1}}^{x_M} \phi_1'\phi_2' dx & \cdots & \int_{x_{M-1}}^{x_M} \phi_1'\phi_{M-1}' dx \\ \int_{x_{M-1}}^{x_M} \phi_2'\phi_1' dx & \int_{x_{M-1}}^{x_M} (\phi_2')^2 dx & \cdots & \int_{x_{M-1}}^{x_M} \phi_2'\phi_{M-1}' dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_{x_{M-1}}^{x_M} \phi_{M-1}'\phi_1' dx & \int_{x_{M-1}}^{x_M} \phi_{M-1}'\phi_2' dx & \cdots & \int_{x_{M-1}}^{x_M} (\phi_{M-1}')^2 dx \end{bmatrix}.$$

Note that for the hat basis functions, each interval has only *two* none-zero basis functions, therefore,

$$A = \begin{bmatrix} \int_{x_0}^{x_1}(\phi_1')^2dx & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} + \begin{bmatrix} \int_{x_1}^{x_2}(\phi_1')^2dx & \int_{x_1}^{x_2}\phi_1'\phi_2'dx & \cdots & 0 \\ \int_{x_1}^{x_2}\phi_2'\phi_1'dx & \int_{x_1}^{x_2}(\phi_2')^2dx & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & \int_{x_2}^{x_3}(\phi_2')^2dx & \int_{x_2}^{x_3}\phi_2'\phi_3'dx & \cdots & 0 \\ 0 & \int_{x_2}^{x_3}\phi_3'\phi_2'dx & \int_{x_2}^{x_3}(\phi_3')^2dx & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \int_{x_{M-1}}^{M}(\phi_{M-1}')^2dx \end{bmatrix}.$$

The none-zero contribution from a particular element is

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

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 also 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 the 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. We can concentrate on the contribution to the stiffness matrix and load vector on this element from the two non-zero hat functions. It is easy to verify the following:

$$\int_{x_i}^{x_{i+1}} (\psi_i')^2 dx = \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2} dx = \frac{1}{h_i}$$

$$\int_{x_i}^{x_{i+1}} \psi_i' \phi_{i+1}' dx \int_{x_i}^{x_{i+1}} -\frac{1}{h_i^2} dx = -\frac{1}{h_i}$$

$$\int_{x_i}^{x_{i+1}} (\psi_{i+1}')^2 dx = \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2} dx = \frac{1}{h_i}$$

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

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

The stiffness matrix A is assembled as follows,

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

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

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 6.1. *If we use a uniform triangulation $x_i = ih$, $h = 1/M$, $i = 0, 1, \cdots, M$ and approximate the following integral by the mid-point rule,*

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

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.

6.4 Matlab programming of the FEM for the 1D model problem

Matlab codes to solve the model problem:

$$-u''(x) = f(x), \quad a \le x \le b; \qquad u(a) = u(b) = 0. \tag{6.10}$$

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 $\int_{x1}^{x2} f(x) hat1 dx$ using the Simpson rule.
- $y = int\_hata2\_f(x1, x2)$ computes the integral $\int_{x1}^{x2} f(x) hat2 dx$ 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 at an arbitrary point xp in the solution domain.

6.4.1 Define the basis functions

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

$$\psi_1^e(x) = \frac{x - x_1}{x_2 - x_1} \tag{6.11}$$

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

$$\psi_2^e(x) = \frac{x_2 - x}{x_2 - x_1} \tag{6.12}$$

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
```

6.4.2 Define $f(x)$

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

6.4.3 The main FEM 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);           % Initialization
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
```

6.4.4 A test example

First we test the results with

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

The exact solution is

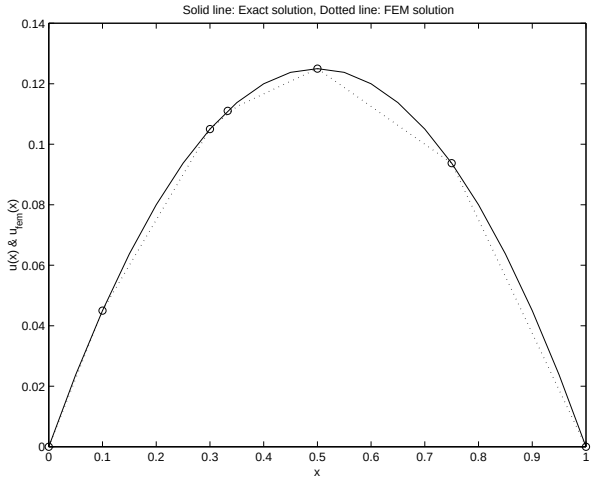
$$u(x) = \frac{x(1-x)}{2}. \tag{6.13}$$

A sample drive Matlab code is listed below:

```
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')
```

Figure 6.3 shows the plots produced by running the code. Figure 6.3 (a) shows both the true solution (the solid line) and the FEM solution (the dashed line). The little 'o's are the FEM solution at the nodal points which is the same as that of the true solution for this example. Figure 6.3 (a) shows the error between the true and the FEM solutions at a few selected points. We see that the error is zero at the nodal points. However, this may not be true for general problems.

(a)



(b)

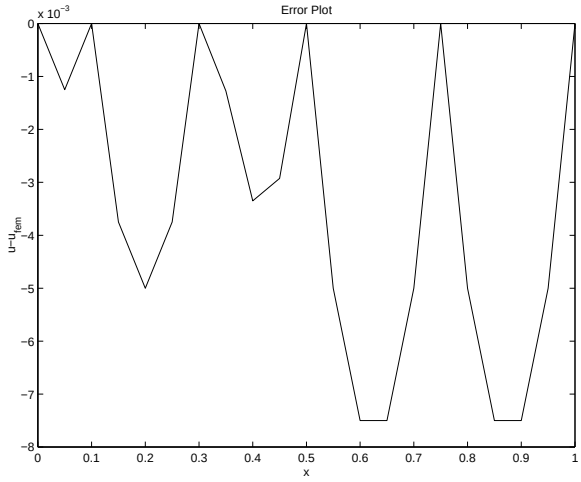


Figure 6.3. *(a) Plot of the true solution (solid line) and the FEM solution (the dashed line). (b): The error plot at some selected points.*

6.5 Exercises

1. This problem deals with the following boundary value problem:

$$-u''(x) + u(x) = f(x), \qquad u(0) = u(1) = 0.$$

   (a) Show that the weak form (variational form) is given by

$$(u', v') + (u, v) = (f, v), \quad \forall\, v(x) \in H_0^1(0,1),$$

   where

$$(u, v) = \int_0^1 u(x)v(x)dx,$$

$$H_0^1(0,1) = \{v(x), \quad v(0) = v(1) = 0, \ \int_0^1 v^2 dx < \infty\}.$$

   (b) Derive the linear system of the equations for the FEM approximation:

$$u_h = \sum_{j=1}^{3} \alpha_j \phi_j(x)$$

   with the following information:

   - $f(x) = 1$.
   - The nodal points and the elements are indexed as:

$$x_0 = 0, \quad x_2 = \frac{1}{4}, \quad x_3 = \frac{1}{2}, \quad x_1 = \frac{3}{4}, \quad x_4 = 1.$$
$$\Omega_1 = [x_3, x_1], \quad \Omega_2 = [x_1, x_4], \quad \Omega_3 = [x_2, x_3], \quad \Omega_4 = [x_0, x_2].$$

   - Use the hat function as the basis functions

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

     Do not re-order the nodal points and the elements.
   - Assemble the stiffness matrix and the load vector *element by element.*

2. This problem needs to modify *drive.m*, *f.m* and *soln.m*. Use the Matlab codes to solve

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

   Try two different triangulations: (a) the one given in *drive.m*; (b) the uniform grid $x_i = i * h$, $h = 1/M$, $i = 0, 1, \cdots, M$. Take $M = 10$. This can be done in Matlab using the command: $x = 0 : 0.1 : 1$.

   Use the grids to solve the problems for the following $f(x)$ or exact $u(x)$:

   (a) $u(x) = \sin(\pi x)$, what is $f(x)$?

   (b) $f(x) = x^3$, what is $u(x)$?

(c) (extra credit) $f(x) = \delta(x - 1/2)$, what is $u(x)$?

Make sure that the error is reasonably small.

3. This problem needs to modify *fem1d.m*, *drive.m*, *f.m* and *soln.m*. Assume we know that

$$\int_{x_i}^{x_{i+1}} \phi_i(x)\phi_{i+1}(x)dx = \frac{h}{6}$$

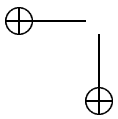
where $h = (x_{i+1} - x_i)/h$, ϕ_i and ϕ_{i+1} are the hat function centered at x_i and x_{i+1} respectively. Use the Matlab codes to solve

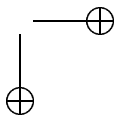
$$-u''(x) + u(x) = f(x), \qquad u(0) = u(1) = 0.$$

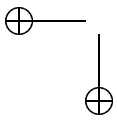
Try to use the uniform grid $x = 0 : 0.1 : 1$ in Matlab, for the following exact $u(x)$:

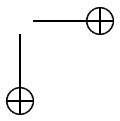
(a) $u(x) = \sin(\pi x)$, what is $f(x)$?

(b) $u(x) = x(1 - x)/2$, what is $f(x)$?









Chapter 7

Theoretical Fundamentals of FEM

Using the FE method, we need to answer these questions:

- What is the right functional space V for the solution?
- What is the right *weak* or *variational* form of a differential equation.
- What kind of basis functions or finite element spaces should we choose?
- How accurate is a FEM solution?

We try briefly to answer these equations in this Chapter. Remember that finite element methods are based on integral forms, not point-wise form as in finite difference methods. We need to generalize the theory corresponding to point-wise form to integral forms.

7.1 Functional space of $C^m(\Omega)$

Functional Space is a *set* of functions with operations. For example,

$$C(\Omega) = C^0(\Omega) = \Big\{ u(x), \quad u(x) \text{ is continuous on } \Omega \Big\} \tag{7.1}$$

is a linear space that contains all continuous functions on Ω. It is a linear space because for any real numbers α and β, we have,

$$\text{if} \quad u_1 \in C(\Omega), \qquad u_2 \in C(\Omega), \quad \text{then} \quad \alpha u_1 + \beta u_2 \in C(\Omega).$$

In the notation, Ω is a domain where the functions are defined, for example, $\Omega = [0,\ 1]$.

The functional space with first order continuous derivatives is defined as

$$C^1(\Omega) = \big\{ u(x), \quad u(x),\ u'(x) \text{ are continuous on } \Omega \big\}, \tag{7.2}$$

and similarly

$$C^m(\Omega) = \Big\{ u(x), \quad u(x), u'(x), \cdots, u^{(m)} \text{ are continuous on } \Omega \Big\}. \tag{7.3}$$

Obviously, we have,

$$C^0 \supset C^1 \supset \cdots \supset C^m \supset \cdots. \tag{7.4}$$

Let m go to the infinity, we define

$$C^{\infty}(\Omega) = \{u(x), \quad u(x) \text{ is indefinitely differentiable on } \Omega\}. \tag{7.5}$$

For examples, e^x, $\sin x$ and elementary functions are in C^{∞}.

7.1.1 High dimensional spaces and multi-index notations

Consider multi-dimensional functions $u(x_1, x_2, \cdots, x_n)$, or $u(\mathbf{x})$, $\mathbf{x} \in R^n$. We use multi-index notations to simplify the expressions of partial derivatives. Let $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$, $\alpha_i \geq 0$ be an integer vector in R^n. For example, if $n = 5$, $\alpha = (1, 2, 0, 0, 2)$ is one of α's. The muilti-index notation is used to express the following partial derivative

$$D^{|\alpha|}u(\mathbf{x} = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}, \quad |\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_n, \quad \alpha_i \geq 0. \tag{7.6}$$

Example 7.1. *When $n = 2$ and $u(\mathbf{x}) = u(x_1, x_2)$, all possible $D^{|\alpha|}u$ are*

$$\alpha = (2, 0), \quad D^{|\alpha|}u = \frac{\partial^2 u}{\partial x_1^2},$$
$$\alpha = (1, 1), \quad D^{|\alpha|}u = \frac{\partial^2 u}{\partial x_1 \partial x_2},$$
$$\alpha = (0, 2), \quad D^{|\alpha|}u = \frac{\partial^2 u}{\partial x_2^2}.$$

The C^m space can be defined as

$$C^m(\Omega) = \left\{u(x_1, x_2, \cdots, x_n), \quad D^{|\alpha|}u \text{ are continuous on } \Omega, \ |\alpha| \leq m\right\}. \tag{7.7}$$

That is all possible derivatives up to order m are continuous on Ω.

Example 7.2. *When $n = 2$, $m = 3$, then u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}, u_{xxx}, u_{xxy}, u_{xyy}, u_{yyy} all have to be continuous on Ω if $u \in C^3(\Omega)$. Note that $C^m(\Omega)$ has infinite dimensions.*

The **distance** in $C^0(\Omega)$ is defined as,

$$d(u, v) = \max_{x \in \Omega} |u(x) - v(x)| \tag{7.8}$$

A linear space with distance defined is called a *metric space*. A distance is an non-negative function of u and v that satisfies,

$$d(u, v) \geq 0, \quad d(u, v) = 0, \quad \text{iff} \quad u \equiv v. \tag{7.9}$$
$$d(u, v + w) \leq d(u, v) + d(u, w), \quad \text{triangle inequality.} \tag{7.10}$$

In the expressions above. the word *iff* means *if and only if*.

The **Norm** in $C^0(\Omega)$ is an non-negative function of u defined by

$$\|u(x)\| = d(u, \theta) = \max_{x \in \Omega} |u(x)|, \tag{7.11}$$

where θ is the zero element. A norm needs to satisfy

$$\|u(x)\| \geq 0, \quad \|u(x)\| = 0, \quad \text{iff} \quad u \equiv 0. \tag{7.12}$$

$$\|\alpha u(x)\| = |\alpha| \|u(x)\|, \quad \alpha \text{ is a number}, \tag{7.13}$$

$$\|u(x) + v(x)\| \leq \|u(x)\| + \|v(x)\|, \quad \text{triangle inequality}. \tag{7.14}$$

A linear space with a norm defined is called a *normed space*. In $C^m(\Omega)$, the distance and the norm are defined as

$$d(u, v) = \max_{0 \leq |\alpha| \leq m} \max_{x \in \Omega} \left| D^{|\alpha|} u(x) - D^{|\alpha|} v(x) \right|, \tag{7.15}$$

$$\|u(x)\| = \max_{0 \leq |\alpha| \leq m} \max_{x \in \Omega} \left| D^{|\alpha|} u \right|. \tag{7.16}$$

7.2 Spaces in integral forms – Sobolev spaces $H^m(\Omega)$

In analogue to point-wise $C^m(\Omega)$ spaces, we can define integral spaces $H^m(\Omega)$ called Sobolev spaces,

$$C^0 \longrightarrow L^2 = H^0, \qquad C^m \longrightarrow H^m.$$

The square-integrable space $H^0(\Omega) = L^(\Omega)$ is defined as

$$L^2(\Omega) = \left\{ u(x), \int_\Omega u^2(x) dx < \infty \right\}, \tag{7.17}$$

corresponding to the point-wise $C^0(\Omega)$ space. It is easy to see that

$$C(0, 1) = C^0(0, 1) \subset L^2(0, 1).$$

Example 7.3. *It is easy to verify that* $u(x) = 1/x^{1/4} \notin C^0$, *but*

$$\int_0^1 \left(\frac{1}{x^{1/4}} \right)^2 dx = \int_0^1 \frac{1}{\sqrt{x}} dx = 2 < \infty.$$

Therefore $u(x) \in L^2(0, 1)$.

The distance in $L^2(\Omega)$ is defined as

$$d(f, g) = \left\{ \int_\Omega |f - g|^2 \, dx \right\}^{1/2}. \tag{7.18}$$

Thus $L^2(\Omega)$ is a metric space. We say that $f \equiv g$ (identical) if $d(f, g) = 0$. The following two functions are identical in $L^2(-2, 2)$

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0, \end{cases} \qquad g(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases}$$

The norm in the $L^2(\Omega)$ space is defined as

$$\|u\|_{L^2} = \|u\|_0 = \left\{ \int_\Omega |u|^2 \, dx \right\}^{1/2}. \tag{7.19}$$

We can prove that these definitions satisfy the requirements of the properties requirements for the distance and the norm.

$L^2(\Omega)$ *is a complete space* means that any Cauchy sequence $\{f_n\}$ in L^2 has a limit in L^2. In other words, there is a $f \in L^2(\Omega)$ such that

$$\lim_{n\to\infty} \|f_n - f\|_{L^2} = 0, \quad \text{or} \quad \lim_{n\to\infty} f_n = f.$$

A Cauchy sequence is a sequence that satisfies the property: for any given positive number ϵ, there is an integer N, such that

$$\|f_n - f_m\|_{L^2} < \epsilon, \qquad \text{if} \quad m \geq N, \quad n \geq N.$$

A complete normed space is called a *Banach* space (a Cauchy sequence converges in terms of the norm). Therefore L^2 is a Banach space.

7.2.1 The inner product in L^2

In R^n, given any two vectors,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in R^n, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in R^n,$$

we know that the inner product is defined as

$$(x, y) = x^T y = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

Similarly, the inner product in the L^2 space is defined as

$$(f, g) = \int_\Omega f(x) g(x) dx, \quad f \in L^2(\Omega),\ g \in L^2(\Omega). \tag{7.20}$$

It satisfies the requirements of the definition of an inner product

$$\begin{aligned} (f, g) &= (g, f), \\ (\alpha f, g) &= (f, \alpha g) = \alpha (f, g) \quad \forall \alpha \in R, \\ (f, g + w) &= (f, g) + (f, w). \end{aligned}$$

for any $f \in L^2(\Omega)$ and $g \in L^2(\Omega)$. With the inner product, the weak form for the simple model problem

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

can be written as

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

and the minimization forms is

$$\min_{v\in H_0^1(0,1)} F(v), \quad F(v)=\frac{1}{2}(v',v')-(f,v).$$

The norm, distance, and the inner product in $L^2(\Omega)$ have the following relations

$$\|u\|=\sqrt{(u,u)}=d(u,\theta)=\left\{\int_\Omega |u|^2\,dx\right\}^{1/2}. \tag{7.21}$$

7.2.2 The Cauchy-Schwartz inequality in $L^2(\Omega)$

For a Hilbert space with a norm (u,v) and its resulted norm $\|u\|=\sqrt{(u,u)}$, the Cauchy-Schwartz inequality is the following,

$$|(u,v)|\le\|u\|\|v\|. \tag{7.22}$$

Below we list some examples of the Cauchy-Schwartz inequality:

$$\left|\sum_{i=1}^n x_iy_i\right|\le\left\{\sum_{i=1}^n x_i^2\right\}^{1/2}\left\{\sum_{i=1}^n y_i^2\right\}^{1/2};$$

$$\left|\sum_{i=1}^n x_i\right|\le\sqrt{n}\left\{\sum_{i=1}^n x_i^2\right\}^{1/2};$$

$$\left|\int_\Omega fg dx\right|\le\left\{\int_\Omega f^2dx\right\}^{1/2}\left\{\int_\Omega g^2dx\right\}^{1/2};$$

$$\left|\int_\Omega f dx\right|\le\left\{\int_\Omega f^2dx\right\}^{1/2}\sqrt{V}.$$

where V is the volume of Ω.

A proof of the Cauchy-Schwartz inequality

Noting that $(u,u)=\|u\|^2$, we construct a quadratic function of α given u and v as follows

$$f(\alpha)=(u+\alpha v,u+\alpha v)=(u,u)+2\alpha(u,v)+\alpha^2(v,v)\ge 0.$$

The quadratic function is non-negative, and therefore the discriminant satisfies

$$\Delta=b^2-4ac\le 0, \quad i.e. \quad 4(u,v)^2-4(u,u)(v,v)\le 0, \quad \text{or} \quad (u,v)^2\le(u,u)(v,v).$$

Taking square root from both sides we then have the Cauchy-Schwartz inequality.

A complete Banach space with an inner product defined is called a *Hilbert space*. So $L^2(\Omega)$ is a Hilbert space (linear space, inner product, complete).

Summary of definitions of spaces

The following diagram shows relations among the definitions of spaces.

Metric Space (distance) $\longrightarrow$ Normed Space (norm) $\longrightarrow$ Banach space (complete) $\longrightarrow$ Hilbert space (inner product).

7.2.3 The $L^p(\Omega)$ spaces

An $L^p(\Omega)$ space is defined as

$$L^p(\Omega) = \left\{ u(x), \int_\Omega |u(x)|^p \, dx < \infty \right\}. \tag{7.23}$$

The distance in $L^p(\Omega)$ is defined as

$$d(f,g) = \left\{ \int_\Omega |f-g|^p \, dx \right\}^{1/p}. \tag{7.24}$$

An $L^p(\Omega)$ is a metric and complete space, so it is also a Banach space. But it is not a Hilbert space because there is no inner product, and its resulting norm can be defined unless $p = 2$.

7.3 Sobolev spaces – related to derivatives in integral forms

Similar to $C^m(\Omega)$, we use $H^m(\Omega)$ to define function spaces with derivatives in integral forms. Such function spaces are part of Sobolev spaces.

If there is no derivative, then we define

$$H^0(\Omega) = L^2(\Omega) = \left\{ v(x), \int_\Omega |v|^2 dx < \infty \right\}. \tag{7.25}$$

7.3.1 The definition of the weak derivatives

We know that if $u(x) \in C^1(0,1)$, then for any function $\phi \in C^1(0,1)$, $\phi(0) = \phi(1) = 0$, we have

$$\int_0^1 u'(x)\phi(x) = u\phi \Big|_0^1 - \int_0^1 u(x)\phi'(x)dx = -\int_0^1 u(x)\phi'(x)dx. \tag{7.26}$$

Such a $\phi(x)$ is called a testing function in $C_0^1(0,1)$. Therefore we can define the weak derivative of $u(x) \in L^2(\Omega) = H^0(\Omega)$ as a function $v(x)$ that satisfies

$$\int_\Omega v(x)\phi(x)dx = -\int_\Omega u(x)\phi'(x)dx \tag{7.27}$$

for all $\phi(x) \in C_0^1(\Omega)$, $\phi(0) = \phi(1) = 0$. If such a function exists, it is called the first order weak derivative of $u(x)$ and it is denoted as $v(x) = u'(x)$.

Similarly, we can define the m-th order weak derivative of $u(x) \in H^0(\Omega)$ as a function $v(x)$ that satisfies

$$\int_\Omega v(x)\phi(x)dx = (-1)^m \int_\Omega u(x)\phi^{(m)}(x)dx \tag{7.28}$$

for all $\phi(x) \in C_0^m(\Omega)$. That is

$$\phi(x) \in C^m(\Omega), \phi(x) = \phi'(x) = \cdots = \phi^{(m-1)}(x) = 0, \quad \text{if} \quad x \in \partial\Omega.$$

If such a function $v(x)$ exists, it is called the m-th order weak derivative of $u(x)$ and it is denoted as $v(x) = u^{(m)}(x)$. High order weak derivatives are defined from lower order derivatives.

7.3.2 Definition of Sobolev spaces $H^m(\Omega)$

Having defined the first partial derivatives, we define the Sobolev space $H^1(\Omega)$ as

$$H^1(\Omega) = \left\{ v(x), \quad D^{|\alpha|} v \in L^2(\Omega),\ |\alpha| \le 1 \right\}. \tag{7.29}$$

For example, $H^1(a,b)$ is defined as

$$H^1(a,b) = \left\{ v(x), \quad a < x < b,\ \int_a^b v^2 dx < \infty;\ \int_a^b (v')^2 dx < \infty \right\}.$$

In two space dimensions, $H^1(\Omega)$ is defined as

$$H^1(\Omega) = \left\{ v(x,y), \quad v \in L^2(\Omega),\ \frac{\partial v}{\partial x} \in L^2(\Omega),\ \frac{\partial v}{\partial y} \in L^2(\Omega) \right\}.$$

Similarly, the Sobolev space $H^m(\Omega)$ is defined as

$$H^m(\Omega) = \left\{ v(\mathbf{x}), \quad D^{|\alpha|} v \in L^2(\Omega),\ |\alpha| \le m \right\}. \tag{7.30}$$

7.3.3 The inner product in H^m spaces

The inner product in $H^0(\Omega)$ is the same as that in $L^2(\Omega)$,

$$(u,v)_{H^0(\Omega)} = (u,v)_0 = \int_\Omega uv dx.$$

The inner product in $H^1(a,b)$ is defined as

$$(u,v)_{H^1(a,b)} = (u,v)_1 = \int_a^b \left(uv + u'v' \right) dx.$$

The inner product in $H^1(\Omega)$ of two variables is defined as

$$(u,v)_{H^1(\Omega)} = (u,v)_1 = \iint_\Omega \left(uv + \frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial v}{\partial y} \right) dxdy.$$

The inner product in $H^m(\Omega)$ of general dimensions is

$$(u,v)_{H^m(\Omega)} = (u,v)_m = \iint_\Omega \sum_{|\alpha| \le m} \left(D^{|\alpha|} u(\mathbf{x}) \right) \left(D^{|\alpha|} v(\mathbf{x}) \right) d\mathbf{x}. \tag{7.31}$$

The norm in $H^m(\Omega)$ of general dimensions is

$$\|u\|_{H^m(\Omega)} = \|u\|_m = \left\{ \int_\Omega \sum_{|\alpha| \le m} \left| D^{|\alpha|} u(\mathbf{x}) \right|^2 d\mathbf{x} \right\}^{1/2}. \tag{7.32}$$

Therefore $H^m(\Omega)$ is a Hilbert space. A norm can be defined from the inner product. For example, in $H^1(a,b)$, the H^1 norm is defined by

$$\|u\|_1 = \left\{ \int_a^b \left(u^2 + u'^2 \right) dx \right\}^{1/2}.$$

The *distance* in $H^m(\Omega)$ of general dimensions is then defined as

$$d(u,v)_m = \|u - v\|_m. \tag{7.33}$$

7.3.4 Relations between $C^m(\Omega)$ and $H^m(\Omega)$ –Sobolev embedding theorem

In one dimensional space, we have

$$H^1(\Omega) \subset C^0(\Omega), \quad H^2(\Omega) \subset C^1(\Omega), \quad \cdots, H^{1+j}(\Omega) \subset C^j(\Omega).$$

Theorem 7.1. *The Sobolev embedding theorem: If $2m > n$, then*

$$H^{m+j} \subset C^j, \quad j = 0, 1, \cdots, \tag{7.34}$$

where n is the dimension of the independent variables of the elements in the Sobolev space.

Example 7.4. *In two dimensional space, we have $n = 2$. The condition $2m > n$ means that $m > 1$. From the embedding theorem, we have*

$$H^{2+j} \subset C^j, \quad j = 0, \longrightarrow H^2 \subset C^0, \quad j = 1, \longrightarrow H^3 \subset C^1, \cdots \tag{7.35}$$

If $u(x, y) \in H^2$, which means u, u_x, u_y, u_{xx}, u_{xy}, and u_{yy} all belong to L^2, we can conclude that $u(x, y)$ is continuous, but u_x and u_y may not be continuous!

Example 7.5. *In three dimensional space, $n = 3$, $2m > n$ means $m > 3/2$. We have the same result as in 2D.*

$$H^{2+j} \subset C^j, \quad j = 0, \longrightarrow H^2 \subset C^0, \quad j = 1, \longrightarrow H^3 \subset C^1, \cdots \tag{7.36}$$

We regard the *regularity* of solutions as the degree of smoothness or lack of it of a class of problems measured in C^m and H^m. If $u(x) \in H^m$ or C^m, then the larger m is, the smoother of the function.

7.4 The FEM analysis for the 1D model problems

For the simple 1D model problem

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

we know that the weak form is

$$\int_0^1 u'v'dx = \int_0^1 fvdx \quad \text{or} \quad (u', u') = (f, v).$$

Since v is arbitrary, we can take $v = f$ or $v = u$, then we get

$$\int_0^1 u'v'dx = \int_0^1 u'^2 dx, \quad \int_0^1 fvdx = \int_0^1 f^2 dx.$$

So both u, u', f, v, and v' should belong to $L^2(0, 1)$. That is $u \in H^1(0, 1)$ and $v \in H^1(0, 1)$. So the solution is in $H^1(0, 1)$ Sobolev space and we should also take v in the same space. From the Sobolev embedding theorem, we also know that $H^1 \subset C^0$, which means the solution is continuous.

7.4.1 Conforming finite element methods

Definition: If the finite element space is a subspace of the solution space, then the finite element method is called a *conforming finite space*. For example, the piecewise linear function over a given a triangulation is a conforming finite element space for the model problem. The finite element method is called a *conforming finite element method*. We mainly discuss conforming finite element methods in this book.

If we put the boundary condition together, then we can define the solution space as

$$H_0^1(0,1) = \left\{ v(x), \quad v(0) = v(1) = 0,\ v \in H^1(0,1) \right\}. \tag{7.37}$$

When we look for a finite element solution in a finite dimensional space V_h, V_h should be a subspace of $H_0^1(0,1)$ for conforming finite element methods, that is,

$$\begin{aligned} V_h = \{ & v_h, \quad v_h(0) = v_h(1) = 0,\ v_h \text{ is continuous piecewise linear over the triangulation} \} \\ & \subset H^0(0,1). \end{aligned}$$

The finite element solution is in V_h, a subspace of $H_0^1(0,1)$. If the solution of the weak form is in $H_0^1(0,1)$ but not in V_h space, then an error is introduced as the result of substituting the solution space with the finite dimensional space. Nevertheless, the finite element solution is the best approximation in V_h in some norm as we will see later.

7.4.2 The FEM analysis for the one dimensional Sturm-Liouville problems

A one-dimensional Sturm-Liouville problem with Dirichlet boundary conditions at two ends has the following form,

$$\begin{aligned} -(p(x)u'(x))' + q(x)u(x) &= f(x), \quad x_l < x < x_r \\ u(x_l) = 0, &\qquad u(x_r) = 0, \\ p(x) \geq p_{min} > 0, &\qquad q(x) \geq q_{min} \geq 0, \\ p(x) \in C(x_l, x_r), &\qquad q(x) \in C(x_l, x_r). \end{aligned} \tag{7.38}$$

The condition on $p(x)$ and $q(x)$ is to guarantee the problem is well-posed, that is, the weak form has a uniques solution. Later we will see that if $f(x) \in L^2(x_l, x_r)$, $p(x) \in C(x_l, x_r)$, and $q(x) \in C(x_l, x_r)$ in addition to the conditions above, we can guarantee an uniques solution to the weak form of the problem. To derive the weak form, we multiply a testing function $v(x)$, $v(x_l) = v(x_r) = 0$ to the both sides of the equations above and integrate from x_l to x_r to get

$$\begin{aligned} \int_{x_l}^{x_r} \left(-(p(x)u')' + qu \right) v dx &= \int_{x_l}^{x_r} f v dx \\ -pu'v \Big|_{x_l}^{x_r} + \int_{x_l}^{x_r} \left(pu'v' + quv \right) dx &= \int_{x_l}^{x_r} f v dx \\ \longrightarrow \int_{x_l}^{x_r} \left(pu'v' + quv \right) dx = \int_{x_l}^{x_r} f v dx, &\quad \forall v \in H_0^1(x_l, x_r). \end{aligned}$$

7.4.3 The bilinear form

We define the bilinear form as

$$a(u,v) = \int_{x_l}^{x_r} \left(pu'v' + quv\right) dx. \tag{7.39}$$

We call it the bilinear form because it is linear for both u and v, that is

$$\begin{aligned} a(\alpha u + \beta w, v) &= \int_{x_l}^{x_r} \left(p(\alpha u' + \beta w')v' + q(\alpha u + \beta w)v\right) dx \\ &= \alpha \int_{x_l}^{x_r} \left(pu'v' + quv\right) dx + \beta \int_{x_l}^{x_r} \left(w'v' + qwv\right) dx \\ &= \alpha a(u,v) + \beta a(w,v), \end{aligned}$$

where α and β are scalars. Similarly

$$a(u, \alpha v + \beta w) = \alpha a(u,v) + \beta a(u,w)$$

Note that the bilinear form looks like an inner product, and it is a different form of an inner product. Also, if $p \equiv 1$ and $q \equiv 1$, then

$$a(u,v) = (u,v)_1.$$

Since $a(u,v)$ itself is an inner product, we can define the corresponding norm, which is called the *energy norm*:

$$\|u\|_a = \sqrt{a(u,u)} = \left\{ \int_{x_l}^{x_r} \left(pu'^2 + qu^2 \right) dx \right\}^{\frac{1}{2}}, \tag{7.40}$$

where the first term is called *the kinetic energy*, and the second term is called *the potential energy*. The Cauchy-Schwartz inequality implies $|a(u,v)| \le \|u\|_a \|v\|_a$.

Use the bilinear form, the notations are much simplified. The week form for Sturm-Liouville problem is

$$a(u,v) = f(v), \quad \forall v \in H_0^1(x_1, x_2). \tag{7.41}$$

The minimization form is

$$\min_{v \in H_0^1(x_l, x_r)} F(v) = \min_{v \in H_0^1(x_l, x_r)} \frac{1}{2} a(v,v) - (f,v). \tag{7.42}$$

Later we will see that all self-adjoint elliptic differential equations have this kind of weak form and minimization from. The finite element method using the Ritz form is the same as the Galerkin form.

7.4.4 The FEM for the 1D S-L problems using piecewise linear basis functions in H^1.

Given *any* finite dimensional space V_s whose basis are

$$\phi_1(x) \in H_0^1, \quad \phi_2(x) \in H_0^1, \quad \cdots, \quad \phi_M(x) \in H_0^1,$$

and

$$\begin{aligned} V_s &= span\{\phi_1, \phi_2, \cdots, \phi_M\} \\ &= \left\{ v_s, \quad v_s = \sum_{i=1}^{M} \alpha_i \phi_i \right\} \subset H_0^1(a,b) \end{aligned}$$

The Galerkin method is (using the weak form) is the following. Let the approximate solution is

$$u_s = \sum_{j=1}^{M} \alpha_j \phi_j. \tag{7.43}$$

The coefficients are chosen such that the weak form is satisfied

$$a(u_s, v_s) = (f, v_s), \quad \forall\, v_s \in V_h.$$

In other words, we enforce the weak form on V_s not on $H_0^1(a,b)$. This is where the error comes in.

For the finite element space, in order to force the weak form on the entire space, we just need to enforce it on the basis functions since *any* element in the space is a linear combination of the basis functions. Therefore we just need to enforce

$$a\left(u_s, \phi_i\right) = (f, \phi_i), \quad i = 1, 2, \cdots, M,$$

or

$$\begin{aligned} a\left(\sum_{j=1}^{M} \alpha_j\, \phi_j, \phi_i\right) &= (f, \phi_i), \quad i = 1, 2, \cdots, M, \\ \sum_{j=1}^{M} a(\phi, \phi_i)\, \alpha_j &= (f, \phi_i), \quad i = 1, 2, \cdots, M \end{aligned}$$

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

$$\begin{bmatrix} a(\phi_1,\phi_1) & a(\phi_1,\phi_2) & \cdots & a(\phi_1,\phi_M) \\ a(\phi_2,\phi_1) & a(\phi_2,\phi_2) & \cdots & a(\phi_2,\phi_M) \\ \vdots & \vdots & \vdots & \vdots \\ a(\phi_M,\phi_1) & a(\phi_M,\phi_2) & \cdots & a(\phi_M,\phi_M) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_M \end{bmatrix} = \begin{bmatrix} (f,\phi_1) \\ (f,\phi_2) \\ \vdots \\ (f,\phi_M) \end{bmatrix}$$

The linear system of equations has the following nice properties

- A is symmetric. That is $\{a_{ij}\} = \{a_{ji}\}$, or $A = A^T$ since $a(\phi_i, \phi_j) = a(\phi_j, \phi_i)$. Note this is only true for self-adjoint elliptic problems. In one dimensions, the second order ODE has the form

  $$-(pu')' + qu = f.$$

  The following equation

  $$-u'' + xu' = f$$

is not self-adjoint, the weak form for this equation is

$$(u', v') + (u', v) = (f, v)$$

We still can use the Galerkin method, but we will have terms like (ϕ_i', ϕ_j) which is different from (ϕ_j', ϕ_i). So the coefficient matrix is not symmetric anymore.

- A is positive definite. That is

  1. $x^T A x > 0$, if $x \neq 0$.
  2. All eigenvalues of A are positive.

  Proof: For any $\eta \neq 0$, we show that $\eta^T A \eta > 0$.

$$\begin{aligned}
\eta^T A \eta = \eta^T (A\eta) &= \sum_{i=1}^{M} \eta_i \sum_{j=1}^{M} a_{ij} \eta_j \\
&= \sum_{i=1}^{M} \eta_i \sum_{j=1}^{M} a(\phi_i, \phi_j) \eta_j \\
&= \sum_{i=1}^{M} \eta_i \sum_{j=1}^{M} a(\phi_i, \eta_j \phi_j) \\
&= \sum_{i=1}^{M} \eta_i \, a\left(\phi_i, \sum_{j=1}^{M} \eta_j \phi_j\right) \\
&= a\left(\sum_{i=1}^{M} \eta_i \phi_i, \sum_{j=1}^{M} \eta_j \phi_j\right) \\
&= a\left(v_s, v_s\right) = \|v_s\|_a^2 > 0
\end{aligned}$$

since $v_s = \sum_{i=1}^{M} \eta_i \phi_i \neq 0$ because η is a nonzero vector and ϕ_i are linear independent.

7.4.5 The local stiffness matrix and load vector using the hat basis functions

The local stiffness matrix using hat basis function is still 2 by 2 matrix that has the form

$$K_i^e = \begin{bmatrix} \int_{x_i}^{x_{i+1}} p\phi_i' 2 & \int_{x_i}^{x_{i+1}} p\phi_i' \phi_{i+1}' dx \\ \int_{x_i}^{x_{i+1}} p\phi_{i+1}' \phi_i' dx & \int_{x_i}^{x_{i+1}} p{\phi_{i+1}'}^2 dx \end{bmatrix} + \begin{bmatrix} \int_{x_i}^{x_{i+1}} q\phi_i^2 & \int_{x_i}^{x_{i+1}} q\phi_i \phi_{i+1} dx \\ \int_{x_i}^{x_{i+1}} q\phi_{i+1} \phi_i dx & \int_{x_i}^{x_{i+1}} q\phi_{i+1}^2 dx \end{bmatrix}$$

and the load vector is

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

The global stiffness matrix and load vector can be assembled by element by element approach.

7.5 Error analysis for FEM

Error analysis for finite element methods usually include two parts. One is error estimates for a given finite element space. The other one is the convergence analysis (a limiting process) that shows the finite element solution converges the true solution to the weak form in some norms as the mesh size h approaches zero. We first recall some notations and set-ups.

1. We are given a weak form $a(u,v) = L(v)$, $u \in V$, $v \in V$ with a known space V, which usually has infinite dimensions.
2. Let u be the solution of the weak form.
3. We denote V_h as a *finite dimensional* and subspace of V, i.e. $V_h \subset V$ (condition for a conforming FEM). It does not have to depend on h.
4. We denote u_h as the solution of the weak form in the subspace V_h.
5. We define $e_h = u(x) - u_h(x)$ as the global error. We want to find a sharp upper bound for $\|e_h\|$ using certain norms.

Note that error arises when we substitute the solution space with a finite dimensional space. In other words, the weak form is satisfied only in the sub-space V_h, not in the solution space V. However, we can prove that the solution that satisfies the weak form in the sub-space V_h is the best approximation to the exact solution u in the finite element space in the energy norm.

Theorem 7.2.

1. *u_h is the* projection *of u onto V_h in terms of energy inner product, that is*

$$u - u_h \perp V_h, \qquad or \quad u - u_h \perp \phi_i,\ i = 1, 2, \cdots, M, \tag{7.44}$$

$$a(u - u_h, v_h) = 0,\ \forall\, v_h \in V_h, \qquad or \quad a(u - u_h, \phi_i) = 0,\ i = 1, 2, \cdots, M \tag{7.45}$$

   assuming that $\{\phi_i\}$ are the basis functions.

2. *Best approximation in the energy norm:*

$$\|u - u_h\|_a \leq \|u - v_h\|_a, \quad \forall v_h \in V_h.$$

Proof:

$$\begin{aligned}
& a(u,v) = (f,v), \quad \forall v \in V, \\
\rightarrow\ & a(u,v_h) = (f,v_h), \quad \forall v_h \in V_h \quad \text{since} \quad V_h \subset V, \\
& a(u_h,v_h) = (f,v_h), \quad \forall v_h \in V_h, \quad \text{since} \quad u_h \quad \text{is the solution} \quad \text{in} \quad V_h, \\
\text{subtract} \rightarrow\ & a(u - u_h, v_h) = 0, \quad \text{or} \quad a(e_h, v_h) = 0, \quad \forall v_h \in V_h.
\end{aligned}$$

Now we prove that u_h is the best approximation in V_h.

$$\begin{aligned}
\|u - v_h\|_a^2 &= a(u - v_h, u - v_h) \\
&= a(u - u_h + u_h - v_h, u - u_h + u_h - v_h), \quad \text{Let} \quad w_h = u_h - v_h \in V_h, \\
&= a(u - u_h + w_h, u - u_h + w_h) \\
&= a(u - u_h, u - u_h + w_h) + a(w_h, u - u_h + w_h) \\
&= a(u - u_h, u - u_h) + a(u - u_h, w_h) + a(w_h, u - u_h) + a(w_h, w_h) \\
&= \|u - u_h\|_a^2 + 0 + 0 + \|w_h\|_a^2, \quad \text{since,} \quad a(e_h, u_h) = 0 \\
&\geq \|u - u_h\|_a^2.
\end{aligned}$$

That is

$$\|u - u_h\|_a \leq \|u - v_h\|_a.$$

Example: For the Sturm-Liouville problem, we have

$$\begin{aligned}
\|u - u_h\|_a^2 &\leq \int_a^b \left(p(u' - u_h')^2 + q(u - u_h)^2\right) dx \\
&\leq p_{max} \int_a^b (u' - u_h')^2 dx + q_{max} \int_a^b (u - u_h)^2 dx \\
&\leq \max\{p_{max}, q_{max}\} \int_a^b \left((u' - u_h')^2 + (u - u_h)^2\right) dx \\
&= C\|u - u_h\|_1^2,
\end{aligned}$$

where $C = \max\{p_{max}, q_{max}\}$. So we obtain

$$\begin{aligned}
&\|u - u_h\|_a \leq \hat{C}\|u - u_h\|_1, \\
&\|u - u_h\|_a \leq \|u - v_h\|_a \leq \bar{C}\|u - v_h\|_1.
\end{aligned}$$

7.5.1 Interpolation functions and error estimates.

Usually the solution is unknown, in order to get the error estimate, we need to choose a special $v_h^* \in V_h$, which is very close to the solution. Then we use the error estimate $\|u - u_h\|_a \leq \|u - v_h^*\|_a$. Usually we choose the *piecewise interpolation function*. That is another reason we choose the finite element space as piecewise linear, quadratic, or cubic functions over the given triangulation.

Linear piecewise interpolation function of one dimension.

Given a triangulation $x_0, x_1, x_2, \cdots, x_M$, the linear piecewise interpolation function is defined as $u_I(x)$:

$$u_I(x) = \frac{x - x_i}{x_{i-1} - x_i} u(x_{i-1}) + \frac{x - x_{i-1}}{x_i - x_{i-1}} u(x_i), \quad x_{i-1} \leq x \leq x_i.$$

It is obvious that $u_I(x) \in V_h$, where $V_h \subset H^1$, the set of continuous piecewise linear functions. Therefore

$$\|u - u_h\|_a \leq \|u - u_I\|_a.$$

Since $u(x)$ is unknown, $u_I(x)$ is unknown as well. However we know the *upper bound* of the interpolation functions.

Theorem 7.3. *Given a function $u(x) \in C^2[a,b]$ and a triangulation $x_0, x_1, x_2, \cdots, x_M$. The continuous piecewise linear function u_I has the following error estimates*

$$\|u - u_I\|_\infty = \max_{x\in[a,b]} |u(x) - u_I(x)| \le \frac{h^2}{8}\|u''\|_\infty, \tag{7.46}$$

$$\|u'(x) - u_I'(x)\|_{L^2(a,b)} \le h\sqrt{b-a}\|u''\|_\infty. \tag{7.47}$$

Proof: Let $\tilde{e_h} = u(x) - u_I(x)$, then

$$\tilde{e_h}(x_{i-1}) = \tilde{e_h}(x_i) = 0.$$

From the Rolle's theorem, there must be at least one point z_i between x_{i-1} and x_i, that is

$$\tilde{e_h}'(z_i) = 0.$$

Therefore

$$\begin{aligned}\tilde{e_h}'(x) &= \int_{z_i}^{x} \tilde{e_h}''(t)dt \\ &= \int_{z_i}^{x} \left(u''(t) - u_I''(t)\right) dt \\ &= \int_{z_i}^{x} u''(t)dt.\end{aligned}$$

We obtain the following error estimate

$$|\tilde{e_h}'(x)| \le \int_{z_i}^{x} |u''(t)|dt \le \|u''\|_\infty \int_{z_i}^{x} dt \le \|u''\|_\infty h,$$

and

$$\begin{aligned}\|\tilde{e_h}'\|_{L^(a,b)} = \|\tilde{e_h}'\|_0 &\le \left\{\|u''\|_\infty^2 \int_a^b h^2 dt\right\}^{\frac{1}{2}} \\ &\le \sqrt{b-a}\|u''\|_\infty h.\end{aligned}$$

Thus we have proved the second equality. To prove the first one, we can assume that $x_{i-1} + h/2 \le z_i \le x_i$, otherwise we can use the other half interval. We use the Taylor expansions to get:

$$\begin{aligned}\tilde{e_h}(x) &= \tilde{e_h}(z_i + x - z_i), \quad \text{assume} \quad x_{i-1} \le x \le x_i, \\ &= \tilde{e_h}(z_i) + \tilde{e_h}'(z_i)(x - z_i) + \frac{1}{2}\tilde{e_h}''(\xi)(x - z_i)^2, \quad x_{i-1} \le \xi \le x_i, \\ &= \tilde{e_h}(z_i) + \frac{1}{2}\tilde{e_h}''(\xi)(x - z_i)^2,\end{aligned}$$

Take $x = x_i$, we have

$$\begin{aligned} 0 = \tilde{e_h}(x_i) &= \tilde{e_h}(z_i) + \frac{1}{2}\tilde{e_h}''(\xi)(x_i - z_i)^2, \\ \tilde{e_h}(z_i) &= -\frac{1}{2}\tilde{e_h}''(\xi)(x_i - z_i)^2, \\ |\tilde{e_h}(z_i)| &\le \frac{1}{2}\|u''\|_\infty (x - z_i)^2 \le \frac{h^2}{8}\|u''\|_\infty. \end{aligned}$$

Note that the largest value of $\tilde{e_h}(x)$ has to be one of z_i's where the derivative is zero.

7.5.2 Error estimates of the finite element methods using the interpolation function.

For one dimensional Sturm-Liouville problem, we have

Theorem 7.4.

$$\|u - u_h\|_a \le Ch\|u''\|_\infty \tag{7.48}$$
$$\|u - u_h\|_1 \le \hat{C}h\|u''\|_\infty, \tag{7.49}$$

where C and $\hat{C}$ are two constants.

Proof:

$$\begin{aligned} \|u - u_h\|_a^2 &\le \|u - u_I\|_a^2 \\ &\le \int_a^b \left(p(u' - u_I')^2 + q(u - u_I)^2\right) dx \\ &\le \max\{p_{max}, q_{max}\} \int_a^b \left((u' - u_I')^2 + (u - u_I)^2\right) dx \\ &\le \max\{p_{max}, q_{max}\} \|u''\|_\infty^2 \int_a^b \left(h^2 + h^4/64\right) dx \\ &\le Ch^2 \end{aligned}$$

The second inequality is obtained because $\|\ \|_a$ and $\|\ \|_1$ are equivalent meaning that

$$c\|v\|_a \le \|v\|_1 \le C\|v\|_a, \qquad \hat{c}\|v\|_1 \le \|v\|_a \le \hat{C}\|v\|_1.$$

7.5.3 Error estimates in point-wise norm.

For one dimensional Sturm-Liouville problem, we can easily prove the following (over-estimate though).

Theorem 7.5.

$$\|u - u_h\|_\infty \le Ch\|u''\|_\infty \tag{7.50}$$
$$\|u' - u_h'\|_\infty \qquad \textit{Does not make sense} \tag{7.51}$$

where C is a constants. Since u_h' is discontinuous at nodal points, the infinity norm makes no sense because it can only be defined for continuous functions.

Proof:

$$
\begin{aligned}
e_h(x) = u(x) - u_h(x) &= \int_a^x e_h'(t)dt \\
|e_h(x)| &\le \int_a^b |e_h'(t)|dt \\
&\le \left\{\int_a^b |e_h'|^2 dt\right\}^{1/2} \left\{\int_a^b 1\, dt\right\}^{1/2} \\
&\le \sqrt{b-a}\left\{\int_a^b \frac{p}{p_{min}}|e_h'|^2 dt\right\}^{1/2} \\
&\le \frac{\sqrt{b-a}}{p_{min}}\|e_h\|_a \\
&\le \frac{\sqrt{b-a}}{p_{min}}\|\tilde{e_h}\|_a \\
&\le Ch\|u''\|_\infty.
\end{aligned}
$$

REMARK: Actually we can prove a better inequality:

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

The finite element method is second order accurate.

7.6 Exercises

1. Take $n = 3$, the number of variables, describe the Sobolev space $H^3(\Omega)$, i.e. $m = 3$, in terms of $L^2(\Omega)$. Also explain the inner product, the norm, the Schwartz inequality, the distance, and the Sobolev embedding theorem in this space as we did in class.

2. Consider the function $v(x) = |x|^\alpha$ on $\Omega = (-1, 1)$ with $\alpha \in \mathcal{R}$. For what values of α is $v \in H^0(\Omega)$? (Consider negative α as well). For what values is $v \in H^1(\Omega)$? in $H^m(\Omega)$? For what values of α is $v \in C^m(\Omega)$?
   Hint: Generally
   $$|x|^\alpha = \begin{cases} x^\alpha & \text{if } x \ge 0 \\ (-x)^\alpha & \text{if } x < 0. \end{cases}$$
   However, if $\alpha = 2k$, where k is a non-negative integer, then
   $$|x|^\alpha = \begin{cases} x^{2k} & \text{if } \alpha = 2k, k > 0 \\ 1 & \text{if } \alpha = 0. \end{cases}$$
   Also
   $$\lim_{x\to 0}|x|^\alpha = \begin{cases} 0 & \text{if } \alpha > 0 \\ 1 & \text{if } \alpha = 0 \\ \infty & \text{if } \alpha < 0, \end{cases} \qquad \int_{-1}^1 |x|^\alpha dx = \begin{cases} \dfrac{2}{\alpha+1} & \text{if } \alpha > -1 \\ \infty & \text{if } \alpha \le -1. \end{cases}$$

3. Are each of the following statements true or false? Justify your answers.

   (a) If $u \in H^2(0,1)$ then u' and u'' are both continuous functions.

   (b) If $u(x,y) \in H^2(\Omega)$, then $u(x,y)$ may not have continuous partial derivatives $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$? Does $u(x,y)$ have first and second order *weak* derivatives? Is $u(x,y)$ continuous in Ω?

4. Consider the Sturm-Louville problem:

$$-\big(p(x)u(x)'\big)' + q(x)u(x) = f(x), \quad 0 \le x \le \pi,$$
$$\alpha u(0) + \beta u'(0) = \gamma, \quad u'(\pi) = u_b,$$

   where

$$0 < p_{min} \le p(x) \le p_{max} \le \infty, \quad 0 \le q_{min} \le q(x) \le q_{max} \le \infty.$$

   (a) Derive the weak form for the problem. Define a bilinear form $a(u,v)$ and a linear form $L(v)$ to simplify it. What is the energy norm?

   (b) What kind of restrictions should we have for α, β, and γ in order that the weak form has a solution?

   (c) Determine the space where the solution resides under the weak form.

   (d) If we look for a finite element solution in a finite dimensional space V_h using a conforming finite element method, should V_h be a subspace of C^0, C^1, C^2?

   (e) Given a triangulation $x_0 = 0 < x_1 < x_2 \cdots < x_{M-1} < x_M = \pi$, if the finite dimensional space is generated by the hat functions, what kind of structure do the local and global stiffness matrix and the load vector have? Is the resulting linear system of equations formed by the global stiffness matrix and the load vector symmetric, positive definite, and banded?

5. Extra Credit: Consider the two-point boundary value problem

$$-u''(x) = f(x), \quad a < x < b, \qquad u(a) = u(b) = 0.$$

   Let $u_h(x)$ be the finite element solution using the piecewise linear space (in $H_0^1(a,b)$) spanned by a mesh $\{x_i\}$. Show that

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

   where C is a constant. **Hint:** First show that $\|u_h - u_I\|_a = 0$, where $u_I(x)$ is the interpolation function in V_h.

Chapter 8

Futher discussion of FEM in 1D: Boundary conditions, High order FEM, and Implementation

8.1 Boundary Conditions.

Typical boundary conditions include the following at one end, say $x = a$.

1. Dirichlet boundary condition, for example, $u(a) = u_a$ is given.
2. Neumann or derivative boundary condition, for example, $u'(a)$ is given.
3. Robin, also called mixed, boundary condition, for example, $\alpha u(a) + \beta u'(a) = c$ is given, where α, β, and γ are known, but $u(a)$ and $u'(a)$ are unknown.

Example: Consider

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

Dirichlet BC at $x = 0$ and Neumann BC at $x = 1$. To derive the weak form

$$\begin{aligned} \int_0^1 -u'' v dx &= \int_0^1 f v dx, \\ -u'v|_0^1 + \int_0^1 u'v' dx &= \int_0^1 f v dx, \\ -u'(1)v(1) + u'(0)v(0) + \int_0^1 u'v' dx &= \int_0^1 f v dx. \end{aligned}$$

We do not know $u'(0)$, so we have to set $v(0) = 0$ to make it vanish. Therefore Dirichlet boundary conditions are called *essential* boundary conditions. On the other hand, since $u'(1) = 0$, the first term is zero, it does not matter what $v(1)$ is. In other words, there is no constraint on $v(1)$. Therefore Neumann boundary conditions are called *natural* boundary condition. Notice that $u(1)$ is unknown. The weak form of the example is the same

$$\begin{aligned} (u', v') &= (f, v), \quad \forall v \in H_E^1(0,1) \\ H_E^1(0,1) &= \left\{ v(x), \quad v(0) = 0, v \in H^1(0,1) \right\}. \end{aligned}$$

8.1.1 General boundary conditions.

Consider the Sturm-Liouville problem

$$-(pu')' + qu = f, \quad a \le x \le b, \tag{8.1}$$

$$u(a) = 0, \quad \alpha u(b) + \beta u'(b) = \gamma, \quad \beta \neq 0, \quad \frac{\alpha}{\beta} \ge 0, \tag{8.2}$$

where α, β, and γ are known constants, but $u(b)$ and $u'(b)$ are unknowns. Integration by parts again we get

$$-p(b)u'(b)v(b) + p(a)u'(a)v(a) + \int_a^b \left(pu'v' + quv\right) dx = \int_a^b fv dx. \tag{8.3}$$

Since we do not know $u'(a)$, we set $v(a) = 0$, this is the essential boundary condition. From the mixed boundary condition we can solve for $u'(b)$

$$u'(b) = \frac{\gamma - \alpha u(b)}{\beta} \tag{8.4}$$

Plug this into the equation (8.3), we get

$$-p(b)v(b)\frac{\gamma - \alpha u(b)}{\beta} + \int_a^b \left(pu'v' + quv\right) dx = \int_a^b fv dx.$$

This is

$$\int_a^b \left(pu'v' + quv\right) dx + \frac{\alpha}{\beta}p(b)u(b)v(b) = \int_a^b fv dx + \frac{\gamma}{\beta}p(b)v(b).$$

This is the weak form (variational form) of the Sturm-Liouville problem with the given boundary conditions.

Define:

$$a(u,v) = \int_a^b \left(pu'v' + quv\right) dx + \frac{\alpha}{\beta}p(b)u(b)v(b) \tag{8.5}$$

$$L(v) = (f,v) + \frac{\gamma}{\beta}p(b)v(b) \tag{8.6}$$

We can prove that:

1. $a(u,v) = a(v,u)$, that is $a(u,v)$ is symmetric.
2. $a(u,v)$ is a bi-linear form: that is

$$a(\alpha u + \beta w, v) = \alpha a(u,v) + \beta a(w,v)$$
$$a(u, \alpha v + \beta w) = \alpha a(u,v) + \beta a(u,w)$$

3. $a(u,v)$ is a inner product. The energy norm is

$$\|u\|_a = \sqrt{a(u,u)} = \left\{ \int_a^b \left(pu'^2 + qu^2\right) dx + \frac{\alpha}{\beta}p(b)u(b)^2 \right\}^{\frac{1}{2}}$$

Now we can see why we need to have $\beta \neq 0$, and $\alpha/\beta \geq 0$.

Using the inner product, the solution of the weak form $u(x)$ satisfies

$$a(u, v) = L(v), \quad \forall v \in H_E^1(a, b) \tag{8.7}$$

$$H_E^1(a, b) = \left\{ v(x), \quad v(a) = 0, \quad v \in H^1(a, b) \right\}. \tag{8.8}$$

Again remember that there is no restriction on $v(b)$. The boundary condition is essential at $x = a$, natural at $x = b$.

The solution u is also the minimizer of the following functional

$$F(v) = \frac{1}{2} a(v, v) - L(v) \tag{8.9}$$

in the space $H_E^1(a, b)$.

8.1.2 Non-homogeneous Dirichlet boundary condition

Now we assume that $u(a) = u_a \neq 0$ in (8.2). In this case, the solution can be decomposed as the sum of the particular solution

$$-(pu')' + qu = 0, \quad a \leq x \leq b, \tag{8.10}$$

$$u(a) = u_a, \quad \alpha u(b) + \beta u'(b) = 0, \quad \beta \neq 0, \quad \frac{\alpha}{\beta} \geq 0, \tag{8.11}$$

and

$$-(pu')' + qu = f, \quad a \leq x \leq b, \tag{8.12}$$

$$u(a) = 0, \quad \alpha u(b) + \beta u'(b) = \gamma, \quad \beta \neq 0, \quad \frac{\alpha}{\beta} \geq 0. \tag{8.13}$$

If we are able to find the particular solution, then we can use the weak form to find the solution corresponding to the homogeneous Dirichlet boundary condition at $x = a$.

8.2 Numerics

In this section, we consider the finite element method using the piecewise linear function over a triangulation $x_1 = a$, x_2, $\cdots$, $x_M = b$ for the Sturm-Liouville problem

$$-(pu')' + qu = f, \quad a \leq x \leq b,$$

$$u(0) = u_a, \quad \alpha u(b) + \beta u'(b) = \gamma, \quad \beta \neq 0, \quad \frac{\alpha}{\beta} \geq 0.$$

We again use the hat functions as the basis functions. We will focus on the treatment of the boundary conditions. Let the finite element solution be

$$u_h(x) = \sum_{i=0}^{M} \alpha_i \phi_i(x). \tag{8.14}$$

The solution is unknown at $x = b$, so it is not surprise to have $\phi_M(x)$ for the natural boundary condition. However, do pay attention to the first term $\phi_0(x)$ that is not in $H_0^1(a, b)$ to deal with the Dirichlet boundary condition.

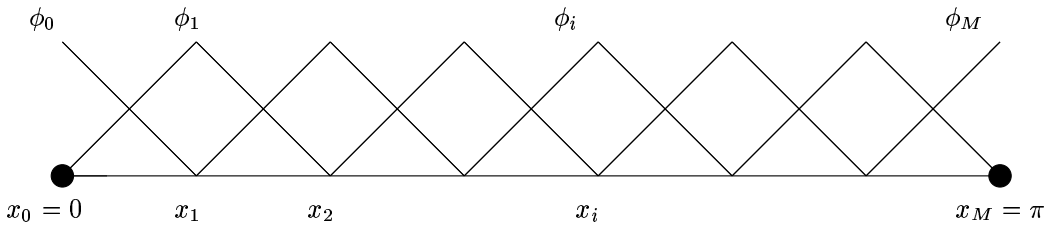


Figure 8.1. *Diagram of a triangulation in one dimension*

The local stiffness matrix is

$$K_i^e = \left[\begin{array}{cc} a(\phi_i,\phi_i) & a(\phi_i,\phi_{i+1}) \\ a(\phi_{i+1},\phi_i) & a(\phi_{i+1},\phi_{i+1}) \end{array} \right]_{(x_i,x_{i+1})} = \left[\begin{array}{cc} \int_{x_i}^{x_{i+1}} p{\phi_i'}^2 dx & \int_{x_i}^{x_{i+1}} p\phi_i'\phi_{i+1}' dx \\ \int_{x_i}^{x_{i+1}} p\phi_{i+1}'\phi_i' dx & \int_{x_i}^{x_{i+1}} p{\phi_{i+1}'}^2 dx \end{array} \right]$$

$$+ \left[\begin{array}{cc} \int_{x_i}^{x_{i+1}} q{\phi_i}^2 dx & \int_{x_i}^{x_{i+1}} q\phi_i\phi_{i+1} dx \\ \int_{x_i}^{x_{i+1}} q\phi_{i+1}\phi_i dx & \int_{x_i}^{x_{i+1}} q\phi_{i+1}^2 dx \end{array} \right] + \frac{\alpha}{\beta} p(b) \left[\begin{array}{cc} \phi_i^2(b) & \phi_i(b)\phi_{i+1}(b) \\ \phi_{i+1}(b)\phi_i(b) & \phi_{i+1}^2(b) \end{array} \right],$$

and the local load vector is

$$F_i^e = \left[\begin{array}{c} L(\phi_i) \\ L(\phi_{i+1}) \end{array} \right] = \left[\begin{array}{c} \int_{x_i}^{x_{i+1}} f\phi_i dx \\ \int_{x_i}^{x_{i+1}} f\phi_{i+1} dx \end{array} \right] + \frac{\gamma}{\beta} p(b) \left[\begin{array}{c} \phi_i(b) \\ \phi_{i+1}(b) \end{array} \right].$$

We can see clearly the contributions from the boundary conditions. Note that the only non-zero contribution of the boundary condition to the stiffness matrix and the load vector is from the last element $[x_{M-1}, x_M]$.

8.2.1 A numerical treatment of the Dirichlet boundary condition

The finite element solution defined on the triangulation can be written as

$$u_h(x) = u_a\phi_0(x) + \sum_{j=1}^{M} \alpha_j \phi_j(x),$$

where α_j, $j = 1, 2, \cdots, M$ are the unknowns. Note that $u_a\phi_0(x)$ is an approximate particular solution.

To use the finite element method to determine the coefficients, we enforce the weak form for $u_h(x) - u_a\phi_0(x)$ which satisfies the homogeneous Dirichlet boundary condition at $x = a$. Notice that $\phi_0(x) \equiv 0$ and $\phi'(x) \equiv 0$ when $x \geq x_1$. We have

$$a\left(u_h(x) - u_a\phi_0(x), \phi_i(x)\right) = L(\phi_i), \quad i = 1, 2, \cdot, M,$$

or

$$\begin{aligned}
a(\phi_1,\phi_1)\alpha_1 + a(\phi_1,\phi_2)\alpha_2 + \cdots + a(\phi_1,\phi_M)\alpha_M &= L(\phi_1) - a(\phi_0,\phi_1)\,u_a \\
a(\phi_2,\phi_1)\alpha_1 + a(\phi_2,\phi_2)\alpha_2 + \cdots + a(\phi_2,\phi_M)\alpha_M &= L(\phi_2) - a(\phi_0,\phi_2)\,u_a \\
\cdots\cdots\cdots\cdots &= \cdots\cdots\cdots\cdots \\
a(\phi_M,\phi_1)\alpha_1 + a(\phi_M,\phi_2)\alpha_2 + \cdots + a(\phi_M,\phi_M)\alpha_M &= L(\phi_M) - a(\phi_0,\phi_M)\,u_a .
\end{aligned}$$

This is equivalent to the following system of equations

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & a(\phi_1,\phi_1) & \cdots & a(\phi_1,\phi_M) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a(\phi_M,\phi_1) & \cdots & a(\phi_M,\phi_M) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_M \end{bmatrix} = \begin{bmatrix} u_a \\ (f,\phi_1) - a(\phi_1,\phi_0)u_a \\ \vdots \\ (f,\phi_M) - a(\phi_M,\phi_0)u_a \end{bmatrix}.$$

This is one of methods to deal with the Dirichlet boundary conditions.

8.2.2 Contribution from Neumann or Mixed boundary condition

The contribution of the mixed boundary condition using the hat basis function is zero until the last element $[x_{M-1}, x_M]$ where $\phi_M(b)$ is not zero. Therefore the local stiffness matrix of the last element is

$$\begin{bmatrix} \displaystyle\int_{x_i}^{x_{i+1}} \left(p{\phi'_{M-1}}^2 + q\phi_{M-1}^2\right) dx & \displaystyle\int_{x_i}^{x_{i+1}} \left(p\phi'_{M-1}\phi'_M + q\phi_{M-1}\phi_M\right) dx \\ \displaystyle\int_{x_i}^{x_{i+1}} \left(p\phi'_M\phi'_{M-1} + q\phi_M\phi_{M-1}\right) dx & \displaystyle\int_{x_i}^{x_{i+1}} \left(p{\phi'_M}^2 + \phi_M^2\right) dx + \frac{\alpha}{\beta}p(b) \end{bmatrix}.$$

The local load vector is

$$F_i^e = \begin{bmatrix} \displaystyle\int_{x_{M-1}}^{x_M} f(x)\phi_{M-1}(x)dx \\ \displaystyle\int_{x_{M-1}}^{x_M} f(x)\phi_M(x)dx + \frac{\gamma}{\beta}p(b) \end{bmatrix}$$

8.2.3 The pseudo-code of the FEM for the Sturm-Liouvill problem using the piecewise linear basis functions.

- Initialize:

```
for i=0, M
    F(i) = 0
    for j=0, M
        A(i,j) = 0
    end
end
```

- Assembling element by element:

  for i=1, M

  $$A(i-1,i-1) = A(i-1,i-1) + \int_{x_{i-1}}^{x_i} \left(p{\phi'_{i-1}}^2 + q\phi_{i-1}^2 \right) dx$$

  $$A(i-1,i) = A(i-1,i) + \int_{x_{i-1}}^{x_i} \left(p\phi'_{i-1}\phi'_i + q\phi_{i-1}\phi_i \right) dx$$

  $$A(i,i-1) = A(i-1,i)$$

  $$A(i,i) = A(i,i) + \int_{x_{i-1}}^{x_i} \left(p{\phi'_i}^2 + q\phi_i^2 \right) dx$$

  $$F(i-1) = F(i-1) + \int_{x_{i-1}}^{x_i} f(x)\phi_{i-1}(x)dx$$

  $$F(i) = F(i) + \int_{x_{i-1}}^{x_i} f(x)\phi_i(x)dx$$

  end

- Deal with the Dirichlet BC:

  $$A(0,0) = 1; \qquad F(0) = u_a$$

  for i=1, M

  $$F(i) = F(i) - A(i,0) * ua;$$

  $$A(i,0) = 0; \qquad A(0,i) = 0.$$

  end

- Deal with the Mixed BC at $x = b$.

  $$A(M,M) = A(M,M) + \frac{\alpha}{\beta} p(b)$$

  $$F(M) = F(M) + \frac{\gamma}{\beta} p(b)$$

- Solve $AU = F$.
- Error analysis.

8.3 High order elements.

To solve the Sturm-Liouville problems or other second order differential equations, we can use the piecewise linear finite dimensional space over a triangulation. The error usually is $O(h^2)$. If we want to improve the accuracy, we can try:

- Refine the triangulation, that is, decrease h.
- Use better and larger finite dimensional spaces, for example, use the piecewise quadratic or the piecewise cubic functions.

  We will use the Sturm-Liouville problem

  $$-\left(p'u\right)' + qu = f, \quad a \le x \le b$$
  $$u(a) = 0, \quad u(b) = 0,$$

as the model problem for the discussions. The other boundary condition can be treated in a similar way as we discussed before. We assume a triangulation

$$x_0 = a,\ x_1,\ \cdots,\ x_M = b, \quad \text{and the elements}$$
$$\Omega_1 = [x_0, x_1], \quad \Omega_2 = [x_1, x_2],\ \cdots,\ \Omega_{M-1} = [x_{M-1}, x_M]$$

are given. We will discuss the piecewise quadratic and piecewise cubic functions. We still require the finite dimensional spaces are in $H^1(a,b)$, so the finite element methods are conforming methods.

8.3.1 Piecewise quadratic basis function

Define

$$V_h = \{v(x), \quad v(x) \text{ is continuous piecewise quadratic over the triangulation}, v(x) \in H^1(a,b)\}$$

Obviously the piecewise linear finite dimensional space is a subspace of the space defined above. So we expect the finite element solution is more accurate compared to the one obtained using the piecewise linear functions.

What is the dimension of the piecewise quadratic basis function?

If we know the dimension of the finite dimensional space, then we can choose a set of basis functions. The dimension of a finite dimensional space sometimes is also called the *the degree of freedom*.

Given a function $\phi(x)$ in V_h, on each element, the quadratic function has the form

$$\phi(x) = a_i x^2 + b_i x + c_i, \quad x_i \le x < x_{i+1}. \tag{8.15}$$

In other words, there are three parameters to determine a quadratic function in the interval $[x_i, x_{i+1}]$. In total, there are M elements, so there are $3M$ parameters. But there are not totally free because they have to satisfy the continuity condition which is

$$\lim_{x \to x_i -} \phi(x) = \lim_{x \to x_i +} \phi(x).$$

More precisely

$$a_{i-1} x_i^2 + b_{i-1} x_i + c_{i-1} = a_i x_i^2 + b_i x_i + c_i, \quad i = 1, 2, \cdots M.$$

There are $M-1$ interior nodal points, so there are $M-1$ restraints. Also $\phi(x)$ should satisfy the boundary conditions $\phi(a) = \phi(b) = 0$. Thus the total degree of the freedom, the dimension of the finite element space is

$$3M - (M-1) - 2 = 2M - 1. \tag{8.16}$$

Construct basis functions for the piecewise quadratic functional space.

We know the dimension of V_h is $2M-1$, if we can construct $2M-1$ basis function that are linear independent, then all the functions in V_h can be expressed in terms of the basis functions. The criteria is similar to the hat functions.

- They have to be continuous piecewise quadratic functions.
- They should have the minimum support, i.e., they are zero almost everywhere.
- We also hope they are similar to the hat function, that is they are unity at one point, but zero on other nodal points.

Since the degree of the freedom is $2M-1$, but there are only $M-1$ interior nodal points, we add M auxiliary points (not nodal points) between x_i and x_{i+1} and define

$$z_{2i} = x_i, \tag{8.17}$$

$$z_{2i+1} = \frac{x_i + x_{i+1}}{2}. \tag{8.18}$$

For example, if the nodal points are $x_0 = 0$, $x_1 = \pi/2$, $x_2 = \pi$, then $z_0 = x_0$, $z_1 = \pi/4$, $z_2 = x_1 = \pi/2$, $z_3 = 3\pi/4$, $z_4 = x_2 = \pi$. Note all the basis function has to be one piece in the interval $[z_{2k}, z_{2k+2}]$, which is an element.

Now we can define the basis function

$$\phi_i(z_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{8.19}$$

First consider the basis function in the first element $[x_0, x_1]$, that corresponding to the interval $[z_0, z_2]$, the boundary point, z_1, the mid point, and z_2 which is a nodal point. We need to construct ϕ_1, and ϕ_2. For $\phi_1(x)$, we have $\phi_1(z_0) = 0$, $\phi_1(z_1) = 1$, and $\phi_1(z_2) = 0$, $\phi_1(z_j) = 0$, $j = 3, \cdots, 2M-1$. In the interval $[z_0, z_1]$, it has to have the form

$$\phi_1 = C(x - z_0)(x - z_2)$$

because z_0 and z_2 are zeros of $\phi_1(x)$. We choose C such that $\phi_1(z_1) = 1$, therefore we get

$$\phi_1(z_1) = C(z_1 - z_0)(z_1 - z_2) = 1, \quad \longrightarrow C = \frac{1}{(z_1 - z_0)(z_1 - z_2)}.$$

So the basis function $\phi_1(x)$ is

$$\phi_1(x) = \begin{cases} \dfrac{(x - z_0)(x - z_2)}{(z_1 - z_0)(z_1 - z_2)} & \text{if } z_0 \le x < z_2, \\ 0 & \text{otherwise.} \end{cases} \tag{8.20}$$

It is easy to verify that $\phi_1(x)$ is continuous piecewise quadratic function in the domain $[x_0, x_M]$.

Generally in the element $[x_i, x_{i+1}]$, the three global basis functions have the form

$$\phi_{2i}(z) = \begin{cases} 0 & \text{if } z < x_{i-1} \\ \dfrac{(z - z_{2i-1})(z - z_{2i-2})}{(z_{2i} - z_{2i-1})(z_{2i} - z_{2i-2})} & \text{if } x_{i-1} \le z < x_i \\ \dfrac{(z - z_{2i+1})(z - z_{2i+2})}{(z_{2i} - z_{2i+1})(z_{2i} - z_{2i+2})} & \text{if } x_i \le z < x_{i+1} \\ 0 & \text{if } x_{i+1} < z. \end{cases}$$

$$\phi_{2i+1}(z) = \begin{cases} 0 & \text{if } x < x_i \\ \dfrac{(z - z_{2i})(z - z_{2i+2})}{(z_{2i+1} - z_{2i})(z_{2i+1} - z_{2i+2})} & \text{if } x_i \le z < x_{i+1} \\ 0 & \text{if } x_{i+1} < z. \end{cases}$$

$$\phi_{2i+2}(z) = \begin{cases} 0 & \text{if } x < x_i \\ \dfrac{(z - z_{2i})(z - z_{2i+1})}{(z_{2i+2} - z_{2i})(z_{2i+2} - z_{2i+1})} & \text{if } x_i \le z < x_{i+1} \\ \dfrac{(z - z_{2i+3})(z - z_{2i+4})}{(z_{2i+2} - z_{2i+3})(z_{2i+2} - z_{2i+4})} & \text{if } x_i \le z < x_{i+1} \\ 0 & \text{if } x_{i+2} < z. \end{cases}$$

8.3.2 Assembling the stiffness matrix and the load vector.

The finite element solution can be written as

$$u_h = \sum_{i=1}^{2M-1} \alpha_i \phi_i(x).$$

The entries of the coefficient matrix is $\{a_{ij}\} = a(\phi_i, \phi_j)$ and the load vector is $F_i = L(\phi_i)$. On each element $[x_i, x_i + 1]$, that is $[z_{2i}, z_{2i+2}]$, there are three non-zero basis functions: ϕ_{2i}, ϕ_{2i+1}, and ϕ_{2i+2}. So the local stiffness matrix has the form:

$$K_i^e = \left[\begin{array}{ccc} a(\phi_{2i}, \phi_{2i}) & a(\phi_{2i}, \phi_{2i+1}) & a(\phi_{2i}, \phi_{2i+2}) \\ a(\phi_{2i+1}, \phi_{2i}) & a(\phi_{2i+1}, \phi_{2i+1}) & a(\phi_{2i+1}, \phi_{2i+2}) \\ a(\phi_{2i+2}, \phi_{2i}) & a(\phi_{2i+2}, \phi_{2i+1}) & a(\phi_{2i+2}, \phi_{2i+2}) \end{array} \right]_{(x_i, x_{i+1})} \tag{8.21}$$

and the local load vector is

$$L_i^e = \left[\begin{array}{c} L(\phi_{2i}) \\ L(\phi_{2i+1}) \\ L(\phi_{2i+2}) \end{array} \right]_{(x_i, x_{i+1})}. \tag{8.22}$$

The stiffness matrix is still symmetric positive definite. But it is denser compared to the hat basis functions. It is still banded matrix with the band width being 5. It is a penta-diagonal matrix. The advantage is more accurate.

8.3.3 Cubic basis functions in $H^1(a, b)$ space.

On each element $[x_i, x_{i+1}]$, the basis function is a cubic

$$\phi(x) = a_i x^3 + b_i x^2 + c_i x + d_i, \quad i = 0, 1, \cdots, M - 1.$$

There are four parameters and we need to add two auxiliary points between x_i and x_{i+1}. The local stiffness matrix is going to be a four by four matrix. I will leave this as a homework problem.

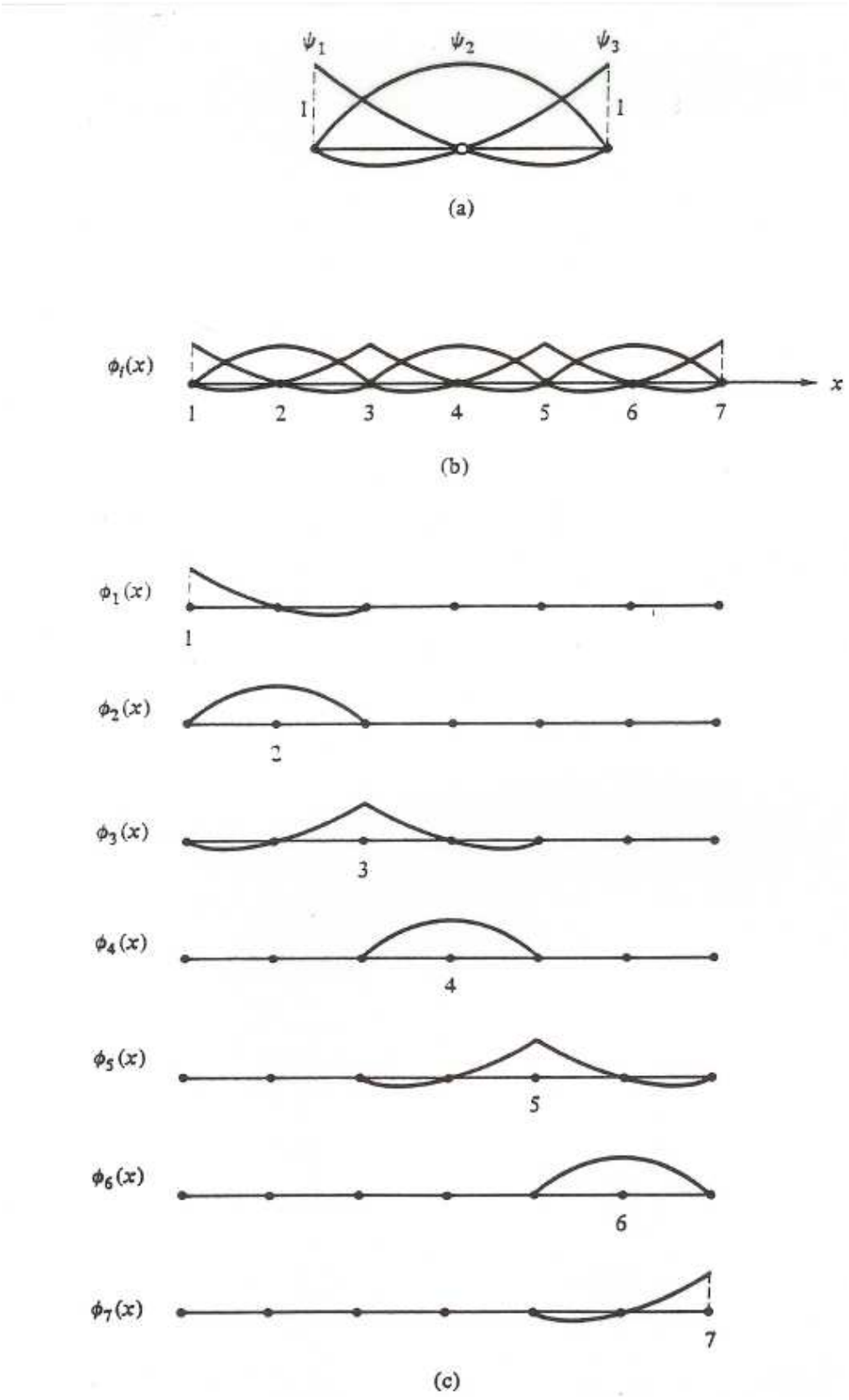


Figure 8.2. *(a) A three-node element with quadratic shape functions. (b) a mesh consisting of three quadratic elements and (c) the basis functions generated by three elements.*

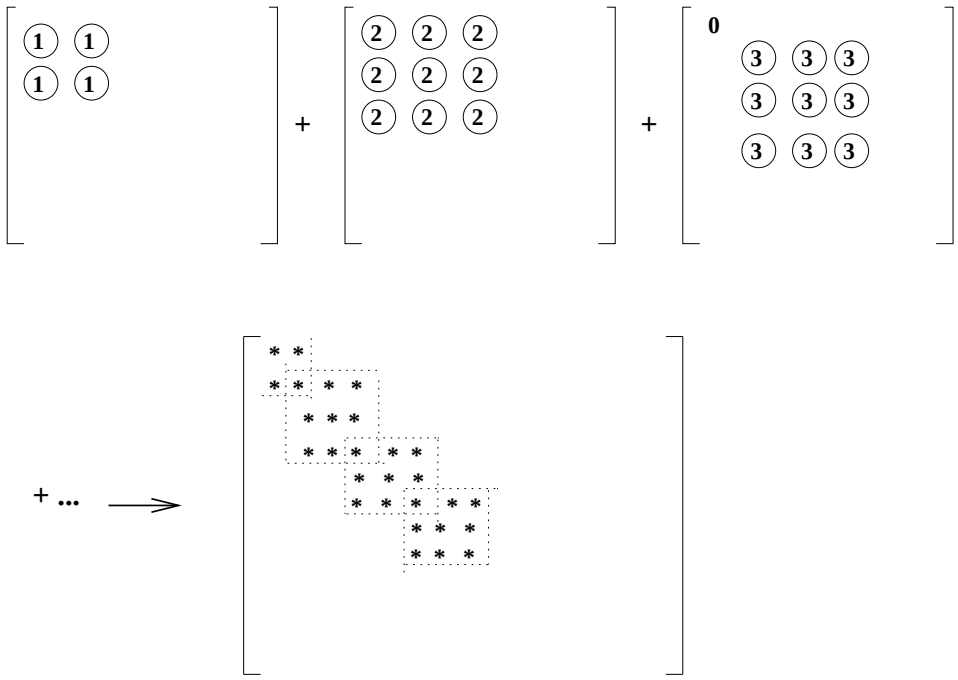


Figure 8.3. *Diagram of assembling the stiffness matrix using piecewise quadratic basis functions.*

8.4 The General finite element method code for 1D problems using Matlab

The Matlab codes are available at:
http://www4.ncsu.edu/~zhilin/TEACHING/MA587/MATLAB/1D

- The problems can be solved are:

$$-(k(x)u')' + c(x)u' + q(x)u = f(x), \quad a \le x \le b.$$

- We use the conforming finite element method.
- Boundary conditions. Dirichlet, Neumann and Mixed boundary conditions at $x = a$ and $x = b$.
- The triangulation is:

$$x_0 = a \le x_1 < x_2 \cdots < x_M = b.$$

  We will elaborate this again later.
- The finite element spaces can be: Piecewise linear, quadratic, and cubic over the triangulation.

- The integration method is the Gaussian quadrature of order one, two, three and four.
- The matrix assembling is done element by element.

8.4.1 Quadrature formulas – Gaussian quadrature

For finite element methods, we need to calculate integrals of the form

$$\int_a^b f(x)dx = \int_{-1}^1 \bar{f}(\xi)d\xi$$

$$\xi = \frac{x-a}{b-a} + \frac{x-b}{b-a}, \quad \text{or} \quad x = a + \frac{b-a}{2}(1+\xi)$$

$$d\xi = \frac{2}{b-a}dx, \quad dx = \frac{b-a}{2}d\xi$$

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^1 f\left(a + \frac{b-a}{2}(1+\xi)\right)d\xi = \frac{b-a}{2}\int_{-1}^1 \bar{f}(\xi)d\xi.$$

The general quadrature formula can be written as

$$\int_{-1}^1 g(\xi)d\xi = \sum_{i=1}^N w_i g(\xi_i).$$

We want to choose ξ_i and w_i so that the quadrature is as accurate as possible. In the mid-point rule, the trapezoidal rule, and the Simpson's methods, ξ_i are pre-defined. In the Gaussian quadrature formula's, both ξ_i and w_i are unknowns. Generally we can choose ξ_i and w_i such that the quadrature formula is exact for $g(x) = 1, x, \cdots, x^{2N-1}$. The number $2N-1$ is called the *algebraic precision* of the quadrature formula. Such quadrature formulas have the following characteristics:

- Very accurate.
- Open formulas, do not need to use two end points.
- No recursive relations for ξ_i and w_i.
- Good for finite element methods because $b-a \sim h$ is small, we just need a few points ξ_i.

We call ξ_i and w_i Gaussian points and weights of the quadrature formulas.

Gaussian quadrature of order 1 (one point):

$$\int_{-1}^1 g(\xi)d\xi = w_1 g(\xi_1).$$

Take $g(\xi) = 1$ and $g(\xi) = \xi$, we get

$$g(\xi) = 1, \int_{-1}^1 g(\xi)d\xi = 2, \quad \longrightarrow 2 = w_1 \cdot 1,$$

$$g(\xi) = \xi, \int_{-1}^1 g(\xi)d\xi = 0, \quad \longrightarrow 0 = w_1\xi_1.$$

Therefore, $w_1 = 2$, $\xi_1 = 0$. The quadrature formula is the mid-point rule.

Gaussian quadrature of order 2 (two points).

Take $g(\xi) = 1$ and $g(\xi) = \xi$, and $g(\xi) = \xi^2$, we get

$$g(\xi) = 1, \int_{-1}^{1} g(\xi)d\xi = 2, \quad \longrightarrow 2 = w_1 + w_2,$$

$$g(\xi) = \xi, \int_{-1}^{1} g(\xi)d\xi = 0, \quad \longrightarrow 0 = w_1\xi_1 + w_2\xi_2,$$

$$g(\xi) = \xi^2, \int_{-1}^{1} g(\xi)d\xi = \frac{2}{3}, \quad \longrightarrow \frac{2}{3} = w_1\xi_1^2 + w_2\xi_2^2,$$

$$g(\xi) = \xi^3, \int_{-1}^{1} g(\xi)d\xi = 0, \quad \longrightarrow 0 = w_1\xi_1^3 + w_2\xi_2^3.$$

Solve the non-linear system of equations we get:

$$w_1 = w_2 = 1, \qquad \xi_1 = -\frac{1}{\sqrt{3}}, \quad \xi_2 = \frac{1}{\sqrt{3}},$$

and the Gaussian quadrature of order 2 is

$$\int_{-1}^{1} g(\xi)d\xi \approx g\left(-\frac{1}{\sqrt{3}}\right) + g\left(\frac{1}{\sqrt{3}}\right). \tag{8.23}$$

We can pre-store these data in two separated matrices:

$$\begin{matrix} & & \xi_i & \end{matrix} \qquad \begin{matrix} & & w_i & \end{matrix}$$

$$\begin{bmatrix} 0 & \frac{-1}{\sqrt{3}} & \frac{-\sqrt{3}}{\sqrt{5}} & -0.8611363116 \\ & \frac{1}{\sqrt{3}} & 0 & -0.3399810436 \\ & & \frac{\sqrt{3}}{\sqrt{5}} & 0.3399810436 \\ & & & 0.8611363116 \end{bmatrix} \begin{bmatrix} 2 & 1 & \frac{5}{9} & 0.3478548451 \\ & 1 & \frac{8}{9} & 0.6521451549 \\ & & \frac{5}{9} & 0.6521451549 \\ & & & 0.3478548451 \end{bmatrix}.$$

Below is the Matlab code setint.m:

```
function [xi,w] = setint

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%  Function setint provides the integration points x(i), and the                %
%  weights of the Gaussian quadrature formula.                                  %
%  Output:                                                                      %
%     x(4,4):  x(:,i) is the Gaussian points of order i.                        %
%     w(4,4):  w(:,i) is the weights of quadrature of order i.                  %
%                                                                               %
% Reference: Finite element, An introduction, Vol. 1 by E.Becker,               %
% G.Carey, and J.Oden, pp. 94.                                                  %
%-------------------------------------------------------------------------------%

  clear x
  clear w

     xi(1,1) = 0;
     w(1,1) = 2; % Gaussian quadrature of order 1

     xi(1,2) = -1/sqrt(3);
```

```
    xi(2,2) = -xi(1,2);
    w(1,2) = 1;
    w(2,2) = w(1,2); % Gaussian quadrature of order 2

    xi(1,3) = -sqrt(3/5);
    xi(2,3) = 0;
    xi(3,3) = -xi(1,3);
    w(1,3) = 5/9;
    w(2,3) = 8/9;
    w(3,3) = w(1,3);        % Gaussian quadrature of order 3

    xi(1,4) = - 0.8611363116;
    xi(2,4) = - 0.3399810436;
    xi(3,4) = -xi(2,4);
    xi(4,4) = -xi(1,4);
    w(1,4) = 0.3478548451;
    w(2,4) = 0.6521451549;
    w(3,4) = w(2,4);
    w(4,4) = w(1,4); % Gaussian quadrature of order 4

    return

%------------------------- END OF SETINT -----------------------------
```

8.4.2 Shape functions

Usually it is easier to transfer the weak form in the new variable ξ, and write the basis function using the new variable ξ. For each element, the weak form, assuming $c(x) = 0$,

$$\int_{x_i}^{x_{i+1}} \left(k(x)\phi_i'\phi_j' + q(x)\phi_i\phi_j \right) dx = \int_{x_i}^{x_{i+1}} f\phi_i(x)dx$$

becomes

$$\frac{x_{i+1} - x_i}{2} \int_{-1}^{1} \left(\bar{k}(\xi)\psi_i'\psi_j' + \bar{q}(\xi)\psi_i\psi_j \right) d\xi = \frac{x_{i+1} - x_i}{2} \int_{-1}^{1} \bar{f}(\xi)\psi_i d\xi$$

where

$$\bar{k}(\xi) = k\left(x_i + \frac{x_{i+1} - x_i}{2}(1 + \xi) \right),$$

so are $\bar{b}$ and $\bar{f}$. ψ_i, ψ_j are the local basis function under the new variables. They are called the shape functions. For piecewise linear functions, there are only two none-zero shape functions

$$\psi_1 = \frac{1 - \xi}{2}, \qquad \psi_2 = \frac{1 + \xi}{2}. \tag{8.24}$$

$$\psi_1' = -\frac{1}{2}, \qquad \psi_2' = \frac{1}{2}, \tag{8.25}$$

see Fig. 8.4.

For the piecewise quadratic functions, there are only three none-zero shape functions

$$\psi_1 = \frac{\xi(\xi - 1)}{2}, \qquad \psi_2 = 1 - \xi^2, \qquad \psi_3 = \frac{\xi(\xi + 1)}{2}, \tag{8.26}$$

$$\psi_1' = \xi - \frac{1}{2}, \qquad \psi_2' = -\xi, \qquad \psi_3' = \xi + \frac{1}{2}, \tag{8.27}$$

see Fig. 8.4.

Notice that when we find the derivatives, there is an extra factor which is

$$\frac{d\phi_i}{dx} = \frac{d\psi_i}{d\xi}\frac{d\xi}{dx} = \psi_i' \frac{2}{x_{i+1} - x_i}$$

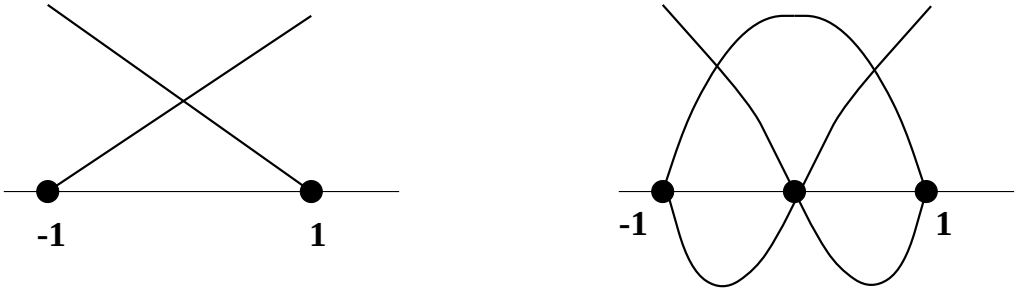


Figure 8.4. *Shape functions under new coordinates ξ. (a). The hat functions. (b). The quadratic function.*

The shape function can be defined in a subroutine:

$$[psi,\ \ dpsi\,] = shape(xi, n)$$

where n is the choice of the elements, for example, $n = 1$ is the linear basis function, $n = 2$ is the quadratic basis function, and $n = 3$ is the cubic basis function. Take $n = 2$ as an example, the outputs are

$psi(1)$, $psi(2)$, $psi(3)$, The value of the three basis functions,
$dpsi(1)$, $dpsi(2)$, $dpsi(3)$, The value of the derivatives.

The Matlab of the subroutine is the following:

```
function [psi,dpsi]=shape(xi,n);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%   Function ''shape'' evaluates the values of the basis functions              %
%   and their derivatives at a point xi.                                        %
%                                                                               %
%   n: The basis function. n=2, linear, n=3, quadratic, n=3, cubic.             %
%   xi: The point where the base function is evaluated.                         %
%   Output:                                                                     %
%   psi:  The value of the base function at xi.                                 %
%   dpsi: The derivative of the base function at xi.                            %
%   Reference: Finite element. An introduction y E.Becker, G.Carey,             %
%   and J.Oden, Vol.1., pp. 95-96.                                              %
%-------------------------------------------------------------------------------%

   switch n
   case 2,
     % Linear base function
     psi(1) = (1-xi)/2;
     psi(2) = (1+xi)/2;
     dpsi(1) = -0.5;
     dpsi(2) = 0.5;
     return

   case 3,
     % quadratic base function
     psi(1) = xi*(xi-1)/2;
     psi(2) = 1-xi*xi;
     psi(3) = xi*(xi+1)/2;
     dpsi(1) =  xi-0.5;
     dpsi(2) = -2*xi;
     dpsi(3) = xi + 0.5;
     return

   case 4,
     % cubic  base function
```

```
    psi(1) = 9*(1/9-xi*xi)*(xi-1)/16;
    psi(2) = 27*(1-xi*xi)*(1/3-xi)/16;
    psi(3) = 27*(1-xi*xi)*(1/3+xi)/16;
    psi(4) = -9*(1/9-xi*xi)*(1+xi)/16;

    dpsi(1) = -9*(3*xi*xi-2*xi-1/9)/16;
    dpsi(2) = 27*(3*xi*xi-2*xi/3-1)/16;
    dpsi(3) = 27*(-3*xi*xi-2*xi/3+1)/16;
    dpsi(4) = -9*(-3*xi*xi-2*xi+1/9)/16;
    return

  end

%-------------------------- END OF SHAPE -----------------------------
```

8.4.3 Main data structure

- Nodal points: $x_1 = a$, x_2, $\cdots$, $x_{nnode} = b$. The number of total nodal points plus the auxiliary points is $nnode$.
- Elements: Ω_1, Ω_2, $\cdots$, Ω_{nelem}. The number of elements is $nelem$.
- Connection between the nodal points and the elements: $nodes(nnode, nelem)$, $nodes(j, i)$ is the j-th index of the notes in the i-th element. For the linear basis function, $j = 1, 2$ since there two nodes in an element. For the quadratic basis function, $j = 1, 2, 3$, since there two nodes and an auxiliary point.

Example. Given the triangulation and the indexing of the nodal points and the elements. For linear basis functions,

$$\begin{aligned} nodes(1,1) &= 1, \quad & nodes(1,2) &= 3, \\ nodes(2,1) &= 3, \quad & nodes(2,2) &= 4, \\ nodes(1,3) &= 4, \quad & nodes(1,4) &= 2, \\ nodes(2,3) &= 2, \quad & nodes(2,4) &= 5 \end{aligned}$$

Example. Given the triangulation and the indexing of the nodal points and the elements. For quadratic basis functions,

$$\begin{aligned} nodes(1,1) &= 4, \quad & nodes(2,1) &= 2, \quad & nodes(3,1) &= 5, \\ nodes(1,2) &= 1, \quad & nodes(2,2) &= 3, \quad & nodes(3,2) &= 4 \end{aligned}$$

8.4.4 Outline of the algorithm

```
function  [x,u]=fem1d
```

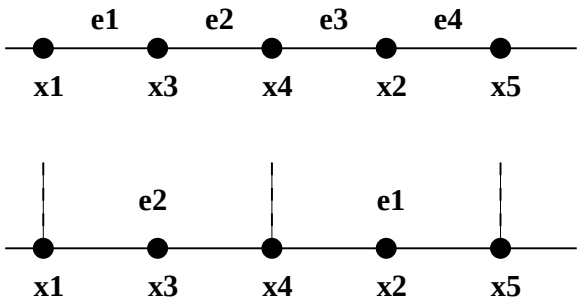


Figure 8.5. *Example of the relation between nodes and elements. (a). The linear basis functions. (b). The quadratic basis function.*

```
        global nnode nelem
        global gk gf
        global xi  w

%%% Output: x are nodal points; u is the finite element solution at
%%%         nodal points.

      [xi,w] = setint;     % Get Gaussian points and weights.

%%% Input data, pre-process

      [x,kbc,ubc,kind,nint,nodes] = prospset;

  %%%  x(nnode): Nodal points,        kbc, ubc: Boundary conditions

  %%%  kind(nelen): Choice of FEM spaces. kind(i)=1,2,3 indicate
  %%%  piecewise linear, quadratic, and cubic FEM space over the
  %%%  triangulation.

  %%%  nint(nelen): Choice of Gaussian quadrature. nint(i)=1,2,3,4
  %%%  indicate Gaussian order 1, 2, 3, 4.

      formkf(kind,nint,nodes,x,xi,w);

  %%%  Assembling the stiffness matrix and the load vector element by
  %%%  element.

      aplyb(kbc,ubc);

  %%%  Deal with the boundary conditions.
```

```
      u = gk\gf;              % Solve the linear system of equations

  %%%  Error analysis ...
```

8.4.5 Assembling element by element

The matlba code is formkf.m

```
   function formkf(kind,nint,nodes,x,xi,w)

.............

    for nel = 1:nelem,
       n = kind(nel) + 1;       % Linear FEM space. n = 2, quadratic n=3, ..

       i1 = nodes(1,nel);       % The first node in nel-th element.
       i2 = nodes(n,nel);       % The last node in nel-th element.
       i3 = nint(nel);          % Order of Gaussian quadrature.
       xic = xi(:,i3);          % Get Gaussian points in the column.
       wc = w(:,i3);            % Get Gaussian weights.

%%% Evaluate the local stiffness matrix ek, and the load vector ef.
       [ek,ef] = elem(x(i1),x(i2),n,i3,xic,wc);

%%% Assembling to the global stiffness matrix gk, and the load vector gf.
       assemb(ek,ef,nel,n,nodes);

    end
```

Evaluation of local stiffness matrix and the load vector.

The Matlab code is elem.m

```
   function [ek,ef] = elem(x1,x2,n,nl,xi,w)
   dx = (x2-x1)/2;

% [x1,x2] is an element [x1,x2]
% n is the choice of FEM space. Linear n=2; quadratic n=3; ...

  for l=1:nl,                          % Quadrature formula that summarize.
    x = x1 + (1.0 + xi(l))*dx;         % Transform the Gaussian points.
    [xk,xc,xb,xf] = getmat(x);         % Get the coefficients at the
                                       % Gaussian points.
    [psi,dpsi] = shape(xi(l),n);       % Get the shape function and
                                       % its derivatives.
```

```
% Assembling the local stiffness matrix and the load vector.
% Notice the additional factor 1/dx in the derivatives.
    for i=1:n,
       ef(i) = ef(i) + psi(i)*xf*w(l)*dx;
       for j=1:n,
          ek(i,j)=ek(i,j)+(xk*dpsi(i)*dpsi(j)/(dx*dx) ...
               +xc*psi(i)*dpsi(j)/dx+xb*psi(i)*psi(j) )*w(l)*dx;
       end
    end
  end
```

Global assembling

The Matlab code is assemb.m

```
    function assemb(ek,ef,nel,n,nodes)
    global gk gf

    for i=1:n,                % Connection between nodes and the elements
      ig = nodes(i,nel);                  % Assemble global vector gf
      gf(ig) = gf(ig) + ef(i);

       for j=1:n,
         jg = nodes(j,nel);          % Assemble global stiffness matrix gk
         gk(ig,jg) = gk(ig,jg) + ek(i,j);
       end
     end
```

Input Data.

The Matlab code is propset.m

```
      function [x,kbc,vbc,kind,nint,nodes] = propset
```

- The relation between the number of nodes, *nnode* and the number of elements *nelem*:
  Linear: $nelem = nnode - 1$.
  Quadratic: $nelem = (nnode - 1)/2$.
  Cubic: $nelem = (nnode - 1)/3$.
- Notes arranged in ascendant order. Equally spaced points are grouped together.
  The Matlab code is datain.m

  function [data] = datain(a,b,nnode,nelem)

The output data has $nrec$ groups

$$data(i,1) = n1, \quad \text{The index of the beginning of nodes.}$$
$$data(i,2) = n2, \quad \text{The number of points in this group.}$$
$$data(i,3) = x(n1), \quad \text{The first nodal point.}$$
$$data(i,4) = x(n1+n2), \quad \text{The last nodal point in this group.}$$

The simple case is

$$data(i,1) = i, \quad data(i,2) = 0, \quad data(i,3) = x(i), \quad data(i,4) = x(i).$$

- The basis functions to be used in each element:

```
for i=1:nelem
   kind(i) = inf_ele = 1, or 2, or 3.
   nint(i) = 1, or 2, or 3, or 4.
   for j=1,kind(i)+1
     nodes(j,i) = j + kind(i)*(i-1);
   end
end
```

8.4.6 The boundary conditions

The Matlab code is aplybc.m. It is determined by an array of two elements $kbc(2)$ and a data array $vbc(2,2)$. At the left boundary

$kbc(1) = 1, \quad vbc(1,1) = u_a, \quad$ The Dirichlet BC at the left end.

$kbc(1) = 2, \quad vbc(1,1) = -k(a)u'(a), \quad$ The Neumann BC at the left end.

$kbc(1) = 3, \quad vbc(1,1) = uxma, \quad vbc(2,1) = uaa, \quad$ The mixed BC of the form :
$k(a)u'(a) = uxma(u(a) - uaa)$.

The boundary condition will affect the stiffness matrix and the load vector and are handled in Matlab codes aplybc.m and drchlta.m.

- The Dirichlet boundary condition $u(a) = u_a = vbc(1,1)$.

```
for i=1:nnode,
    gf(i) = gf(i) - gk(i,1)*vbc(1,1);
    gk(i,1) = 0;    gk(1,i) = 0;
end
gk(1,1) = 1;    gf(1) = vbc(1,1);
```

  where gk is the global stiffness matrix, and gf is the global load vector.

- The Neumann boundary condition $u'(a) = u_{xa}$ is given. The boundary condition can be re-written as $-k(a)u'(a) = -k(a)u_{xa} = vbc(1,1)$. We only need to change the load vector.

```
gf(1) = gf(1) + vbc(1,1);
```

- The mixed boundary condition $\alpha u(a) + \beta u'(a) = \gamma$, $\beta \neq 0$. The boundary condition can be re-written as

$$\begin{aligned} k(a)u'(a) &= -\frac{\alpha}{\beta}k(a)\left(u(a) - \frac{\gamma}{\alpha}\right) \\ &= u_{xma}\left(u(a) - u_{aa}\right) = \mathbf{vbc}(\mathbf{1},\mathbf{1})\left(u(a) - \mathbf{vbc}(\mathbf{2},\mathbf{1})\right) \end{aligned}$$

We need to change both the global stiffness matrix and the global load vector.

```
gf(1) = gf(1) + vbc(1,1)*vbc(2,1);
gk(1,1) =  gk(1,1) + vbc(1,1);
```

Examples

1. $u(a) = 2$, we should set $kbc(1) = 1$, $vbc(1,1) = 2$.
2. $k(x) = 2 + x^2$, $a = 2$, $u'(2) = 2$. Since $k(a) = k(2) = 6$, $-k(a)u'(a) = -12$, we should set $kbc(1) = 2$, $vbc(1,1) = -12$.
3. $k(x) = 2 + x^2$, $a = 2$, $2u(a) + 3u'(a) = 1$. Since

$$\begin{aligned} 3u'(a) &= -2u(a) + 1, \\ 6u'(a) &= -4u(a) + 2 = -4\left(u(a) - \frac{1}{2}\right), \end{aligned}$$

so we should set $kbc(1) = 3$, $vbc(1,1) = -4$, $vbc(2,1) = 1/2$.

Similar at the right boundary condition $x = b$, we should have

$kbc(2) = 1$, $\quad vbc(1,2) = u_b$, $\quad$ The Dirichlet BC at the right end.

$kbc(2) = 2$, $\quad vbc(1,2) = k(b)u'(b)$, $\quad$ The Neumann BC at the right end.

$kbc(2) = 3$, $\quad vbc(1,2) = uxmb$, $\quad vbc(2,2) = ubb$, $\quad$ The mixed BC of the form : $-k(b)u'(b) = uxmb(u(b) - ubb)$.

The boundary condition will affect the stiffness matrix and the load vector and are handled in Matlab codes aplybc.m and drchlta.m.

- The Dirichlet boundary condition $u(b) = u_b = vbc(1,2)$.

```
for i=1:nnode,
    gf(i) = gf(i) - gk(i,nnode)*vbc(1,2);
    gk(i,nnode) = 0;    gk(nnode,i) = 0;
end
gk(nnode,nnode) = 1;    gf(nnode) = vbc(1,1).
```

- The Neumann boundary condition $u'(b) = u_{xb}$ is given. The boundary condition can be re-written as $k(b)u'(b) = k(b)u_{xb} = vbc(1,2)$. We only need to change the load vector.

```
gf(nnode) = gf(nnode) + vbc(1,2);
```

- The mixed boundary condition $\alpha u(b) + \beta u'(b) = \gamma$, $\beta \neq 0$. The boundary condition can be re-written as

$$\begin{aligned} -k(b)u'(b) &= \frac{\alpha}{\beta} k(b) \left(u(b) - \frac{\gamma}{\alpha} \right) \\ &= u_{xmb} \left(u(b) - u_{bb} \right) = \mathbf{vbc}(\mathbf{1}, \mathbf{2}) \left(u(b) - \mathbf{vbc}(\mathbf{2}, \mathbf{2}) \right) \end{aligned}$$

  We need to change both the global stiffness matrix and the global load vector.

```
gf(nnode) = gf(nnode) + vbc(1,2)*vbc(2,2);
gk(nnode,nnode) =  gk(nnode,nnode) + vbc(1,2);
```

8.4.7 An example:

To check the code, we always try to compare the numerical results with some known exact solution. We take an exact solution as

$$u(x) = \sin x, \qquad a \leq x \leq b.$$

The material parameters are

$$k(x) = 1 + x, \quad c(x) = \cos x, \quad q(x) = x^2.$$

The right hand side can be calculated as

$$f(x) = (ku')' + cu' + qu = (1 + x) \sin x - \cos x + \cos^2 x + x^2 \sin x.$$

These functions are defined in the Matlab code getmat.m

```
function [xk,xc,xq,xf] = getmat(x);
  xk = 1+x; xc = cos(x); xq = x*x;
  xf = (1+x)*sin(x)-cos(x)+cos(x)*cos(x)+x*x*sin(x);
```

The triangulation is defined in the Matlab code datain.m. All other parameters used for the finite element method are defined in the Matlab code propset.m which include the following:

- The boundary $x = a$ and $x = b$, for example, $a = 1$, $b = 4$.
- The number of nodal points, for example, $nnode = 41$.
- The choice of basis functions. If we use the same basis function, then we can set, for example, $inf\_ele = 2$, which is the quadratic basis function $kind(i) = inf\_ele$;.
- The number of elements. If we use uniform elements, then $nelem = (nnode - 1)/inf\_ele$. We need to make it an integer.
- The choice of Gaussian quadrature formula, $nint(i) = 4$, for an example. The order of Gaussian quadrature formula should be the same or higher order than the basis functions. Otherwise it may not converge! For examples, if we use linear element, *i.e.* $inf\_ele = 1$, then we can choose $nint(i) = 1$, or $nint(i) = 2$ etc.
- Determine the boundary conditions, $kbc(1)$ and $kbc(2)$ and $vbc(i, j)$, $i, j = 1, 2$. Note that the linear system of equation is singular if both boundary conditions are Neumann because the solution either does not exist or is not unique.

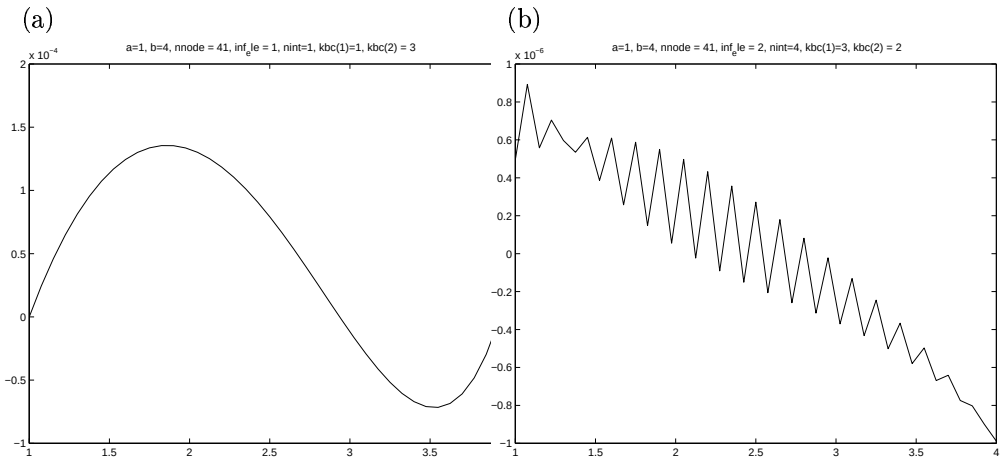


Figure 8.6. *Error of the finite element solutions at nodal poins. (a): The result is obtained with piecewise linear basis function and Gaussian quadrature of order one in the interval* $[1,4]$. *Dirichlet boundary condition at* $x = a$ *and mixed boundary condition* $3u(b)+4u'(b) = \gamma$ *from the exact solution* $u(x) = \sin x$ *are used. The magnitude of the error is order of* $O(10^{-4})$. *(b): Mixed boundary condition* $3u(a) + 4u'(a) = \gamma$ *at* $x = a$ *and the Neumann boundary condition at* $x = b$ *from the exact solution are used. The result is obtained with piecewise quadratic basis function and Gaussian quadrature of order four. The magnitude of the error is order of* $O(10^{-6})$.

To run the program, simple type the following in the Matlab:

```
[x,u]= fem1d;
```

To find out the detailed usage of the FEM code, read README carefully. Fig.8.6 gives the error plots of two different boundary conditions.

8.5 The finite element method for higher order equations in one-dimension

Example:

$$u'''' + q(x)u = f(x), \quad 0 \le x \le 1,$$
$$I : u(0) = u'(0) = 0, \quad u(1) = u'(1) = 0.$$

$$\text{or} \quad II \quad u(1) = 0, \quad u''(1) = 0,$$

$$\text{or} \quad III \quad u''(1) = 0, \quad u'''(1) = 0.$$

The weak form: Multiply a testing function $v(x) \in V$ and integrate by parts to get

$$\int_0^1 (u'''' + q(x)u)vdx = \int_0^1 fvdx,$$
$$u'''v\mid_0^1 - \int_0^1 u'''v'dx + \int_0^1 quvdx = \int_0^1 fvdx,$$
$$u'''v\mid_0^1 - u''v'\mid_0^1 + \int_0^1 \left(u''v'' + quv\right)dx = \int_0^1 fvdx,$$
$$u'''(1)v(1) - u'''(0)v(0) - u''(1)v'(1) + u''(0)v'(0) + \int_0^1 \left(u''v'' + quv\right)dx = \int_0^1 fvdx.$$

For the first case $u(0) = u'(0) = 0$, $u(1) = u'(1) = 0$, we do not know $u'''(0)$, $u'''(1)$, $u''(0)$, and $u''(1)$, therefore we should set

$$v(0) = v'(0) = v(1) = v'(1) = 0. \tag{8.28}$$

They are essential boundary conditions at $x = 0$ and $x = 1$. The weak form is

$$a(u, v) = f(v), \tag{8.29}$$

where the bi-linear form and the linear form are

$$a(u, v) = \int_0^1 \left(u''v'' + quv\right)dx, \tag{8.30}$$
$$L(v) = \int_0^1 fvdx. \tag{8.31}$$

Since the weak form involve second order derivatives, the solution space is

$$H_0^2 = \left\{v(x), \quad v(0) = v'(0) = v(1) = v'(1) = 0, v \in L^2, v' \in L^2, v'' \in L^2\right\}. \tag{8.32}$$

From the Sobolev embedding theorem we know that $H^2 \subset C^1$.

For the boundary condition $u(1) = u''(1) = 0$, we still have $v(1) = 0$ since $u'''(1)$ is unknown. However, there is no restriction on $v'(1)$ and the solution space is

$$H_E^2 = \left\{v(x), \quad v(0) = v'(0) = v(1) = 0, v \in H^2(0, 1)\right\}. \tag{8.33}$$

For the boundary condition $u''(1) = u'''(1) = 0$, there are no restrictions on both $v(1)$ and $v'(1)$ and the solution space is

$$H_E^2 = \left\{v(x), \quad v(0) = v'(0) = 0, v \in H^2(0, 1)\right\}. \tag{8.34}$$

For non-homogeneous *natural or mixed boundary conditions*, the weak form and the linear form may be different. For non-homogeneous *essential boundary conditions*, the weak form and the linear form will be the same. We have to do something to adjust the essential boundary conditions.

8.5.1 The finite element method.

Given a triangulation

$$0 = x_0 < x_1 < x_2 < \cdots < x_M = 1,$$

we want to construct a finite dimensional space V_h. For conforming finite element methods, $V_h \in H^2(0,1)$. Therefore we can not use the piecewise linear functions since they are in the Sobolev space H^1 but not in H^2.

How about piecewise quadratic functions? Theoretically, we can find a finite dimensional space of piecewise quadratic functions that is a subset of H^2. But nobody likes to use it because the basis function has very large support that involve at least six nodes. The most successful conforming finite dimensional space is the piecewise cubic functions over the triangulation

$$V_h = \left\{ v(x), \quad v(x) \quad \text{is piecewise cubic function}, \quad v \in H^2(0,1) \right\}. \tag{8.35}$$

The degree of freedom. On each element, we need four parameters to determine a cubic function. For essential boundary conditions at both $x = a$ and $x = b$, the total parameters are $4M$ for M elements. At each interior nodal point, the cubic is continuous and its derivative is also continuous, in addition, there are four boundary conditions, therefore, the dimension of the finite element space is

$$4M - 2(M-1) - 4 = 2(M-1).$$

Construct the basis functions

Since the derivative has to be continuous, we use piecewise Hermite interpolation and construct the basis function in two categories. The first category is

$$\phi_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \quad \text{and} \quad \phi_i'(\mathbf{x_j}) = \mathbf{0}, \quad \text{for any } x_j. \end{cases} \tag{8.36}$$

In other words, the basis functions in this group have unity at one node, and are zero at other nodes. The derivatives are zero at all nodes. To construct the local basis function in the element $[x_i, x_{i+1}]$, we can set

$$\phi_i(x) = \frac{(x - x_{i+1})^2 \left(a(x - x_i) + 1\right)}{(x_i - x_{i+1})^2}.$$

It is obvious that $\phi_i(x_i) = 1$, $\phi_i(x_{i+1}) = \phi_i'(x_{i+1}) = 0$, *i.e.*, x_{i+1} is a double root of the polynomial. We use $\phi'(x_i) = 0$ to find the coefficient a, so finally we will get

$$\phi_i(x) = \frac{(x - x_{i+1})^2 \left(\dfrac{2(x - x_i)}{(x_{i+1} - x_i)} + 1\right)}{(x_i - x_{i+1})^2}. \tag{8.37}$$

The global basis function can be written

$$\phi_i(x) = \begin{cases} 0 & \text{if } x \le x_{i-1} \\ \dfrac{(x-x_{i-1})^2\left(\dfrac{2(x-x_i)}{(x_{i-1}-x_i)}+1\right)}{(x_i-x_{i-1})^2} & \text{if } x_{i-1} \le x \le x_i \\ \dfrac{(x-x_{i+1})^2\left(\dfrac{2(x-x_i)}{(x_{i+1}-x_i)}+1\right)}{(x_i-x_{i+1})^2} & \text{if } x_i \le x \le x_{i+1} \\ 0 & \text{if } x_{i+1} \le x. \end{cases} \tag{8.38}$$

There are $M-1$ such basis functions. The second group of basis functions satisfy

$$\bar{\phi}_i'(x_j) = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \bar{\phi}_{\mathbf{i}}(\mathbf{x_j}) = \mathbf{0}, \text{for any } x_j. \tag{8.39}$$

In other words, the basis functions in this group are zero at all nodes, and the derivatives have unity at one node, and are zero at other nodes. To construct the local basis function in the element $[x_i, x_{i+1}]$, we can set

$$\bar{\phi}_i(x) = C(x-x_i)(x-x_{i+1})^2$$

since x_i and x_{i+1} are zeros of the cubic and x_{i+1} is a double root of the cubic. The constant C is chosen such that $\psi_i'(x_i) = 1$, thus we finally get:

$$\bar{\phi}_i(x) = \frac{(x-x_i)(x-x_{i+1})^2}{(x_i-x_{i+1})^2}. \tag{8.40}$$

The global basis function is then

$$\bar{\phi}_i(x) = \begin{cases} 0 & \text{if } x \le x_{i-1} \\ \dfrac{(x-x_i)(x-x_{i-1})^2}{(x_i-x_{i-1})^2} & \text{if } x_{i-1} \le x \le x_i \\ \dfrac{(x-x_i)(x-x_{i+1})^2}{(x_{i+1}-x_i)^2} & \text{if } x_i \le x \le x_{i+1} \\ 0 & \text{if } x_{i+1} \le x. \end{cases} \tag{8.41}$$

8.5.2 Shape functions.

There are four shape functions in the interval $[-1, 1]$, they are

$$\psi_1(\xi) = \frac{(\xi-1)^2(\xi+2)}{4}, \tag{8.42}$$

$$\psi_2(\xi) = \frac{(\xi+1)^2(-\xi+2)}{4}, \tag{8.43}$$

$$\psi_3(\xi) = \frac{(\xi-1)^2(\xi+1)}{4}, \tag{8.44}$$

$$\psi_4(\xi) = \frac{(\xi+1)^2(\xi-1)}{4}. \tag{8.45}$$

Note: There are two basis functions that are *centered* at each node. Such note is called a *double node*. The finite element solution can be written as

$$u_h(x) = \sum_{i=1}^{M-1} \alpha_j \phi_j(x) + \sum_{i=1}^{M-1} \beta_j \bar{\phi}_j(x). \tag{8.46}$$

After we find the coefficients α_j and β_j, then we have

$$u_h(x_j) = \alpha_j\,, \qquad u_h'(x_j) = \beta_j\,.$$

There are four non-zero basis functions on each element $[x_i, x_{i+1}]$. If we arrange the basis function in the order of $\phi_1, \phi_2, \psi_1, \psi_2, \phi_3, \phi_3, \psi_3, \cdots$, then the local stiffness matrix has the form

$$\begin{bmatrix} a(\phi_i,\phi_i) & a(\phi_i,\phi_{i+1}) & a(\phi_i,\bar{\phi}_i) & a(\phi_i,\bar{\phi}_{i+1}) \\ a(\phi_{i+1},\phi_i) & a(\phi_{i+1},\phi_{i+1}) & a(\phi_{i+1},\bar{\phi}_i) & a(\phi_{i+1},\bar{\phi}_{i+1}) \\ a(\bar{\phi}_i,\bar{\phi}_i) & a(\bar{\phi}_i,\phi_{i+1}) & a(\bar{\phi}_i,\bar{\phi}_i) & a(\bar{\phi}_i,\bar{\phi}_{i+1}) \\ a(\bar{\phi}_{i+1},\bar{\phi}_i) & a(\bar{\phi}_{i+1},\bar{\phi}_{i+1}) & a(\bar{\phi}_{i+1},\bar{\phi}_i) & a(\bar{\phi}_{i+1},\bar{\phi}_{i+1}) \end{bmatrix}_{(x_i,x_{i+1})}.$$

The global stiffness matrix is still banded matrix with band width being 6.

Question:

1. Can we use this finite dimensional space to solve the second order differential equation

$$-(pu')' + qu = f?$$

2. Consider the general fourth order two-point boundary value problems

$$a_4 u'''' + a_3 u''' + a_2 u'' + a_1 u' + a_0 u = f, \quad a \le x \le b$$

with the following mixed boundary conditions

$$2u'''(a) - u''(a) + \gamma_1 u'(a) + \rho_1 u(a) = \delta_1 \tag{8.47}$$

$$u'''(a) + u''(a) + \gamma_2 u'(a) + \rho_2 u(a) = \delta_2 \tag{8.48}$$

$$u(1) = 0 \tag{8.49}$$

$$u'(1) = 0. \tag{8.50}$$

Derive the weak form for this problem. **Hint:** Solve for $u'''(a)$ and $u''(a)$ from (8.57) and (8.58). The weak form should only involve up to second order derivatives.

8.6 Lax-Milgram Lemma, Existence and Uniqueness of FEM

We want to know the following for finite element methods:

- What kind of weak form should we use?
- Existence and uniqueness of the solution.
- Minimization form or weak form?
- Uniform treatments for any dimensions.

Reference: C. Johnson's book: *Numerical solution of Partial Differential Equations by the Finite Element Method.*

8.6.1 General settings: Assumptions, conditions

Let V be a Hilbert space with inner product $(\ ,\ )_V$ and the norm $\|u\|_V = \sqrt{(u,u)_V}$. Examples include C^m, Sobolev spaces H^1, H^2, etc. Assume that there is a *bilinear* form,

$$a(u,v), \quad V \times V \longmapsto R,$$

and a linear form

$$L(v), \quad V \longmapsto R,$$

that satisfy the following conditions

1. $a(u,v)$ is symmetric, i.e. $a(u,v) = a(v,u)$.
2. $a(u,v)$ is continuous with both u and v which means there is a constant γ such that

   $$|a(u,v)| \leq \gamma \|u\|_V \|v\|_V,$$

   for any $u \in V$ and $v \in V$. If this condition is true, $a(u,v)$ is a called a bounded operator. The least lower bound of such γ is called the norm of the operator $a(u,v)$.
3. $a(u,v)$ is V-elliptic which means that there is a constant α such that

   $$a(v,v) \geq \alpha \|v\|_V^2$$

   for any $v \in V$. It is also called *coercive.*
4. L is continuous which means there is a constant Λ such that

   $$|L(v)| \leq \Lambda \|v\|_V,$$

   for any $v \in V$.

8.6.2 Conclusions: Lax-Milgram Lemma

Under the conditions (2)-(4), there *exists* a *unique* element $u \in V$ such that

$$a(u, v) = L(v), \quad \forall\, v \in V.$$

Furthermore, if the condition (1) is also true, that is $a(u, v)$ is symmetric, then the followings are true

1. $\|u\|_V \leq \dfrac{\Lambda}{\alpha}$.
2. u is also the unique global minimizer of

$$F(v) = \frac{1}{2}a(v, v) - L(v).$$

Sketch of the proof. The Riesz's representation theorem states: *Any bounded linear operator, e.g. $L(v)$, in a Hilbert space, a linear space with an inner product, e.g. $a(u, v)$, is a unique special inner product of an element in the space.* In our case, the Hilbert space is V; the inner product in V is $a(u, v)$; the bounded linear operator is $L(v)$, therefore there is unique element u^* in V such that

$$L(v) = a(u^*, v), \quad \forall\, v \in V.$$

The a-norm is equivalent to V norm. From the continuity condition of $a(u, v)$, we have

$$\|u\|_a = \sqrt{a(u, u)} \leq \sqrt{\gamma \|u\|_V^2} = \sqrt{\gamma}\, \|u\|_V.$$

From the V-elliptic condition we have

$$\|u\|_a = \sqrt{a(u, u)} \geq \sqrt{\alpha \|u\|_V^2} = \sqrt{\alpha}\, \|u\|_V.$$

Therefore

$$\sqrt{\alpha}\, \|u\|_V \leq \|u\|_a \leq \sqrt{\gamma}\, \|u\|_V.$$

or

$$\frac{1}{\sqrt{\gamma}}\, \|u\|_a \leq \|u\|_V \leq \frac{1}{\sqrt{\alpha}} \|u\|_a.$$

$F(u^*)$ is the global minimizer. For any $v \in V$,

$$\begin{aligned}
F(v) = F(u^* + v - u^*) = F(u^* + w) &= \frac{1}{2}a(u^* + w, u^* + w) - L(u^* + w) \\
&= \frac{1}{2}\left(a(u^* + w, u^*) + a(u^* + w, w)\right) - L(u^*) - L(w) \\
&= \frac{1}{2}\left(a(u^*, u^*) + a(w, u^*) + a(u^*, w) + a(w, w)\right) - L(u^*) - L(w) \\
&= \frac{1}{2}a(u^*, u^*) - L(u^*) + \frac{1}{2}a(w, w) + a(u^*, w) - L(w) \\
&= F(u^*) + \frac{1}{2}a(w, w) - 0 \\
&\geq F(u^*).
\end{aligned}$$

Proof of the stability.

$$\alpha\|u^*\|_V^2 \le a(u^*,u^*) = L(v^*) \le \Lambda\|u^*\|_V.$$

Therefore

$$\alpha\|u^*\|_V^2 \le \Lambda\|u^*\|_V, \qquad \Longrightarrow \qquad \|u^*\|_V \le \frac{\Lambda}{\alpha}.$$

Remark: Lax-Milgram Lemma is often used to prove the existence and uniqueness of the solutions of partial differential equations.

8.6.3 An example of Lax-Milgram theorem.

Consider the one dimensional Sturm-Loivill problem again,

$$\begin{aligned} &-(pu')' + qu = f, \quad a \le x \le b, \\ &u(0) = 0, \quad \tilde{\alpha}u(b) + \tilde{\beta}u'(b) = \tilde{\gamma}, \quad \tilde{\beta} \ne 0, \quad \frac{\tilde{\alpha}}{\tilde{\beta}} \ge 0. \end{aligned}$$

The bi-linerar form is

$$a(u,v) = \int_a^b \left(pu'v' + quv\right) dx + \frac{\tilde{\alpha}}{\tilde{\beta}} p(b)u(b)v(b),$$

and the linear form is

$$L(v) = (f,v) + \frac{\tilde{\gamma}}{\tilde{\beta}} p(b)v(b).$$

The space is $V = H_E^1(a,b)$. To show the conditions of the Lax-Milgram theorem, we need to have the Poincare inequality:

Theorem 8.1. *If $v(x) \in H^1$ and $v(a) = 0$, then*

$$\int_a^b v^2 dx \le (b-a)^2 \int_a^b |v'(x)|^2 dx, \quad \text{or} \quad \int_a^b |v'(x)|^2 dx \ge \frac{1}{(b-a)^2} \int_a^b v^2 dx. \quad (8.51)$$

Proof:

$$v(x) = \int_a^x v'(t)dt, \qquad \Longrightarrow$$

$$|v(x)| \le \int_a^x |v'(t)|dt \le \left\{\int_a^x |v'(t)|^2 dt\right\}^{1/2} \left\{\int_a^x dt\right\}^{1/2} \le \sqrt{b-a}\left\{\int_a^b |v'(t)|^2\right\}^{1/2}$$

Therefore

$$v^2(x) \le (b-a)\int_a^b |v'(t)|^2 dt, \qquad \Longrightarrow$$

$$\int_a^b v^2(x)dx \le (b-a)\int_a^b |v'(t)|^2 dt \int_a^b dx \le (b-a)^2 \int_a^b |v'(x)|dx.$$

Verify the Lax-Milgram Lemma conditions for the Sturm-Loiville problem.

- It is obvious that $a(u,v) = a(v,u)$.
- The bi-linear form is continuous:

$$\begin{aligned}
|a(u,v)| &= \left| \int_a^b \left(pu'v' + quv \right) dx + \frac{\tilde{\alpha}}{\tilde{\beta}} p(b)u(b)v(b) \right| \\
&\leq \max\{\, p_{max},\, q_{max} \,\} \left(\int_a^b \left(|u'v'| + |uv| \right) dx + \frac{\tilde{\alpha}}{\tilde{\beta}} |u(b)v(b)| \right) \\
&\leq \max\{\, p_{max},\, q_{max} \,\} \left((u,v)_1 + \frac{\tilde{\alpha}}{\tilde{\beta}} |u(b)v(b)| \right) \\
&\leq \max\{\, p_{max},\, q_{max} \,\} \left(\|u\|_1 \|v\|_1 + \frac{\tilde{\alpha}}{\tilde{\beta}} |u(b)v(b)| \right).
\end{aligned}$$

  Notice that

$$\begin{aligned}
|u(b)v(b)| &= \left| \int_a^b u'(x)dx \int_a^b v'(x)dx \right| \\
&\leq (b-a) \sqrt{\int_a^b |u'(x)|^2 dx} \sqrt{\int_a^b |v'(x)|^2 dx} \\
&\leq (b-a) \sqrt{\int_a^b \left(|u'(x)|^2 + |u'(x)|^2 \right) dx} \sqrt{\int_a^b \left(|v'(x)|^2 + |v(x)|^2 \right) dx} \\
&\leq (b-a) \|u\|_1 \|v\|_1 .
\end{aligned}$$

  Therefore

$$|a(u,v)| \leq \max\{\, p_{max},\, q_{max} \} \left(1 + \frac{\tilde{\alpha}}{\tilde{\beta}} (b-a) \right) \|u\|_1 \|v\|_1 .$$

  That is

$$\gamma = \max\{\, p_{max},\, q_{max} \} \left(1 + \frac{\tilde{\alpha}}{\tilde{\beta}} (b-a) \right).$$

- $a(v,v)$ is V-elliptic.

$$\begin{aligned}
a(v,v) &= \int_a^b \left(p(v')^2 + qv^2 \right) dx + \frac{\tilde{\alpha}}{\tilde{\beta}} p(b)v(b)^2 \\
&\geq \int_a^b p(v')^2 dx \\
&\geq p_{min} \int_a^b (v')^2 dx \\
&= p_{min} \left(\frac{1}{2} \int_a^b (v')^2 dx + \frac{1}{2} \int_a^b (v')^2 dx \right) \\
&\geq p_{min} \left(\frac{1}{2} \frac{1}{b-a} \int_a^b v^2 dx + \frac{1}{2} \int_a^b (v')^2 dx \right) \\
&= p_{min} \, \min \left\{ \frac{1}{2(b-a)}, 1 \right\} \|v\|_1^2 .
\end{aligned}$$

That is

$$\alpha = p_{min} \min\left\{\frac{1}{2(b-a)}, 1\right\}.$$

- $L(v)$ is continuous

$$\begin{aligned} L(v) &= \int_a^b f(x)v(x)dx + \frac{\tilde{\gamma}_1}{\tilde{\beta}}p(b)v(b) \\ &\leq (f,v)_0 + \left|\frac{\tilde{\gamma}_1}{\tilde{\beta}}\right| p(b)\|v\|_1 \\ &\leq \|f\|_0\|v\|_1 + \left|\frac{\tilde{\gamma}_1}{\tilde{\beta}}\right| p(b)\|v\|_1 \\ &\leq \left(\|f\|_0 + \left|\frac{\tilde{\gamma}_1}{\tilde{\beta}}\right| p(b)\right)\|v\|_1. \end{aligned}$$

That is

$$\Lambda = \|f\|_0 + \left|\frac{\tilde{\gamma}_1}{\tilde{\beta}}\right| p(b).$$

Therefore we have proved that the conditions of the Lax-Milgram lemma are true with certain assumptions such as $p(x) \geq p_{min} > 0$, $q(x) \geq 0$, etc. We can conclude that there is unique solution in $H_e^1(a,b)$ to the original differential equation. The solution also satisfies:

$$\|u\|_1 \leq \frac{\|f\|_0 + \left|\tilde{\gamma}/\tilde{\beta}\right| p(b)}{p_{min} \min\left\{\frac{1}{2(b-a)}, 1\right\}}.$$

8.6.4 Abstract finite element method.

With the same settings, we assume that V_h is a finite dimensional subspace of V. Let $\{\phi_1, \phi_2, \cdots \phi_M\}$ be a basis for V_h. We can now formulate the following discrete analogue of the problem. Find $u_h \in V_h$ such that

$$a(u_h, v) = L(v), \qquad \forall v \in V_h, \tag{8.52}$$

or

$$F(u_h) \leq F(v), \qquad \forall v \in V_h. \tag{8.53}$$

Using the weak form, the weak form in V_h is equivalent to

$$a(u_h, \phi_i) = L(\phi_i), \qquad i = 1, \cdots, M. \tag{8.54}$$

Let the solution u_h be

$$u_h = \sum_{j=1}^{M} \alpha_j \phi_j.$$

Then from the weak form we get

$$a\left(\sum_{j=1}^{M}\alpha_j\phi_j,\phi_i\right)=\sum_{j=1}^{M}\alpha_j a(\phi_j,\phi_i)=L(\phi_i),\qquad i=1,\cdots,M,$$

which in matrix form is

$$AU=F,$$

where $U\in R^M$, $F\in R^M$ with $F(i)=L(\phi_i)$, and A is an $M\times M$ matrix with elements $A(i,j)=a(\phi_j,\phi_i)$. Since any element in V_h can be written as

$$v=\sum_{i=1}^{M}\eta_i\phi_i,$$

we have

$$a(v,v)=a\left(\sum_{i=1}^{M}\eta_i\phi_i,\sum_{j=1}^{M}\eta_j\phi_j\right)=\sum_{i,j=1}^{M}\eta_i a(\phi_i,\phi_j)\eta_j=\eta^T A\eta>0$$

as long as $\eta^T=\{\eta_1,\cdots\eta_M\}\neq 0$. Therefore A is symmetric positive definite.

The minimization form in the discrete case is

$$\frac{1}{2}U^TAU-F^T\mathbf{U}=\min_{\eta\in R^M}\left[\frac{1}{2}\eta^TA\eta-F^TU\right].\tag{8.55}$$

Existence and uniqueness of discrete problem. Since A is symmetric positive definite, A is invertible. There is a unique solution to the discrete weak form. Also from the conditions of Lax-Milgram lemma, we have

$$\alpha\|u_h\|_V^2\le a(u_h,u_h)=L(u_h)\le\Lambda\|u_h\|_V,$$

which gives

$$\|u_h\|_V\le\frac{\Lambda}{\alpha}.$$

Error estimates. Let the error vector be $e_h=u-u_h$, then

- $a(e_h,v_h)=(e_h,v_h)_a=0,\quad\forall v_h\in V_h$.
- $\|u-u_h\|_a=\sqrt{a(e_h,e_h)}\le\|u-v_h\|_a,\quad\forall v_h\in V_h$. That is u_h is the best approximation to u in the energy norm.
- $\|u-u_h\|_V\le\frac{\gamma}{\alpha}\|u-v_h\|_V,\quad\forall v_h\in V_h$, which gives the error estimates in the V norm.

Sketch of the proof: From the weak form, we have

$$a(u,v_h)=L(v_h),\quad a(u_h,v_h)=L(v_h),\quad\Longrightarrow\quad a(u-u_h,v_h)=0.$$

This means the finite element solution is the projection of u on to the space V_h. It is shown to be the best solution in V_h in the energy norm from the following:

$$\begin{aligned}\|u-v_h\|_a^2 &= a(u-v_h, u-v_h) = a(u-u_h+w_h, u-u_h+w_h)\\ &= a(u-u_h, u-u_h) + a(u-u_h, w_h) + a(w_h, u-u_h) + a(w_h, w_h)\\ &= a(u-u_h, u-u_h) + a(w_h, w_h)\\ &\geq \|u-u_h\|_a^2,\end{aligned}$$

where $w_h = u_h - v_h \in V_h$. Finally from the condition (3), we have

$$\begin{aligned}\alpha\|u-u_h\|_V^2 &\leq a(u-u_h, u-u_h) = a(u-v_h) + a(u-u_h, w_h)\\ &= a(u-u_h, u-u_h+w) = a(u-u_h, u-v_h)\\ &\leq \gamma\|u-u_h\|_V\|u-v_h\|_V.\end{aligned}$$

The last inequality is obtained from condition (2).

8.7 Exercises

1. (Purpose: Review abstract finite element methods.) Consider the Sturm-Liouville problem

$$\begin{aligned}-u'' + u &= f, \quad 0 \leq x \leq \pi,\\ u(0) = 0, &\qquad u(\pi) + u'(\pi) = 1.\end{aligned}$$

   Let V_f be the finite dimensional space

$$V_f = span\,\{\, x,\ \sin(x), \sin(2x)\,\}.$$

   Find the best approximation to the solution of the weak form from V_f in the energy norm. You can use either analytic derivation or computer software packages, for example, Maple, Matlab, SAS, etc. to solve the problem. Take $f = 1$ for the computation. Compare with this approach with the finite element method using the hat basis functions.

2. Consider the Sturm-Liouville problem

$$\begin{aligned}-(pu')' + qu &= f, \quad a \leq x \leq b,\\ u(a) = 0, &\qquad u(b) = 0.\end{aligned}$$

   Given a triangulation $a = x_0 < x_1 \cdots < x_M = b$ and a finite element space

   $V_h = \{\ v(x),\ v(x)$ is piecewise cubic function over the triangulation, $v(x) \in H_0^1(a,b)\}$

   (a) Find the dimension of V_h.

   (b) Find all non-zero shape functions $\psi_i(\xi)$, $-1 \leq \xi \leq 1$ and plot them.

   (c) What is the size of the local stiffness matrix and load vector? Sketch of the assembling process.

(d) List a few advantages and disadvantages of this finite element space compared with the piecewise continuous linear finite dimensional spaces, i.e., the hat functions.

3. You need to down-load the files of one dimensional FEM code from the class homepage. Let

$$u(x) = e^x \sin x, \quad p(x) = 1 + x^2, \quad q(x) = e^{-x}, \quad c(x) = 1,$$

and $f(x)$ is determined from the following differential equation

$$-(pu')' + c(x)u' + qu = f, \quad a \le x \le b.$$

Use this example to get familiar with the FEM package by trying the following boundary conditions.

(a) Dirichlet BC at $x = a$ and $x = b$, where $a = -1$, $b = 2$.

(b) Neumann BC at $x = a$ and Dirichlet BC at $x = b$, where $a = -1$, $b = 2$.

(c) Mixed BC, $\gamma = 3u(a) - 5u'(a)$, at $x = a = -1$, and Neumann BC at $x = b = 2$.

Tabulate the errors in the infinity norm

$$e_M = \max_{0 \le i \le M} |u(x_i) - U_i|$$

at the nodes and auxiliary points as the follows

M	Basis	Gaussian	error	e_M/e_{2M}

for different $M = 4, 8, 16, 32, 64$ (nnode=M+1), or the **closest integers** if necessary, with linear. quadratic, and cubic basis functions, **NO** plots please. What are the convergence orders? Note: the method is second, third, or fourth, order convergent if the ratio e_M/e_{2M} approaches 4, 8, 16 respectively.

For the **last case**, (1) print out the stiffness matrix for the *linear basis function* with $M = 5$. Is it symmetric? (2) Plot the computed solution against the exact one and the error plot for the case of the linear basis function. Take enough points to plot the exact solution so that we can see the whole picture. (3) Also plot the error versus $h = 1/M$ in the log-log scale for three different basis. The slope of such a plot is the convergence order of the method employed. For this problem, you will only produce *five* plots for the last case.

Extra Credit: Find the energy norm, H^1 norm, and L^2 norm of the error and do the grid refinement analysis.

4. Use the Lax-Milgram Lemma to show whether the following two-point value problem has a uniques solution or not.

$$\begin{aligned} &-u'' + q(x)u = f, \qquad 0 \le x \le 1, \\ &u'(0) = u'(1) = 0 \end{aligned} \tag{8.56}$$

where $q(x) \geq q_{min} > 0$. What happens if we relax the condition to $q(x) \geq 0$? Give counter examples if necessary.

5. Consider the general fourth order two-point boundary value problems

$$a_4 u'''' + a_3 u''' + a_2 u'' + a_1 u' + a_0 u = f, \quad a \leq x \leq b$$

with the following mixed boundary conditions

$$2u'''(a) - u''(a) + \gamma_1 u'(a) + \rho_1 u(a) = \delta_1 \tag{8.57}$$
$$u'''(a) + u''(a) + \gamma_2 u'(a) + \rho_2 u(a) = \delta_2 \tag{8.58}$$
$$u(1) = 0 \tag{8.59}$$
$$u'(1) = 0. \tag{8.60}$$

Derive the weak form for this problem. **Hint:** Solve for $u'''(a)$ and $u''(a)$ from (8.57) and (8.58). The weak form should only involve up to second order derivatives.

6. Given the following equation

$$-(pu')' + qu - \lambda u = 0 \tag{8.61}$$
$$u(0) = 0, \quad u(\pi) = 0. \tag{8.62}$$

   (a) Find the weak form of the problem.

   (b) Check whether the conditions of the Lax-Milgram Lemma are satisfied. Which condition is violated? Is the solution unique for arbitrary λ? *Note: It is obvious that $u = 0$ is a solution. For some λ, we can find non-trivial solution $u(x) \neq 0$. Such λ is called one eigenvalue of the system, and the non-zero solution is called eigenfunction corresponding to the eigenvalue.* The problem to find the eigenvalues and the eigenfunctions is called an eigenvalue problem.

   (c) Find all the eigenvalues and eigenfunctions when $p(x) = 1$ and $q(x) = 0$. **Hint:** $\lambda_1 = 1$ and $u(x) = \sin(x)$ is one pair of the solutions.

7. Using the one dimensional FEM package with linear basis function with a uniform grid to solve the following eigenvalue problem:

$$-(pu')' + qu - \lambda u = 0, \qquad 0 \leq x \leq \pi$$
$$u(0) = 0, \quad u'(\pi) + \alpha u(\pi) = 0,$$
$$\text{where} \quad p(x) \geq p_{min} > 0, \quad q(x) \geq 0, \quad \alpha \geq 0.$$

   (a) $p(x) = 1$, $q(x) = 1$, $\alpha = 1$.

   (b) $p(x) = 1 + x^2$, $q(x) = x$, $\alpha = 3$.

   Try to solve the eigenvalue problem with $M = 5$ and $M = 20$. Print out the eigenvalues but not the eigenfunctions. Plot all the eigenfunctions in a single plot

for $M = 5$ and two typical eigenfunctions for $M = 20$. So totally you will produce only 6-plots.

Hint: The approximate eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_M$ and the eigenfunction $u_{\lambda_i}(x)$ are the generalized eigenvalues of

$$Ax = \lambda Bx,$$

where A is the stiffness matrix, and $B = \{b_{ij}\}$ with $b_{ij} = \int_0^\pi \phi_i(x)\phi_j(x)dx$. You can either numerically or analytically to generate the matrix B. In matlab, you can use $[V, D] = EIG(A, B)$ to find the generalized eigenvalues and the corresponding eigenvectors. For a computed eigenvalue λ_i, the corresponding eigenfunction is

$$u_{\lambda_i}(x) = \sum_{j=1}^{M} \alpha_{i,j}\phi_j(x)$$

where $[\alpha_{i,1}, \alpha_{i,2}, \cdots, \alpha_{i,M}]^T$ is the eigenvector corresponding to the generalized eigenvalue.

Note: if we can find the eigenvalues and corresponding eigenfunctions, the solution to the differential equation then can be expanded in terms of the eigenfunctions, similar to the Fourier series.

8. (Application problem): Consider nuclear fuel element of spherical form, consisting of a sphere of "fissionable" material surrounded by a spherical shell of aluminum "cladding" as shown in the figure. We wish to determine the temperature distribution in the nuclear fuel element and the aluminum cladding.

   The governing equations for the two regions are the same, with the exception that there is no heat source term for the aluminum cladding. We have

$$-\frac{1}{r^2}\frac{d}{dr}r^2 k_1 \frac{dT_1}{dr} = q, \quad 0 \le r \le R_F$$
$$-\frac{1}{r^2}\frac{d}{dr}r^2 k_2 \frac{dT_2}{dr} = 0, \quad R_F \le r \le R_C$$

   where subscripts 1 and 2 refer to the nuclear fuel element and cladding, respectively. The heat generation in the nuclear fuel element is assumed to be of the form

$$q_1 = q_0\left[1 + c\left(\frac{r}{R_F}\right)^2\right]$$

   where q_0 and c are constants depending on the nuclear material. The boundary conditions are

$$kr^2\frac{dT_1}{dr} = 0, \quad r = 0, \quad \text{natural BC},$$
$$T_2 = T_0, \quad r = R_C$$

   where T_0 is a constant. Note the temperature at $r = R_F$ is continuous. Derive a weak form for this problem (**Hint:** multiply r^2 to both sides). Use two linear elements

$[0, R_F]$, and $[R_F, R_C]$, to determine the finite element solution, and compare the nodal temperature, $T(0)$ and $T(R_f)$, with the exact solution

$$T_1 = T_0 + \frac{q_0 R_F^2}{6k_1}\left\{\left[1-\left(\frac{r}{R_F}\right)^2\right] + \frac{3}{10}c\left[1-\left(\frac{r}{R_F}\right)^4\right]\right\} + \frac{q_0 R_F^2}{3k_2}\left(1+\frac{3}{5}c\right)\left(1-\frac{R_F}{R_C}\right)$$

$$T_2 = T_0 + \frac{q_0 R_F^2}{3k_2}\left(1+\frac{3}{5}c\right)\left(\frac{R_F}{r}-\frac{R_F}{R_C}\right)$$

Take $T_0 = 80$, $q_0 = 5$, $k_1 = 1$, $k_2 = 50$, $R_F = 0.5$, $R_C = 1$, $c = 1$ for plotting and comparison.

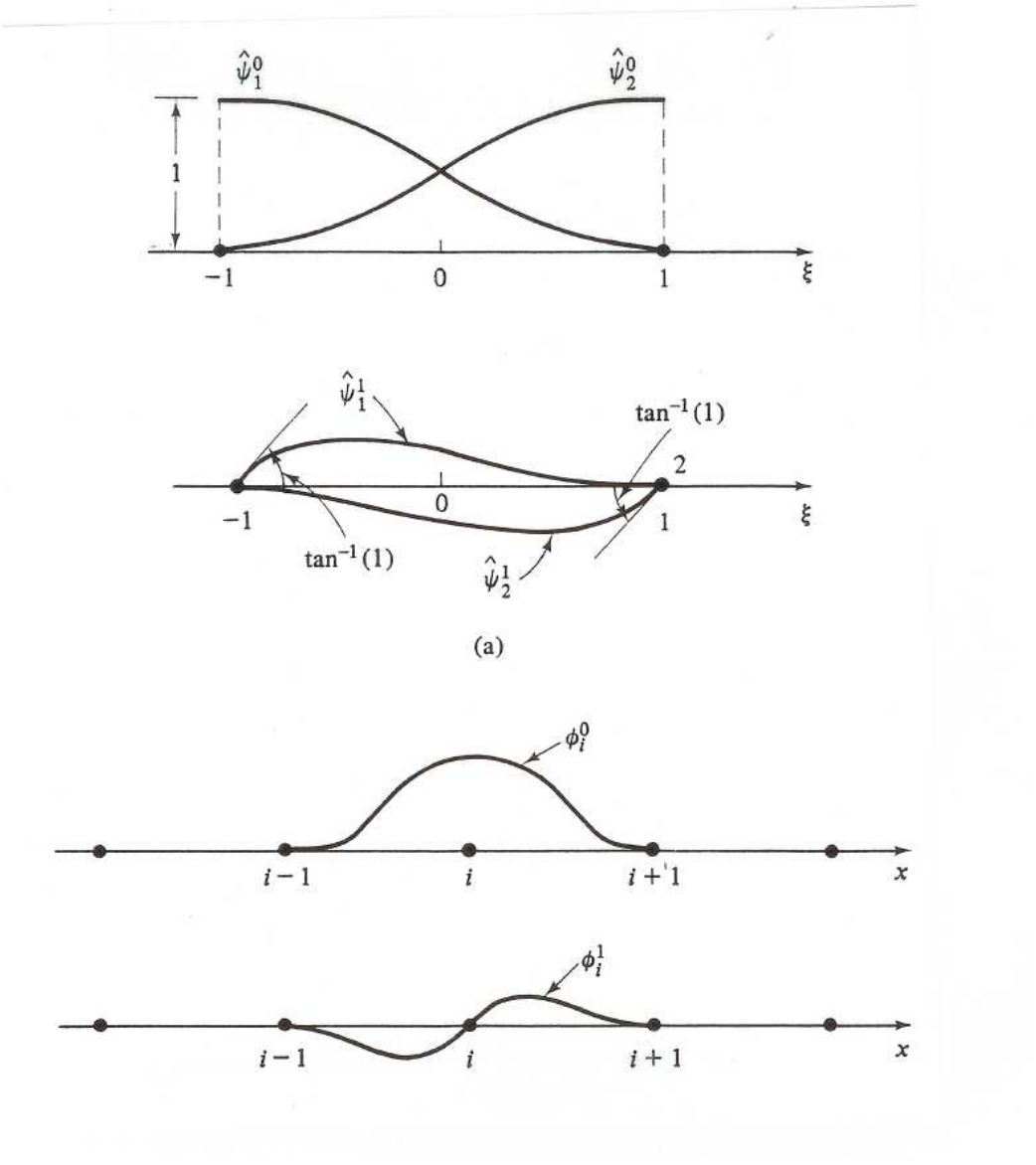
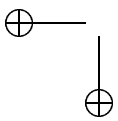
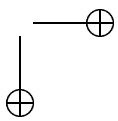


Figure 8.7. *Hermite cubic shape function on a master element and (b) corresponding global functions at a node i in the mesh.*





Bibliography

[1] J. Adams, P. Swarztrauber, and R. Sweet. Fishpack: Efficient Fortran subprograms for the solution of separable elliptic partial differential equations. http://www.netlib.org/fishpack/.

[2] D. Braess. *Finite Elements.* Cambridge University Press, 1997.

[3] S.C. Brenner and L.R. Scott. *The Mathematical Theory of Finite Element Methods.* Springer New York, 2002.

[4] D. Calhoun. A Cartesian grid method for solving the streamfunction-vorticity equation in irregular regions. *J. Comput. Phys.*, 176:231–275, 2002.

[5] G. F. Carey and J. T. Oden. *Finite Element, I-V.* Prentice-Hall, Inc, Englewood Cliffs, 1983.

[6] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[7] P. G. Ciarlet. *The finite element method for elliptic problems.* North Holland, 1978, and SIAM Classic in Applied Mathematics 40, 2002.

[8] D. De Zeeuw. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *J. Comput. Appl. Math.*, 33:1–27, 1990.

[9] L. C. Evans. *Partial Differential Equations.* AMS, 1998.

[10] G. Golub and C. Van Loan. *Matrix computations.* The Johns Hopkins University Press, 2nd ed., 1989.

[11] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations.* Cambridge, 2008.

[12] Jr. J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* SIAM, 1996.

[13] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Cambridge University Press, 1987.

[14] R. J. LeVeque. Clawpack and Amrclaw – Software for high-resolution Godunov methods. 4-th Intl. Conf. on Wave Propagation, Golden, Colorado, 1998.

[15] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations, Steady State and Time Dependent Problems.* SIAM, 2007.

[16] Z. Li and M-C. Lai. The immersed interface method for the Navier-Stokes equations with singular forces. *J. Comput. Phys.*, 171:822–842, 2001.

[17] Z. Li and C. Wang. A fast finite difference method for solving Navier-Stokes equations on irregular domains. *J. of Commu. in Math. Sci.*, 1:180–196, 2003.

[18] M. Minion. A projection method for locally refined grids. *J. Comput. Phys.*, 127:158–178, 1996.

[19] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations.* Cambridge press, 1995.

[20] Y. Saad. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[21] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method.* Prentice-Hall, 1973.

[22] J. C. Strikwerda. *Finite Difference Scheme and Partial Differential Equations.* Wadsworth & Brooks, 1989.

[23] J. W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods.* Springer New York, 1995.

Index