FINITE ELEMENT METHOD IN 1D – PART 1

We have in previous courses in mathematics solved differential equations both analytically (with pen and paper) and numerically (with a computer). Usually we have to be satisfied with a numerical solution, at least when the problem is a bit more complicated. In this computer lab we shall study the finite element method for the numerical solution of boundary value problems.

We only consider problems in one dimension for this lab. However, the methods we consider here can easily be extended to problems in two or three dimensions.

In the next computer lab, we will consider time-dependent problems, the heat and wave equations.

1. Boundary value problems

We consider the following second order differential equation on the interval (a, b),

$$-(cu')' = f \quad \text{in} \quad (a, b) \tag{1a}$$

where c and f are given functions. We consider different types of boundary conditions at the endpoints of the interval,

$$\begin{aligned} u(a) &= u_a \\ c(b)u'(b) &= q_b, \end{aligned} \tag{1b}$$

where u_a and q_b are given scalar values. The boundary condition where the value of the solution is prescribed, in our case $u(a) = u_a$, is called a *Dirichlet boundary condition*. The boundary condition, where the flux is prescribed, in our case $c(b)u'(b) = q_b$, is called a *Neumann boundary condition*.

The first step to solve the boundary value problem (1) is to reformulate it as a variational problem. To do this, we multiply the equation in (1a) with a smooth test function v and integrate over the interval (a, b),

$$-\int_a^b (cu')'v\,dx = \int_a^b fv\,dx. \tag{2}$$

Now we carry out integration by parts on the left hand side to get

$$\begin{aligned} -\int_a^b (cu')'v\,dx &= -\big[cu'v\big]_a^b + \int_a^b cu'v'\,dx \\ &= -c(b)u'(b)v(b) + c(a)u'(a)v(a) + \int_a^b cu'v'\,dx. \end{aligned}$$

If we choose the test function v such that $v(a) = 0$, and make use of the Neumann boundary condition at the right endpoint b, we can rearrange the terms in (2) to

obtain

$$\begin{aligned}\int_a^b cu'v'\,dx &= \int_a^b fv\,dx + \overbrace{c(b)u'(b)}^{=q_b}v(b) - c(a)u'(a)\overbrace{v(a)}^{=0} \\ &= \int_a^b fv\,dx + q_b v(b).\end{aligned} \tag{3}$$

Since v was an arbitrary test function with $v(a) = 0$, equation (3) must hold for any (reasonable) choice of v. To make a more precise statement, we introduce the function spaces[1]

$$V = \big\{v : \int v^2 + (v')^2\,dx < \infty \text{ and } v(a) = u_a\big\}, \tag{4a}$$

$$\widehat{V} = \big\{v : \int v^2 + (v')^2\,dx < \infty \text{ and } v(a) = 0\big\}. \tag{4b}$$

With this notation, the weak (or variational) formulation of the problem reads: Find $u \in V$ such that

$$\int_a^b cu'v'\,dx = \int_a^b fv\,dx + q_b v(b) \qquad \forall v \in \widehat{V}. \tag{5}$$

The weak (or variational) formulation (5) has several advantages over the original (strong) formulation (1). For example, we only have first order derivatives in the variational formulation and no derivative on the coefficient c. However, the two formulations are not completely equivalent – a solution to the variational problem is not necessarily a solution to the original problem, and vice versa.

2. Deriving the finite element method

We have derived the variational problem (5), which is easier to solve than the original problem. However, we still do not have a problem formulation that we can solve numerically, because the space of all possible solutions, V, is infinite dimensional. In order to make the problem solvable on a computer, we need to discretise it. That is, we must replace the infinite-dimensional space V with a finite-dimensional subspace V_h.

Let $a = x_1 < x_1 < \cdots < x_n = b$, be a partition of the interval $[a, b]$ into $n - 1$ subintervals of length $h_j = x_{j+1} - x_j$. Let V_h be the space of functions

$$V_h = \big\{v_h \in V : v_h \text{ is continuous on } [a, b] \text{ and linear on } (x_j, x_j),\ j = 1, \ldots, n-1\big\}.$$

The space $\widehat{V}_h \subset \widehat{V}$ is defined analogously.

The most convenient basis for V_h and $\widehat{V}_h$ are the so-called hat functions. The basis functions $\{\varphi_i\}_{i=1}^n$ are defined by the nodal property

$$\varphi_j(x_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

[1]The definitions given here are intentionally vague and not strictly mathematically correct, to keep the theory to a minimum. The derivatives in (4) are *generalised derivatives*, and do not need to be defined everywhere. The V as defined here is not a linear space, in order to keep the definition close to the practical implementation, but in a rigorous exposition of the finite element method the problem would be recast to replace inhomogeneous Dirichlet boundaries with homogeneous ones.

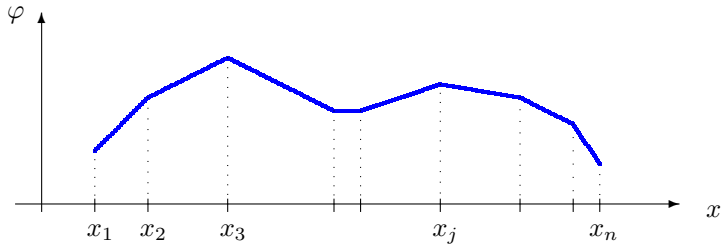

FIGURE 1. The finite element functions are continuous and piecewise linear.

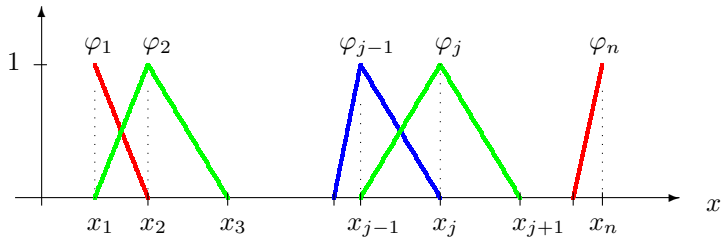

FIGURE 2. Finite element basis functions.

The expansion coefficients in this basis are the nodal values. That is, for $v_h \in V_h$, we have

$$v_h = \sum_{i=1}^{n} v_h(x_i)\varphi_i.$$

An explicit definition of the basis functions is also given[2]

$$\varphi_1(x) = \begin{cases} \dfrac{x_1 - x}{h_1} & \text{if } x_1 \le x \le x_2 \\ 0 & \text{otherwise} \end{cases}$$

$$\varphi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h_{i-1}} & \text{if } x_{i-1} \le x \le x_i \\ \dfrac{x_{i+1} - x}{h_i} & \text{if } x_i \le x \le x_{i+1} \\ 0 & \text{otherwise} \end{cases} \qquad i = 2, 3, \ldots, n-1$$

$$\varphi_n(x) = \begin{cases} \dfrac{x - x_n}{h_n} & \text{if } x_n \le x \le x_n \\ 0 & \text{otherwise.} \end{cases}$$

[2]Note that the basis functions here are not differentiable in the classical sense, but they are differentiable in the generalised sense – there is a function φ' such that for any smooth ψ, $\psi(a) = \psi(b) = 0$, the following integration by parts formula holds:

$$\int_a^b \varphi\psi'\,dx = -\int_a^b \varphi'\psi\,dx.$$

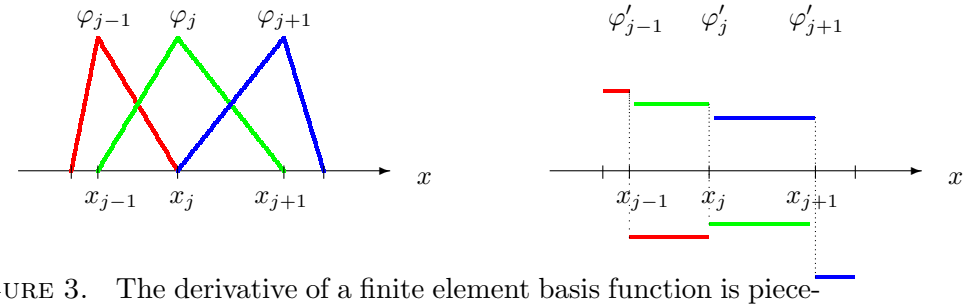

FIGURE 3. The derivative of a finite element basis function is piecewise constant.

The finite element method for the boundary value problem (1) can now be formulated: Find $u_h \in V_h$ such that

$$\int_a^b c u_h' v_h' \, dx = \int_a^b f v_h \, dx + q_b v_h(b) \qquad \forall v \in \widehat{V}_h. \tag{6}$$

The solution u_h can be written $u_h = \sum_{j=1}^n \xi_j \varphi_j$, where $\xi_j = u_h(x_j)$ are the unknown nodal values of the solution. Since the identity in (6) holds for all $v_h \in \widehat{V}_h$ if and only if it holds for all basis functions φ_i, we can write (6) as follows: Find $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots \xi_n)$ such that

$$\sum_{j=1}^n \xi_j \underbrace{\int_a^b c \varphi_j' \varphi_i' \, dx}_{=a_{i,j}} = \underbrace{\int_a^b f \varphi_i \, dx + q_b \varphi_i(b)}_{=b_i} \qquad i = 1, 2, \ldots, n. \tag{7}$$

With $\boldsymbol{A}$ as the matrix with entries $a_{i,j}$ and $b = (b_1, b_2, \ldots, b_n)$, this is a linear system in the standard form

$$\boldsymbol{A}\boldsymbol{\xi} = \boldsymbol{b}, \tag{8}$$

which can be solved with standard linear algebra methods.

3. FINITE ELEMENT ASSEMBLY

We will now look in detail at the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$, and how their entries are computed. Since the basis functions φ_i, $i = 1, \ldots, n$, are piecewise linear, their derivatives are piecewise constant, with

$$\varphi_i'(x) = \begin{cases} \dfrac{1}{h_{i-1}} & \text{if } x_{i-1} \le x \le x_i \\ \dfrac{-1}{h_i} & \text{if } x_i \le x \le x_{i+1} \\ 0 & \text{otherwise} \end{cases} \qquad i = 2, 3, \ldots, n-1$$

and with ϕ_1' and ϕ_n' being similar with the obvious modifications near the endpoints of the interval. See also Figure 3.

Note that $a_{i,j} = a_{j,i}$ and $a_{i,j} = 0$ if $|i - j| > 1$, so we need only compute a few of the matrix entries. For simplicity, we assume that the partitioning of the interval is

uniform, that is $h_i = h = (n-1)^{-1}$, $i = 1, \dots, n-1$. If we use the trapezoidal rule[3] to approximate the integrals over each subinterval (x_i, x_{i+1}), we get

$$\begin{aligned}
a_{1,1} &= \frac{1}{h^2}\int_{x_1}^{x_2} c\,dx \approx \frac{c(x_1)+c(x_2)}{2h} \\
a_{i,i} &= \frac{1}{h^2}\int_{x_{i-1}}^{x_{i+1}} c\,dx \approx \frac{c(x_{i-1})+2c(x_i)+c(x_{i+1})}{2h} \qquad i = 2, \dots, n-1 \\
a_{n,n} &= \frac{1}{h^2}\int_{x_{n-1}}^{x_n} c\,dx \approx \frac{c(x_{n-})+c(x_n)}{2h} \\
a_{i,i+1} = a_{i+1,i} &= -\frac{1}{h^2}\int_{x_i}^{x_{i+1}} c\,dx \approx -\frac{c(x_i)+c(x_{i+1})}{2h} \qquad i = 1, \dots, n-1.
\end{aligned}$$

The entries in the vector $\boldsymbol{b}$ can also be computed with the same quadrature rule. We get

$$\begin{aligned}
b_1 &= \int_{x_1}^{x_2} f\varphi_1\,dx \approx \frac{h}{2}f(x_1) \\
b_i &= \int_{x_{i-1}}^{x_{i+1}} f\varphi_i\,dx \approx hf(x_i) \qquad i = 2, \dots, n-1 \\
b_n &= \int_{x_{n-1}}^{x_n} f\varphi_n\,dx + q_b \approx \frac{h}{2}f(x_n) + q_b.
\end{aligned}$$

4. Handling Dirichlet boundary conditions

A careful comparison of the definition of the finite element spaces V_h and $\widehat{V}_h$ and basis $\{\varphi_i\}_{i=1}^n$ reveals that we have one too many basis functions – only functions with $v_h(a) = u_a$ are permitted in V_h. To remedy this, we need to eliminate one unknown from the linear system $\boldsymbol{A\xi} = \boldsymbol{b}$ by fixing the value at $x = a$.

The way we do this is simply replacing the equation for $i = 1$ in (7) with $\xi_1 = u_a$. In the linear system $\boldsymbol{A\xi} = \boldsymbol{b}$, this means replacing row 1 of the matrix of $\boldsymbol{A}$ with the row 1 of the identity matrix $\boldsymbol{I}_n$, and replacing b_1 with u_a.

After applying the Dirichlet boundary condition, the linear system $\boldsymbol{A\xi} = \boldsymbol{b}$ has a tridiagonal structure as follows (where it is empty there are only zeros):

$$\begin{bmatrix}
1 & & & & & & \\
a_{2,1} & a_{2,2} & a_{2,3} & & & & \\
 & \ddots & \ddots & \ddots & & & \\
 & & a_{i,i-1} & a_{i,i} & a_{i,i+1} & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\
 & & & & & a_{n,n-1} & a_{n,n}
\end{bmatrix}
\begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_i \\ \vdots \\ \xi_n \\ \xi_n \end{bmatrix}
=
\begin{bmatrix} u_a \\ hf_2 \\ \vdots \\ hf_i \\ \vdots \\ hf_n \\ hf_n/2 + q_b \end{bmatrix}$$

[3] Recall that with $h = x_2 - x_1$, the trapezoidal rule reads

$$\int_{x_1}^{x_2} v(x)\,dx \approx \frac{h}{2}v(x_1) + \frac{h}{2}v(x_2).$$

$n-1$	4	8	16	32	64
error	4.1399e-03	1.0207E-03	2.5429E-04	6.3516E-05	1.5876E-05
rate	n/a	2.0200	2.0050	2.0013	2.0003

TABLE 1. Estimated errors and covergence rates. The estimated convergence rates approaches the exact value of two when the size of the subintervals decreases.

5. Code verification

When have implemented a numerical scheme for the mathematical model, we need to make sure that it works correctly. It is easy to make mistakes when deriving the algorithm, or when implementing it in code. Approximations we make, such as replacing an exact integral with a quadrature rule, or using inexact solvers, may cause a loss of accuracy. Therefore, it is important that we test, in a rigorous manner, that the scheme works as well as expected.

The method of manufactured solutions in one way to test that the code works. We will choose an exact solution u to a boundary value problem (1) and derive f, u_a and q_b from the known solution. These will be inputs used to compute an approximate numerical solution u_h. Since the exact solution u is known, we can investigate the error $u - u_h$.

In order to evaluate the error in a rigorous manner, we make use of error estimates for the finite element method. For the method considered here, the following estimate holds.

$$\|u - u_h\|_2 \leq Ch^2. \tag{9}$$

Here, C is a constant that depends on the interval (a, b), the coefficient c, and the exact solution u, but not on h, and $\|\cdot\|_2$ is the norm

$$\|v\|_2 = \sqrt{\int_a^b v^2\,dx}$$

The factor h^2 in (9) means that finite element method has second order convergence in the mesh resolution h. More generally, we assume an error bound of the form

$$\|u - u_h\|_2 \leq Ch^r,$$

where r is order of convergence, which may be different from two if the method is not correctly implemented. When the exact solution is known, then r can be estimated from computed errors. For example, if we reduce the size of the subintervals from $2h$ to h, the order of convergence r can be estimateted:

$$\frac{\|u - u_{2h}\|_2}{\|u - u_h\|_2} \sim \frac{C(2h)^r}{Ch^r} \sim 2^r \quad \Longrightarrow \quad r \sim \log_2\left(\frac{\|u - u_{2h}\|_2}{\|u - u_h\|_2}\right). \tag{10}$$

If the estimated rates from (10) are consistent with the expected convergence rate of two from (9), it is a strong indication that the algorithm has been correctly implemented. To be confident about the convergence estimated rates, we need to compute them for a range of decreasing values of h, and it is common practice to tabulate the results as in Table 1.

Exercise 1. We consider the following boundary value problem

$$-u'' = (\pi^2/4)\sin(\pi x/2) \quad \text{in} \quad (0,1) \tag{11a}$$

with the boundary conditions

$$\begin{aligned} u(0) &= 0 \\ u'(1) &= 0. \end{aligned} \tag{11b}$$

We will solve (11) numerically with the finite element method. Throughout this exercise, we assume that the partitioning is uniform, i.e. $x_{i+1} - x_i = h = (n-1)^{-1}$, $i = 1, \ldots, n$.

(a) Compute, by hand, the matrix entries $a_{i,j}$ (without accounting for boundary conditions) when $c(x) = 1$. Compare to the matrix you would get from a finite difference stencil.

(b) Write a MATLAB function `[A] = StiffnessMatrix(N, c)` that computes the stiffness matrix $\boldsymbol{A}$ (without accounting for the boundary conditions), using the trapezoidal rule to approximate the integrals. You can assume that the domain is the unit interval $(0,1)$ and that the partitioning is uniform. The function inputs could be the number of subintervals $N = n-1$ and the coefficient function c, and the output should be a matrix in sparse format.

Test `StiffnessMatrix` with $c(x) = 1$ and check that you get the same values that you computed in (a).

(c) Write a MATLAB function `[b] = LoadVector(N, f)` that computes the load vector $\boldsymbol{b}$, without the accounting for the boundary conditions. The inputs can be the number of subintervals $N = n-1$ and the source term f. Check that the values are correct.

(d) Write a function MATLAB function that applies Dirichlet and Neumann type boundary conditions to the matrix A and vector b. For example, this could be a funcion `[A, b] = ApplyBCs(A, b, l_t, l_v, r_t, r_v)` where the other function inputs are

- `l_t` – type of boundary condition at $x = 0$ (Dirichlet or Neumann)
- `l_v` – boundary value at $x = 0$, i.e. the value $u(0)$ for a Dirichlet boundary condition or the flux $-cu'(0)$ for a Neumann boundary condition
- `r_t` – type of boundary condition at $x = 1$ (Dirichlet or Neumann)
- `r_v` – boundary value at $x = 1$, i.e. the value $u(1)$ for a Dirichlet boundary condition or the flux $cu'(1)$ for a Neumann boundary condition

(e) Use the functions you have implemented in (b)–(d) to compute a finite element solution to (11). Estimate the error, and plot the numerical solution together with the exact solution $u = \sin(\pi x/2)$. The code in the appendix can be used to estimate the error.

(f) Compute the errors for $N = 4, 8, 16, 32, 64$. Make a table similar to Table 1. Do you get second order convergence?

Exercise 2. We consider the following boundary value problem

$$-\left((1+x^2)u'\right)' = 10x^3 \quad \text{in} \quad (0,1) \tag{12a}$$

with the boundary conditions

$$\begin{aligned} u'(0) &= 0 \\ u(1) &= 1. \end{aligned} \tag{12b}$$

This problem has a unique solution, which is

$$u(x) = \frac{5}{2}\left(x - \frac{x^3}{3} - \arctan(x) - \frac{4}{15} + \frac{\pi}{4}\right). \tag{13}$$

Use the functions you implemented in Exercise 1 to solve this problem numerically. Plot the numerical solution together with the exact solution, and make table of estimated convergence rates.

Appendix A. Computing the error

The error in the numerical solution can be computed as

$$\|u - u_h\|_2 = \sqrt{\int_a^b (u - u_h)^2 \, dx}.$$

It is impractical to evaluate this integral exactly, so instead we use a numerical scheme to compute an approximate value. If the accuracy of the quadrature degree is too low, it might not provide a good estimate of the error. The code below is an example of the how the error can be computed, using a three-point Gaussian rule on each subinterval.

```
function e = Errornorm(u_e, u_h)
%ERRORNORM This function computes L2 error
%   Computes an estimated distance between the
%   exact solution and the numerical solution.
%   The integral is approximated with three-point
%   Gauss-Legendre quadrature rule
%       u_e   : Exact solution (function)
%       u_h   : Numerical solution (vector of nodal values)

n = length(u_h);       % number of subintervals
h = 1.0 / (n - 1);     % length of subintervals

% rule for reference interval [0, 1]
t = [ (1 - sqrt(3/5)) / 2 ; 1 / 2 ; (1 + sqrt(3/5)) / 2 ];
w = [ 5/18 ; 8/18 ; 5/18 ];

e2 = 0;
for i = 1:(n-1)
    % evaluate at quadrature points
    v_e = u_e( (i-1) * h  + h * t );
    v_h = (1-t) * u_h(i) + t * u_h(i+1);

    % square difference and sum with weights
    e2  = e2 + sum( (v_e -v_h).^2 .* w );
end
e = sqrt( h * abs(e2) );
end
```