LECTURE NOTES: FINITE ELEMENT METHOD

AXEL MÅLQVIST

1. Motivation

The finite element method has two main strengths.

1.1. **Geometry.** Very *complex geometries* can be used. This is probably the main reason why finite element methods are used so frequently in industrial applications. Simulations of flow around cars and airplanes, oil reservoir simulation where the computational domain has cracks, structural mechanics applications where stress and strain are computed for complicated devices, such as a car engine, are just a few examples.

1.2. **Strong mathematical foundation.** The finite element method has a very strong foundation in the theory of functional analysis developed in the first half of the 20th century. This gives us access to powerful tools when *analyzing the error of the approximate solution*. We recommend you to read the courses *Finita element metoder* and *Funktionalanalys* if you want to learn more.

2. Model problem

We consider the simplest possible second order boundary value problem, the Poisson equation.

2.1. **The Poisson equation in one dimension.**

$$\begin{cases} -u''(x) &= f(x), \quad x \in [0,1], \\ u(0) &= 0, \\ u(1) &= 0, \end{cases} \tag{2.1}$$

where $f(x)$ is a given function, defined on the interval $[0,1]$. We note that u belongs to the following space of functions,

$$V = \{v \in C([0,1]) : v(0) = v(1) = 0\}. \tag{2.2}$$

Date: September 22, 2009.

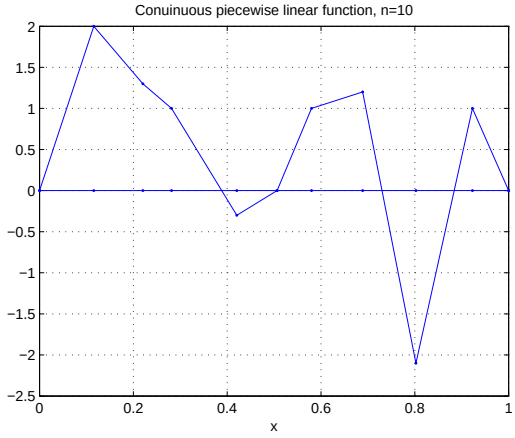

FIGURE 1. A continuous piecewise linear function.

2.2. **Applications.**

- An elastic bar fixed at the end points, Hooke's law $\sigma = Eu'$ and equilibrium equation $-\sigma' = f$, u is tangential displacement, σ is the stress, E is the module of elasticity, and f is a given load acting in the tangential direction.
- Stationary heat conduction, Fourier's law $-q = ku'$ and conservation of energy $q' = f$, where u is temperature, q heat flow, k heat conductivity, and f is a given heat source.

3. Continuous piecewise linear functions

In any numerical method it is crucial to discretize the computational domain $[0, 1]$.

3.1. **Discretization.** We let $0 = x_0 < x_1 < \cdots < x_j < \cdots < x_n = 1$ and define $h_j = x_j - x_{j-1}$, $j = 1, \ldots, n$, to be the local mesh size. We would like to compute an approximation u_h to u which is continuous and piecewise linear on the mesh given by the points $\{x_j\}_{j=0}^n$, see an example in Figure 1. We let V_h be the space of continuous piecewise linear basis functions,

(3.1)
$$V_h = \{v \in C([0,1]) : v(0) = v(1) = 0 \text{ and } v(x), x \in [x_j, x_{j+1}] \text{ is linear for all } j \in 0, \ldots, n-1\}.$$

We note that $V_h \subset V$.

3.2. **Basis for V_h.** The space V_h is a function space with $n-1$ degrees of freedom, the values at the inner nodes. There is a simple basis for V_h consisting of continuous piecewise linear functions that are one in one node and zero in all other nodes, see Figure 2. We will denote these functions $\phi_j(x)$, where $j = 1, \ldots, n-1$. We have,

$$V_h = \text{span}(\{\phi_j\}_{j=1}^{n-1}). \tag{3.2}$$

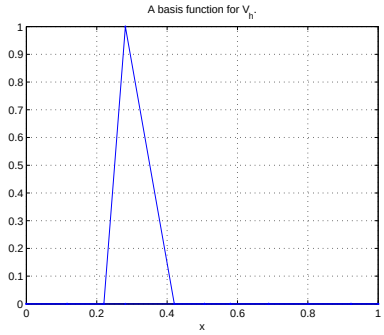

FIGURE 2. A basis function (hat-function) for V_h.

Note that $\dim(V_h) = n - 1$. In fact each function in V_h can be identified with a vector in $\mathbf{R}^{n-1}$, namely the vector of function values at the nodes $[v(x_1), v(x_2), \ldots, v(x_{n-1})]$, and each vector in $\mathbf{R}^{n-1}$ can be identified with a function in V_h, $v = \sum_{j=1}^{n-1} v(x_j)\phi_j(x)$.

4. The weak form

It is easy to see that a function $u_h \in V_h$ can not solve equation (2.1) in the usual sense. A solution to equation (2.1) has to have two bounded derivatives, functions in V_h only have one. In fact, there are a lot of situations when even the exact solution u to a physical problem, which we model by a second order partial differential equation, does not belong the space of functions with bounded second derivative. One such example is when the boundary conditions are discontinuous or the boundary is non-smooth.

In order to work with these problems we introduce the concept of weak (or variational) solutions to partial differential equations. The weak form of a second order partial differential equation is derived using the following three steps.

(1) Multiply the equation by a test function $v \in V$.
(2) Integrate over the computational domain $[0, 1]$.
(3) Integrate by parts in order to move the second derivative to the test function.

If we do this for the model problem (2.1) we get,

(4.1)
$$\int_0^1 f(x)v(x)\,dx = \int_0^1 -u''(x)v(x)\,dx = \int_0^1 u'(x)v'(x)\,dx - [u'(x)v(x)]_{x=0}^{x=1} = \int_0^1 u'(x)v'(x)\,dx.$$

Note that the boundary terms vanish due to the boundary conditions for functions in V. The weak form of equation (2.1) reads: find $u \in V$ such that,

$$\int_0^1 u'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx. \tag{4.2}$$

for all $v \in V$. If an equation has a classical solution (with two bounded derivatives), this solution is always also a weak solution.

The main reason for introducing the weak solution concept is that we on longer need to have two bounded derivatives in order to have a solution. It is enough that $u'(x)$ takes values so the integral in the weak form is bounded. We also note that a function $u_h \in V_h$ fits the weak form framework since the first derivative of a continuous piecewise linear function is piecewise constant and thereby integrable.

5. The finite element method

The finite element method is very much linked to the weak form of the original equation. Instead of searchng for a solution in the infinite dimensional space V we only consider the space of continuous piecewise linear functions. The finite element formulation of equation (2.1) reads: find $u_h \in V_h$ such that,

$$\int_0^1 u_h' v_h' \, dx = \int_0^1 f v_h \, dx, \quad \text{for all } v_h \in V_h. \tag{5.1}$$

We will now show how this abstract equation can be transformed into a linear system of equations.

5.1. **Represent V_h with the span of the hat-functions.** The first step is to remember that we have a simple basis for V_h, the hat-functions $\{\phi_j\}_{j=1}^{n-1}$. Since any function in V_h can be written as a linear combination of hat-functions we can write,

$$u_h(x) = \sum_{j=1}^{n-1} c_j \phi_j(x), \quad \text{where } c_j \in \mathbf{R}. \tag{5.2}$$

We also note that instead of testing equation (5.1) with all possible functions $v_h \in V_h$ it is enough to test with the basis functions $\{\phi_i\}_{i=1}^{n-1}$, remember that any function in V_h can be written as a linear combination of the basis functions. If we plug this into equation (5.1) we get: find $u_h = \sum_{j=1}^{n-1} c_j \phi_j$ such that,

$$\int_0^1 u_h' \phi_i' \, dx = \int_0^1 \sum_{j=1}^{n-1} c_j \phi_j' \phi_i' \, dx = \int_0^1 f \phi_i \, dx, \quad \text{for all } i = 1, \ldots, n-1. \tag{5.3}$$

We can switch the order of the sum and the integral to get: find $u_h = \sum_{j=1}^{n-1} c_j \phi_j$ such that,

$$\sum_{j=1}^{n-1} c_j \int_0^1 \phi_j' \phi_i' \, dx = \int_0^1 f \phi_i \, dx, \quad \text{for all } i = 1, \ldots, n-1. \tag{5.4}$$

5.2. **How to transform this into a linear system of equations.** We are now very close to transforming this into a computational method. We introduce a vector $\vec{c} = [c_1, c_2, \ldots, c_{n-1}]$ a vector $\vec{b} = [\int_0^1 f\phi_1 \, dx, \ldots, \int_0^1 f\phi_{n-1} \, dx]$ and a matrix K with elements $K_{i,j} = \int_0^1 \phi_j' \phi_i' \, dx$. We can now write equation (5.4) on matrix form,

$$K\vec{c} = \vec{b}. \tag{5.5}$$

In order to compute the finite element solution $u_h = \sum_{j=1}^{n-1} c_j \phi_j$ we need to solve the linear system in equation (5.5) to get the coefficients $\{c_j\}_{j=1}^{n-1}$. To solve the system (5.5) we need to compte the vector $\vec{b}$ and the matrix $\vec{K}$ in the following way,

$$\vec{b}_j = \int_0^1 f(x)\phi_i(x)\,dx, \tag{5.6}$$

$$K_{i,j} = \int_0^1 \phi_j'(x)\phi_i'(x)\,dx, \tag{5.7}$$

for $0 < i, j < n$. Note that each basis function ϕ_i only overlap with at most two other basis functions so the K matrix will be very sparse, i.e. it has mainly zero entries.

In the one dimensional case it is very easy to compute the elements in K. Since $\phi_j = 1$ in node x_j, zero in all other nodes, and linear we know that,

$$\phi_j'(x) = \frac{1}{h_j}, \quad \text{for all } x \in [x_{j-1}, x_j], \tag{5.8}$$

$$\phi_j'(x) = -\frac{1}{h_{j+1}}, \quad \text{for all } x \in [x_j, x_{j+1}], \tag{5.9}$$

$$\phi_j'(x) = 0, \quad \text{for all } x \in [0, x_{j-1}] \cup [x_{j+1}, 1]. \tag{5.10}$$

This means that,

$$K_{i,i} = \int_{x_{i-1}}^{x_i} \frac{1}{h_i^2}\,dx + \int_{x_i}^{x_{i+1}} \frac{1}{h_{i+1}^2}\,dx = \frac{1}{h_i} + \frac{1}{h_{i+1}}, \quad \text{for all } i = 1, \ldots, n-1 \tag{5.11}$$

$$K_{i,i-1} = -\int_{x_{i-1}}^{x_i} \frac{1}{h_i^2}\,dx = -\frac{1}{h_i}, \quad \text{for all } i = 2, \ldots, n-1 \tag{5.12}$$

$$K_{i,i+1} = -\int_{x_i}^{x_{i+1}} \frac{1}{h_{i+1}^2}\,dx = -\frac{1}{h_{i+1}}, \quad \text{for all } i = 1, \ldots, n-2, \tag{5.13}$$

In case $h_i = 1/n := h$ for all $i = 1, \ldots, n$, K can be constructed in Matlab easily, `n=10;e=ones(n+1,1);K=spdiags([-e 2*e -e],-1:1,n+1,n+1)*n;`. It remains to construct the vector $\vec{b}$. In order to do this we need to compute integrals $\int_0^1 f(x)\phi_i(x)\,dx$. We do this is the case when $f = 1$. We have,

$$\vec{b}_i = \int_0^1 \phi_i(x)\,dx = \frac{h_i + h_{i+1}}{2}, \quad \text{for all } i = 1, \ldots, n-1. \tag{5.14}$$

In the case $h_i = 1/n := h$ we write `b=ones(n+1,1)/n;` in Matlab. We get the final solution using the `c=mldivide(K,b);` command i.e. *backslash*.

6. Error estimation

In this section we will show how the distance between u and u_h can be bounded in terms of known data. First we need to decide how to measure distance between functions in a function space V.

6.1. **Norms in function spaces.** In order to measure distances between functions we need to introduce a norm. There are several options,

- The euclidian norm, $\|v\|_{\text{Eucl}} = (\sum_{j=1}^{n-1} |v(x_j)|^2)^{1/2}$, which only measures the function values in the nodes.
- The $L^2([0,1])$-norm $\|v\|_2 = (\int_0^1 |v(x)|^2\,dx)^{1/2}$.
- The energy norm (H^1-norm) $\|v'\|_2 = (\int_0^1 |v'(x)|^2\,dx)^{1/2}$.

It turns out that the energy norm is particularly useful when measuring distances between the finite element approximation and the exact solution.

6.2. **Scalar products in function spaces and the Galerkin orthogonality.** In order to prove a bound of the error in energy norm we need to introduce a scalar product between functions in the space V. We let,

$$(u,v) = \int_0^1 u \cdot v\,dx, \quad \text{for all } u,v \in V, \tag{6.1}$$

this is called the L^2 scalar product. Using this new scalar product notation the weak form and the finite element method reads: find $u \in V$ such that,

$$(u',v') = (f,v), \quad \text{for all } v \in V. \tag{6.2}$$

and, find $u_h \in V_h$ such that,

$$(u_h',v_h') = (f,v_h), \quad \text{for all } v_h \in V_h. \tag{6.3}$$

Since $V_h \subset V$ we note that,

$$((u-u_h)',v_h') = 0, \quad \text{for all } v_h \in V_h. \tag{6.4}$$

This is referred to as the *Galerkin orthogonality*. It means that the error $u - u_h \in V$ is orthogonal to the subspace V_h. In Figure 3 we see an analogy to this in $\mathbf{R}^3$ where the subspace is the plane, $z = 0$. The Galerkin orthogonality gives that the finite element solution is a projection (called the Ritz or elliptic projection) of the exact solution u into V_h, that we can compute without knowing u.

We now want to bound the error in the energy norm $\|(u-u_h)'\|_2$. There is one function in V_h for which we do know how close it is to u measured in the energy norm. We will start by studying this function before we analyze the finite element solution.

6.3. **The nodal interpolation.** We let $\tilde{u}_h$ denote the nodal interpolant of u, see Figure 4. The nodal interpolant is defined by the following formula,

$$\tilde{u}_h = \sum_{j=1}^{n-1} u(x_j)\phi_j(x) \in V_h. \tag{6.5}$$

In order to bound the distance between u_h and u it turns out that we first need to bound the distance between $\tilde{u}_h$ and u.

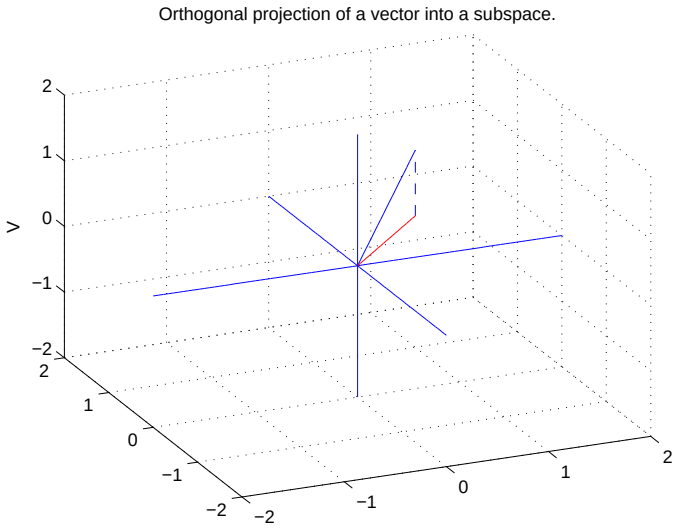

FIGURE 3. Projection of a vector down to a subspace.

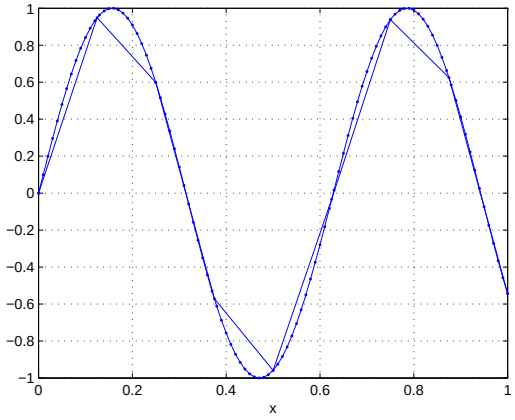

FIGURE 4. The function $sin(10x)$ with interpolant using 8 nodes.

We will now bound the difference between $\tilde{u}_h$ and u in terms of h and f. Let $x \in [x_j, x_{j+1}]$. We then have,

$$\tilde{u}_h' = \frac{u(x_{j+1}) - u(x_j)}{x_{j+1} - x_j}. \tag{6.6}$$

If we assume that $u'(x)$ is continuous there exists a $\chi \in [x_j, x_{j+1}]$ such that $u'(\chi) = \tilde{u}_h'$, see Figure 5. We let $x \in [x_j, x_{j+1}]$ and Taylor expand $u'(x)$ at $x = \chi$. We get,

$$|u'(x) - \tilde{u}_h'(x)| = |u'(\chi) + u''(\hat{x})(x - \chi) - \tilde{u}_h'(x)| = |u''(\hat{x})(x - \chi)|, \tag{6.7}$$

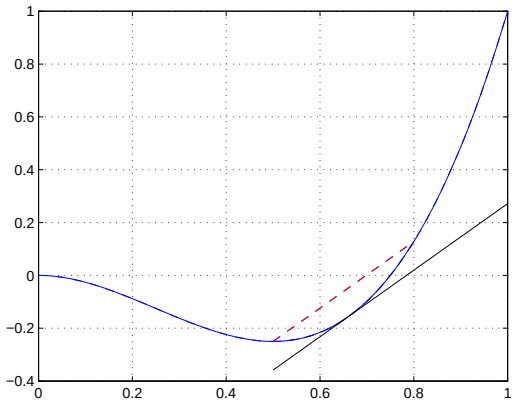

FIGURE 5. There is a $\chi \in [x_j, x_{j+1}]$ such that $u'(\chi) = \tilde{u}_h'$.

for some $\hat{x} \in [x, \chi] \in [x_j, x_{j+1}]$. We can now identify $u''(\hat{x}) = -f(\hat{x})$ and get,

$$|u'(x) - \tilde{u}_h'(x)| = |x - \chi||f(\hat{x})| \leq \max_{y \in [x_j, x_{j+1}]} |h_j f(y)|, \quad \text{for all } x \in [x_j, x_{j+1}]. \tag{6.8}$$

If we let $h = h_j$ for all $x \in [x_{j-1}, x_j]$ i.e. h is a piecewise constant function indicating the local size of the mesh we have,

$$|u'(x) - \tilde{u}_h'(x)| = \max_{y \in [0,1]} |hf(y)|, \quad \text{for all } x \in [0, 1]. \tag{6.9}$$

Since this holds for all $x \in [0, 1]$ we also have,

$$\|(u - \tilde{u}_h)'\|_2 \leq \max_{y \in [0,1]} |h(y)f(y)|, \tag{6.10}$$

using equation (5.1).

6.4. **A priori error estimate.** We are now ready to derive a bound of the error in energy norm. We will need a lemma called the *Cauchy-Schwarz* inequality that is very useful in mathematics.

Lemma 6.1. *For any two members $v, w \in V$ we have,*

$$|(v, w)| \leq \|v\|_2 \|w\|_2 \tag{6.11}$$

Proof. Let $\alpha = -(v, w)/(w, w)$ and assume $(v, w) \neq 0$, otherwise the result is trivially true. We have,

$$\begin{aligned} 0 &\leq (v + \alpha w, v + \alpha w) = (v, v) + 2\alpha(v, w) + |\alpha|^2(w, w) \\ &= (v, v) - 2|(v, w)|^2/(w, w) + |(v, w)|^2/(w, w) = (v, v) - |(v, w)|^2/(w, w), \end{aligned} \tag{6.12}$$

By multiplication with (w, w) followed by addition of $|(v, w)|$ we get,

$$|(v, w)|^2 \leq (v, v) \cdot (w, w), \tag{6.13}$$

in particular,

$$|(v,w)| \leq \|v\|_2 \|w\|_2, \tag{6.14}$$

since $(v,v) = \|v\|_2^2$ for all $v \in V$. □

As seen the proof is very general and in fact it applies to any function space for which a scalar product is defined. We can now present the main result, which is often referred to as an *a priori* estimate of the error since it can be done without knowing the computed approximation u_h, i.e. prior to the computation.

Theorem 6.1. *The error between the solutions to equations (2.1) and (5.1) measured in energy norm can be bounded as follows,*

$$\|(u-u_h)'\|_2 \leq \max_{y\in[0,1]} |h(y)f(y)|. \tag{6.15}$$

Proof. We start from the left hand side and use the Galerkin orthogonality,

$$\|(u-u_h)'\|_2^2 = (u-u_h, u-u_h) = (u-u_h, u) - (u-u_h, u_h) = (u-u_h, u-v_h), \tag{6.16}$$

for any $v_h \in V_h$. We now use the Cauchy-Schwarz inequality,

$$\|(u-u_h)'\|_2^2 = ((u-u_h)', (u-v_h)') \leq \|(u-u_h)'\|_2 \|(u-v_h)'\|_2, \tag{6.17}$$

we divide by $\|(u-u_h)'\|_2$ to get,

$$\|(u-u_h)'\|_2 \leq \|(u-v_h)'\|_2, \quad \text{for all } v_h \in V_h. \tag{6.18}$$

This means that the finite element solution is best in energy norm among the functions in V_h. In particular it is better then the interpolant $\tilde{u}_h$ i.e.,

$$\|(u-u_h)'\|_2 \leq \|(u-\tilde{u}_h)'\|_2 \leq \max_{y\in[0,1]} |h(y)f(y)|. \tag{6.19}$$

□

We note that the finite element solution converges to the exact solution as $h \to 0$.

7. Adaptivity

Given any mesh and the function f we can immediately calculate the error bound $\max_{y\in[0,1]} |h(y)f(y)|$. It is very likely that a constant h (i.e. equiv-distributed mesh points) will not give an optimal mesh in the sense that the number of nodes needed to guarantee a desired accuracy will not be the smallest possible.

One often want the error to be smaller that some given *tolerance*, which usually is a small number compared to the solution itself. Assume we have such a value `tol` at hand. We can now design an adaptive algorithm that will choose the mesh function h so that a minimum number of nodes are needed to meet the tolerance `tol`.

We choose a mesh function $h(y)$ such that $|h(y)f(y)| \leq tol$ for all $y \in [0,1]$. This is possible since $f(y)$ is known. Then we solve the problem using this mesh and automatically we know that the error $\|(u-u_h)'\|_2 \leq tol$.

We have now adapted the mesh for the particular problem we are solving. If an other function f where given we would have to construct a new optimal mesh.

8. Modification to the model problem

In this section we show how the finite element method deals with other types of boundary conditions and coefficients that are not constant.

8.1. **Boundary conditions.** One advantage with the finite element method is that it, also in higher dimensions, is very easy to implement any of the three standard type boundary conditions as well as mixed combinations of the three,

- Dirichlet boundary conditions, $u(0) = g_1$ and $u(1) = g_2$.
- Neumann boundary conditions, $u'(0) = g_1$, or $u'(1) = g_2$.
- Robin boundary conditions, $u'(0) + g_1 u(0) = g_2$ and $u'(1) + g_3 u(1) = g_4$.

We will now give an example of how the system of equations modifies when we use the following model problem,

$$\begin{cases} -u''(x) &= f(x), \quad x \in [0,1], \\ u(0) &= g_1, \\ u'(1) &= g_2, \end{cases} \tag{8.1}$$

We need to construct a new space for the exact and the approximate solutions. We let,

$$V^g = \{v \in C([0,1]) : v(0) = g\}, \tag{8.2}$$

and

$$V_h^g = \{v \in C([0,1]) : v(0) = g \text{ and } v(x), x \in [x_j, x_{j+1}] \text{ is linear for all } j \in 0, \ldots, n-1\}, \tag{8.3}$$

We start by constructing the weak form of the equation by multiplying with a test function and integrating by parts. We end up with: find $u \in V^{g_1}$ such that,

$$(f, v) = (-u'', v) = (u', v') - [u'(x)v(x)]_{x=0}^{x=1} = (u', v') - g_2 v(1), \quad \text{for all } v \in V^0. \tag{8.4}$$

Note that the test functions v are zero and not g_1 at $x = 0$. The reason for this is that the solution is known there $u = g_1$ so we do not need more degrees of freedom in order to compute the solution (get a square matrix). The corresponding finite element method reads: find $u_h \in V_h^{g_1}$ such that,

$$(u_h', v_h') = (f, v_h) + g_2 v_h(1), \quad \text{for all } v_h \in V_h^0. \tag{8.5}$$

If we let $u_h = \sum_{j=1}^n c_j \phi_j + g_1 \phi_0$, note that $u_h(0) = g_1$ so we need to include this in the formula for u_h, $\phi_0(x)$ being the half hat-function that is equal to one at $x = 0$ and zero at all other nodes, and plug this into the equation we get, find $u_h = \sum_{j=1}^n c_j \phi_j + g_1 \phi_0 \in V_h^{g_1}$ such that,

$$\sum_{j=1}^n c_j (\phi_j', \phi_i') = (f, \phi_i) + g_2 \phi_i(1) - g_1 (\phi_0', \phi_i'), \quad \text{for all } i \in 1, \ldots, n. \tag{8.6}$$

We note that the matrix K will be identical to last time except that it is now a $n \times n$ matrix instead of a $n-1 \times n-1$. This is because the solution is not know at the right end

point. The vector $\vec{b}$ will be modified in the top and at the bottom in the following way,

$$b_1 = (f, \phi_1) - g_1(\phi_0', \phi_1') = (f, \phi_1) + \frac{g_1}{h_1}, \tag{8.7}$$

$$b_i = (f, \phi_i), \quad \text{for all } i = 1, \ldots, n-1, \tag{8.8}$$

$$b_n = (f, \phi_n) + g_2\phi_n(1) = (f, \phi_n) + g_2, \tag{8.9}$$

since $(\phi_0', \phi_1') = -1/h_1$, $(\phi_0', \phi_i') = 0$ for all $i > 1$, $\phi_i(1) = 0$ if $i > n$, and $\phi_n(1) = 1$.

8.2. **Non-constant coefficients and numerical integration.** We study the following model problem,

$$\begin{cases} -(a(x)u'(x))' &= f(x), \quad x \in [0,1], \\ u(0) &= 0, \\ u(1) &= 0, \end{cases} \tag{8.10}$$

where $f(x)$ and $a(x)$ are given functions, defined on the interval $[0,1]$. We assume that $a(x) \geq \alpha > 0$ for some fix constant α. The finite element formulation of equation (8.10) reads: find $u_h \in V_h$ such that,

$$(au_h', v_h') = (f, v_h), \quad \text{for all } v_h \in V_h. \tag{8.11}$$

The load vector $\vec{b}$ will be identical to last time, $\vec{b}_j = \int_0^1 f\phi_j\,dx$ but the matrix K will have slightly different entries, $K_{i,j} = \int_0^1 a\phi_j'\phi_i'\,dx$. We now have to compute these integrals numerically.

From previous courses you know several method for approximately computing integrals. The mid point rule, the trapezoidal rule, and Simpsons formula are three such examples. To demonstrate the midpoint rule in this setting we show how one element in the matrix K is computed. Let $i = j+1$. We have,

$$K_{j+1,j} = \int_{x_j}^{x_{j+1}} a\phi_j'\phi_{j+1}'\,dx = -\frac{1}{h_j^2}\int_{x_j}^{x_{j+1}} a\,dx \approx -\frac{a(\frac{x_j+x_{j+1}}{2})}{h_j}. \tag{8.12}$$

Higher order methods will give higher accuracy. The optimal choice of points to evaluate a in, if a is a polynomial are the *Gauss points*.

9. Higher dimensional domains

All the analysis presented here extends to two and three dimensional problems. Interation by parts will in higher dimensions be replaced by Green's formula i.e. the higher dimensional version of the finite element method will read: find $u_h \in V_h$ such that,

$$(\nabla u_h, \nabla v_h) = (\nabla u, \nabla v_h) = (-\nabla^2 u, v_h) + \langle \partial_n u, v_h\rangle = (f, v_h), \quad \text{for all } v_h \in V_h, \tag{9.1}$$

where $\langle v, w\rangle$ is an integral on the boundary of the domain and $\partial_n v$ is the normal derivative on the boundary of the function v. These terms vanish since $v_h = 0$ on the boundary.

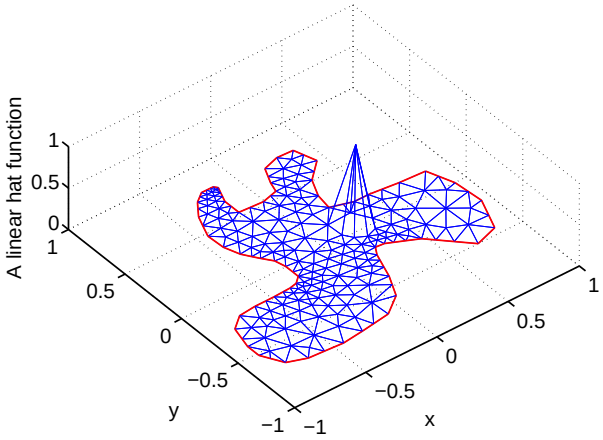

FIGURE 6. The figure from Lab1.

9.1. **Basis functions in two dimensions.** The mesh will consist of triangles for a two dimensional problem, see Figure 6. The basis functions in two dimensions are again continuous, piecewise linear, equal to one in one node, and equal to zero in all other nodes, see Figure 6. These basis functions are often referred to as tent functions for obvious reasons. Again we see that very few of these function overlap, which leads to very sparse matrices. In two dimensions it is crucial to store the mesh information in a convenient way. Usually the mesh is defined through three matrices. One with all coordinates

$$p = \begin{bmatrix} x_0 & x_1 & \dots & x_n \\ y_0 & y_1 & \dots & y_n \end{bmatrix}, \tag{9.2}$$

one keeping track of which nodes belongs to which triangle

$$t = \begin{bmatrix} 1 & 2 & \dots & 12 \\ 2 & 6 & \dots & 11 \\ 5 & 7 & \dots & 14 \end{bmatrix}, \tag{9.3}$$

meaning that triangle one consists of the nodes 1, 2, and 5 and so on, and one keeping track of the boundary,

$$e = \begin{bmatrix} 1 & 2 & \dots & 16 \\ 2 & 3 & \dots & 10 \end{bmatrix}. \tag{9.4}$$

i.e. node 1 and 2 makes up one boundary segment and node 2 and 3 another. A three dimensional mesh will usually be built up of tetrahedrons. The mesh structure used is similar to the two dimensional case and the basis function are constructed in the same way as before.

10. Other equations

The finite element framework generalizes to any partial differential equations. It is not as easy to derive error estimates for complicated equations and sometime the method needs to be modified in order to give reliable results.

10.1. **Time dependent problems.** When solving time dependent problems finite elements can also be used in time. However, in this introductory text we will only consider finite elements in space. The study the following model problem (the heat equation),

$$\begin{cases} \dot{u}(x,t) - u''(x,t) &= f(x,t), \quad x \in [0,1], \\ u(0,t) &= 0, \\ u(1,t) &= 0, \\ u(x,0) &= u_0(x), \end{cases} \tag{10.1}$$

We multiply by a test function in V and integrate by parts to get the weak form,

$$(\dot{u}, v) + (u', v') = (f, v), \quad \text{for all } v \in V. \tag{10.2}$$

The corresponding finite element method reads: find $u_h = \sum_{j=1}^{n-1} c_j(t)\phi_j(x)$ such that,

$$\sum_{j=1}^{n-1} \dot{c}_j(t)(\phi_j, \phi_i) + c_j(t)(\phi_j', \phi_i') = (f, \phi_i), \tag{10.3}$$

for all $i = 1, \ldots, n-1$. Note that the coefficients now depend on time, $c_j(t)$. This is because we have not discretized in time yet. This transforms into a system of equations including the new matrix M defined elementwise by $M_{i,j} = (\phi_j, \phi_i)$. We get,

$$M\dot{\vec{c}} + K\vec{c} = b. \tag{10.4}$$

We can now discrtize in time using a standard technique, e.g. implicit Euler to get,

$$M\frac{\vec{c}_n - \vec{c}_{n-1}}{\Delta t_n} = b - K\vec{c}_n, \quad \vec{c}_0 = u_0, \tag{10.5}$$

where Δt_n is the local time step. If we rearrange the terms we get,

$$\vec{c}_n = (M + \Delta t_n K)^{-1}(\Delta t_n \vec{b} + M\vec{c}_{n-1}), \tag{10.6}$$

i.e. we need to invert a matrix in every time step.

11. Conclusion and repetition

We end these lectures with some conclusions.

- The finite element method does not need simple geometry which makes it very useful in engineering applications in two and three dimensions.
- The finite element solution approximates the weak solution of a partial differential equation rather then the classical solution to the original problem.
- The finite element solution is optimal in energy norm for the class of problems we consider.

- We present an a priori error estimate that shows the the finite element solution converges to the exact solution as the mesh size $h \to 0$.
- We show how an adaptive algorithm can be constructed given an error estimate. The adaptive algorithm directs most computational effort to where it is needed.
- It is easy to implement different boundary conditions for the finite element method.
- The analysis extends directly to higher dimensions. Keeping track of the unstructured mesh will become increasingly difficult in higher dimensions but this is manageable.
- We show how the method can be applied to discretize the space for a time dependent problem.

Department of Information Technology, Uppsala University
E-mail address: axel.malqvist@it.uu.se
URL: http://user.it.uu.se/~axelm