Project TMA4220: Finite Element Method

XXX, YYY

1 Theory

1.1 The problem

The heat equation on a general domain is given as

$$k\nabla^2 u(\mathbf{x},t) = \frac{\partial u(\mathbf{x},t)}{\partial t}, \tag{1}$$

with the Dirichlet condition $u(\mathbf{x},t) = g(\mathbf{x},t)$ on the boundary, for some given function g. If we insetad look at the stationary case, the right hand side of (1) becomes zero. In addition if instead of looking at the inhomogenous Dirichlet boundary condition.

Let $\Omega \subset \mathbb{R}^n$ be bounded, open set, and let $u(\mathbf{x})$ satisfy the boundary value problem

$$\begin{aligned} \nabla^2 u(\mathbf{x}) &= 0 && , \quad u(\mathbf{x}) \in \Omega \\ u(\mathbf{x}) &= u_0 && , \quad u \in \partial\Omega_D \\ k\frac{\partial u(\mathbf{x})}{\partial n} &= -h\,(u(\mathbf{x}) - u_{amb}), && \quad u \in \partial\Omega_R. \end{aligned}$$

Here the temperature u_0 and the ambient temperature u_{amb} are fixed, while h is the heat transfer coefficient between two materials. $\partial\Omega_D$ and $\partial\Omega_R$ denotes the part Dirichlet and Robin part of the boundary $\partial\Omega = \partial\Omega_D \cup \Omega_R$, respectively. In the Robin boundary condition $\mathbf{n}$ is the outward normal unit vector. The term $\partial u/\partial n$ is the heat flux, and is in this model proportional to the temperature difference between the ambient temperature and the solution. The minus sign emphasizes that the heat is flowing in the opposite direction of the temperature gradient, which is reasonable.

1.2 Weak form

The finite element method requires the PDE to to be formulated in terms of its weak form. We introduce the set of test functions $C_0^\infty(\Omega)$. This is the set of all infinitely differentiable functions with compact support in Ω. $V = H_0^1(\Omega)$ is the usual Sobolov space. By Green's formula we have

$$\int_\Omega v\nabla^2 u \,\mathrm{d}\Omega = \int_{\partial\Omega} v\frac{\partial u}{\partial n}\,\mathrm{d}\gamma - \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}\Omega = 0. \tag{2}$$

The boundary integral is split up into the $\partial\Omega_D$ and $\partial\Omega_R$. If then $v \in V$, then the integral over $\partial\Omega_D$ vanishes and, by collecting terms, (2) reduces to

$$\frac{k}{h}\int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}\Omega + \int_{\partial\Omega_R} uv \,\mathrm{d}\gamma = u_{amb}\int_{\partial\Omega_R} v \,\mathrm{d}\gamma. \tag{3}$$

Note that because of the inhomogeneous Dirichlet boundary conditions $u \notin V$, even though the proper weak formulation should state: Find $u \in V$ such that (3) is satisfied $\forall v \in V$. Formally one should explicitly introduce a lifting function R such that a new function $\mathring{u} = u - R$ satisfies the homogeneous Dirichlet conditions, and formulate the weak form in terms of $\mathring{u}$. However, in our numerical scheme we will get the desired solution by setting the solution of our linear system to the known Dirichlet boundary values.

From (3) we see that our bilinear form $a(\cdot,\cdot)$ and linear functional $F(\cdot)$ takes the form

$$\begin{aligned} a(u,v) &= \frac{k}{h}\int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}\Omega + \int_{\partial\Omega_R} uv \,\mathrm{d}\gamma \\ F(v) &= u_{amb}\int_{\partial\Omega_R} v \,\mathrm{d}\gamma. \end{aligned} \tag{4}$$

Given a set of points $\{\mathbf{x}_i\}_{i=1}^n$ in Ω, we now define a set of nodal functions $\{\varphi_i(\mathbf{x})\}_{i=1}^n$, such that $\varphi_i(\mathbf{x}_j) = \delta_{ij},\ i,j = 1,\ldots,n$. Now we define $V_h = \operatorname{span}\{\varphi_i(\mathbf{x})\}_{i=1}^n$, and let $u_h(\mathbf{x}) = \sum_{i=1}^n u_i\varphi_i(\mathbf{x}) \in V_h$. The Galerkin problem can now be formulated as "Find $u_h \in V_h$ such that $a(u_h, v) = F(v)\,\forall v \in V_h$". Since $a(\cdot,\cdot)$ is bilinear, $F(\cdot)$ is linear and u_h is a linear combination of the basis functions $\{\varphi_i(\mathbf{x})\}_{i=1}^n$, this problem is equivalent to the linear system

$$A\mathbf{u} = \mathbf{b}, \quad A_{ij} = a(\varphi_i, \varphi_j),\ b_i = F(\varphi_i), \tag{5}$$

with $\mathbf{u}$ being the vector with elements corresponding to the coefficients u_i of u_h. We note that since we have not yet enforced the Dirichlet boundary conditions, the solution is not unique, hence A is not invertible in its current form.

1.3 Barycentric coordinates

2 Numerical implementation

2.1 Basis functions

The choice of the type of basis functions is important when implementing the finite element method. In this project we have settled for the simplest type, namely the linear type. In higher dimensions one would usually resort to reference functions in a reference space, however for linear basis functions looking at the physical space is "cool". The linear functions in $\mathbb{R}^3$ is of the form

$$\varphi_i(x,y,z) = a_i x + b_i y + c_i z + d_i,$$

where the index i refers to the node. In a nodal basis we want $\varphi_i(\mathbf{x}_j) = \delta_{ij},\ i,j = 1,\ldots,n$, and in a mesh consisting of tetrahedral elements the coefficients can be calculated per element. For

instance, if $\mathcal{K}_i$ is the element being the tetrahedron given by the nodes $\{j, k, l, m\}$. Then the coefficients in φ_j is given by the solution of the linear system

$$\begin{bmatrix} x_j & y_j & z_j & 1 \\ x_k & y_k & z_k & 1 \\ x_l & y_l & z_l & 1 \\ x_m & y_m & z_m & 1 \end{bmatrix} \begin{bmatrix} a_j \\ b_j \\ c_j \\ d_j \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{6}$$

2.2 The numerical integrals

$a(u, v)$ in (4) contains a the gradient of the basis functions φ. However because of the linearity of the basis functions this integral will be simply be the the volume of each element, multiplied by a constant. Let $\boldsymbol{\varphi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \ldots, \varphi_4(\mathbf{x}))^T$ be the vector of basis function on an element $\mathcal{K}$. Then each entry in the symmetric matrix

$$\boldsymbol{J}_{\boldsymbol{\varphi}} \boldsymbol{J}_{\boldsymbol{\varphi}}^T \int_{\mathcal{K}} \mathrm{d}V$$

represent the contribution from each combination of $\boldsymbol{\nabla}\varphi_i \cdot \boldsymbol{\nabla}\varphi_j$ on $\mathcal{K}$. Here $\boldsymbol{J}_{\boldsymbol{\varphi}}$ is the Jacobian of $\boldsymbol{\varphi}$. The two remaining integrals in (4) are surface integrals over the boundary of Ω. We can use the function `quadrature2D` to integrate over planes in $\mathbb{R}^3$ by projecting down onto $\mathbb{R}^2$. Let $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \in \mathbb{R}^3$. The plane going through all three points have normal vector $\mathbf{n} = (\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)$. Projecting down on the xy-plane then yields

$$z = z(x, y) = \frac{1}{n_3} \left(\mathbf{n} \cdot \mathbf{p}_1 - n_1 x - n_2 y \right).$$

The differential $\mathrm{d}\gamma$ on the plane then becomes

$$\mathrm{d}\gamma = \frac{\|\mathbf{n}\|}{|n_3|} \mathrm{d}x\mathrm{d}y.$$

The whole boundary of the mesh consists of planes normal to the all the axes, so we actually need project onto both the xz- and yz-plane as well for the different parts of the boundary.

When calculating the contribution from the integrals over the boundary in (4) we need to know which nodes are on the boundary. It is natural to keep the list of the elements and the boundary surfaces separate, and adding the contribution from each integral in separate loops in the program. The linear basis functions φ represents a plane in $\mathbb{R}^4$, however as each surface element is a triangle, we only have three points for the system (6). The last point can be chosen arbitrarily, but in order to avoid problems with linear dependence in the row space of (6), it is easiest to choose the last point outside of the domain Ω. This what was done in this project.

2.3 The Dirichlet boundary

The size of the linear system (5) increases with the number of nodes in the mesh, it is important to keep the number of dimensions in system to a minimum. Since the solution on the Dirichlet boundary $\partial\Omega_D$ are known by definition, these nodes need not be in the vector $\mathbf{u}$ in (5). Instead the contribution in each row corresponding to u_i, where $\mathbf{x}_i \notin \partial\Omega_D$, can be moved to the right-hand side of the system, reducing the dimension by the number of nodes on $\partial\Omega_D$. We refer to the code as of how this was done.

Because of the nodal basis, the integrals in (4) will be zero between all non-neighboring nodes, and as a result the matrix A will be sparse. However, as the finite elements works by updating A and b in (5) while looping over all the elements, it is not known beforehand which entries of A and b which will be non-zero. Using MATLAB's `sparse` functionality was therefore not beneficial in our case. On the contrary, we do not reject the possibility that there may be other ways to exploit the sparsity of A.

References

[1] The Finite Element Method (FEM). `https://www.comsol.com/multiphysics/finite-element-method`.

[2] TMA4220 Programming project - part 1. `https://www.math.ntnu.no/emner/TMA4220/2016h/project/Problem_set.pdf`.

[3] Author. *Title*. Publisher, 2016.

[4] Alfio Quarteroni. *Numerical Models for Differential Problems (MS&A)*. Springer, 2014.