Project TMA4220: Finite Element Method

XXX, YYY

1 Introduction

Heat sinks are a commonly used heat exchangers with the purpose of transferring heat that are generated by electronic or mechanical devices [1]. In computers it is useful to be able to regulate the temperature of components like the central processing unit (CPU) or the graphic processing unit (GPU). These units typically generate a lot of heat and it becomes necessary for optimal operation to dissipate this heat away to keep the temperature at an optimal level. This is typically done by using a heat sink together with a fan as seen in **Figure 1**.

When designing heat sinks one typically tries to maximize surface area in contact with the cooling medium surrounding it, such as air. There are several factors that affects the performance of a heat sink. In essence air velocity, choice of material, design and surface treatment are the most important ones. When choosing material one should look for good heat conductors which means that heat can be transferred though the material quickly. In many cases copper is used because of its excellent thermal efficiency.

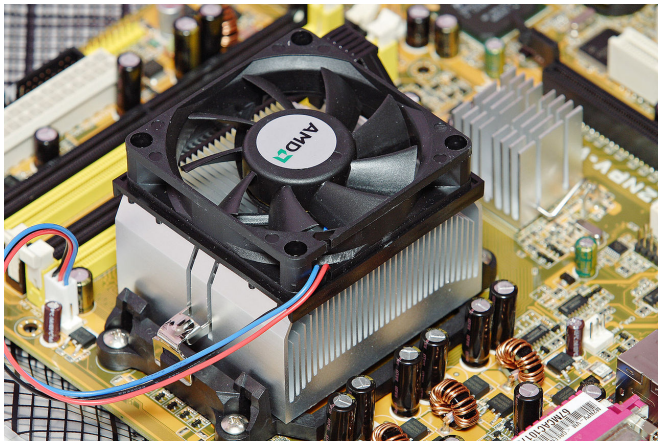


Figure 1: Heat sink and fan on processing unit. Source: [1]

In this project we will be trying to use a Finite Element Method to evaluate the effect of increasing the surface area on a simple heat sink design in two different ways. One ways is to extend the length of the fins, and the other is to increase the number of fins. We will keep the geometric properties of the heat sink base constant, meaning that we will only vary the geometry of the fins themselves. This report is divided into three main parts. In the *Theory* part we will elaborate on the heat equation and show a finite element analysis which results in a linear system of equations. In the *Numerical Implementations* part we will go through and explain the steps of implementing the numerical interpretation. In the *Results and Discussion* we will present our results together with a discussion of their interpretation. Finally we will sum up our findings and suggest improvements in a conclusion.

2 Theory

2.1 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

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

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.

2.2 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

$$\int_\Omega v\nabla^2 u \,\mathrm{d}\Omega = \int_{\partial\Omega} v\frac{\partial u}{\partial n}\,\mathrm{d}\gamma - \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}\Omega = 0. \tag{3}$$

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, (3) reduces to

$$\frac{k}{h}\int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}\Omega + \int_{\partial\Omega_R} uv \,\mathrm{d}\gamma = u_{amb}\int_{\partial\Omega_R} v\,\mathrm{d}\gamma. \tag{4}$$

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

$$\begin{aligned} a(u,v) &= \frac{k}{h}\int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \,\mathrm{d}\Omega + \int_{\partial\Omega_R} uv \,\mathrm{d}\gamma \\ F(v) &= u_{amb}\int_{\partial\Omega_R} v\,\mathrm{d}\gamma. \end{aligned} \tag{5}$$

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

$$A\mathbf{u} = \mathbf{b}, \quad A_{ij} = a(\varphi_i, \varphi_j),\ b_i = F(\varphi_i), \tag{6}$$

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.

2.3 Barycentric coordinates and numerical integration

We only state the formulas for evaluating integrals numerically on simplexes using barycentric quadrature points. For an in-depth cover of this topic, we refer to [7]. Let $\hat{\mathcal{K}}$ be the reference simplex in $\mathbb{R}^n$ with corners $\hat{\mathbf{e}}_1,\ldots,\hat{\mathbf{e}}_n$ and $\mathbf{0}$ and $\mathcal{K}$ be the physical simplex with corners $\mathbf{p}_0,\ldots,\mathbf{p}_n$. The integral transformation then yields

$$\int_{\mathcal{K}} f(\mathbf{x})\,\mathrm{d}V = \int_{\hat{\mathcal{K}}} f(\hat{\mathbf{x}})\frac{1}{n!}\frac{\operatorname{Volume}(\mathcal{K})}{\operatorname{Volume}(\hat{\mathcal{K}})}\,\mathrm{d}\hat{V}.$$

The volume of the simplex $\mathcal{K}$ is [3]

$$\text{Volume}(\mathcal{K}) = \frac{1}{n!} \left| \det \left(\mathbf{p}_1 - \mathbf{p}_0 \ \dots \ \mathbf{p}_n - \mathbf{p}_0 \right) \right|.$$

We also note that $\text{Volume}(\hat{\mathcal{K}}) = 1/n!$. Each point in $\mathcal{K}$ can be described with a barycentric coordinate $\boldsymbol{\lambda} = (\lambda_0, \dots, \lambda_n)$, with the mapping to the physical space $\mathbf{x} = \lambda_0 \mathbf{p}_0 + \cdots + \lambda_n \mathbf{p}_n$. Given a set of m barycentric quadrature points, $\boldsymbol{\lambda}^1, \dots, \boldsymbol{\lambda}^m$, and corresponding weights, $\rho_1, \dots, \rho_m$, the numerical integration formula becomes

$$\int_{\mathcal{K}} f(\mathbf{x}) \, \mathrm{d}V \approx \text{Volume}(\mathcal{K}) \sum_{i=1}^{m} \rho_m f(\lambda_0^i \mathbf{p}_0 + \cdots + \lambda_n^i \mathbf{p}_n).$$

This quadrature rule is used `quadrature2D` and `quadrature3D`, while `quadrature1D` relies on gaussian quadrature. The table of barycentric quadrature points used in the code can be found in [5].

3 Numerical implementation

3.1 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 enough. 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 $\varphi_i(\mathbf{x}_j) = \delta_{ij}$, $i, j = 1, \dots, n$, 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

$$\begin{bmatrix} x_j & y_j & z_j & 1 \\ x_k & y_k & z_k & 1 \\ x_l & y_l & z_l & 1 \\ x_m & y_m & z_m & 1 \end{bmatrix} \begin{bmatrix} a_j \\ b_j \\ c_j \\ d_j \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{7}$$

3.2 The numerical integrals

$a(u, v)$ in (5) 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 $\boldsymbol{\varphi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \dots, \varphi_4(\mathbf{x}))^T$ 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 (5) 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 would actually need project onto both the xz- and yz-plane as well for different parts of the boundary. Instead we rotate the mesh $\pi/4$ around both

the x and the y-axis, using the rotation matrix

$$R_x\left(\frac{\pi}{4}\right) R_y\left(\frac{\pi}{4}\right) = \frac{1}{2}\begin{bmatrix}\sqrt{2} & 0 & 0\\ 0 & 1 & -1\\ 0 & 1 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 1\\ 0 & \sqrt{2} & 0\\ -1 & 0 & 1\end{bmatrix},$$

yielding no boundary surface being normal to the xy-plane.

When calculating the contribution from the integrals over the boundary in (5) 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 (7). The last point can be chosen arbitrarily, but in order to avoid problems with linear dependence in the row space of (7), we choose $\mathbf{n}$, which is linearly independent of $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$.

3.3 The Dirichlet boundary

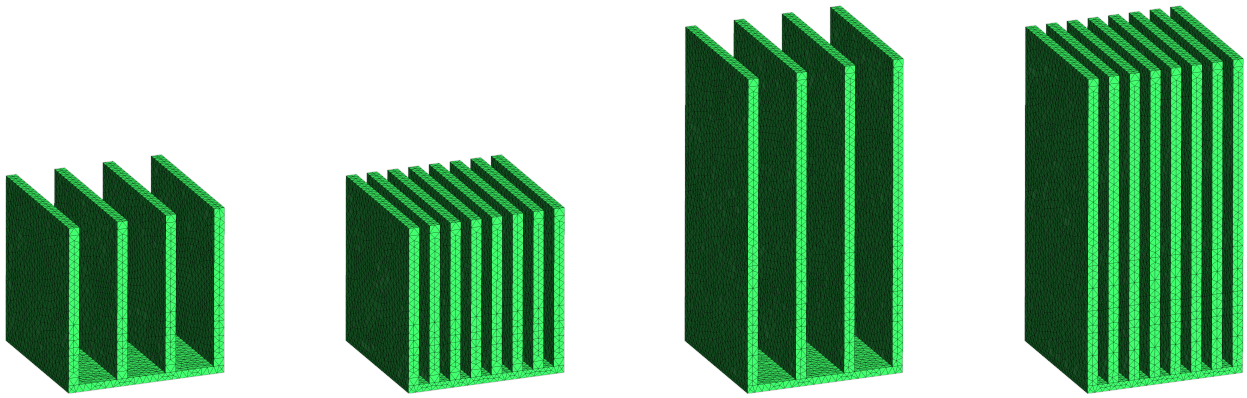
The size of the linear system (6) increases with the number of nodes in the mesh, it is important to keep the number of dimensions in system to a minimum. Since the solution on the Dirichlet boundary $\partial\Omega_D$ is known by definition, these nodes need not be in the vector $\mathbf{u}$ in (6). 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 (5) 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 (6) 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.

A heat sink has a geometry which is generally difficult to model with a 3D mesh. The surface should be as large as possible in order to maximize contact with the surrounding air. The fins should therefore be as many and as thin as possible. Because of this, tetrahedral elements may not be optimal for the type of geometries considered in this report. For instance, Comsol [2] recommends to use brick or prism elements in cases where the geometry consists of thin layers.

3.4 Generating the mesh

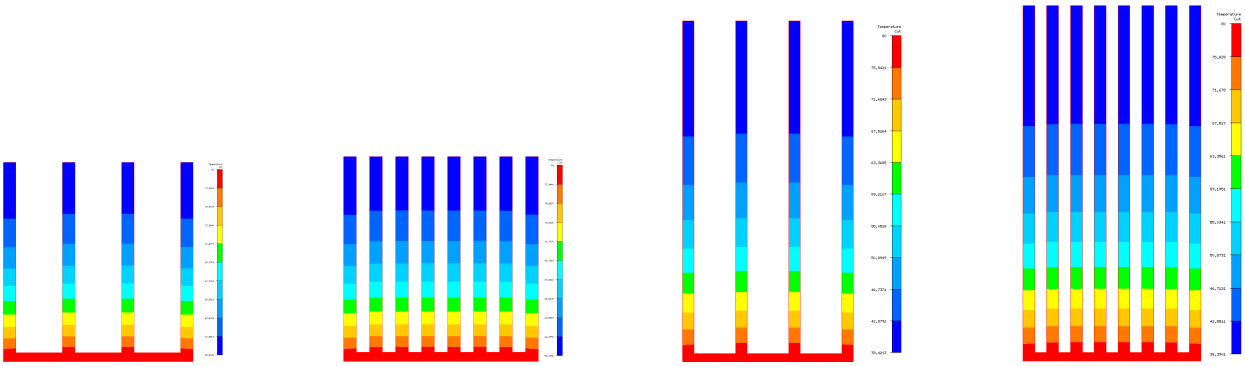
For our purposes we did not need a complex geometry so we opted for a simple geometry for the heat sinks. To create and control the geometry as we wanted, we used the software *Gmsh*. For practical reasons we chose the size of the heat sink base to be 1.05×1.05. For the height of the base and the thickness of the fins we chose 0.05 and 0.07. These values does not change through the experiment. What we varied was the number of fins and the height of the fins. We tried 4 and 8 fins and heights of 1.0 and 2.0.



(a) 4 fins, height: 1 **(b)** 8 fins, height: 1 **(c)** 4 fins, height: 2 **(d)** 8 fins, height: 2

Figure 2: These are the different meshes we tried

4 Results and discussion



(a) 4 fins, 1 height **(b)** 8 fins, 1 height **(c)** 4 fins, 2 height **(d)** 8 fins, 2 height

Figure 3: Temperatures in the middle of the heat sink for the different meshes.

4.1 Visualization of results

Visualizing results from simulations is important in order to understand the result. when having a function in three dimensions it is necessary to make choices of what part of the result which should be displayed. The natural way of doing this is either with iso-surfaces or cutting-planes. The geometries in 2 are somewhat symmetric, especially along the fins, which suggests that the temperature field should be more or less constant along this axis. In 3D visualizing software this is easier to see, and in our solution this is indeed the case. Therefore figure 3 represents a cutting plane normal to the find, which should give a good representation of the temperature field in the whole geometry.

4.2 Assumptions and Physical Interpretation

In our analysis we have made some assumptions and initial choices for the boundary conditions to make a somewhat simplified mathematical model. For the temperature of the bottom of the heat sink base we enforced a constant Dirichlet boundary condition of 80 °C. This choice was made because it is an estimate of the temperature of a very warm processor working at maximum capacity. By enforcing Dirichlet boundary condition we disallow the hypothetical processor to actually cool down, which is not necessarily a realistic assumption. When looking at a computer processor it would be natural to want to keep the temperature below some maximum threshold, and look at how the heat sink would need to be engineered in order to have a sufficient heat loss below this temperature. In that case some kind of Neumann boundary condition on the bottom would maybe be better suited if you want to see such a cooling effect.

Another modeling choice made regarding the Robin conditions on the rest of the boundary is that the ambient temperature in the description of the boundary value problem (2) is also constant. This means that even in the tight spaces in between the fins the air temperature is constant. You could think of it as an average temperature on the boundary. One could maybe improve on this assumption by enforcing different ambient temperature on the outer boundary and the boundary in between the fins. Besides you would probably need a very powerful fan to transport the heat away to keep the ambient temperature at 20 degrees as assumed here.

If there were many more fins and they were a lot thinner and closer together it would be a good point to take into account the heat radiation from one fin to another. Also there would be challenges regarding the mesh. Using tetrahedron elements one could expect the elements to become very irregular. A solution could be to use brick prism elements, or alternatively a 2D mesh of the flat fins.

In Figure 3 we can see the temperatures for the different meshes. The temperature ranges from about 59 °C to 80 °C for figure 3a and 3b, and from 39 °C to 80 °C for figure 3c and 3d. The Dirichlet boundary on the bottom works like a thermal reservoir, supplying as much heat as possible, so the temperature profiles along the fins are virtually identical between fins of the same height, regardless of the number of fins.

The total heat loss to the surrounding air is

$$\int_{\partial\Omega_R} \frac{\partial u}{\partial n} \, \mathrm{d}\gamma.$$

Since the heat flux in Robin boundary condition in equation (2) is proportional to the temperature difference, and we want the to maximize the heat loss, a high temperature is preferred. The heat sink in figure 2b and 2c have both the same surface area and the same volume. However from figure 3b we see that 2b generally have a higher temperature. The result is that 2b in total has a greater heat loss. It would appear that, given a certain amount of material, the heat sink should be designed with many short, thin fins, as opposed to a fewer and higher fins.

References

[1] Heat sink. `https://en.wikipedia.org/wiki/Heat_sink`. Accessed: 2016-08-11.

[2] Meshing Your Geometry: When to Use the Various Element Types. `https://www.comsol.com/blogs/meshing-your-geometry-various-element-types`.

[3] Simplex. `https://en.wikipedia.org/wiki/Simplex`.

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

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

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

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