Investigating the Thermal Management of a Microprocessor using a Numerical Approach

Sunday 9th May, 2021

L. Versini

***Abstract*—We solve the steady-state heat equation numerically to investigate the average temperature across a full-speed operating microprocessor which erogates $0.5\ \mathrm{W/mm^3}$ of power density. We model heat to be diffused into the environment via natural or forced convection (using a fan) and we illustrate the algorithms we used to solve for the temperature under both conditions. We found that without heat sink, the system reaches an average temperature of $(7301 \pm 2)\ ^\circ\mathrm{C}$ which is well above the operating temperature of $80\ ^\circ\mathrm{C}$. With the addition of a heat sink made by 28 fins $30\mathrm{mm}$ high, the average temperature under natural convection is $(453 \pm 2)\ ^\circ\mathrm{C}$. By considering the same system under forced convection boundary condition, a temperature of $(79.9 \pm 2)\ ^\circ\mathrm{C}$ is achieved, which is within the operating temperature. We then show that this shape is the optimal one in terms of surface area occupied by the sink. Finally we compare a triangular and a rectangular fin heat sink and note that the latter one is more effective in dissipating heat.**

I. Introduction

THE miniaturisation of electronics can imply the increase in the thermal power produced per unit volume and an uncontrolled temperature rise can lead to a reduction of the circuit's performances [1]. It is therefore of interest trying to control the temperature developed within a circuit. We solve the heat equation for the maximum temperature achieved by a full speed operating microprocessor (power density $0.5\ \mathrm{W/mm^3}$) with and without the addition of a heat sink. We model the heat transfer at surfaces with the equations showed in Sec. II-A2 but we otherwise neglect fluid dynamic effects.

II. Theory and Computational Method

At the core of our simulation is the solution of the steady state heat equation (6) (which is a partial differential equation, PDE), via numerical methods. In this section we briefly introduce the physics behind our model and we show how we moved from the continuous PDE to the discrete case.

A. Underlying Physics

1) Fourier Law conduction: Consider the rod depicted in Fig. 1 which has thermal conductivity κ. Fourier Law of conduction [2, p. 571] expresses the heat flux $\phi_{A\to B}$ from body A (left) to body B (center) as

$$\phi_{A\to B} = \kappa \frac{T_a - T_b}{h}, \tag{1}$$

where h is the distance between bodies A and B. The overall heat flux through B is given by the contribution of heat coming from A minus the outgoing energy into C (colder body). Hence,

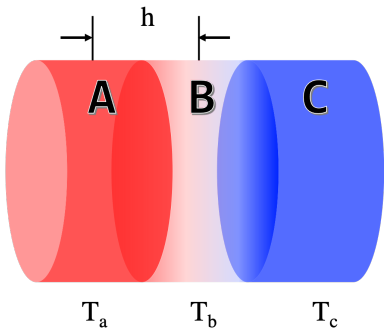


Fig. 1: Rod with a non-zero temperature gradient. Heat flows from warmer slice A into slice B and from B into C according to Fourier's Law of conduction, where temperatures are such that $T_a > T_b > T_c$. Distance between slices A-B-C is h.

$$\phi_{total} = \phi_{A\to B} - \phi_{B\to C} = \kappa \frac{-2T_b + T_a + T_c}{h} \tag{2}$$

We note that in the steady-state, we have $\phi_{total} = 0$.

2) Heat dissipation: The heat generated in the microprocessor is dissipated at the solid-air interface. We assume convection to be the dominating process and two empirical formulas are given in [3] to model heat flux ϕ at the boundary in case of natural convection (N) and forced convection (F) via a fan:

$$\phi = \gamma_{N/F}\ (T_{surf} - T_{env}) \tag{3}$$

$$\gamma_N = 1.31\ (T_{surf} - T_{env})^{1/3} \tag{4}$$

$$\gamma_F = 11.4 + 5.7v \tag{5}$$

Where T_{env} and T_{surf} are respectively environment and object surface's temperatures and v is the speed of the wind in $(\mathrm{m/s})$.

B. Setting up the numerical problem

Given the above equations, we want to solve the heat equation in the microprocessor-heatsink system with Neumann Boundary conditions [4, p. 752]. The system contains heat sources (i.e. the microprocessor) described by the power density $q_{(x,y,z)}$ and a conductivity κ that may vary in space.

The heat equation in the steady state can be written as follows [3]:

$$\kappa \nabla^2 T = k \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right) = -q_{(x,y,z)} \tag{6}$$

This has to be true at any point in space. In the above, we further simplify the math by looking at the cross section of a very long microprocessor for which we can take $\frac{\partial^2 T}{\partial z^2} \approx 0$ (cfr. [3]). Moreover, we note that since the boundary conditions depend on $(T_{surf} - T_{env})$, we can perform a change of variable $T' = T_{surf} - T_{env}$, solve eq. (6) for the difference in temperature and add an offset of T_{env} at the end. For the remaining, we will assume this change of variable and drop the superscript in T'.

1) Numerical representation: To solve PDE (6) numerically, we divide the system into a 2-dimensional grid of pixels, each one with its own temperature $T_{i,j}$, heat capacity $\kappa_{i,j}$ and source term $q_{i,j}$. Moreover, we need to approximate the Laplacian operator ∇^2 with a discrete *stencil*. We use the finite difference scheme described in [5, p. 407] and eq. (6) becomes:

$$\kappa_{i,j} \frac{-4T_{i,j} + T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{h^2} = -q_{i,j} \tag{7}$$

Where h is the pixel size. This equation is rearranged into:

$$\kappa_{i,j}(-4T_{i,j} + T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}) = -q_{i,j}\ h^2 \tag{8}$$

We physically interpret this equation as the sum of the heat fluxes from the four sides of the pixel $T_{i,j}$ using Fourier conduction law introduced in Sec. II-A1.

2) Handling the boundary: In this subsection we describe the form of eq. (7) at boundaries between two different solids and between solid-air. We consider the example illustrated in Fig. 2 which is a (3×3) grid of solid, composed by two different materials (blue filling and white filling) with thermal conductivities κ_1 and κ_2. The system is surrounded by air, and the heat generated by the sources (S) is dissipated via convection (Sec. II-A2).

We consider the left hand side of eq. (7) evaluated at the point $(0, 1)$. The contributions from $(0, 0)$ and $(0, 2)$ are straightforward to evaluate since all three points have the same κ_1. The contribution to the total heat flux from the point $(1, 1)$ which has thermal conductivity κ_2 needs to take energy conservation into account. Indeed, we must have that $\phi_{(1,1)\to(0,1)} = -\phi_{(0,1)\to(1,1)}$, but since Fourier Law depends on the thermal conductivity of the particular material, we need to find a single value that can be used for both $\phi_{(1,1)\to(0,1)}$ and $\phi_{(0,1)\to(1,1)}$. In our simulation we choose to use the average $\bar{\kappa} = (\kappa_1 + \kappa_2)/2$.

Finally, we need to consider the last contribution from $(-1, 1)$. That pixel is labelled with "Air" and it is where Neumann boundary conditions are applied. We consider $(-1, 1)$ to be a *virtual* extension of the material with conductivity κ_1

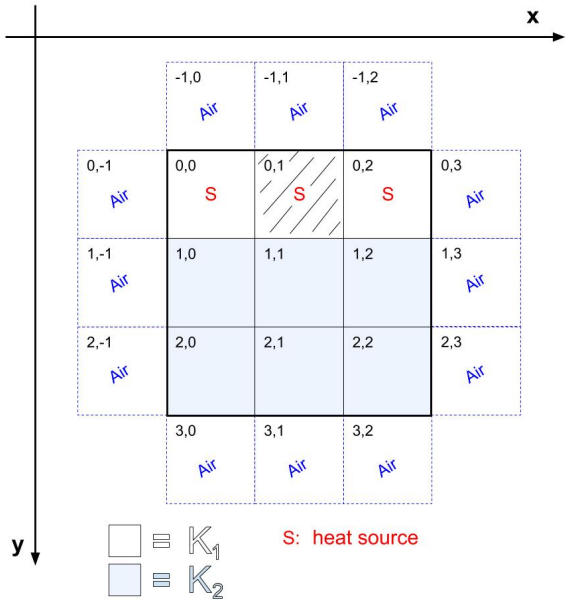


Fig. 2: Thick black line defines a sample (3×3) pixel grid representing two materials with different thermal conductivities (κ_1 and κ_2). Pixels outside of the black perimeter show the interface matter-air. Pixels labelled with "S" are source of heat. Origin placed in the top left corner, following Python's matrix indexing convention.

and we set its temperature $T^*_{-1,1}$ such that the condition on the gradient of $T_{0,1}$ is satisfied. Hence, using (1) and (3)

$$\kappa_1 \frac{T_{0,1} - T^*_{-1,1}}{h} = \gamma\ T_{0,1} \tag{9}$$

and rearranging the above we obtain an expression for $\kappa_1 T^*_{-1,1}$:

$$\kappa_1 T^*_{-1,1} = \kappa_1 T_{0,1} - \gamma h T_{0,1} \tag{10}$$

Hence, summing up the discussion above, we can write eq. (7) at $(0, 1)$ as:

$$-3\kappa_1 T_{0,1} - \bar{\kappa} T_{0,1} + \bar{\kappa} T_{1,1} + \kappa_1 T_{0,2} + \kappa_1 T_{0,0} + \kappa_1 T^*_{-1,1} = -q_{i,j} h^2 \tag{11}$$

Analogous equations for the remaining 8 pixels can written following this procedure. By solving the system of coupled equations that comes out if it, the values of temperatures $T_{i,j}$ can be found. It is important to note that even though eq. (11) looks linear with temperature, in the case of natural convection the factor γ_N contained in $T^*_{-1,1}$ (eq. (10)) has a $(T_{0,1})^{1/3}$ dependency, hence breaking linearity.

Eq. (11) can be written in natural units by dividing both sides of the equation by $q_0 h^2$ where q_0 is some reference power density (which in the system we simulated would be $0.5\ \mathrm{W/mm^3}$). We could then solve for $T' = [k_0/(q_0 h^2)]T$ which is dimensionless and where k_0 is a reference conductivity. However, since at our scale the values appearing in eq. (11) turn out to be in a similar order of magnitude - from $O(1)$ to $O(10^4)$ - we decided to use SI units instead.

III. Algorithm Choice and Validation

In this section we motivate the choice of algorithms to solve the system of coupled equations introduced in Sec. II-B2. These differ according to the *natural convection* and the *forced convection* cases (Sec. II-A2). The algorithms were implemented in Python 3 and ran on a MacOs machine. A linear algebra routine was implemented to perform matrix inversion and to solve linear systems, using either *LU* decomposition or the iterative *Jacobi* method according to which one was faster in a particular problem. However, where speed was crucial (e.g. matrices of order O(1000)) *NumPy* functions *numpy.linalg.solve* and *numpy.linalg.inv* were used instead [6].

1) Forced Convection: As mentioned in Sec. II-B2, in the case of forced convection the system of coupled equations turns out to be linear. Hence the problem is in the form:

$$\mathbf{M}\ \vec{\mathbf{T}} = \vec{\mathbf{b}} \tag{12}$$

Where $\vec{\mathbf{T}}$ is the unrolled 1D vector of temperatures containing every pixel in the 2D microprocessor-heatsink system and the matrix $\mathbf{M}$ encapsulates the left hand side of eq. (8). Vector $\vec{\mathbf{b}}$ contains the source terms in the right hand side of eq. (8).

2) Natural Convection: In the case of natural convection, γ_N is not linear with temperature, hence leading to a set of coupled non-linear equations. In order to solve them, we tried two different approaches, a *relaxation* method and *Newtons's* method. The first one was obtained by moving the T^* terms in eq. (11) to the right hand side, hence obtaining a linearly dependent left hand side which can be expressed by a matrix-vector multiplication $\mathbf{M}\cdot\vec{\mathbf{T}}$ and a $\vec{\mathbf{b}}_{(\vec{\mathbf{T}})}$ vector on the right which is a function of temperature since it contains the $-\kappa T^*_{i,j}$ terms. This suggested the following iterative algorithm,

$$\vec{\mathbf{T}}^{(n+1)} = \mathbf{M^{-1}}\ \vec{\mathbf{b}}_{(\vec{\mathbf{T}}^{(n)})} \tag{13}$$

where $\mathbf{M^{-1}}$ has to be computed only once during the iterations. This approach turned out to be quite slow in convergence (see Sec. III-A). The multidimensional *Newton's method* was far more efficient: by writing all the terms in eq. (11) on the left hand side, we converted the simultaneous solution of a system of equations to a multi-dimensional root finding problem. This was solved using Newton's algorithm as showed in [5, p. 275]. We note that in this problem the *Hessian* matrix is computed analytically. The detailed implementation can be found in the code [7].

A. Performance

We now compare the performance of the three algorithms introduced in the previous section. The time required to solve the PDE can be split into two tasks, the first one is initialising the system, which requires an iteration through all points of the 2D system in order to fill in the relevant matrices. By memorising relevant information in arrays, this time consuming step is performed only once (see code [7]). Then, there is the actual solution that is performed via a direct or an iterative method. For the latter, we stopped the iteration when the fractional change of the maximum temperature was less than $2\cdot10^{-11}$. Since the first task is common to all three algorithms, in Fig. 3 we show the time taken to perform the solution part as a function of $\vec{\mathbf{T}}$ vector size.

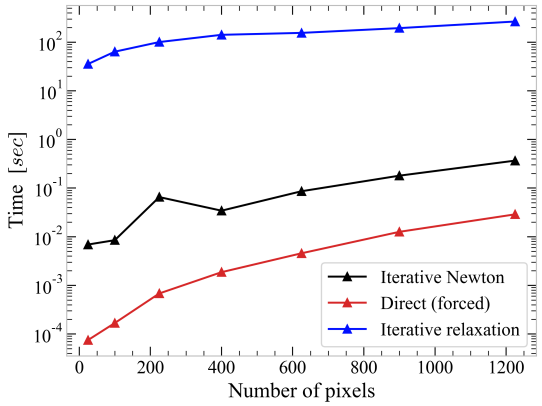


Fig. 3: Shows the time needed for the algorithms to solve the coupled system of equations in the case of Direct method (forced convection), Newton's method (natural convection) and relaxation (natural convection). The algorithms were run using the linear algebra package *numpy.linalg* to achieve better performance.

We can see that the ending part of all plots is approximately linear in log scale, hence suggesting an exponential-like growth. Within the range of pixels we worked on, Newton's method is faster than relaxation, leading an improvement of about three orders of magnitude. However, we also see that the Black line is slightly steeper, hence suggesting that the advantage of Newton's method over the relaxation might not be retained in the long run. Indeed, even though in this context Newton's method converges in a much shorter number of iterations ($O(10)$ against $O(100,000)$ for relaxation), it requires to solve a linear system of equations at every step, which can be computationally demanding in the long run, whereas the relaxation method only requires inverting a matrix once at the beginning of the computation.

In order to validate the results, the algorithms were cross-checked (a relaxation method was coded for the forced convection case too) and the order of magnitude of some of the calculations was compared with an available online simulator [8]. Moreover, we ensured that our results were not resolution dependent by comparing the answers at 4 pixels per mm^2 (used throughout the investigation) and at 16 pixels per mm^2.

IV. Investigation

The aim of the investigation was assessing under what physical settings can a full-power operating microprocessor work below $80\ ^{\circ}\mathrm{C}$ [3], given an outside temperature of $T_{env} = 20\ ^{\circ}\mathrm{C}$.

We analysed the system microprocessor-ceramic plate first on its own and then coupled with a heat sink, under both natural and forced convection conditions. Finally, we simulated a triangular-shaped fin heat sink.

The microprocessor is modelled as a $1\times14\ \mathrm{mm}^2$ rectangle and "infinitely" long in the depth dimension (cfr. approx-

imations made in Sec. II-B) and at top speed it produces $0.5\ \mathrm{W/mm^3}$. The microprocessor is centered on a $2\times20\ \mathrm{mm}^2$ ceramic case which has a supporting function. A heat sink of variable fin number is showed in Fig. 7a. More details about the sizes of the elements can be found in [3]. The conductivities κ of the various components are summarised in the following table.

Component	Thermal Conductivity [W/m K]
Microchip	150
Ceramic case	230
Heat sink	250

A. Without Heat Sink

We simulated the system without heat sink under natural convection conditions. The average temperature across the microprocessor in the steady state case was measured to be $< T >= (7301 \pm 2)\ ^{\circ}\mathrm{C}$, where the error on the mean value is the standard deviation of the temperatures measured at all pixels in the microchip. This is more than five times higher than the melting point of Silicon ($T_{melt} = 1414\ ^{\circ}\mathrm{C}$) [9]. Realistically, we could expect the microprocessor to break before reaching this temperature. Moreover, if for any reason natural convection had to be obstructed (e.g. due to obstacles), this value would be much higher since "dead air" is not a good conductor [2, p. 571]. Hence, it is necessary to use some kind of device with high thermal conductivity to draw heat out of the microprocessor and distribute it on a wider area.

B. Heat Sink and Natural Convection

We simulated the system under the same conditions, this time adding an Aluminium heat sink as showed in Fig. 7a, with a total of 28 fins $30\ \mathrm{mm}$ high. By simulating the system under natural convection, we measured an average temperature of $< T >= (453 \pm 2)\ ^{o}C$ across the microchip.

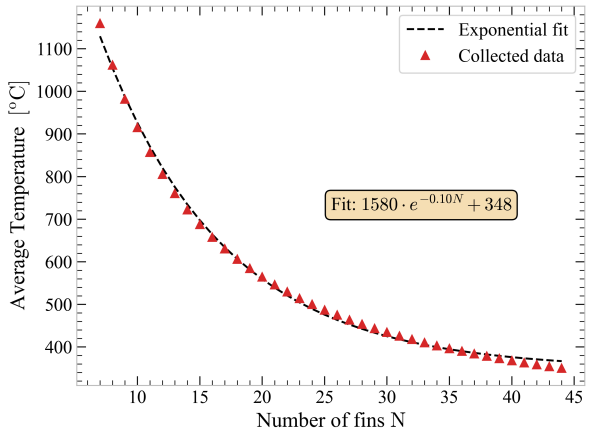


Fig. 4: Shows the average microprocessor temperature as a function of the number of fins N. We note that gradient of the line decreases as N increases suggesting that adding fins gets less and less effective as the number grows up. An exponential fit was performed using *scipy.optimize.curve_fit* and the resulting fit is showed in the yellow rectangle.

Even though we see that the addition of a heat sink leads to a sensible improvement in the average temperature, this is still too high. We note that the distribution of temperatures along the fins is not homogeneous, with the central and warmer fins contributing more to the cooling than the ones at the sides according to eq. (3). Hence, we expect that increasing the number of fins will get decreasingly helpful and this can be seen in Fig. 4 which shows the average temperature as a function of fin number (fin height $30\ \mathrm{mm}$). By performing an exponential fit (using *scipy.optimize.curve_fit*), we can see that the fitted function has an asymptote at $348\ ^{\circ}\mathrm{C}$. Even if this does not precisely reflect the behaviour of the data points, it still suggests that a large number of fins would be required to reach a safe operating temperature ($80\ ^{\circ}\mathrm{C}$). Similar considerations can be done for the fin height.

Since the central part of the heat sink is warmer, we might think that keeping the number of fins N fixed but reducing the fin spacing would lead to lower temperatures, as there would be a higher density of fins at the center. However, it is also true that reducing the fin distance translates into reducing the total surface exposed to air since the base which contributes to cooling shrinks too. Hence, this may or may not lead to lower temperatures across the microchip according to the trade off between these two effects. In the particular case of Fig. 7b (fin height: $30\ \mathrm{mm}$), increasing the fin distance to $b = 2\ \mathrm{mm}$ leads to a higher average temperature. However if we consider the same system with a fin height of $20\ \mathrm{mm}$, then more importance is given to the base of the heat sink and having a fin distance of $b = 2\ \mathrm{mm}$ turns out to be more convenient.

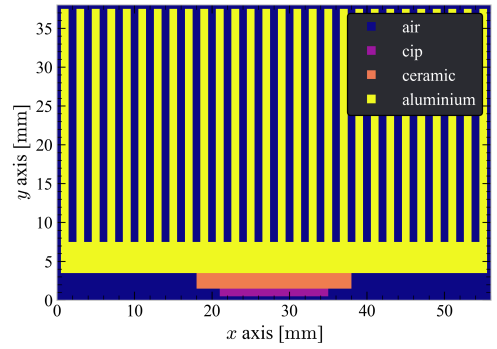
C. Heat sink and Forced Convection

In the previous section we came to the conclusion that natural convection alone does not allow to achieve good operating temperatures if we want to keep the heat sink within practically useful sizes. We now analyse what happens when a fan is employed to force air through the structure. In this case the boundary condition is given by the formula for forced convection in Sec. II-A2, in which we took wind velocity $v = 20\ m/s$. By considering the same system showed in Fig. 7a, we obtained an average temperature across the microchip of $< T >= (79.9 \pm 2)^{\circ}\mathrm{C}$, which we accept although the standard deviation implies that there are points inside the microchip where the temperature is higher than $80^{\circ}\mathrm{C}$.

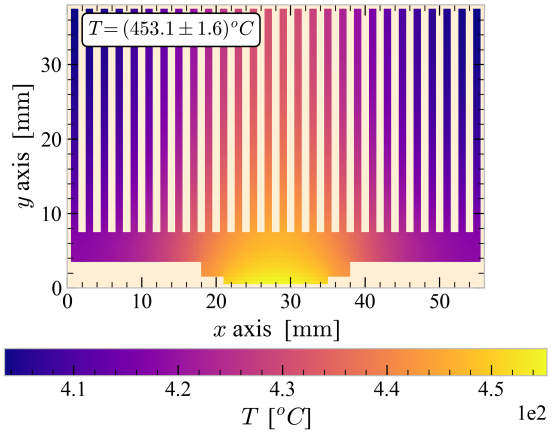
The choice of the number of fins and heights is degenerate: there are multiple combinations that allow good working temperatures. One selection criterion might be the one of choosing the shape that optimizes the volume occupied by the microchip. In Fig. 6 we show the average temperature across the microprocessor for a combination of fin heights and fin numbers. We can indeed notice that the optimal area within the safe-temperature zone (not white-bordered cells) is achieved for a fin height of $30\ \mathrm{mm}$ and fin number of 28 (as the one showed in Fig. 7a).

D. Shape of fins

Another improvement to the heat sink might be achieved by changing the shape of the fins. Taking inspiration from



(a)



(b)

Fig. 5: Top: diagram showing the dimensions of the system composed by microprocessor (violet on the bottom), ceramic case (Orange), heat sink (Yellow) and surrounded by air (Blue). The heat sink is composed by 28 fins 30 mm high. Bottom: shows the heat map obtained by solving the sourced steady state diffusion equation for the system in the above frame with natural convection boundary conditions. The average and standard deviation of the temperatures measured across the microprocessor is reported in the white box and it is above the operating temperature of 80°C.

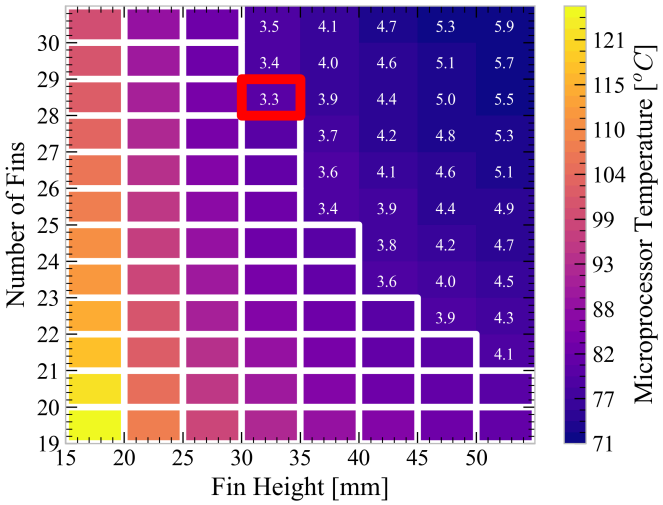


Fig. 6: The plot shows the average temperature across the microprocessor for variable number of fins and fin height in the heat sink. White bordered cells are above the maximum operating temperature of 80°C. Numbers in the top right are proportional to the sectional surface area of the system. The red bordered cell shows the combination of fin height and fin number that minimises the area while keeping the average temperature below 80°C.

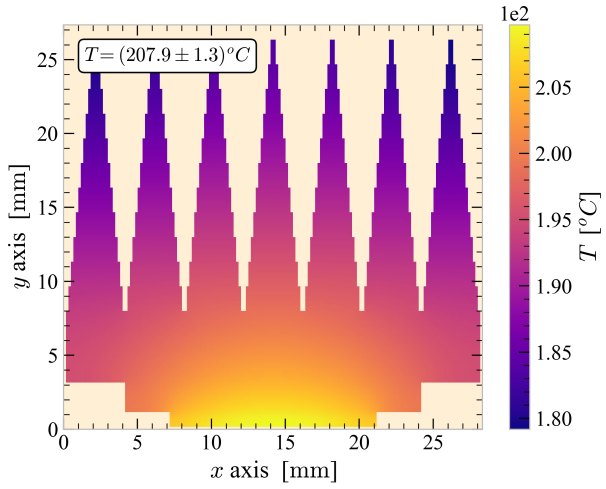
the work of Chen [10], we compared the simulations run on rectangular and triangular shaped fins. Fig. 7 shows that for the dimensions we chose, rectangular fins achieve better results than triangular ones. Indeed whereas a triangular shape leads to a higher area exposed for fin, it also has a larger thermal resistance since the width of the fin decreases hence obstructing the heat flow. However, this answer might be case dependent and more investigation at different relative dimensions is needed before drawing a conclusion.

V. Conclusion

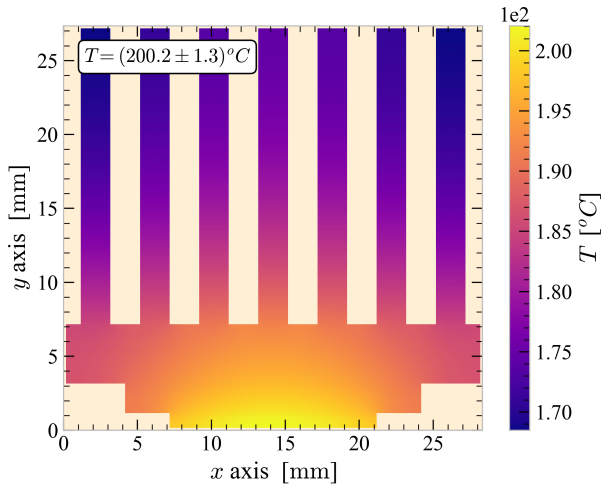
We showed an approach to set the steady state heat equation numerically (Sec. II-B2) and we showed how a direct and iterative methods can be used to solve for the heat map (Sec. III). We then applied this technique to analyse the temperature profile within a full-speed operating microprocessor and we came to the conclusion that both a heat sink and a fan are required to achieve operating temperatures below 80°C whilst limiting the size of the sink. We finally showed how these algorithms can be applied to different shaped fins and that within our model rectangular fins appear to be more effective than triangular ones. All of the above results should be validated experimentally and we further note that our approach does not directly describe fluid dynamic effects nor tries to model whether a certain fin pattern is suitable for the air pump to work properly. The effect of these considerations might be object of future investigations.

References

[1] Mohammad Reza Salimpour, Morteza Sharifhasan, and Ebrahim Shirani, "Constructal Optimization of Microchannel Heat Sinks With Noncircular Cross Sections," vol. 34:10, pp. 863–874, Apr. 2013.
[2] R. A. F. Hugh D. Young, *University Physics*, fourteenth ed. Pearson, 2016.
[3] Mark Schott and Paul Dauncey, "Computational Physics 2020/21 Projects: Submission Guidelines," 2020.
[4] K. F. Riley, M. P. Hobson, S. J. Bence, *Mathematical methods for Physics and Engineering*, 3rd ed. Cambridge University Press, 2006.
[5] Mark Newman, *Computational Physics*, first (revised and expanded) ed. University of Michigan, 2013.
[6] Opensource, "numpy.linalg." [Online]. Available: https://numpy.org/doc/stable/reference/routines.linalg.html
[7] Lorenzo Versini, "Computational Physics Code," 2020.
[8] "Heatsink calculator." [Online]. Available: https://www.heatsinkcalculator.com/blog/
[9] "Royal Society of Chemistry: Silicon." [Online]. Available: https://www.rsc.org/periodic-table/element/14/silicon



(a)



(b)

Fig. 7: Shows two heat sinks with 7 fins $25\ \mathrm{mm}$ high, but with different shape. The resolution used was 36 pixels per mm^2. We note that the triangular fins heat sink reaches an average temperature across the microprocessor which is $7^{\mathrm{o}}\mathrm{C}$ higher than in the rectangular shaped case. This shows that in this case the benefits brought by having a larger surface exposed to air cannot compensate for the higher thermal resistance caused by the shrinking of the fins on the top.

[10] Yongping Chen, Chenbin Zhang, Mingheng Shi, and Jiafeng Wu, "Three-dimensional numerical simulation of heat and fluid flow in noncircular microchannel heat sinks," vol. 36, no. 9, pp. 917–920, Nov. 2009.