corintis

Simulation Hackathon write-up

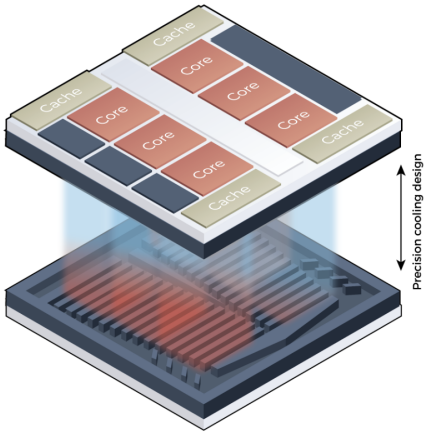
---

1. Challenge Description

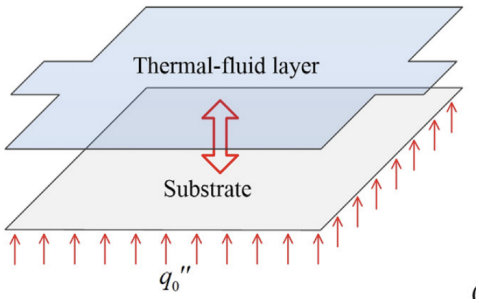
- You will find in the script *stokes.py* a real life example of how to use topology optimization to design 2D microfluidic devices and cool down a microchip based on a PDE-constrained minimisation problem. Currently, the algorithm heavily relies on knowledge from NumCSE, NumPDE and Advanced NumCSE, but you are free to test other methods if they are not familiar/don't like the current problem definition.
- This script will serve as base camp for you to improve the method and/or the design by writing their own code and run it locally on your machine. You will also have access to a google cloud platform instance during the challenge.
- You will be able to explore different methods to improve the code or write your own algorithm from scratch. You are free to explore in the direction that you want. We will describe the criteria of evaluation at the beginning of the challenge.
- We expect you to improve the code along three different axis:
  - Make it faster
    - Improve the simulation and/or optimization algorithm
    - Apply Machine Learning and Reduced Order Models to accelerate the optimization. A dataset for training will be provided.
    - (for example bridging C++ from python)
  - Make it physically better
    - Push to higher Reynolds numbers
    - Implement diverse turbulence models
  - Make it bigger
    - Smart meshing strategies
    - GPU implementation
    - Multi-scale optimisation (multigrid)
- If you don't like any of the ideas that are proposed, you are allowed to go in any direction that you want, and might for example work on cool visualisations, statistical analysis tool of the designs and flow/thermal outputs or even solving a 3D problem.

2. Cooling of Microfluidic devices

The cooling of electronic chips is of central importance in HPC and is responsible on average of 40% of the energy consumption of modern data centers. The goal of this challenge is to design optimised microfluidic devices that have a high heat extraction coefficient to make future electronic chips more performant and sustainable. In this challenge, we will consider microfluidic devices that are directly inside of the chip, as can be seen in the figure below.



(a) A so called "in-chip" microfluidic device, optimised for the heat map of the chip



(b) The conceptual 2 layers model. Note that substrate and channels are inversed with respect to the figure on the left.

The goal of the optimisation is to reduce the peak temperature hot-spots in the chip by strategically creating an efficient topology that:

- Maximise solid contact in the areas where there are hot-spots. Like capillaries in your hand/feet, small channels allow for a high heat transfer coefficient.
- Minimize solid contact in the areas where there are no hot-spots, as to reduce the pressure drop that the fluid will encounter there. This zones are like very large veins close to your heart that can make a lot of blood flow, but have very low heat transfer transfer coefficient.

Broadly speaking, one wants very tiny channels around the hot-spots (capillaries) and very large channels in the lower temperature zones (big veins) to maximise the heat transfer while keeping a reasonable pressure drop. An example of such an optimisation can be seen in figure 1.

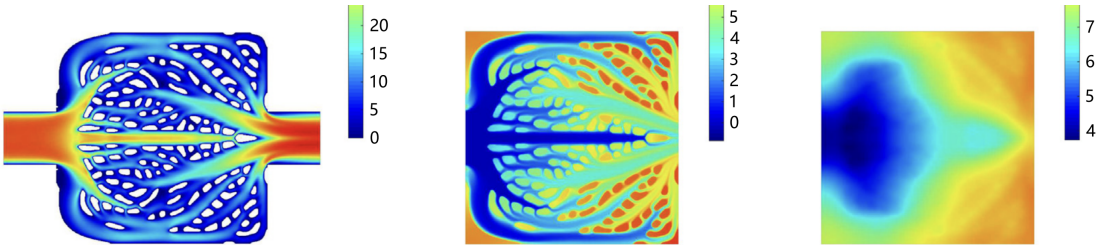


Figure 1: An optimised microfluidic design. The left plot is the velocity magnitude, the middle the temperature in the channel layer, and the right the temperature in the substrate layer. In the left plot, white zones correspond to solid regions

2.1. PDE-constrained optimisation problem

The problem can be formally defined as a PDE-constrained optimisation problem, known in the mathematics community as an inverse problem. Two PDEs are involved: the steady state Navier-Stokes equations for the fluid flow inside of the device, and an advection-diffusion PDE for the thermal problem (the conjugate heat equation). We further define a scalar field $\rho(\mathbf{x}) : \Omega \to \{0.0, 1.0\}$ that is 0.0 where we have a channel wall, and 1.0 where we have a fluid region to represent the topology. This field is called the *design variable*. The solution of the steady state Navier-Stokes equations $\mathbf{up}(\mathbf{x}; \rho(\mathbf{x})) : \Omega \to \mathbb{R}^3$ (u, v, p) and the solution

of the thermal equation $\mathbf{t}(\mathbf{x}; \rho(\mathbf{x})) : \Omega \to \mathbb{R}^2$ (substrate temperature, channel temperature) will then depend on this grid-like parameter as it defines the topology of the problem. An example of how this field could like can be seen in figure 2.

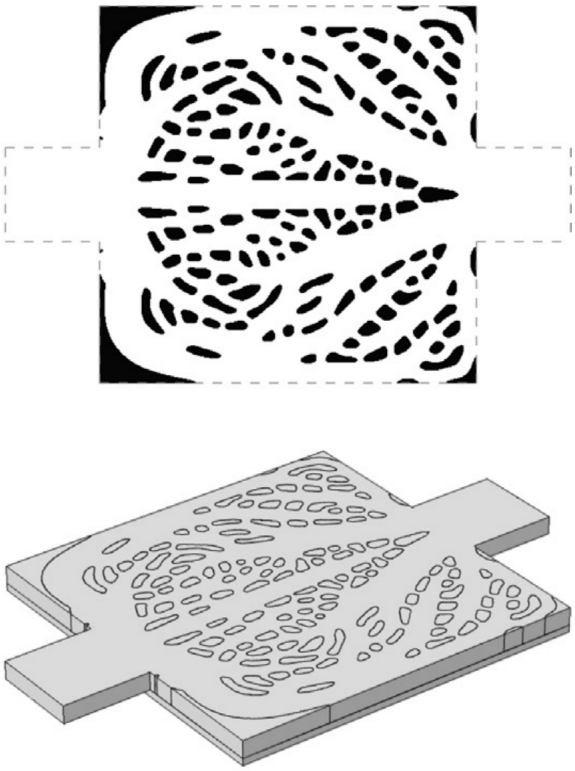


Figure 2: Design Variable $\rho(\mathbf{x}) : \mathbb{R}^2 \to \{0.0, 1.0\}$
and its 3d representation.

We then consider an objective functional (a mapping that takes in a function and outputs a scalar value) $J(\rho) : L^2(\Omega) \to \mathbb{R}$ that represents the maximum temperature of our solution. The goal of the optimisation is then to minimize this functional. Overall, the optimisation algorithm can be summarized as:

Algorithm 1 Topology Optimisation Algorithm

1. Define an initial guess for the topology

$$\rho(\mathbf{x}) : \Omega \rightarrow \{0.0, 1.0\}$$

2. Compute the steady-state Navier-Stokes solution

$$\mathbf{up}(\mathbf{x}; \rho(\mathbf{x})) : \Omega \rightarrow \mathbb{R}^3 \ (u, v, p)$$

3. Compute the two layer thermal solution for the conjugate heat equation (advection-diffusion equation), where the heat is convected by the fluid solution in the channel layer.

$$\mathbf{t}(\mathbf{x}; \rho(\mathbf{x})) : \Omega \rightarrow \mathbb{R}^2 \text{ (substrate temperature, channel temperature)}$$

4. Compute the maximum temperature in the substrate layer

$$J(\rho) : L^2(\Omega) \rightarrow \mathbb{R}$$

5. *Update your design* as to minimize the objective function and loop back to (2) until some convergence criteria is achieved.

The question is then the following: how does one update the design? An easier intuition is to start from the discrete setting. Let's now assume some discretisation of our functions (for example a grid based one, where you would keep each of the values of your function in a vector of size $N = N_x * N_y$), such that instead of handling objects in the infinite dimensional space function space $L^2(\Omega)$ we go back to the finite settings of $\mathbb{R}^N$. Our functional becomes then an objective function $J(\rho) : \mathbb{R}^N \rightarrow \mathbb{R}$, that maps between the design space and the maximal substrate temperature. There are then a variety of algorithms to minimize this N-dimensional objective function, among which the most intuitive one is the gradient descent. The idea is to compute the *derivative* of the objective function J with respect to ρ, and then tweak the current ρ (=our current topology) in the (negative) gradient direction such that we achieve a better score. This would stay an "usual" optimisation problem if we didn't have an additional constraint: we have to minimize the objective $J(\rho)$ while keeping the physics satisfied. We will for now assume that *keeping the physics satisfied* is equivalent to solving some general non-linear equations $\mathbf{g}(\mathbf{x}, \rho) = \mathbf{0}$. In explicit, we can write our problem as

$$\min_{\rho \in L^2(\Omega)} J(\rho) \tag{1}$$

$$\text{subject to } \mathbf{g}(\mathbf{x}, \rho) = \mathbf{0} \tag{2}$$

For a given PDE, as the Navier-Stokes equations or the conjugate heat equation, one can write it into its weak form $F(u) = 0$. This formulation allows to see that a numerical solving of a PDE is equivalent to a minimisation problem on an infinite dimensional function space - that will always be reduced to some finite dimensional problem through *discretisation* (for example a basis functions expansion). Eq. 1 should make more clear why we call this a *PDE-constrained optimisation problem*.

Now that the problem is fully formally defined, one can move forward with trying to find its solution. There exists a variety of algorithm to solve such a problem, and there is no one fit-for-all solution method. The method that is used to solve this problem in the script *stokes.py* is called the method of moving asymptotes (MMA), which requires to

compute the gradient of the functional $J(\rho)$. In topology optimisation, the adjoint method is often used to compute this quantity due to the high number of input parameters and the low-dimensionality of the functional.

3. Details of the model

3.1. The porous 2D incompressible steady-state Navier-Stokes equation

$$\frac{6}{7}\rho_l \mathrm{div}(\boldsymbol{u}\otimes\boldsymbol{u}) = -\boldsymbol{\nabla}p + \mathrm{div}(2\mu_l\epsilon(\boldsymbol{u})) - (\frac{5\mu_l}{2H_t^2} + \alpha(\rho))\boldsymbol{u} \tag{3}$$

$$\mathrm{div}(\boldsymbol{u}) = 0. \tag{4}$$

$$\alpha(\rho) = \frac{1-\rho}{1+q_k\rho}(\alpha_s - \alpha_f) + \alpha_f \tag{5}$$

Here, ρ_l is the liquid density and μ_l the liquid viscosity, both of which are assumed to be constant. The term $\epsilon(\boldsymbol{u}) = \frac{1}{2}(\boldsymbol{\nabla u} + (\boldsymbol{\nabla u})^T)$ is the viscous strain rate tensor. The term in 8, $\frac{5\mu_l}{2H_t^2}\boldsymbol{u}$, is referred to as the "wall drag" or "plate drag" term, representing the impact of adjacent vertical planes above and below the fluid, which hinder the flow. This term originates from the derivation of the viscous strain tensor in the 2D model. α_s and α_f represent the inverse permeability of the solid and fluid respectively, and thus should be 1.0 and 0.0 theoretically. For numerical reasons, we choose α_s to be a very small number, $\alpha_s = H_t/H_{\mathrm{tfactor}}$. To be well-defined, the problem should be completed with appropriate boundary conditions.

In the code, you will implement the non-dimensional equation where $\hat{u} = \frac{u}{U}$, and the spatial coordinates have been rescaled by L, $\hat{x} = \frac{x}{L}$, $\hat{y} = \frac{y}{L}$, $\hat{p} = \frac{p}{\rho_l U^2}$, $Re = \frac{\rho_l U L}{\mu_l}$:

$$\frac{6}{7}\mathrm{div}(\hat{\boldsymbol{u}}\otimes\hat{\boldsymbol{u}}) = -\boldsymbol{\nabla}\hat{p} + \mathrm{div}(\frac{2}{\mathrm{Re}}\epsilon(\hat{\boldsymbol{u}})) - \frac{1}{Re}(\frac{5L^2}{2H_t^2} + \hat{\alpha}(\rho))\hat{\boldsymbol{u}} \tag{6}$$

$$\mathrm{div}(\hat{\boldsymbol{u}}) = 0. \tag{7}$$

For now, the code implements the incompressible stokes equations (good approximation for small Reynolds number)

$$-\boldsymbol{\nabla}\hat{p} + \mathrm{div}(\frac{2}{\mathrm{Re}}\epsilon(\hat{\boldsymbol{u}})) - \frac{1}{Re}(\frac{5L^2}{2H_t^2} + \hat{\alpha}(\rho))\hat{\boldsymbol{u}} = 0 \tag{8}$$

$$\mathrm{div}(\hat{\boldsymbol{u}}) = 0. \tag{9}$$

3.2. The 2 layer thermal model

The velocity obtained from 8 is then used as input in a double-layer model system of equations:

$$\begin{cases} \frac{2}{3}\rho_l C_f(\boldsymbol{u}\cdot\boldsymbol{\nabla}T_t) - \mathrm{div}(\frac{49k_t}{52}\boldsymbol{\nabla}T_t) - \frac{1}{2H_t}\frac{h_b h_t}{h_b+h_t}(T_b - T_t) = 0, \\ -\mathrm{div}(\frac{k_b}{2}\boldsymbol{\nabla}T_b) + \frac{1}{2H_b}\frac{h_b h_t}{h_b+h_t}(T_b - T_t) - \frac{1}{2H_b}q_{in} = 0. \end{cases} \tag{10}$$

where ρ_l (liquid density), C_f (heat capacity of the fluid), H_t and H_b (half-thicknesses of the channel and substrate layers), $\boldsymbol{u}$ (fluid velocity), k_t and k_b (the thermal conductivities of the channel and die layers) and q_{in} (heat source in the die layer) are inputs of the problem while T_t and T_b (average temperatures in the channel and die layers) are the unknowns to

be computed. In general subscript _t stands for the top (channel) layer and subscript _b for the bottom (substrate) solid layer. Additionally, input variables h_t and h_b are defined as :

$$h_t = \frac{35k_t}{26H_t}, \quad h_b = \frac{k_b}{H_b}. \tag{11}$$

Of course the problem 12 should be completed with appropriate boundary conditions. (isolated system and define the inlet temperature where the velocity is entering the domain typically)

In the code, we use the modified the equation to account for the scaling of the mesh and the navier-stokes problem, where the mesh has been rescaled by a factor L and the navierstokes problem outputs a non dimensional flow velocity $\hat{u} = \frac{u}{U}$

$$\begin{cases} \frac{2}{3}\rho_l C_f \frac{U}{L}(\hat{\boldsymbol{u}} \cdot \boldsymbol{\nabla} T_t) - \frac{1}{L^2}\mathrm{div}(\frac{49k_t}{52}\boldsymbol{\nabla} T_t) - \frac{1}{2H_t}\frac{h_b h_t}{h_b + h_t}(T_b - T_t) = 0, \\ -\frac{1}{L^2}\mathrm{div}(\frac{k_b}{2}\boldsymbol{\nabla} T_b) + \frac{1}{2H_b}\frac{h_b h_t}{h_b + h_t}(T_b - T_t) - \frac{1}{2H_b}q_{in} = 0. \end{cases} \tag{12}$$

4. Base ideas

Currently the code implements a stokes-conjugated heat equation. We would recommend for you to start by transforming it to a Navier-Stokes - conjugated heat code. You can then start experimenting along the three axis that are described at the beginning of this document. The main reference that you can use (for details about algorithm and model) is this paper about topology optimization of microchannel heat sinks using a two-layer model.

Here are the kind of go to ideas that you could try to explore:

- Make the viscosity dependent on the temperature of the fluid.
- Add a concentration advection-diffusion PDE to simulate a mixture of two fluids that have two different viscosities.
- Train a ML model that can replace the forward simulation step
- Experiment with different optimisation algorithms
- Use a multigrid solver for the forward step
- Make a cool visualisation tool for the output / interactive app where topology optimisation is run continuously in the background
- ...